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

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

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація комп’ютерної системи управління АЕС критичного застосування”

Виконав здобувач вищої освіти
ІІ курсу, групи КН-22М-2
ОПП «Комп’ютерні науки»
спеціальності 122 «Комп’ютерні науки»
_____ Вінтенко Б. Ю.
«____» _____ 2023 р.

Керівник проекту
доктор технічних наук, професор
_____ Смірнов О.А.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _12 _____*"Інформаційні технології"*_____

Спеціальність 122 *"Комп'ютерні науки"*

Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерні науки"*_

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

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Вінтенку Борису Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація комп'ютерної системи управління АЕС критичного застосування*

2. Керівник роботи *Смірнов Олексій Анатолійович, доктор. техн. наук, професор*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 33-13 від 04.08.2023 року

3. Строк подання студентом роботи до захисту *10.12.2023 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація комп'ютерної системи управління АЕС критичного застосування*

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

1. Призначення та область використання.	*6. Наукова новизна.*
2.Перегляд аналогічних існуючих систем.	*7. Економічна ефективність розробленої програми.*
3. Опис і обґрунтування проектних рішень.	*8. Заходи з охорони праці та техніки безпеки.*
4. Етапи програмування системи.	*9. Висновки.*
5. Впровадження системи в промислову експлуатацію	

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

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Показники економічної ефективності	*1 аркуш*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В.	05.10.2023	14.11.2023
Охорона праці	Оришака О.В.	06.10.2023	16.11.2023

7. Дата видачі завдання « 6 » вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2023 р.	
3.	Розробка моделі компонентів	20.10.2023 р.	
4.	Розробка структур даних	25.10.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2023 р.	
6.	Програмування алгоритмів	10.11.2023 р.	
7.	Розрахунок економічної ефективності	13.11.2023 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2023 р.	
9.	Оформлення ПЗ	17.11.2023 р.	
10.	Попередній захист роботи	10.12.2023 р.	

Дата видачі завдання
« 6 » вересня 2023 р.

Підпис керівника
_____ Смірнов О. А.
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2023 р.

Підпис здобувача
_____ Вінтенко Б. Ю.
(прізвище та ініціали)

АНОТАЦІЯ

Вінтенко Б. Ю. Дослідження та програмна реалізація комп'ютерної системи управління АЕС критичного застосування. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи аварійного та попереджувального захисту реактора на АЕС.

Метою розробки є дослідження вимог до програмного забезпечення, яке використовується в комп'ютерних системах контроля і управління технологічними процесами на атомних електростанціях, і викладення досвіду розробки і інтеграції в ПТК цього програмного забезпечення.

Об'єктом дослідження є розробка програмного забезпечення, яке використовується в комп'ютерних системах контроля і управління технологічними процесами на атомних електростанціях.

Предметом дослідження є методи та галузеві стандарти розробки програмного забезпечення критичного застосування та їх практичне застосування під час реалізації програмного забезпечення, що входить до складу ПТК управління технологічними процесами на АЕС, і який направлений зміну умов спрацювання керуючої системи безпеки.

Методи дослідження базуються на вимогах нормативних документів, що стосуються до розробки програмного забезпечення, важливого для безпеки, та методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи аварійного та попереджувального захисту реактора на АЕС.

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

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

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

Програму розроблено в середовищі C++.

Ключові слова: комп’ютерні науки, програмне забезпечення критичного застосування, АЕС

ABSTRACT

Vintenko B.U. Research and software implementation of the safety-critical computer system for nuclear power plants. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this graduation thesis for the second (master's) level of higher education, software is developed, which is intended for the reactor safety protection system used on nuclear power plants.

The purpose of the development is the research and software implementation of the reactor safety protection system used on nuclear power plants.

The object of the research is the software development process for the reactor safety protection system used on nuclear power plants.

The subject of the study is the methods of software development for the reactor safety protection system used on nuclear power plants.

Research methods are based on methods of requirements analysis, methods requirements specification design, methods of software development.

The result of the work is the software implementation of the reactor safety protection system used on nuclear power plants.

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

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

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

The program is developed in the C++ environment.

Keywords: computer science, safety-critical software, NPP

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

АЕС	–	атомна електростанція
АЗ	–	аварійний захист
АСУ ТП	–	автоматизована система управління технологічними процесами
БЩУ	–	блочний щит управління
БВА	–	блок введення аналоговий
БВД	–	блок введення дискретний
БДН	–	блок діагностики
БРС	–	блок релейних схем
БФЗ	–	блок формування захисту
КСУ	–	комп’ютерна система управління
МКУ	–	модуль корекції уставок
ПЗ	–	програмне забезпечення
ПТК	–	програмно-технічний комплекс

ВСТУП

Актуальність теми. Ядерна енергетика залишається найважливішим постачальником електроенергії України. Але безпека цієї галузі стосується не тільки державних питань, але і міжнародних організацій. В наслідок цього вводяться в дію «Конвенції про ядерну безпеку» [1], які передбачають розробки Державних програм підтримки ядерної безпеки. Життєвий цикл безпечного використання атомних електростанцій, що наведений в ряді нормативних документів (зокрема НП 306.2.141 [2]), передбачає проведення модернізації конструкцій, систем і елементів. Основною задачею модернізації є підтримка і підвищення проектного рівня безпеки АЕС, виходячи з вимог норм, правил та стандартів з ядерної та радіаційної безпеки, міжнародної практики та досвіду експлуатації. Одним із пунктів модернізації АЕС є розробка модернізованих систем контролю і управління, до складу яких входять технічні і програмні засоби.

Так як ядерна енергетика відноситься до сфери діяльності важливої для безпеки, проектування систем контролю і управління є складною задачею із-за вимог безпеки.

Важливість підвищення безпеки АЕС вимагає використання новітніх наукових досліджень, сучасних методів обґрунтування безпеки використання програмних кодів, застосування досвіду і рекомендацій міжнародних організацій, наприклад МАГАТЕ. Програмне забезпечення, що використовується на АЕС, повинно проходити повну валідацію і атестацію перед його застосуванням в експлуатації. Окрім вимог безпеки, до програмного забезпечення пред'являють інші вимоги, наприклад, захищеність від несанкціонованого доступу чи зручність використання операторами.

При модернізації АЕС все більшу частину функцій покладають на програмні засоби ПТК. Це обумовлено зручністю використання, гнучкістю в налаштуваннях під особливості конкретного об'єкту, можливістю ведення архіву. Але згідно нормативних документів, що встановлюють вимоги до комп'ютерних

систем управління атомних електростанцій, різні функції, що виконуються системою, мають різні класи і категорії безпеки і, відповідно, різні вимоги.

Однією із задач під час розробки програмного забезпечення для АЕС є визначення категорії безпеки тих функцій, які буде виконувати програмне забезпечення, що розроблюється. Метою дотримання вимог визначеної категорії безпеки направлене на розробку програмного забезпечення високого ступеня надійності.

При розробці програмного забезпечення для комп'ютерних систем контролю і управління АЕС треба дотримуватися наступних принципів:

– дослідження найкращої практики, що була раніше використана;

– вибору методу проектування;

– застосування модульності;

– проведення верифікації на кожному етапі;

– ведення чіткої документації з можливістю легкої перевірки;

– проведення валідаційного тестування.

Велика кількість нормативних документів, що встановлюють вимоги до життєвого циклу програмного забезпечення, яке розроблюється для комп'ютерних систем контролю і управління АЕС, несе інформацію лише про загальні вимоги і принципи розробки. Методи дотримання принципів розробки і вибір способів реалізації полягають на розробника.

Мета й завдання дослідження. Метою роботи є дослідження вимог до програмного забезпечення, яке використовується в комп'ютерних системах контроля і управління технологічними процесами на атомних електростанціях, і викладення досвіду розробки і інтеграції в ПТК цього програмного забезпечення.

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

– визначення функціонального призначення програмного забезпечення;

– огляд існуючих систем-аналогів, технологій, архітектур, програмних рішень;

– визначення категорії, класу безпеки і вимог до програмного забезпечення, що розробляється;

– ознайомлення з комп'ютерною системою контроля і управління, в складі якої повинно функціонувати програмного забезпечення, що розробляється;

– визначення вимог до інтерфейсу користувача, каналів комунікації з технічним обладнанням, кіберзахисту і модифікованості;

– розробка програмного забезпечення, що реалізовує визначене функціональне призначення і дотримується виконання встановлених вимог;

– інтеграцію програмного забезпечення у ПТК.

Об'єктом дослідження є розробка програмного забезпечення, яке використовується в комп'ютерних системах контроля і управління технологічними процесами на атомних електростанціях.

Предметом дослідження є методи та галузеві стандарти розробки програмного забезпечення критичного застосування та їх практичне застосування під час реалізації програмного забезпечення, що входить до складу ПТК управління технологічними процесами на АЕС, і який направлений зміну умов спрацювання керуючої системи безпеки і виконує функції:

– зміни значень контрольованих параметрів, при яких відбувається спрацювання захисту (іменується уставкою);

– ініціації спрацювання захисту оператором;

– блокування спрацювання вибраних умов захисту (здебільшого використовується під час проведення технічного обслуговування енергоблоку).

Методи дослідження базуються на вимогах нормативних документів, що стосуються до розробки програмного забезпечення, важливого для безпеки, та методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– досліджені нормативні документи за якими класифікують функції, що виконуються комп'ютерними системами контролю і управління АЕС;

– визначені категорії і вимоги, що висуваються до програмного забезпечення, що розробляється для ПТК важливих для безпеки;

– окреслена специфікація вимог до проектування ПЗ, важливого для безпеки;

– розроблена методика керування конфігурацією;

– програмне забезпечення реалізоване відповідно до нормативних вимог.

Практична цінність отриманих результатів полягає в тому, що розроблені і запроваджені технічні рішення дозволяють успішно виконувати функції безпеки, покладені на розроблене ПЗ.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Таким чином, дослідження та програмна реалізація комп’ютерної системи керування АЕС, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення програмного забезпечення

Створення програмного забезпечення для інформаційних і керуючих систем, важливих для безпеки АЕС – це частина Державної програми повузлової заміни технічних засобів АСУ ТП, яка створена на основі «Конвенції про ядерну безпеку».

Для магістерської роботи вибрана розробка програмного забезпечення, яке в складі ПТК виконує функції:

– зміни значень контрольованих параметрів, при яких відбувається спрацювання захисту (іменується уставкою);

– ініціації оператором спрацювання захисту (тестування захистів «АЗ», «ПЗ-1», «ПЗ-2»);

– блокування спрацювання вибраних умов захисту (блокування захистів) – здебільшого використовується під час проведення технічного обслуговування енергоблоку).

Введемо ряд понять, які будуть використовуватися в магістерській роботі.

Безпека АЕС – властивість не перевищувати встановлені межі радіаційного впливу на людей і навколишнє середовище.

Верифікація – процес, направлений на підтвердження відповідності встановленим вимогам.

Відмова - подія, що призводить до порушення працездатності системи (компонента) до втрати здатності виконувати одну або декількох необхідних функцій (НП 306.2.202 [3]).

Життєвий цикл - сукупність стадій створення, тестування, впровадження і використання системи (компонента) в інтервалі часу, що починається з моменту розробки концепції й визначення технічних вимог і закінчується під час виведення системи (компонента) з експлуатації у зв'язку з неможливістю або недоцільністю подальшого використання за призначенням (НП 306.2.202).

Захист – функції управління, спрямовані на усунення небезпеки та/або на запобігання виникненню можливої небезпеки.

Інтерфейс - спільна межа двох або більше об'єктів разом із сукупністю правил і вимог до цих об'єктів.

Інтерфейс «людина-машина» – взаємодія людини з інформаційними і керуючими системами («машина»).

Категорія функції - градація, яка характеризує ступінь важливості функції для безпеки на підставі оцінки її призначення та/або наслідків, спричинених невиконанням або помилковим виконанням функції.

Модернізація – покращення характеристик конструкцій, систем і елементів, що направлені на підвищення безпеки, надійності, безвідмовної роботи і техніко-економічних показників.

Оператори/оперативний персонал – персонал, що здійснює експлуатацію АЕС.

Уставка – це задане значення контрольованого параметра, вихідної події, або зовнішнього фактору впливу, досягнення якого є умовою виконання запропонованої дії.

Функціональна безпека - властивість системи (компонента) атомної станції, що полягає у здатності виконувати всі потрібні функції, важливі для безпеки, зберігати потрібні властивості та відповідати заданим характеристикам в усіх передбачених проектом режимах й умовах експлуатації.

Функція безпеки – конкретна мета, яка повинна бути досягнена для забезпечення безпеки.

Програмне забезпечення, що розроблюється для виконання визначених функцій, отримало назву «Програма «МКУ». МКУ – модуль корекції уставок. Тобто це ПЗ направлене на заміну технічних засобів реалізації аналогічних функцій. Корекція параметрів уставок відбувалась з допомогою підстроювальних резисторів – налаштування перетворення контрольованого параметру, ініціювання оператором спрацювання захисту відбувалась за допомогою

фізичних ключів/перемикачів. Блокування спрацювання вибраних умов захисту (блокування захисту) – функція була відсутня і рекомендована при проведенні модернізації.

Основними цілями створення програми «МКУ» є:

– виконання вищенаведених функцій в модернізованому ПТК, що виконує функції аварійного і попереджувального захисту АЕС;

– розробка зручного інтерфейсу «людина - машина» для операторів АЕС;

– ведення архівів операцій, які виконує оператор;

– інтеграція в апаратну частину ПТК;

– комунікація з іншим програмним забезпеченням ПТК.

Програма «МКУ» вирішує наступні задачі:

– запобігання помилки персоналу під час проведення регламентних робіт на АЕС;

– зручності роботи оперативного персоналу;

– контролю обміну даними з технічними засобами ПТК;

– підлаштування під особливості кожного конкретного об’єкту – алгоритми функціонування і «Карта уставок» на різних енергоблоках можуть відрізнятися;

– обмеження доступу несанкціонованого функціонування.

Програма «МКУ» орієнтована на:

– кваліфікований оперативний персонал АЕС, який вивчив принцип дії, структуру і режими роботи ПТК, в складі якого функціонує програма;

– оперативний персонал, що проходить навчання на тренажері «Тренажер системи аварійного і попереджувального захисту для системи управління і захисту реактора лабораторії «Технічне обслуговування і ремонт системи управління і захисту і обладнання контролю нейтронного потоку».

Перелік уставок, що змінюється, і перелік захистів, що блокуються, формується на основі:

– документу «Карти уставок аварійних і попереджувальних захистів» конкретного енергоблоку АЕС;

– технічного завдання на розробку, виготовлення, прийомку і поставку конкретного ПТК;

– документу «Альбом технологічних алгоритмів АЗ-ПЗ».

Тобто в програмі «МКУ» повинна бути закладена можливість змінювати набір змінних уставок і алгоритмів (захистів), що блокуються. Ці змінні дані зберігаються в окремому файлі «*.mky», який редагується програмою-редактором «MKYEditor».

Примітка – Розробка програма-редактор «MKYEditor» не є метою даної магістерської роботи і розглядається в якості інформаційного додатку до розуміння того, що програма «МКУ» є гнучкою і може застосовуватися в інших ПТК, важливих для безпеки.

Захист від помилки персоналу і зручність роботи персоналу досягається наступним чином:

– поле уставки, що змінюється, заблоковані захисти чи кнопка ініціації захисту підсвічуються кольором;

– діапазон змінення уставки обмежується і, в разі введення хибного значення, появляється попереджувальне повідомлення;

– зміни не будуть проведені до підтвердження змін кнопкою «Запис»;

– у вікні блокувань захистів відображається загальна кількість встановлених блокувань.

Для контролю обміну даними з технічними засобами ПТК програма «МКУ» веде статистику переданих і прийнятих пакетів даних.

Обмеження доступу несанкціонованого функціонування реалізовується шляхом зчитування стану апаратного ключа доступу до проведення робіт і парольним доступом користувача до роботи з програмою «МКУ».

1.2 Область застосування

Програмне забезпечення, розроблене у рамках випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти, призначене для використання у складі систем «АЗ-ПЗ», важливих для безпеки на АЕС.

Система «АЗ-ПЗ» розроблена у рамках модифікації системи аварійного і попереджувального захистів АЕС, яка реалізує погоджені з ГКЯРУ наступні заходи: Концептуальне рішення про виконання «Комплексу заходів, що направлені на запобігання відмови управляючих систем», впровадження «Комплексної зведеної програми підвищення безпеки», технічне рішення «Про запровадження додаткових захистів АЗ, ПЗ при заміні обладнання в першому і другому комплектах системи «АЗ-ПЗ». В модифікованій системі аварійний і попереджувальний захисти здійснюються двома комплектами системи «АЗ-ПЗ» по сигналам датчиків.

Розглянута система «АЗ-ПЗ» у відповідності з документом «Альбом технологічних алгоритмів АЗ-ПЗ» формує сигнали «АЗ», «ПЗ-1», «ПЗ-2» – сигнали аварійного захисту і попереджувального захисту першого і другого ступеню. Сигнали захисту поступають в інші підсистеми СУЗ:

– у вигляді вихідних дискретних управляючих сигналів;

– у вигляді вихідних цифрових інформаційних сигналів.

Система «АЗ-ПЗ» вміщує виконання функцій безпеки і нормальної експлуатації. У відповідності з нормативними документами, що регламентують розробку систем для АЕС, вимоги для забезпечення функцій безпеки більш жорсткі, ніж для забезпечення функцій нормальної експлуатації. Тобто програмні і технічні складові частини, які направлені на виконання функцій різного класу безпеки, рекомендується розділяти, бо дотримання всією системою вимог більш жорсткого класу поведе за собою неоправдане збільшення ціни.

Тобто визначення класу безпеки функцій, які будуть виконувати ТЗ чи ПЗ, це одне із важливих питань на етапі створення проекту.

Міжнародна практика показує, що все більше модернізацій АЕС застосовують заміну технічних засобів, як, наприклад, релейні схеми, програмними засобами, які, окрім виконання основних управляючих функцій, дають можливість автоматизувати проведення технічного обслуговування систем управління АЕС. При цьому технічне обслуговування розділяється на декілька видів в залежності від типу, частоти проведення і призначення. Одним із видів застосування програми «МКУ» є автоматизація процесів при проведені технічного обслуговування систем управління і захисту АЕС.

Окрім АЕС, комп'ютерна система контроля і управління, побудована за аналогічними принципами і вимогами, як система «АЗ-ПЗ», може застосовуватися в інших галузях і направленнях. Наприклад, аналогічні технічні і програмні рішення можуть бути застосовані для:

– систем управління і захисту тепловими АЕС;

– систем управління і захисту дослідницьких реакторів;

– систем управління і захисту тренажерів АЕС;

– інших комп'ютерних систем контроля і управління, важливих для безпеки.

Основною метою розробки комп'ютерних систем контроля і управління є:

– визначення та усунення проблем безпеки об'єкту;

– вдосконалення та оптимізація технічного обслуговування, ремонту, випробовувань систем та елементів об'єкту з метою забезпечення їх надійності, яка достатня для підтримання досягнутого рівня безпеки;

– зосередження уваги на системах і елементах, які мають головний вплив на безпеку об'єкту.

Переваги застосування комп'ютерних систем контролю і управління:

– можливість проведення змін без необхідності заміни апаратної частини;

– автоматизують деяку роботу оперативного персоналу, наприклад, ведення архівів проведених робіт під час використання автоматизованої системи;

– оптимізують роботу операторів під час підлаштування перетворення параметрів;

– наочність відображення інформації під час застосування ризик-інформованого прийняття рішень операторами;

– значно зменшують кількість технічних засобів;

– забезпечують зручний і наглядний інтерфейс «людина- машина».

Таким чином, очевидно, що розробка програмного забезпечення, яке відповідає вимогам, приведеним в нормативних документах на розробки систем, важливих для безпеки, вирішує нагальні Державні і Міждержавні питання про підвищення безпечного використання ресурсів.

2 ПЕРЕГЛЯД ІСНУЮЧИХ АНАЛОГІЧНИХ СИСТЕМ

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

Інтерфейси та засоби зміни уставок

Під час експлуатації КСУ часто виникає необхідність вносити корекції до технологічних алгоритмів їх роботи. Для цього в проекті передбачаються параметри, які можуть бути змінені оперативним персоналом в допустимих діапазонах. Такі параметри називаються уставками. Уставки можуть бути задані для різних параметрів, таких, як температура, тиск, рівень рідини, швидкість тощо. Вони є основою для подальшого керування і регулювання процесу.

Інтерфейси користувача для зміни значень уставок в автоматизованих системах керування технологічними процесами (АСУ-ТП) різняться в залежності від конкретного обладнання та програмного забезпечення, що використовується в системі. Однак основні типи інтерфейсів включають:

1. «Графічний інтерфейс (GUI)». GUI - це найпоширеніший тип інтерфейсу для зміни уставок. Він включає в себе вікна, кнопки, меню, графічні елементи та інші компоненти, які дозволяють операторам легко взаємодіяти з системою. За допомогою GUI оператори можуть відображати поточні значення параметрів, вводити нові уставки, налаштовувати графіки та діаграми, а також виконувати інші дії з контролем процесу;

2. «Вбудований інтерфейс». Деякі пристрої можуть мати спеціалізовані елементи керування (кнопки, індикатори), використовуючи які, користувач може проводити корекцію параметрів;

3. «Веб-інтерфейс». Веб-інтерфейси дозволяють операторам взаємодіяти з системою через веб-браузер. Це дозволяє змінювати уставки з будь-якого пристрою з доступом до Інтернету, що робить цей тип інтерфейсу зручним для віддаленого керування технологічними процесами;

4. «Мобільні додатки». Деякі АСУ-ТП мають мобільні додатки, які дозволяють операторам змінювати уставки зі смартфонів або планшетів. Це особливо корисно для операторів, які перебувають в русі або віддалено від центрального контрольного пункту;

5. «Інтерфейс через API». В деяких випадках системи мають програмне API, яке дозволяє інтегрувати їх з іншими програмами або автоматизованими скриптами для зміни уставок. Це особливо корисно для автоматизованого керування і збору даних.

Інтерфейси користувача розробляються з урахуванням специфіки конкретної системи та потреб користувачів. Вони можуть бути спеціалізованими для певних галузей, таких як промисловість, енергетика, транспорт, медицина тощо, і включати різні функціональні можливості для ефективного керування технологічними процесами.

Приклад графічного інтерфейсу зміни уставок приведений на рисунку 2.1.

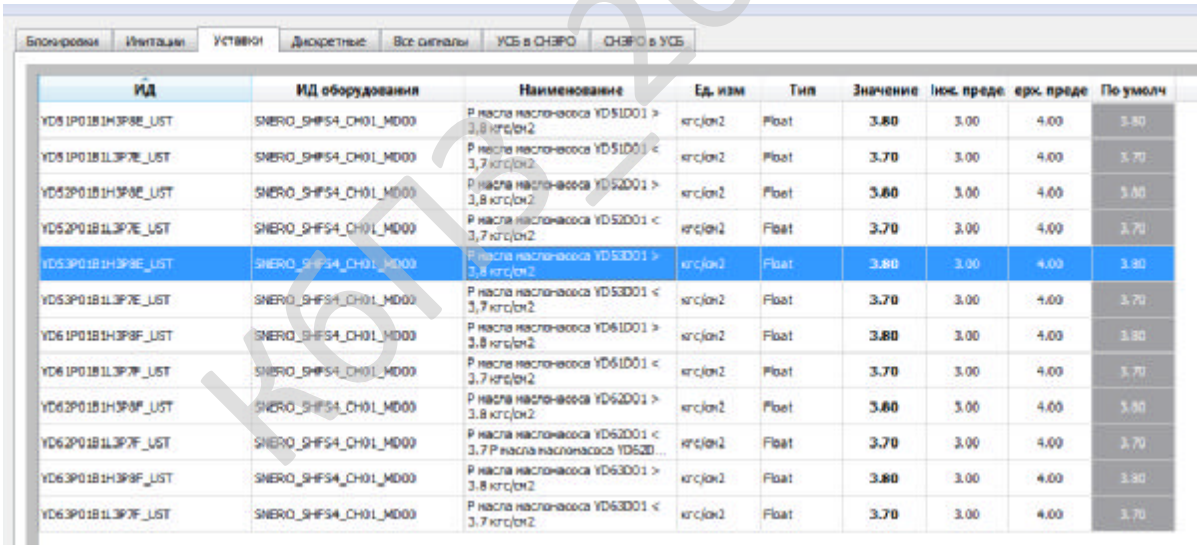

Рисунок 2.1 – Графічний інтерфейс зміни уставок

Приклад вбудованого інтерфейсу зміни уставок приведений на рисунку 2.2.

Рисунок 2.2 – Вбудований інтерфейс зміни уставок

Приклад веб-інтерфейсу зміни уставок приведений на рисунку 2.3.

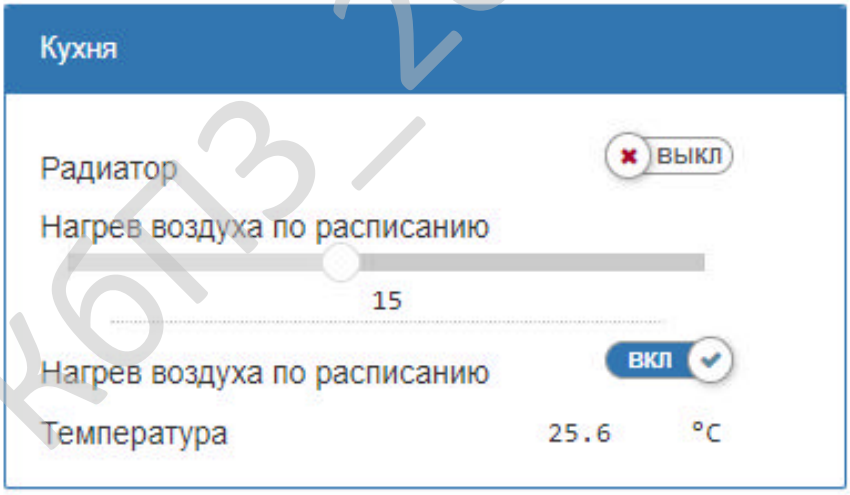

Рисунок 2.3 – Веб-інтерфейс зміни уставок

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

До програмного забезпечення, що розробляється при виконанні магістерської роботи, існують вимоги функціональної безпеки [4], [5]. Функції систем контролю та управління АЕС класифікуються в категорії, що означають

важливість функції для безпеки. Важливість функції для безпеки оцінюється наслідками її відмови тоді, коли необхідне її виконання, та наслідками хибного спрацювання.

Методи класифікації систем на категорії безпеки встановлені в стандарті IEC61226 (Nuclear power plants - Instrumentation and control important to safety - Classification of instrumentation and control functions) [6]. Згідно з ним, існують три категорії КСУ за важливістю для безпеки – «А», «В» та «С»:

– категорія «А» використовується для систем, що грають основну роль в забезпеченні безпеки АЕС і відмова яких може безпосередньо привести до аварійних умов;

– категорію «В» мають системи, що грають додаткову роль в забезпеченні безпечних умов, зокрема у випадку, коли робота функцій цих систем виключить необхідність спрацювання функцій категорії «А». До цієї категорії також відносяться функції, відмова або хибне спрацювання яких може викликати або підсилити аварійний стан. Через наявність ешелону функцій безпеки категорії «А» вимоги до безпеки систем категорії «В» можуть бути не такими строгими, і функціональність таких систем може бути розширеною;

– до категорії «С» відносять функції, що грають додаткову чи допоміжну роль. Вони мають відношення до забезпечення безпеки АЕС, але можуть безпосередньо не використовуватися для реагування на аварійні ситуації.

Програмне забезпечення комп'ютерних систем управління АЕС вимагає високого рівня безпеки та надійності, оскільки навколишнє середовище, в якому вони функціонують, вельми критичне. Міжнародні стандарти IEC60880 [7] та IEC62138 [8] є одними з ключових документів, які регулюють розробку програмного забезпечення для АЕС та визначають вимоги до функціональної безпеки.

Вимоги стандарту IEC60880 застосовуються до програмного забезпечення, що використовуються в КСУ АЕС для виконання функцій безпеки категорії «А». Це ПЗ, яке використовується для функцій захисту, а також

запобігання розвитку проектних аварій і приведення станції в безпечний контрольований стан під час аварійних ситуацій, має бути створене з високим ступенем якості та надійності, пройти повний цикл верифікації та атестації. Мета вимог стандарту IEC60880 – розробка ПЗ, що відповідає критеріям якості та надійності. Вимоги стандарту відносяться до всіх етапів життєвого циклу програмного забезпечення, включаючи специфікацію вимог, проектування, вибір інструментальних засобів та мов, розробку, верифікацію, валідацію, модифікацію тощо.

Стандарт IEC62138 доповнює стандарт IEC 60880, який містить вимоги до програмного забезпечення КСУ, що виконує функції безпеки категорії «А» і застосовується до програмного забезпечення, що використовуються в КСУ АЕС для виконання функцій безпеки категорії «В» і «С». Мета даного стандарту – максимально знизити імовірність прихованих програмних дефектів, що можуть привести до системних відмов через одиничні або множинні програмні відмови (тобто відмови з загальної причини).

Програма, яка розробляється в рамках магістерської роботи, відноситься до класу «В» функціональної безпеки. Це означає, що програма повинна відповідати вимогам, встановленим для цього конкретного класу, забезпечуючи високий рівень безпеки і захисту від можливих помилок та відмов.

Ця програма повинна відповідати наступним критеріям:

– «Підхід "згори-вниз" (Top-Down Approach)» – цей підхід передбачає початок розробки програми з визначення загальних архітектурних рішень та інтерфейсів, а потім поетапно деталізує їх до більш конкретних модулів і компонентів. У контексті управління АЕС, це означає, що спочатку розробляються загальні вимоги до системи, а потім розбиваються на менші функціональні блоки, які реалізуються окремо;

– «Модульність (Modularity)» – програма повинна бути розроблена з використанням модульного підходу, де кожен функціональний блок (модуль) відповідає за конкретну функцію або завдання. Модулі повинні бути незалежними і

можуть бути розроблені та протестовані окремо. Це полегшує підтримку, розширення та відлагодження програми;

– «Модифікованість (Maintainability)» – повинна бути забезпечена можливість легкої модифікації програми, оскільки системи управління АЕС можуть вимагати оновлень та змін через зміни у вимогах, законодавстві чи технологічних досягненнях. Це включає в себе чистий, читабельний код, документацію і правильно спроектовану архітектуру, яка легко змінюється без руйнування інших частин програми;

– «Наявність тестування (Testing)» – повинна бути забезпечена можливість тестування програми, щоб перевірити її правильну роботу та безпеку. Тестування включає в себе розробку тестових наборів для перевірки різних аспектів програми, від функціональності до безпеки. Тестування має бути проведено на всіх рівнях програми, включаючи окремі модулі та інтеграцію між ними.

– «Керування конфігурацією (Configuration Management)» – кожна зміна в програмі має бути акуратно задокументована, а змінені версії мають зберігатися і відслідковуватися для забезпечення стабільності та відновлення попередніх версій у випадку необхідності;

– «Наявність документації (Documentation)» – документація грає важливу роль у розробці програми для систем АЕС. Це включає в себе технічну документацію, яка описує архітектуру, роботу та внутрішню структуру програми, а також настанову користувача та документацію щодо безпеки. Документація має бути докладною та зрозумілою, щоб спростити розуміння та ефективне використання програми.

Керування *конфігурацією* та документація допомагають не тільки забезпечити стабільну роботу програми, але і спростити аудит, підтримку та розвиток програмного продукту у майбутньому.

Вибір мови програмування

Мова програмування C++ є однією з найпопулярніших та найважливіших мов у світі програмування, особливо в контексті промислового програмування. Цей розділ присвячений обґрунтуванню вибору мови C++ для застосування у промисловості та опису факторів, що роблять її настільки популярною у цьому секторі.

Висока продуктивність та ефективність: однією з ключових переваг мови C++ є її висока продуктивність та ефективність. Вона надає програмістам великий контроль над ресурсами комп'ютера, такими як пам'ять і процесор, що дозволяє оптимізувати роботу програм для промислових систем, де кожен цикл процесу і кожен байт пам'яті є критичними.

Низький рівень абстракції: C++ надає можливість працювати на низькому рівні абстракції, що особливо корисно у вбудованих системах та реального часу. Програмісти можуть безпосередньо маніпулювати апаратними ресурсами, що дозволяє оптимізувати швидкодіючі промислові додатки.

Підтримка многопоточності та паралельного програмування: у промисловості часто потрібно обробляти багато завдань паралельно. C++ надає багато засобів для роботи з многопоточністю та паралельним програмуванням, що дозволяє оптимізувати використання багатоядерних процесорів та підвищити продуктивність систем.

Велика кількість бібліотек: C++ має велику кількість бібліотек і фреймворків, призначених спеціально для промислового програмування. Це дозволяє програмістам швидко розробляти додатки та системи, які вимагають певних функціональних можливостей, таких як обробка сигналів, збору даних або керування промисловим обладнанням.

Підтримка системного програмування: у промисловому середовищі часто потрібно розробляти системне програмне забезпечення, таке як операційні системи, драйвери апаратного забезпечення та інші низькорівневі компоненти. Мова C++ відома своєю здатністю до системного програмування.

Широке застосування в індустрії: C++ використовується в широкому спектрі промислових галузей, включаючи авіацію, автомобільну промисловість, медичну техніку, телекомунікації, виробництво та багато інших. Ця універсальність дозволяє програмістам легко переходити між різними проектами та галузями.

Довговічність та стабільність: C++ вже пройшла час і зарекомендувала себе як надійна мова програмування для промислових систем. Програмні продукти, написані на C++, можуть працювати стабільно та довго, що особливо важливо для критичних галузей.

У підсумку, мова програмування C++ відзначається своєю продуктивністю, ефективністю, можливістю працювати на низькому рівні абстракції та великим набором бібліотек. Все це робить її ідеальним інструментом для розробки промислових програмних систем, які вимагають високої надійності та продуктивності.

Для розробки програмного забезпечення обрано середовище розробки Microsoft Visual Studio 2022.

Використання середовища розробки Visual Studio для програмування на мові C++ має численні переваги, які роблять процес розробки зручним і продуктивним. Ось деякі з них:

– інтегроване середовище: Visual Studio надає інтегроване середовище розробки (IDE), яке включає в себе всі необхідні інструменти для програміста. Ви можете працювати зі своїм кодом, збирати, налагоджувати та відлагоджувати програми, всі ці процеси об'єднані в одному зручному інтерфейсі;

– підтримка мови C++: Visual Studio має потужну підтримку мови C++. Вона включає в себе синтаксичне підсвічування, автоматичне доповнення коду, перевірку помилок та підказки, що полегшують написання правильного коду;

– відладка: Visual Studio надає потужні засоби для відлагоджування програм на C++. Ви можете легко встановлювати точки зупинки, аналізувати

значення змінних, використовувати візуальний відлагоджувальник для крокування по коду тощо;

– візуальний редагувальник ресурсів: у Visual Studio є вбудований редагувальник ресурсів, що дозволяє легко створювати та редагувати ресурси, такі, як іконки, курсори, меню і діалогові вікна, що є корисним для розробки графічних інтерфейсів;

– підтримка багатьох платформ: Visual Studio підтримує розробку програм для різних платформ, включаючи Windows, Android, iOS, Linux і багато інших. Це робить його універсальним інструментом для розробки на C++;

– інтеграція з іншими інструментами: Visual Studio легко інтегрується з іншими інструментами, такими як системи керування версіями (наприклад, Git), бази даних, тестові фреймворки та інші. Це полегшує спільну роботу великих розробницьких команд;

– розширюваність: Visual Studio можна розширювати за допомогою різних плагінів та розширень. Ви можете налаштувати середовище розробки відповідно до своїх потреб, додавши необхідні інструменти і функції;

– велика спільнота користувачів і підтримка: Visual Studio має велику та активну спільноту користувачів, що означає, що ви завжди можете знайти відповіді на свої питання та підтримку в разі виникнення проблем.

Узагальнюючи, Visual Studio є потужним інструментом для розробки програм на мові C++. Воно надає ряд переваг, які полегшують роботу програмістам, дозволяючи їм швидше та ефективніше створювати високоякісні промислові програми.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для використання у складі програмно-технічного комплексу системи безпеки реактора, що використовується на АЕС.

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

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

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві у відповідності до стандартів розробки програмного забезпечення критичного застосування. Розробити функціональну та структурну схеми системи;

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

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

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Для розробки програмного забезпечення КСУ АЕС необхідним етапом є проведення аналізу вимог. Вимоги до програмного забезпечення є частиною вимог до КСУ в цілому. Ці вимоги походять з наступних джерел:

– вимоги наукових та проектних організацій, що здійснюють розробку та проектування технологічних процесів на АЕС;

– вимоги експлуатуючої організації АЕС, якою в Україні є державна компанія «Енергоатом»;

– вимоги персоналу АЕС, який здійснює безпосередню експлуатацію та обслуговування КСУ;

– вимоги державних контролюючих органів;

– досвід експлуатації аналогічних систем попередніх поколінь.

Документ, який описує вимоги до програми, називається технічним завданням і є першоджерелом для розробки ПЗ.

В даному розділі будуть описані вимоги до ПЗ «Модуль корекції уставок» («МКУ»), що входять до вимог на розробку системи аварійного та попереджувального захисту реактора (система «АЗ-ПЗ»).

Вимоги до програмного забезпечення розподіляються на наступні групи:

– функціональні вимоги;

– нефункціональні вимоги;

– вимоги до інтерфейсу користувача;

– вимоги до комунікацій;

– вимоги до захищеності;

– вимоги до модифікованості;

– вимоги до переносимості.

Загальні відомості про систему «АЗ-ПЗ»

Програма «Модуль корекції уставок» (далі – «МКУ») використовується у складі систем «АЗ-ПЗ». Система «АЗ-ПЗ» складається з трьох основних шаф, в яких реалізовані незалежні канали аналізу параметрів роботи ядерного реактора та формування сигналів аварійного та попереджувального захисту. Вихідні сигнали спрацювання алгоритмів та сформовані сигнали захисту кожного каналу збираються у мажоритовані вихідні сигнали за принципом «два з трьох» («2/3»). Це означає, що помилкове спрацювання захисту в одному каналі не призведе до видачі хибного сигналу зупинки або обмеження потужності реактора, а також що хибне неспрацювання одного каналу не призведе до відмови у формуванні сигналу захисту при виникненні аварійних умов.

Кожний канал системи «АЗ-ПЗ» містить шасі, що містять наступні основні логічні модулі:

– блоки діагностування («БДН»);

– блоки введення аналогових сигналів («БВА»);

– блоки введення дискретних сигналів («БВД»);

– блоки формування захисту («БФЗ»);

– блоки релейних схем («БРС»).

Блоком, який безпосередньо виконує аналіз вхідних параметрів та формування сигналів захисту згідно технологічних алгоритмів роботи реактора, є «БФЗ». Цей блок містить в собі декілька вбудованих мікропроцесорів типу «MSP-430», що реалізують функції діагностування, математичних перетворень параметрів (наприклад значення тиску до значення температури насичення), а також обробки уставок та блокувань алгоритмів. Безпосередньо обробка вхідних параметрів та технологічних алгоритмів реалізовані на базі програмованої логічної електронної схеми «Cyclone» компанії «Altera» без використання програмного коду.

Технологічні особливості різних енергоблоків, а також різні режими експлуатації ядерного реактора (пуск, робота на потужності, зупинка) можуть

вимагати корегування деяких компонентів алгоритмів захисту. Через це система «АЗ-ПЗ» має забезпечити можливість оперативного корегування уставок (аналогових параметрів) та блокування окремих алгоритмів захисту. Для реалізації цієї можливості необхідна розробка ПЗ високого рівня, яке надасть можливість оператору АЕС змінювати вказані параметри. Ця програма буде виконуватися на панельному комп'ютері, що встановлений в технологічній шафі кожного каналу системи «АЗ-ПЗ».

Клас функціональної безпеки програми «МКУ»

Згідно аналізу стандарту IEC61226, програмне забезпечення категорії «А» має безпосереднє відношення до формування функцій безпеки АЕС. Таким є програмне забезпечення мікропроцесорів, що входять до складу модулів каналу захисту, та проекти ПЛІС модуля «БФЗ», оскільки саме вони відповідають за формування сигналів аварійного та попереджувального захисту реактора.

Програма «МКУ» безпосередньо не виконує функції захисту реактора, проте може впливати на правильність задання значень уставок та технологічних блокувань системи захисту реактора. Виходячи з цього, можна зробити висновок, що ця програма може бути віднесена до класу функціональної безпеки «В», і його розробка має бути виконана згідно вимог міжнародного стандарту IEC62138.

Функціональні вимоги до ПЗ «МКУ»

Програма «МКУ» призначена для виконання наступних функцій:

– задавання значень уставок алгоритмів захисту згідно з переліком;

– блокування окремих алгоритмів захисту згідно з переліком;

– увімкнення режиму тестування вихідних сигналів «АЗ», «ПЗ-1», «ПЗ-2».

Програма повинна обмінюватися інформацією з модулем «БФЗ» відповідного каналу системи «АЗ-ПЗ». Після запуску програми та під час її роботи необхідно постійно зчитувати з модуля «БФЗ поточні встановлені

значення уставок та блокувань алгоритмів і відображати їх в інтерфейсі користувача.

При отриманні від інтерфейсу користувача команди на запис значення уставки або встановлення блокування, програма має надіслати до модуля «БФЗ» команду запису. Після завершення процесу запису програма повинна відобразити нове записане значення параметру.

Після отримання від інтерфейсу користувача команди на тестування вихідного сигналу «АЗ», «ПЗ-1» та «ПЗ-2» програма повинна надіслати до модуля «БФЗ» команду на увімкнення тестування та утримувати цю команду у ввімкненому стані протягом 5 секунд.

Для комунікації програми з модулем «БФЗ» має використовуватися послідовний інтерфейс. Специфікація даного інтерфейсу описана у підрозділі, що стосується комунікації.

Для спрощення роботи оператора програма повинна мати зручний графічний інтерфейс користувача.

З метою забезпечення безпеки програма має реалізовувати алгоритм парольного захисту.

У різних поколіннях систем «АЗ-ПЗ» на АЕС застосовуються панельні комп'ютери різних конфігурацій. В частині систем застосовуються ПК під керуванням операційної системи «Windows CE», в частині - «Windows 7 Embedded» та «Windows 10 Embedded». Необхідно забезпечити функціонування програми у відповідних середовищах роботи.

Вимоги до корекції уставок

В режимі корекції уставок програма повинна постійно зчитувати з модуля «БФЗ» поточні встановлені значення уставок. При отриманні команди на запис від інтерфейсу користувача програма повинна перевірити, чи входить введене користувачем значення до дозволеного діапазону, і в разі входження – відправити команду на запис. У випадку, коли введене значення виходить за межі діапазону, запис такого значення забороняється.

Список уставок має динамічно завантажуватися з файлу проекту.

Типовий список уставок, які можуть змінюватися користувачем, наведений в таблиці 3.1.

Таблиця 3.1 – Перелік уставок, що змінюються програмою

Позначення	Найменування	Діапазон
АЗ t1	Тгн1 > Тгн ном + 8 °С	310.2–329.7 °С
АЗ t2	Тгн2 > Тгн ном + 8 °С	310.2–329.7 °С
АЗ t3	Тгн3 > Тгн ном + 8 °С	310.2–329.7 °С
АЗ t4	Тгн4 > Тгн ном + 8 °С	310.2–329.7 °С
ПЗ t1	Тгн1 > Тгн ном + 3 °С	310.2–329.7 °С
ПЗ t2	Тгн2 > Тгн ном + 3 °С	310.2–329.7 °С
ПЗ t3	Тгн3 > Тгн ном + 3 °С	310.2–329.7 °С
ПЗ t4	Тгн4 > Тгн ном + 3 °С	310.2–329.7 °С
АЗ L1	Lпг1 < Lnom	1330–1900 мм
АЗ L2	Lпг2 < Lnom	1330–1900 мм
АЗ L3	Lпг3 < Lnom	1330–1900 мм
АЗ L4	Lпг4 < Lnom	1330–1900 мм

Вимоги до встановлення блокувань алгоритмів

В режимі блокування алгоритмів програма повинна постійно зчитувати з модуля «БФЗ» поточні встановлені значення блокувань. При отриманні команди на запис від інтерфейсу користувача програма повинна відправити до модуля «БФЗ» команду на увімкнення чи вимкнення обраного користувачем блокування.

Список блокувань має динамічно завантажуватися з файла проекту.

Типовий список блокувань, які можуть встановлюватися користувачем, наведений в таблиці 3.2.

Таблиця 3.2 – Перелік блокувань, що змінюються програмою

Позначення	Найменування
АЗ_1	T < 20 с
АЗ_4	N > Nзад
АЗ_6	dTs < 10°C
АЗ_7	Раз < 140 кгс/см², Тгн > 260 °C
АЗ_8	Раз < 148 кгс/см², N>75%
АЗ_9	dP ГЦН 4-2.5 кгс/см², t < 5 с
АЗ_11	Відключення 1/2, 1/3, 2/4 ГЦН
АЗ_12	Відключення 2/4 ГЦН при N>75%
АЗ_13	S>5 балів
АЗ_18	Тгн > Тном + 8°C
АЗ_23	Відсутні 220 В на 2-х вводах СУЗ
	…
ПЗ-1_1	T < 20 с
ПЗ-1_3	N > Nзад
ПЗ-1_4	Раз > 172 кгс/см²
ПЗ-1_5	Тгн > Тном + 3°C
ПЗ-1_6	Ргпк > 70 кгс/см²
ПЗ-2_1	N > Nзад
ПЗ-2_2	Раз > 165 кгс/см²
ПЗ-2_3	Падіння ОР СУЗ
ПЗ-2_4	ЛЕ > ЛЕдоп
ПЗ-2_5	КК ТВЕЛ < доп
ПЗ-2_6	T < 40 с

Вимоги до тестування вихідних сигналів

В режимі тестування вихідних сигналів програма повинна постійно зчитувати з модуля «БФЗ» стан команд тестування. При отриманні команди на

тестування від інтерфейсу користувача програма повинна відправити до модуля «БФЗ» команду на увімкнення чи вимкнення обраного користувачем вихідного сигналу.

Типовий список вихідних сигналів тестування, які можуть встановлюватися користувачем, наведений в таблиці 3.3.

Таблиця 3.3 – Перелік вихідних сигналів тестування

Позначення	Найменування
АЗ_1	АЗ
ПЗ_1	ПЗ-1
ПЗ_2	ПЗ-2

Вимоги до інтерфейсу користувача

Програма має графічний інтерфейс користувача. Цей інтерфейс повинен відповідати наступним вимогам:

– візуально виділяти обраний користувачем режим роботи: зміна значень уставок, встановлення блокувань, тестування вихідних сигналів;

– відображати список та поточне значення параметрів відповідно до обраного режиму;

– відображати допустимий діапазон введення значень уставок;

– дозволяти користувачеві ввести нове значення уставки для запису;

– дозволяти користувачеві ввести значення блокуванння алгоритму;

– дозволяти користувачеві визначати вихідний сигнал, що тестується;

– перед внесенням змін видавати оператору запит на підтвердження дій;

– відображати стан та кількість пакетів даних, що передані та прийняті від модуля «БФЗ»;

– відображати стан апаратного ключа доступу, яким дозволяється зміна параметрів оператором;

– дозволяти введення даних оператором з сенсорного екрану без використання клавіатури;

– відображати найменування АЕС, номер енергоблоку, та інформацію про виробника.

Вимоги до керування конфігурацією програми

Програма повинна забезпечувати керування конфігурацією. Будь-яка версія програми повинна бути ідентифікована, тобто містити номер версії, номер збірки та дату збірки.

Вимоги до каналу комунікації

Програма повинна взаємодіяти з апаратним забезпеченням системи «АЗ-ПЗ» з використанням послідовного інтерфейсу. Обмін даними програма має здійснювати з модулем «БФЗ». Швидкість обміну даними становить 19200 біт на секунду, 8 біт на байт, без контролю парності.

Ініціатором обміну виступає програма «МКУ», надсилаючи до модулю «БФЗ» пакет наступного формату:

```
struct OutPacket_t
{
  BYTE Size;
  BYTE Command;
  BYTE Param_Reserved;
  short Data;
  WORD Reserved;
  BYTE Crc;
};
```

Поле Size містить розмір пакету в байтах.

Поле Command містить код команди: 1 – «Escape», 2 – «Choice», 3 – «WriteSetting», 4 – «Enter», 5 – «Status»;

Поле Data містить значення уставки, що записується.

Поле CRC містить 8-розрядну контрольну суму пакету.

У відповідь модуль «БФЗ» повинен надіслати до програми «МКУ» пакет наступного формату:

```
struct InPacket_t
{
  BYTE Size;
  BYTE StateCode;
  BYTE ParamNo;
  BYTE Flags;
  short Data;
```

```
    BYTE Reserved;
    BYTE Crc;
};
```

Поле Size містить розмір пакету в байтах.

Поле StateCode містить код режиму меню:0 – «Menu», 1 – «Setting», 2 – «Block», 3 – «Test», 4 – «Edit»;

Поле ParamNo містить номер параметра (уставки, блокування чи тесту), що обраний для читання або запису в даний момент;

Поле Flags містить бітові ознаки стану модуля керування уставками «БФЗ». Біт 0 означає встановлений апаратний ключ доступу, біт 1 – непроініціалізовані значення уставок, біт 2 – непроініціалізовані значення блокувань;

Поле Data містить значення уставки, що прийняте з модуля «БФЗ»;

Поле CRC містить 8-розрядну контрольну суму пакету.

Система команд, яка використовується для взаємодії програми «МКУ» та модуля «БФЗ», використовує принцип системи меню та команд переміщення по ньому.

Після увімкнення живлення «БФЗ» знаходиться в режимі «Меню». При надсиланні до «БФЗ» команду «Choice», процесор «БФЗ» послідовно переходить до пунктів меню «Устави», «Блокування», «Тестування» та відправляє до «МКУ» пакет, в якому в полі ParamNo вказаний поточний пункт меню. Для входу в меню потрібного режиму до «БФЗ» надсилається команда «Choice».

Після входу в потрібний режим («Уставки», «Блокування» чи «Тестування») команда «Choice» використовується для послідовного пошуку потрібного параметру (уставки, блокування чи тесту). Номер поточного параметру приходить в полі ParamNo, поточне значення – в полі Data.

Після знаходження уставки, значення якої потрібно змінити, надсилається команда «WriteSetting», нове значення уставки має міститися в полі Data вихідного пакету. Для підтвердження запису необхідно надіслати команду «Enter».

Після знаходження блокування, значення якої потрібно змінити, надсилається команда «Enter», що інвертує стан поточного блокування. Для підтвердження запису необхідно повторно надіслати команду «Enter».

Після знаходження вихідного сигналу, спрацювання якого потрібно протестувати в режимі «Тестування»,, надсилається команда «Enter», що активує режим тестування. Для підтвердження запису необхідно повторно надіслати команду «Enter». Для утримання сигналу в режимі тестування необхідно надсилати до «БФЗ» команди тестування протягом 5 секунд.

Для переходу по меню режимів на рівень вгору використовується команда «Escape».

Для реалізації контролю цілісності даних до складу кожного пакету входить поле з розміром пакету та контрольною сумою. У випадку, якщо розмір або контрольна сума невірна – пакет не береться до обробки.

Вимоги до кіберзахисту

Для захисту від несанкціонованої зміни параметрів на системі «АЗ-ПЗ» використовується багаторівневий захист. Перший рівень захисту передбачає фізичний доступ до приміщення з системою та відкриття дверей шафи, при цьому на блочному щиті керування (БЩУ) блоку спрацьовує сигналізація. Наступний рівень захисту являє собою апаратний ключ, який оперативний персонал повинен встановити до електронного замка кожної шафи, де функціонує програма «МКУ». Програма «МКУ» має забезпечити власний рівень захисту інформації за допомогою пароля. Пароль має складатися з набору чисел. Для зміни значення уставки, перемикання блокування алгоритму, тестування вихідного сигналу, зміни налаштувань, а також виходу з програми має пропонуватися ввести пароль.

Для аналізу роботи програми програмою має вестися файл журналу, в який мають заноситися дані про невдалі спроби авторизації та інші помилки, що виникли в процесі роботи.

Вимоги до модифікованості

Програма повинна виконуватися на різних АЕС та різних енергоблоках. В зв'язку з тим, що на різних енергоблоках присутні різні технологічні відмінності та може встановлене технологічне обладнання різних виробників, список уставок, блокувань алгоритмів та вихідних сигналів може змінюватися. Через це програма має мати можливість модифікації списку та діапазонів уставок, списку блокувань, сигналів тестування тощо. Для забезпечення цієї можливості дані, які можуть бути змінені, мають зберігатися в окремому файлі проекту. Користувач має мати можливість обирати файл проекту. Програма повинна функціонувати відповідно до списку параметрів, що встановлюються файлом конфігурації проекту.

Загальні вимоги до проектування ПЗ

В об'ємі даної роботи з проектування та розробки ПЗ «МКУ» мають бути виконані основні процеси життєвого циклу (ЖЦ) ПЗ відповідно до стандарту IEC62138:

– розробка чіткої специфікації вимог;

– розробка специфікації ПЗ;

– реалізація ПЗ;

– тестування.

Процес проектування ПЗ має відповідати наступним вимогам:

– керування конфігурацією;

– підхід «згори-донизу»;

– чітка і зрозуміла структура;

– модульність;

– розділення ПЗ та конфігураційних даних;

– можливість модифікації;

– можливість самоконтролю.

Виходячи з опису вимог до програми, першим етапом є створення специфікації вимог у вигляді групи таблиць.

Для реалізації чіткої структури ПЗ, а також підходу «згори-вниз» вимоги будуть розподілені за ієрархічним принципом на окремі групи.

Для кожної групи вимог, крім груп верхнього рівня, буде вказано ідентифікатор «батьківської» вимоги.

Для кожної вимоги в таблиці будуть вказані наступні поля:

– ідентифікатор;

– опис;

– ідентифікатор дочірньої групи вимог («розподіл» до груп);

– тип тестування (верифікація («V»), функціональне тестування («F»), аудит («A»)).

Специфікація вимог

Специфікація вимог відповідає підходу проектування «згори-донизу», що вимагається стандартом IEC62138.

Специфікація вимог до програми «МКУ» приведена в таблицях 3.4-3.15.

Таблиця 3.4 – Специфікація основних вимог

ІД	Найменування	Розподіл	Тестування
RQ.M.01	Задавання значень уставок алгоритмів захисту	RQ.U	F
RQ.M.02	Блокування окремих алгоритмів захисту	RQ.B	F
RQ.M.03	Увімкнення режиму тестування вихідних сигналів	RQ.TR	F
RQ.M.04	Комунікація з модулем «БФЗ»	RQ.COM	F
RQ.M.05	Зручний інтерфейс користувача	RQ.UI	A
RQ.M.06	Захищеність	RQ.SEC	V
RQ.M.07	Підтримка ОС Windows CE та Windows 7/10 Embedded	RQ.OS	V

Таблиця 3.5 – Специфікація вимог до зміни уставок

ІД	Найменування	Розподіл	Тестування
RQ.U.01	Програма повинна постійно зчитувати з модуля «БФЗ» поточні встановлені значення уставок	RQ.COM	F
RQ.U.02	Програма повинна постійно відображати поточні встановлені значення уставок	RQ.TUST	F
RQ.U.03	При отриманні команди на запис від інтерфейсу користувача програма повинна перевірити, чи входить введене користувачем значення до дозволеного діапазону	RQ.TUST	F
RQ.U.04	В разі входження – відправити команду на запис	RQ.COM	F
RQ.U.05	Введене значення виходить за межі діапазону, запис такого значення забороняється.	RQ.TUST	F
RQ.U.06	Список уставок має динамічно завантажуватися з файла проекту	RQ.ENV.01	F
RQ.U.07	Під час виконання зміни значення уставки необхідно заборонити будь-які інші зміни параметрів	RQ.TUST	F

Таблиця 3.6 – Специфікація вимог до встановлення блокувань

ІД	Найменування	Розподіл	Тестування
RQ.B.01	Програма повинна постійно зчитувати з модуля «БФЗ» поточні встановлені значення блокувань	RQ.COM	F
RQ.B.02	Програма повинна постійно відображати з поточні встановлені значення блокувань	RQ.TBL	F
RQ.B.03	Отримані команди на запис від інтерфейсу користувача програма повинна відправити до модуля «БФЗ» команду на увімкнення чи вимкнення	RQ.COM	F
RQ.B.04	Список блокувань має динамічно завантажуватися з файла	RQ.ENV.02	F
RQ.B.05	Під час виконання блокування необхідно заборонити будь-які інші зміни параметрів	RQ.TBL	F

Таблиця 3.7 – Специфікація вимог до тестування вихідних сигналів

ІД	Найменування	Розподіл	Тестування
RQ.TR.01	Програма повинна постійно зчитувати з модуля «БФЗ» стан команд тестування	RQ.COM	F
RQ.TR.02	Отриманні команди на запис від інтерфейсу користувача програма повинна відправити до модуля «БФЗ» команду на увімкнення тестування на 5 секунд	RQ.TTR	F
RQ.TR.03	Під час виконання тестування необхідно заборонити будь-які зміни параметрів	RQ.TTR	F

Таблиця 3.8 – Специфікація вимог до інтерфейсу користувача

ІД	Найменування	Розподіл	Тестування
RQ.UI.01	Програма має графічний інтерфейс користувача	–	A
RQ.UI.02	Програма має містити вкладку «Уставки»	RQ.TUST	A
RQ.UI.03	Програма має містити вкладку «Блокування»	RQ.TBL	A
RQ.UI.04	Програма має містити вкладку «Тестування»	RQ.TTR	A
RQ.UI.05	Програма має містити вкладку «Параметри»	RQ.TPA	A
RQ.UI.06	Відображати стан та кількість пакетів даних, що передані та прийняті від модуля «БФЗ»;	–	A
RQ.UI.07	Дозволяти введення даних оператором з сенсорного екрану без використання клавіатури	–	A
RQ.UI.08	Відображати стан ключа доступу	–	A

Таблиця 3.9 – Специфікація вимог до вкладки «Уставки»

ІД	Найменування	Розподіл	Тестування
RQ.TUST.01	Відображати список уставок	–	F
RQ.TUST.02	Відображати поточне значення уставок	–	F
RQ.TUST.03	Відображати допустимий діапазон введення значень уставок	–	F
RQ.TUST.04	Дозволяти користувачеві ввести нове значення уставки для запису	–	F
RQ.TUST.05	Перед внесенням змін видавати оператору запит на підтвердження дій	–	F

Таблиця 3.10 – Специфікація вимог до вкладки «Блокування»

ІД	Найменування	Розподіл	Тестування
RQ.TBL.01	Відображати список блокувань	–	F
RQ.TBL.02	Відображати поточне значення блокувань	–	F
RQ.TBL.03	Дозволяти користувачеві ввести нове значення блокування	–	F
RQ.TBL.04	Перед внесенням змін видавати оператору запит на підтвердження дій	–	F

Таблиця 3.11 – Специфікація вимог до вкладки «Тестування»

ІД	Найменування	Розподіл	Тестування
RQ.TTR.01	Відображати список вихідних сигналів тестування	–	F
RQ.TTR.02	Відображати поточне значення вихідних сигналів тестування	–	F
RQ.TTR.03	Дозволяти користувачеві увімкнути тестування заданого вихідного сигналу	–	F
RQ.TTR.04	Перед внесенням змін видавати оператору запит на підтвердження дій	–	F

Таблиця 3.12 – Специфікація вимог до вкладки «Параметри»

ІД	Найменування	Розподіл	Тестування
RQ.TPA.01	Вкладка повинна відображати інформацію про конфігурацію програми	–	A
RQ.TPA.02	Вкладка повинна відображати інформацію про поточний проект	–	A
RQ.TPA.03	Вкладка повинна відображати команду зміни паролю	–	A
RQ.TPA.04	Вкладка повинна відображати команду вибору поточного проекту	–	A
RQ.TPA.05	Вкладка повинна відображати команду виклику вікна статистики	–	A
RQ.TPA.06	Вкладка повинна відображати команду виходу	–	A

Таблиця 3.13 – Специфікація вимог комунікації з модулем БФЗ

ІД	Найменування	Розподіл	Тестування
RQ.COM.01	Обмін інформацією через COM-порт	–	F
RQ.COM.02	Відправка до «БФЗ» пакетів згідно з протоколом	–	V
RQ.COM.03	Прийом від «БФЗ» пакетів згідно з протоколом	–	V
RQ.COM.04	Аналіз цілісності пакетів та даних	–	F

Таблиця 3.14 – Специфікація вимог до захищеності

ІД	Найменування	Розподіл	Тестування
RQ.SEC.01	Прийом та відображення стану апаратного ключа	RQ.UI RQ.COM	F
RQ.SEC.02	Зміна уставок за паролем та ключем	RQ.TUST	F
RQ.SEC.03	Зміна блокувань за паролем та ключем	RQ.TBL	F
RQ.SEC.04	Тестування за паролем та ключем	RQ.TTR	F
RQ.SEC.05	Зміна налаштувань проекту за паролем	RQ.TPA	F
RQ.SEC.06	Вихід з програми за паролем	RQ.TPA	F
RQ.SEC.07	Зміна пароля	RQ.TPA	F

Таблиця 3.15 – Специфікація вимог до середовища

ІД	Найменування	Розподіл	Тестування
RQ.ENV.01	Інформація про проект зберігається у файлі	RQ.U RQ.B	V
RQ.ENV.02	Робота під Windows CE 320x240	–	V
RQ.ENV.03	Робота під Windows 7/10 640x480	–	V
RQ.ENV.04	Незалежність основного коду від ОС	–	V
RQ.ENV.05	Незалежність файлу проекту від ОС	–	V

3.2 Розробка структурної схеми

На рисунку 3.1 зображена структурна схема програми «МКУ».

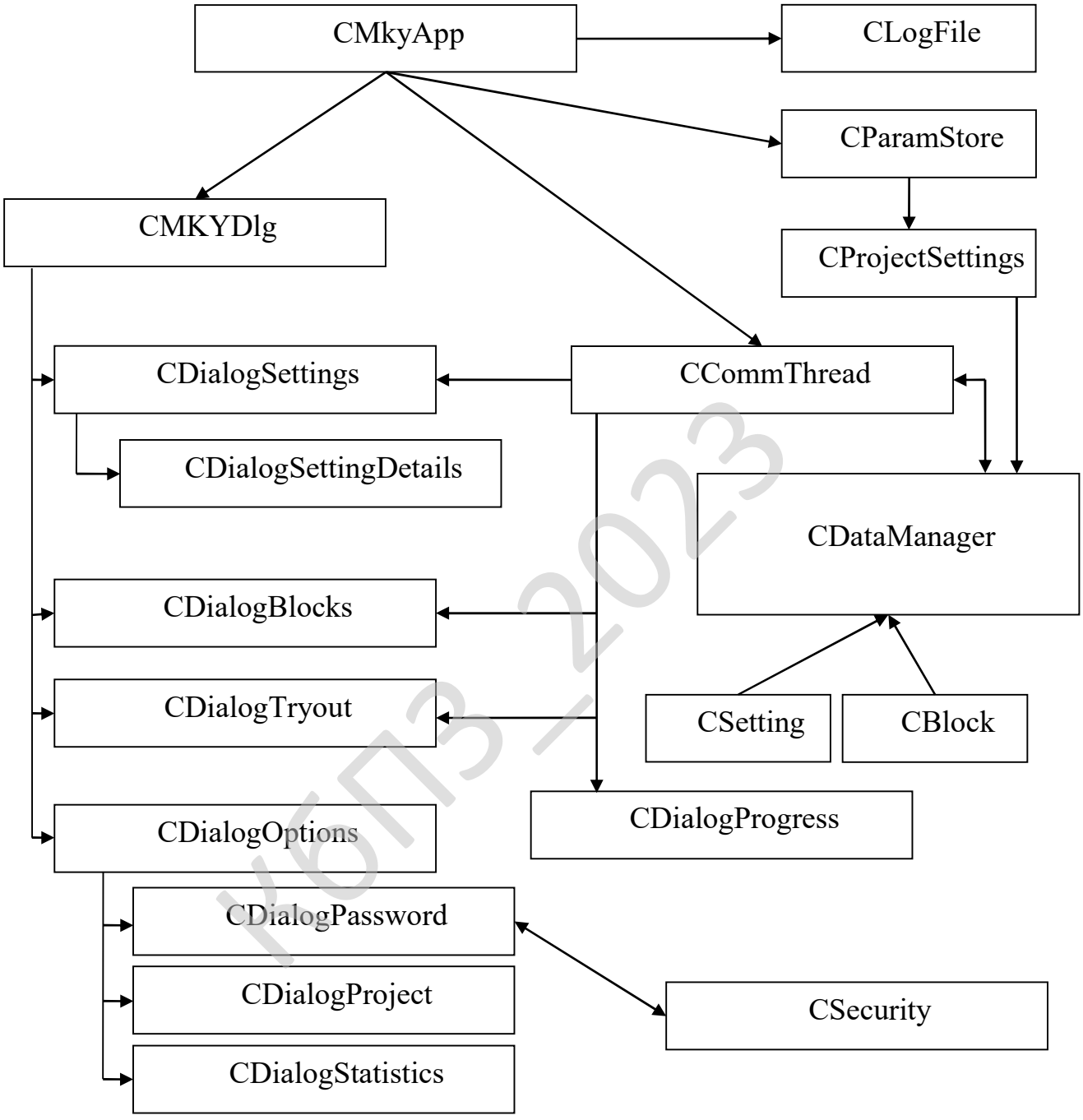

Рисунок 3.1 – Структурна схема програми

Структурна схема програми визначає склад модулів і класів програми. В програмі «МКУ» структурна схема співставляється зі структурою специфікації вимог до програми. Цим реалізується вимога стандарту проектування ПЗ

IEC62138 щодо чіткої структури програми, її модульності та принципу проектування «згори-донизу».

Проект програми «МКУ» складається з набору файлів. Модульність програми забезпечується як розташуванням програмного коду в окремих файлах, так і можливостями об’єктно-орієнтованого програмування мови С++ - використанням класів, інкапсуляції та патернів проектування.

Для досягнення однозначної модульності програми кожний клас програми розміщується в окремому файлі.

Кожний клас розташований на визначеному рівні взаємодії з іншими класами програми.

В таблиці 3.16 наведена інформація про кожний клас програми, його зв’язок з іншими класами та посилання на вимоги, наведені в специфікації вимог, що реалізуються даним класом.

Таблиця 3.16 – Класи програми «МКУ»

Найменування	Опис
MKY2023.cpp MKY2023.h class CMKYApp : public CWinApp	Головний клас програми, що запускає всі процеси та створює головний діалог програми
MKYDlg.cpp MKYDlg.h class CMKYDlg : public CDialog	Головний діалог програми, що створює функціональні діалоги та службові елементи інтерфейсу
CommThread.cpp CommThread.h class CCommThread struct InPacket_t struct OutPacket_t	Комунікаційний потік програми

Продовження таблиці 3.16

DataManager.cpp DataManager.h class CDataManager	Менеджер даних програми (уставок та блокувань)
MKYStatusBarCtrl.cpp MKYStatusBarCtrl.h CMKYStatusBarCtrl:public CStatusBarCtrl	Клас відображення статусного рядка програми
Block.cpp Block.h class CBlock	Клас інформації про блокування
Setting.cpp Setting.h class CSetting	Клас інформації про уставку
DialogSettings.cpp DialogSettings.h class CDialogSettings : public CDialog	Діалог відображення та корекції уставок
DialogBlocks.cpp DialogBlocks.h class CDialogBlocks : public CDialog	Діалог відображення та корекції блокувань
DialogTryout.cpp DialogTryout.h class CDialogTryout : public CDialog	Діалог перевірки вихідних сигналів захисту

Продовження таблиці 3.16

DialogOptions.cpp DialogOptions.h class CDialogOptions : public CDialog	Діалог налаштувань програми
DialogSettingDetails.cpp DialogSettingDetails.h class CDialogSettingDetails : public CDialog	Діалог детальної інформації про уставку: назву, діапазон тощо
DialogProgress.cpp DialogProgress.h class CDialogProgress : public CDialog	Діалог відображення статусу процесу обміну
DialogPassword.cpp DialogPassword.h class CDialogPassword : public CDialog	Діалог зміни паролю
DialogProject.cpp DialogProject.h class CDialogProject : public CDialog	Діалог вибору конфігурації проекту
DialogStatistics.cpp DialogStatistics.h class CDialogStatistics : public CDialog	Діалог відображення статистики
ProjectSettings.cpp ProjectSettings.h struct CProjectSettings	Клас, який зберігає параметри проекту

Продовження таблиці 3.16

ParamStore.cpp ParamStore.h class CParamStore	Клас, який зберігає налаштування користувача
Security.cpp Security.h class CSecurity	Клас, який реалізує функції парольної безпеки
LogFile.cpp LogFile.h class CLogFile	Клас, який виконує створення та запис файлу журналу
ExListCtrl.cpp ExListCtrl.h class CExListCtrl : public CListCtrl	Службовий клас для розширених можливостей елемента керування «Список»
MKYListCtrl.cpp MKYListCtrl.h class CMKYListCtrl:public CExListCtrl	Специфікований службовий клас для розширених можливостей елемента керування «Список»
ColorButton.cpp ColorButton.h class CColorButton : public CButton	Службовий клас для розширених можливостей елемента керування «Кнопка»
AutoCS.h Класи: auto_cs	Клас критичної секції з автоматичним розблокуванням

3.3 Розробка функціональної схеми

Програма має шість основних функціональних блоків:

1. Головне вікно;
2. Вкладка «Уставки»;
3. Вкладка «Блокування»;
4. Вкладка «Тестування»;
5. Вкладка «Параметри»;
6. Комунікаційний потік.

В даному розділі будуть детально розглянути функціональні дії, які можуть виконуватися за допомогою функціональних блоків.

Головне вікно

Головне вікно програми являє собою діалог, на якому присутній елемент керування «Tab Control» (набір вкладок), що містить на собі функціональні вкладки. Цей елемент забезпечує можливість зручного перемикання між основними функціями програми.

В нижній частині головного діалогу відображається рядок статусу. Цей рядок виконує дві функції: відображення стану обміну інформацією з модулем «БФЗ» за вказаним послідовним портом, а також стан апаратного ключа доступу. При наявності зв’язку відображається назва послідовного порту та об’єм переданих даних, при відсутності – повідомлення про таймаут з позначенням його червоним кольором. Стан ключа доступу відображається повідомленням на сірому фоні, якщо ключ не встановлений, на зеленому фоні – якщо ключ встановлений.

Вкладка «Уставки»

Вкладка «Уставки» є першою в списку вкладок, з якими працює користувач. Ця вкладка призначена для перегляду та зміни значень уставок каналу системи «АЗ-ПЗ».

Ця вкладка містить наступні елементи:

– елемент «список», що містить три колонки і відображає список уставок, що завантажені з файлу проекту. У першій колонці відображається короткий ідентифікатор уставки, наприклад «АЗ t1». У другій колонці відображається значення уставки, яке прийняте від модуля «БФЗ» через комунікаційний потік, наприклад «323.0». Точність відображення значення сигналу задається у проекті, для уставок типу «температура» за замовчуванням становить 1 знак після десяткової коми, для уставок типу «рівень» - 0 знаків після коми. У випадку, якщо відсутній зв'язок з модулем «БФЗ», в колонці «Значення» відображається три знаки питання («???»). У третій колонці відображається текстове найменування уставки, наприклад «Т гн 1 > Тгн ном + 8°С». Для вибору уставки, значення якої необхідно змінити, користувачеві необхідно виділити в даному списку потрібну уставку;

– поле введення значення обраної уставки, розташоване в нижній частині вкладки. В цьому полі користувач має ввести число, яке необхідно записати. У випадку, якщо запис уставки недоступний (уставка не обрана в списку, немає зв'язку з модулем «БФЗ» або відсутній апаратний ключ доступу), це поле недоступне для редагування;

– кнопка «Запис». При натисканні цієї кнопки перевіряється, чи входить введене користувачем число до допустимого діапазону. У випадку, якщо число не входить до діапазону – видається повідомлення про помилку. Якщо значення підходить за діапазоном – до комунікаційного потоку надсилається команда запису з номером уставки та новим значенням;

– екранна клавіатура для введення значення уставки, являє собою набір кнопок з цифрами «0»-«9», знаком «-», десятковою крапкою та кнопкою очистки. Ця клавіатура призначена для введення значення уставок користувачем на сенсорному екрані без використання фізичної клавіатури.

При відображенні вкладки на екрані за таймером відбувається періодичне оновлення значень уставок, які приймаються від модуля «БФЗ».

Таким чином, за допомогою вкладки «Уставки» виконується функція інформування користувача про поточне значення уставок, а також функція задання нових значень уставок.

Вкладка «Блокування»

Вкладка «Блокування» є другою в списку вкладок, з якими працює користувач. Ця вкладка призначена для перегляду, встановлення або зняття блокувань окремих алгоритмів захисту каналу системи «АЗ-ПЗ».

Ця вкладка містить наступні елементи:

– елемент «список», що містить три колонки і відображає список блокувань алгоритмів, завантажені з файлу проекту. У першій колонці відображається короткий ідентифікатор блокування, наприклад «АЗ_1» або «ПЗ-1_1». У другій колонці відображається стан блокування алгоритму, прийнятий від модуля «БФЗ» через комунікаційний потік, наприклад «ТАК» або «НІ». У випадку, якщо відсутній зв’язок з модулем «БФЗ», в колонці «Значення» відображається три знаки питання («???»). У третій колонці відображається текстове найменування блокування алгоритму, наприклад «Т < 10с». Для вибору блокування, значення якого необхідно змінити, користувачеві необхідно виділити в даному списку потрібний елемент;

– опція встановлення значення блокування («чекбокс»), розташована в нижній частині вкладки. В цьому полі користувач може встановити або зняти опцію блокування. У випадку, якщо запис блокування недоступний (елемент не обраний в списку, немає зв’язку з модулем «БФЗ» або відсутній апаратний ключ доступу), це поле недоступне для зміни.

При відображенні вкладки на екрані за таймером відбувається періодичне оновлення станів блокувань алгоритмів, які приймаються від модуля «БФЗ».

Таким чином, за допомогою вкладки «Блокування» виконується функція інформування користувача про поточне блокування алгоритмів, а також функція встановлення або скидання блокувань.

Вкладка «Тестування»

Вкладка «Тестування» є третьою в списку вкладок, з якими працює користувач. Ця вкладка призначена для імітації формування вихідних сигналів захисту каналу системи «АЗ-ПЗ».

Ця вкладка містить три основні кнопки: «АЗ», «ПЗ-1» та «ПЗ-2». В залежності від конфігурації проекту вкладка може містити четверту кнопку «P<». Назви вихідних сигналів також можуть бути зміненими через редагування файлу проекту.

Кнопки реалізують функцію перевірки вихідних сигналів. Вони є доступними для натискання тільки у випадку, якщо наявний зв'язок з модулем «БФЗ» та встановлений апаратний ключ доступу. При натисканні будь-якої кнопки програма відправляє до комунікаційного потоку команду на тестування сигналу протягом 5 секунд.

Вкладка «Параметри»

Вкладка «Параметри» є службовою і останньою в списку вкладок програми. Вона містить наступні функціональні елементи керування:

– текстові поля з інформацією про назву АЕС, номер енергоблоку;

– текстові поля з інформацією про версію програми та дату її збірки;

– кнопку виклику функції зміни паролю;

– кнопку виклику функції вибору файлу проекту зі списком уставок та блокувань;

– кнопку виклику функції відображення статистики з'єднання (кількості переданих та прийнятих пакетів);

– кнопку виходу з програми.

Таким чином, функціональна вкладка «Параметри» забезпечує доступ користувача до службових функцій програми.

Комунікаційний потік

Для реалізації обміну інформацією програми «МКУ» з процесором модуля «БФЗ» програма містить окремий потік, що реалізується в класі

«CComThread». Після запуску потік відкриває послідовний порт. Для реалізації функції прийому та передачі даних програма містить дві функції – Send() та Receive(). Функція Send() надсилає до «БФЗ» вихідний пакет, який заповнений відповідно до поточного стану комунікаційного потоку (пошук режиму, пошук уставки, пошук блокування тощо). Після надсилання потік очікує прийняття від «БФЗ» вхідного пакету, що містить інформацію про стан потрібного об’єкту або поточний режим. При наявності вірного розміру та контрольної суми пакету комунікаційний потік оновлює інформацію про об’єкт у масиві станів об’єктів класу CDataManager та надсилає наступний вихідний пакет.

Для керування комунікаційним потоком клас CComThread містить функції операцій:

```
void InitiateBlocksRead();
void InitiateSettingsRead();
void InitiateBlockToggle(int Index);
void InitiateSettingChange (
          int Index, double Value, int ChangeStepCount);
void InitiateTryoutMode();
void InitiateTryout(int Index, int MaxIndex);
```

Для реалізації безпечної взаємодії класу CComThread з інтерфейсним потоком використовуються критичні секції.

Функціональна схема програми зображена на рисунку 3.2.

Рисунок 3.2 – Функціональна схема програми

3.4 Розробка діаграми процесів

У програмі можуть відбуватися наступні процеси:

– зчитування та відображення поточних значень уставок;

– зчитування та відображення поточних значень блокувань;

– авторизація;

– введення та запис нового значення обраної уставки;

– введення та запис нового значення обраного блокування;

– введення та відправка команди на тестування вихідного сигналу;

– надсилання та прийом пакетів даних до модуля «БФЗ».

У програмі реалізовано два потоки: інтерфейсний потік та потік обміну даними. Для синхронізації даних використовуються критичні секції.

В інтерфейсному потоці відбуваються наступні процеси:

– періодичне оновлення та відображення значень параметрів на елементах керування;

– обробка дій користувача.

В комунікаційному потоці виконуються наступні процеси:

– відкриття послідовного порту;

– обробка інформації про поточний режим обміну та тип параметрів, що обробляються;

– перемикання режиму обміну (запис або читання, типи об’єктів, рівні меню);

– відправка вихідних пакетів;

– прийом вхідних пакетів;

– розбір даних від вхідних пакетів та оновлення значень масивів параметрів програми.

На рисунку 3.3 зображена діаграма процесів програми.

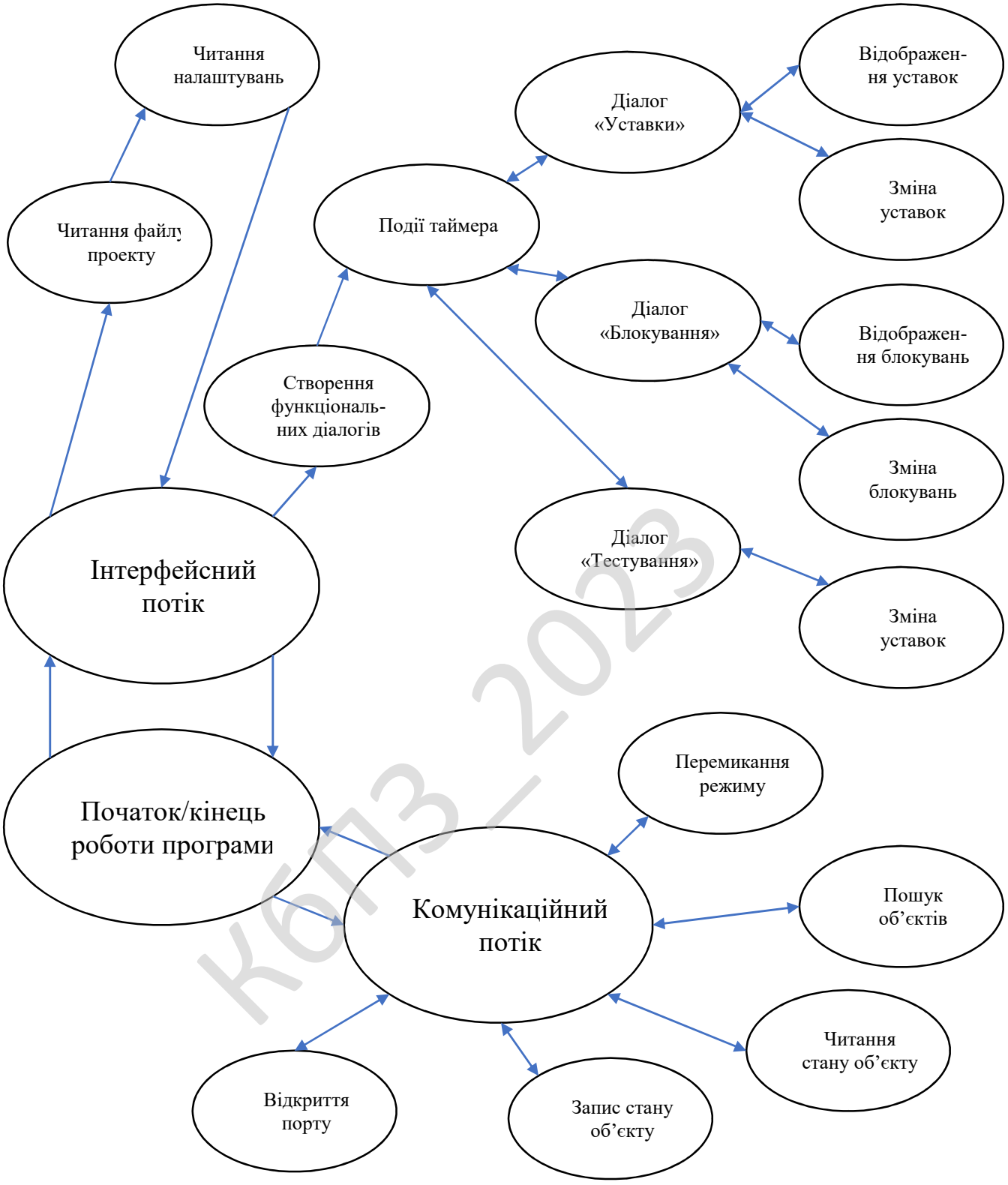

Рисунок 3.3 – Діаграма процесів програми

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

4.1 Блок–схеми та опис алгоритмів функціонування системи

Програма «МКУ» розподілена на два потоки, що виконуються паралельно: інтерфейсний потік та потік обміну інформацією. Синхронізація доступу до даних між потоками відбувається за допомогою критичних секцій. Нижче наведено опис основних алгоритмів роботи кожного потоку.

Інтерфейсний потік

Після запуску програми створюється екземпляр головного класу програми CMkyApp. При створенні даного екземпляру також відбувається створення екземплярів основних об’єктів програми: CDataManager (зберігання та робота з списком уставок та блокувань), CSecurity (робота з авторизацією), CProjectSettings (налаштування проекту), CComThread (комунікаційний потік).

Після створення екземпляру головного класу відбувається завантаження налаштувань програми з реєстру та завантаження файлу проекту зі списком уставок та блокувань. Після цього створюється та запускається комунікаційний потік.

Після запуску комунікаційного потоку виконується ініціалізація інтерфейсу користувача. Створюється екземпляр класу основного діалогу програми CMKYDlg. Цей діалог створює дочірні екземпляри класів функціональних діалогів програми – CDialogSettings (зміна уставок), CDialogBlocks (зміна блокувань), CDialogTryout (тестування вихідних сигналів), CDialogOptions (діалог налаштувань). Функція OnInitDialog кожного класу діалогу відповідає за створення необхідних елементів керування.

Після створення елементів інтерфейсу клас головного діалогу запускає періодичний таймер. Подія таймера передається всім дочірнім діалогам. У функції обробки події таймера відбувається аналіз поточного стану уставок, блокувань та сигналів тестування. У випадку, коли дані змінюються

комунікаційним потоком – відбувається оновлення відображення стану об’єктів в елементах керування (списках, кнопках тощо).

При виконанні користувачем операцій по задаванню значень уставок, блокувань тощо, класи функціональних діалогів відправляють команди зміни значень до комунікаційного потоку для обробки.

Всі об’єкти програми створюються статично і існують протягом всього часу роботи програми.

На рисунку 4.1 зображена блок-схема інтерфейсного потоку програми.

Комунікаційний потік

Після створення класу комунікаційного потоку CComThread та його запуску відбувається відкриття послідовного порту, який вказано в налаштуваннях програми. Після цього в пам’яті створюються два буфери для зберігання вхідного та вихідного пакетів даних. Основна функція потоку виконує дві основні операції: відправка пакету даних (функція Send()) та прийом даних (функція Receive()).

Клас CComThread містить посилання на об’єкт класу CDataManager для оновлення інформації про стан об’єктів програми. Також він містить множини станів роботи (enum EState) та множини команд (enum ECommand), а також інформацію про поточний етап обміну та отриману команду.

Формування вихідного пакету функцією «Send» відбувається в залежності від поточного етапу. Наприклад, якщо відбувається читання значень уставок – потік відправляє команди «Choice» і за допомогою функції Receive() чекає у відповідь пакет зі станом уставки та її номером, потім відправляє наступну команду «Choice» і циклічно повторює прийом. У випадку, якщо приходить команда на запис – командою «Choice» відбувається пошук уставки з необхідним номером і відправляється команда «WriteSetting». У випадку, якщо потрібно змінити тип об’єктів – відправляється команда «Escape», командою «Choice» відбувається пошук потрібного режиму, вхід в потрібний режим командою «Enter» тощо.

При завершенні програми послідовний порт закривається та потік завершує свою роботу.

На рисунку 4.2 зображена блок-схема комунікаційного потоку.

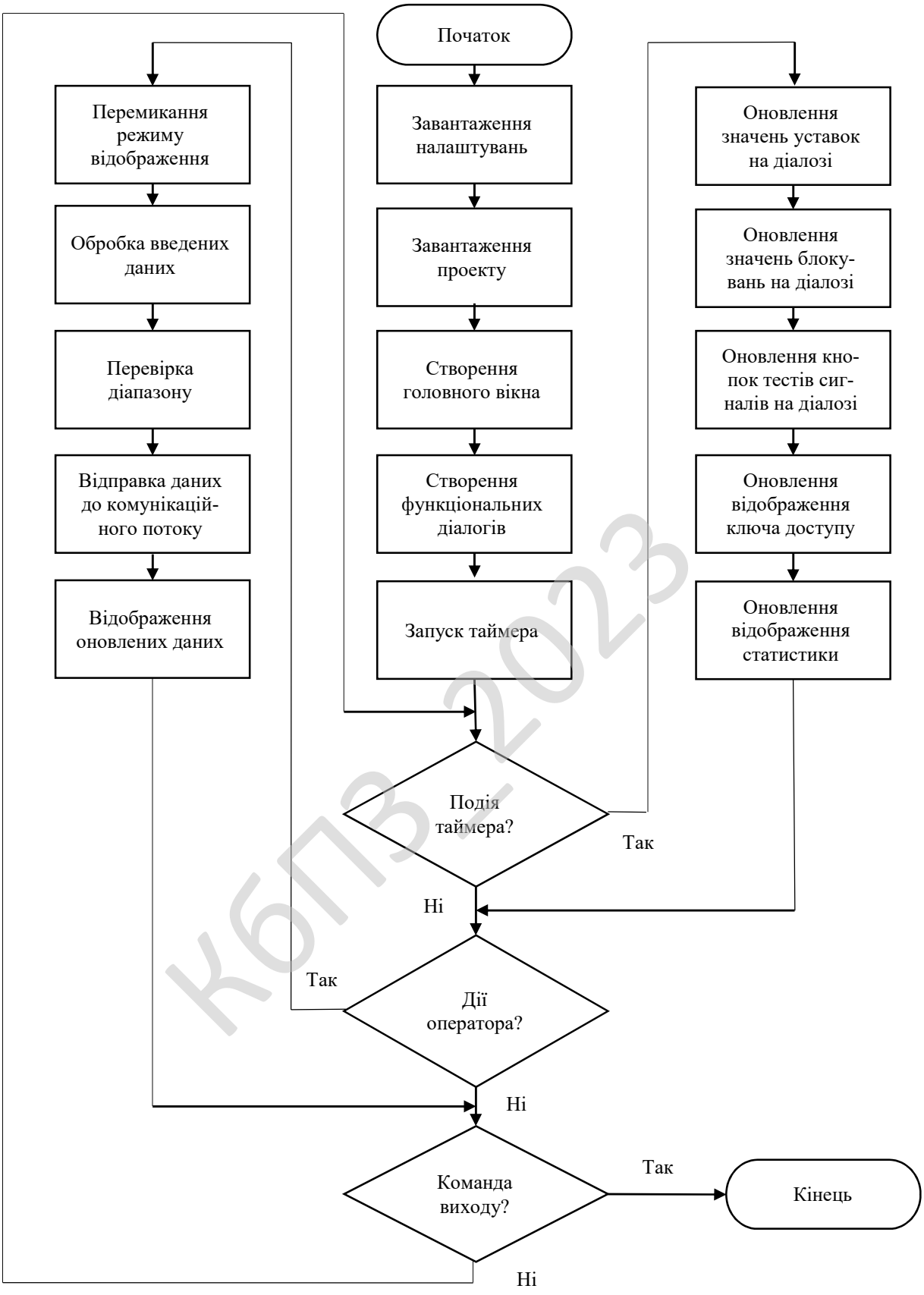

Рисунок 4.1 – Блок-схема інтерфейсного потоку

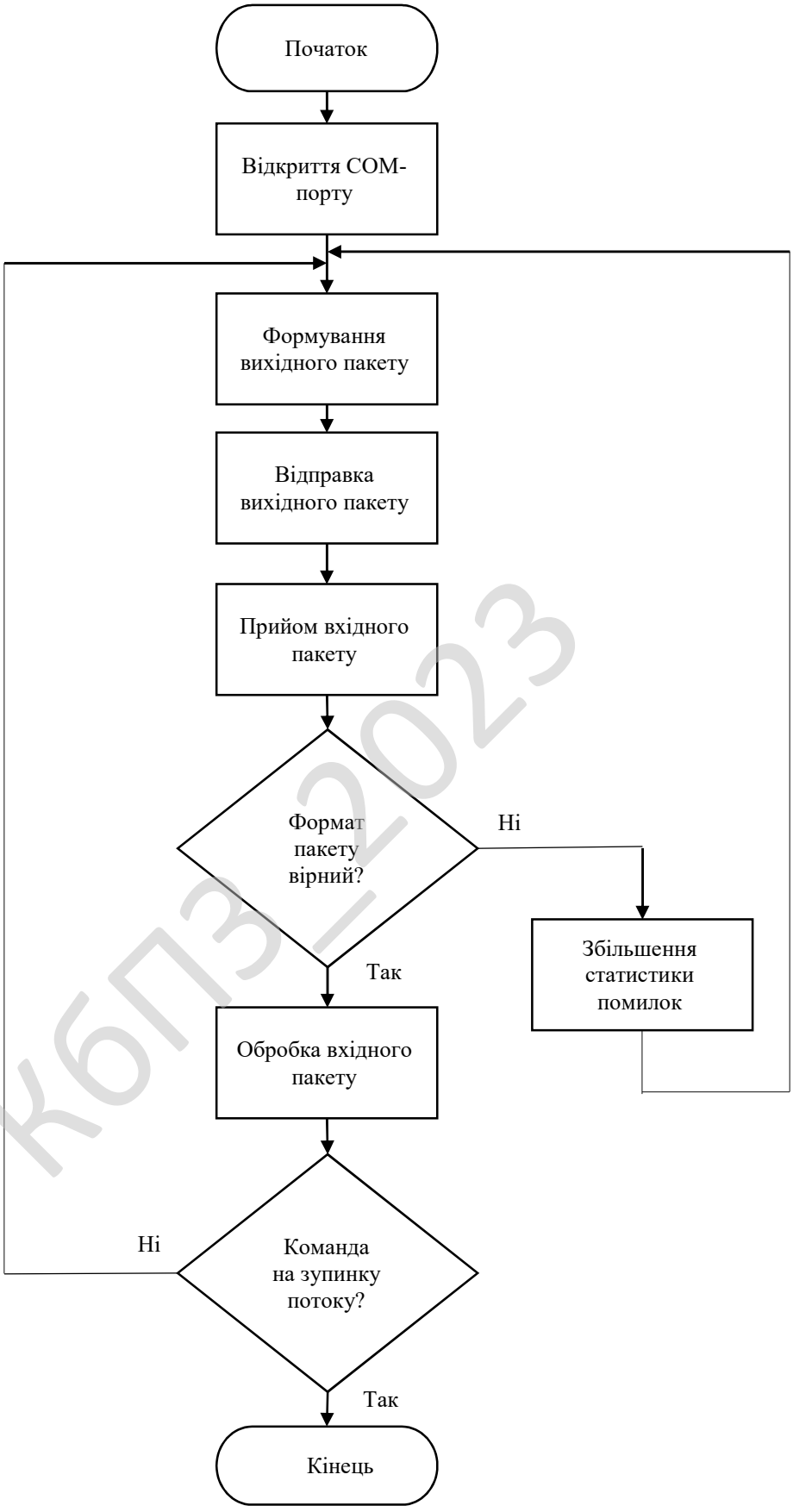

Рисунок 4.2 – Блок-схема комунікаційного потоку

Наведемо код ініціалізації основних об’єктів програми.

```
class CMKYApp : public CWinApp
{
public:
  CMKYApp();
  ~CMKYApp();
protected:
  virtual BOOL InitInstance();
  DECLARE_MESSAGE_MAP()
private:
  // Base objects
  //
  CParamStore m_ParamStore;
  CLogFile m_LogFile;
  CDataManager m_DataManager;
  CSecurity m_Security;
  CCommThread m_CommThread;
  CProjectSettings m_ProjectSettings;
};
BOOL CMKYApp::InitInstance()
{
  CWinApp::InitInstance();
  AfxEnableControlContainer();

  // Standard initialization
  SetRegistryKey(_T("MKY2023"));

  // Loading settings
  //
  BOOL ConfigurationOk = m_ParamStore.Load();
  if (ConfigurationOk == TRUE)
  {
       ConfigurationOk = m_ProjectSettings.Load(m_ParamStore.ProjectFile(),
m_ParamStore.ShowReserved(), m_DataManager);

       if (ConfigurationOk == FALSE)
       {
            CString str;
            str.Format(_T("Помилка завантаження проекту [%s]!"),
m_ParamStore.ProjectFile());
            AfxMessageBox(str, MB_OK | MB_ICONERROR);
```

```
            }
        }
        // Запуск комунікації
        //
        m_CommThread.Execute(m_ParamStore.Port());
        BOOL ConnectedOK = m_CommThread.IsConnected();
        if (ConnectedOK == FALSE)
        {
            AfxMessageBox(_T("Помилка відкриття порту ") + m_ParamStore.Port(),
MB_OK | MB_ICONERROR);
        }
        // Create Main Window
        //
        {
            CMKYDlg dlg(m_ParamStore, m_LogFile, m_DataManager, m_Security,
m_CommThread, m_ProjectSettings);
            m_pMainWnd = &dlg;
            dlg.DoModal();
        }
        return FALSE;
    }
```

Наведемо код основної функції комунікаційного потоку.

```
AFX_THREADPROC CCommThread::ThreadProc( LPVOID pParam )
{
    CCommThread* pThis = (CCommThread*)pParam;
    while (!pThis->bTerminateThread)
    {
        Sleep(50);
        pThis->Send();
        Sleep (100);
        do{
        }while (pThis->Receive());
    }
    TRACE0("CCommThread terminated.\n");
    return 0;
}
```

Наведемо фрагмент коду функції Send() – формування вихідного пакету.

```
void CCommThread::Send()
{
    double Value;
```

```
switch (State)
{

case Idle:
     OutPacket.Command = COMMAND_CHOICE;
     break;

case Initialize:
     OutPacket.Command = COMMAND_STATUS;
     break;

case SearchingModeLevel:
     OutPacket.Command = COMMAND_ESCAPE;
     break;

case SearchingMode:
case ChangeMode:
     OutPacket.Command = COMMAND_CHOICE;
     break;

case EnteringMode:
     OutPacket.Command = COMMAND_ENTER;
     break;

case BlocksReading:
case SettingsReading:
     OutPacket.Command = COMMAND_CHOICE;
     break;

case SearchingItem:
     OutPacket.Command = COMMAND_CHOICE;
     break;

case BlockChanging:
     OutPacket.Command = COMMAND_ENTER;
     break;

case SettingChanging:
     OutPacket.Command = COMMAND_SETTING;
     Value = SettingNewValue / CurrentSettingMultiplier;
     OutPacket.Data = CSetting::Round(Value);
```

```
            if (!TimeoutWriteToFlash)
            {
                TimeoutCounter = TIMEOUT_VALUE_FLASH;
                TimeoutWriteToFlash = TRUE;
                TRACE0("Setting - TIMEOUT_VALUE_FLASH\n");
            }

            break;

        case TryoutRunning:
            OutPacket.Command = COMMAND_ENTER;
            break;

        case SaveChanges:
            OutPacket.Command = COMMAND_CHOICE;
            break;

        case EditConfirming:
            OutPacket.Command = COMMAND_ENTER;

            if (!TimeoutWriteToFlash)
            {
                TimeoutCounter = TIMEOUT_VALUE_FLASH;
                TimeoutWriteToFlash = TRUE;
                TRACE0("EditConfirm - TIMEOUT_VALUE_FLASH\n");
            }
            break;
        }
        if (OutPacket.Command == COMMAND_ENTER)
            EnterCount++;
        if (OutPacket.Command == COMMAND_ESCAPE)
            EscapeCount++;

        OutPacket.Size = sizeof (OutPacket);
        OutPacket.Crc = CountCRC ((BYTE*)&OutPacket, sizeof (OutPacket));
        DWORD wr;
        WriteFile(hCom, &OutPacket, sizeof(OutPacket), &wr, NULL);

        if (wr != sizeof (OutPacket))
        {
```

```
            ErrorCount++;
            return;
        }
            PacketSent++;
    }
```

4.2 Захист розробленого програмного забезпечення

В даній магістерській роботі захищеність програмного забезпечення розглядається в двох аспектах: захищеність даних під час обміну з апаратними засобами та захищеність від неавторизованих дій користувача.

Перший аспект – захищеність даних, якими програма обмінюється з апаратними модулями БФЗ полягає в тому, що кожний пакет даних, який надсилається до модуля або приймається від модуля, захищений контрольною сумою за алгоритмом CRC-8. Через те, що пакети мають дуже малий розмір, 8-бітна розрядність контрольної суми є достатньою для контроля цілісності пакетів даних. Якщо контрольна сума, яка отримана в пакеті, не співпадає з розрахунковою – такий пакет ігнорується.

Другий аспект – захищеність програми від неавторизованих або некоректних дій користувача. Він полягає в наступних пунктах:

– апаратний ключ доступу для зміни параметрів. Цей ключ встановлюється в шасі, де встановлені логічні модулі ПТК, та дозволяє проводити зміну параметрів тільки коли знаходиться в положенні «Увімкнено»;

– програмний пароль, який встановлюється користувачем і запитується під час кожної операції зміни уставок або блокування алгоритмів захисту.

Таким чином, розроблене програмне забезпечення є захищеним як з точки зору обміну з апаратними засобами, так і з точки зору взаємодії з користувачем.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Умови виконання програми

– панельний комп'ютер Advantech TPC –642S / SE або сумісний з IBM PC (32- або 64-розрядної архітектури);

– операційна система Microsoft Windows CE 3.0, що зберігається в ПЗУ комп'ютера, або Microsoft Windows 10/11;

– інтерфейс RS-232, що з'єднує порт COM комп'ютера з блоком БФЗ.

ПРИМІТКА: конфігурація комп'ютера може бути змінена в залежності від типу КСУ і використовуваних компонентів.

Установка програми на комп'ютер з Windows CE

Призначення файлів програми наведено в таблиці 5.1.

Таблиця 5.1 – Призначення файлів програми

\IPSM(Harddisk)\Startup\MKY.exe	Програма MKY
\IPSM(Harddisk)\MKYDatabase*.mky	Бази даних проектів
\IPSM(Harddisk)\MKY.cfg	Налаштування программи

Для встановлення або оновлення програми необхідно перезаписати наведені файли на панельному комп’ютері Для цього необхідно виконати наступні дії.

– ПРИ ВИМКНЕНОМУ ПК встановити в панель управління знімний накопичувач;

– встановити на мережеву карту статичну IP-адресу, маска підмережі 255.255.255.0;

– переписати ВЕСЬ вміст каталогу IMAGE з флешки в каталог \IPSM (\Harddisk);

– перезапустити ПК;

– після перезапуску задати ім'я проекту (вкладка "Параметри"). Список проектів завантажується з каталогу MKYDatabase;

– перезапустити програму (ярлик на робочому столі);

– ПРИ ВИМКНЕНОМУ ПК витягнути знімний накопичувач.

Налаштування програми

Щоб переглянути інформацію про версію програми і змінити параметри, виберіть вкладку «Параметри». На екрані з'явиться вікно, зображене на рисунку 5.1.

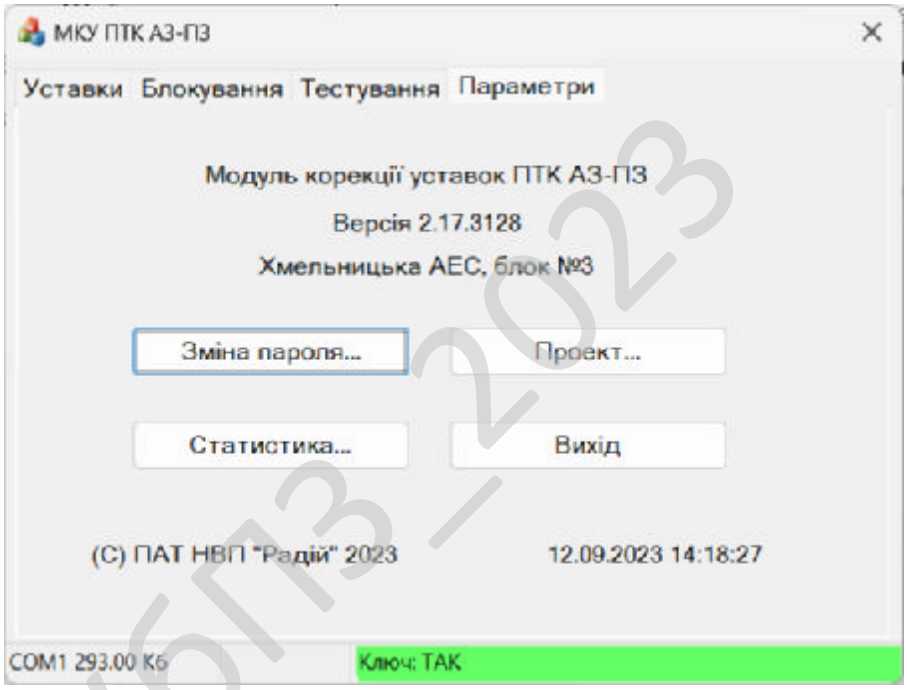

Рисунок 5.1 – Вкладка налаштувань

Для зміни пароля доступу натиснути кнопку «Змінити пароль ...». Запит пароля показаний на рисунку 5.2. Пароль складається з цифр, клавіша «С» використовується для скидання введеного рядка.

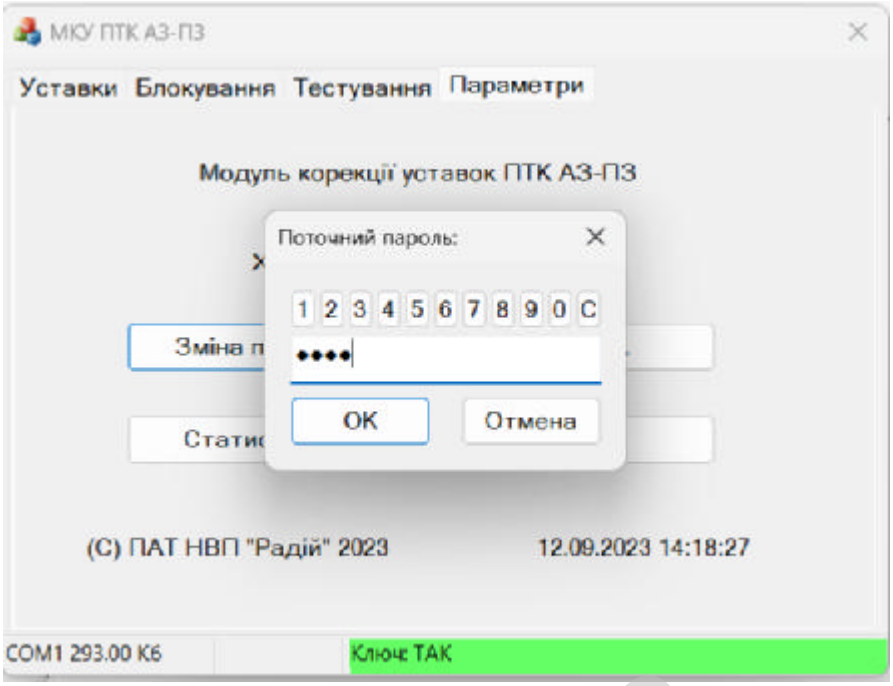

Рисунок 5.2 – Зміна паролю

Після успішного введення пароля на екрані з'явиться повідомлення «Пароль змінено».

Щоб видалити пароль, потрібно ввести поточний пароль і залишити поле пароля порожнім, коли буде запропоновано новий.

Щоб вибрати файл зі списком налаштувань і блокувань, натисніть кнопку "Проект ...". У вікні, що з'явилося (рисунок 5.3) Виберіть потрібний файл. Зміни вступлять в силу після перезапуску програми.

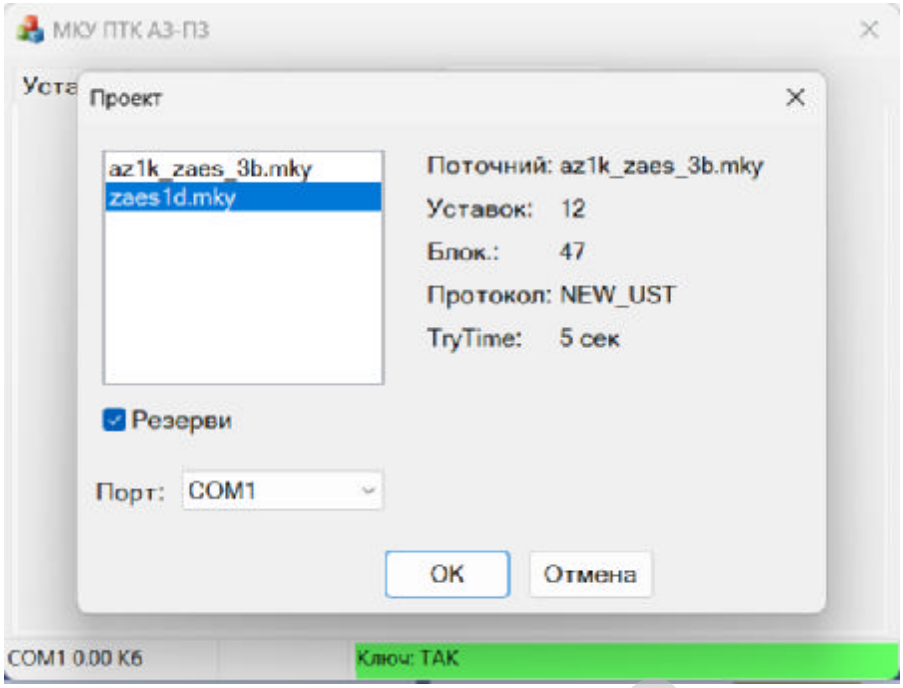

Рисунок 5.3 – Вибір проекту

Щоб переглянути кількість переданих і прийнятих пакетів, а також кількість помилок, натисніть кнопку «Статистика ...». Вікно статистики показано на рисунку 5.4.

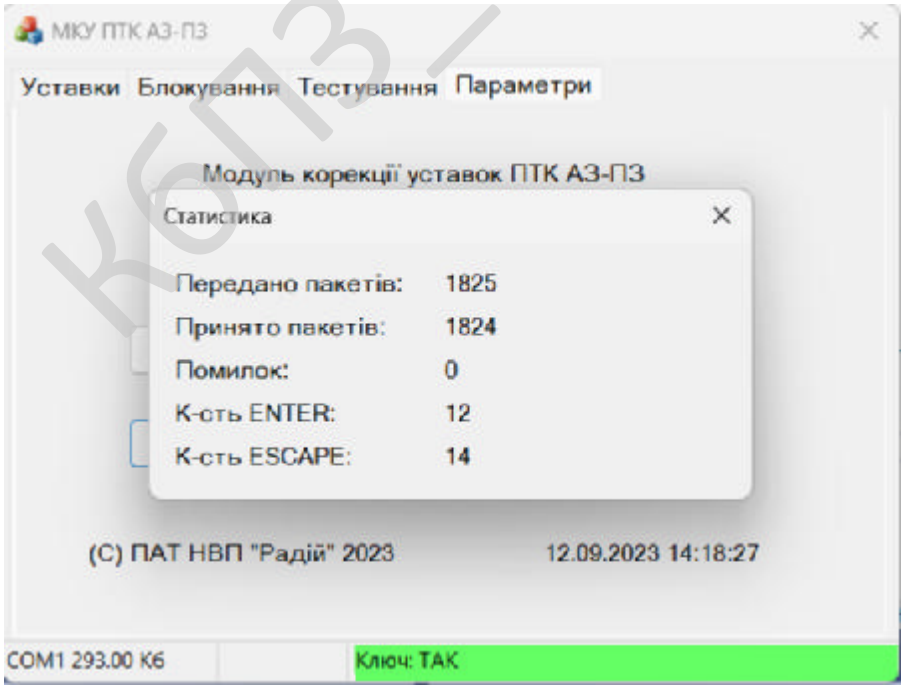

Рисунок 5.4 – Вибір проекту

Опис інтерфейсу програми

Зміна значення уставок, блокування захистів (алгоритмів) і тестування захистів здійснюється за допомогою програми «МКУ», запущеної на панельному комп'ютері, розташованому в кожній шафі формування сигналів (ШФС) – кожному каналі. Зміни вносяться індивідуально для кожного каналу. Програма виконується протягом усього періоду експлуатації системи АЗ-ПЗ. Зовнішній вигляд головного вікна програми "МКУ" показаний на рисунку 5.5.

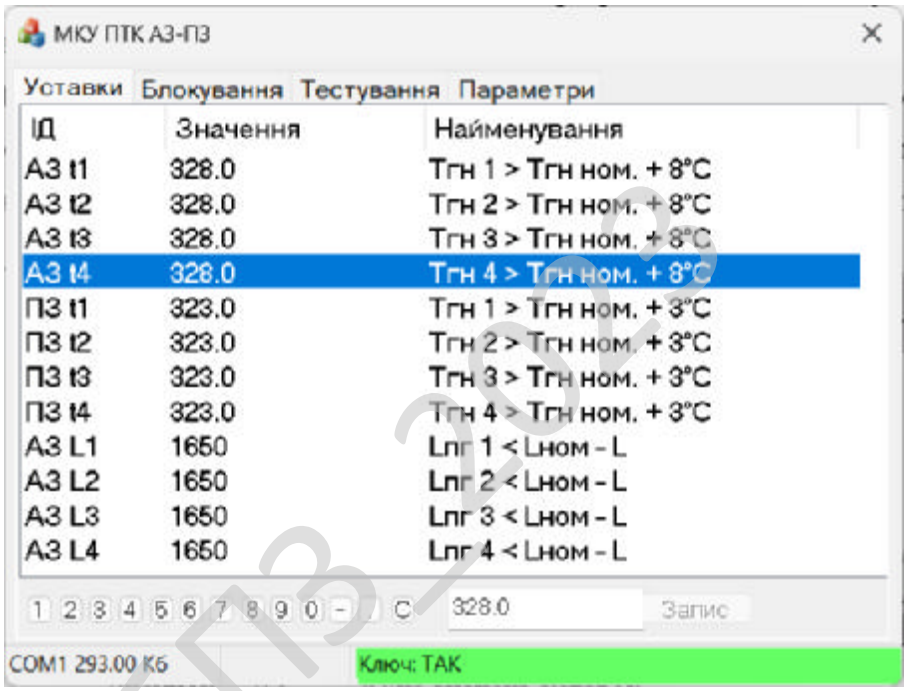

Рисунок 5.5 – Зміна значень уставок

Головне вікно програми «МКУ» містить:

– вкладка «Уставки» – введення режиму перегляду/зміни значень заданих. При виборі цієї вкладки відкривається вікно для перегляду/зміни поточних значень уставок;

– вкладка «Блокування» – увійти в режим перегляду/зміни стану блокувань. При виборі цієї вкладки відкривається вікно для перегляду/зміни поточного стану блокувань (алгоритмів);

– вкладка «Тестування» – вхід в режим тестування захисту. При виборі цієї вкладки відкриється вікно для перевірки спрацювання захисту;

– вкладка «Параметри» – містить інформацію про версію програми «МКУ», а також кнопку зміни пароля доступу, вибору проекту і перегляду статистики;

– робоче поле, змінне для кожного режиму;

– рядок стану, розташований внизу головного вікна програми і відображає параметри:

– найменування COM-порту, стан каналу зв'язку і обсяг даних, що передаються і приймаються від блоку БФЗ;

– кількість включених замків;

– стан ключа доступу. «YES» – відповідає встановленому ключу доступу, «NO» – відповідає відсутності ключа доступу.

Ключ доступу встановлюється в «Зону доступу ключів», розташовану під чи поряд з панельним комп'ютером (визначається конструктивними особливостями ШФС).

Зміна значень уставок, станів блокування і тестування захистів, налаштувань можливі тільки при наявності ключа доступу. При відсутності ключа доступу можливий тільки перегляд поточних значень уставок, станів блокування.

Перехід з одного режиму в інший (вибір відповідної вкладки програми) неможливий при установці ключа доступу. Перед перемиканням з одного режиму в інший необхідно прибрати ключ доступу.

Перед зміною будь-якого параметра (значення заданої точки, стану блокування, тесту) відображається запит пароля. Зміни можна внести після правильного введення пароля. Вам не буде запропоновано повторно ввести пароль, доки ключ доступу не буде вилучено. Перевстановлення ключа тягне за собою повторне введення пароля.

Зміна значень уставок

У вікні перегляду/зміни діючих значень уставок (рисунок 5.5) розташовуються такі елементи:

– список, що містить список налаштувань змінних;

– цифрова клавіатура для введення нового значення вибраної заданої точки;

– поле, що відображає значення вхідної уставки;

– кнопка "Запис" – запис значення уставки, введеної з клавіатури. При натисканні з'являється вікно для підтвердження заміни значення обраної уставки. Після підтвердження починається процес запису уставки.

Зміна значень блокувань

У вікні перегляду/зміни поточного стану захисних блокувань (рисунок 5.6) розташовані такі елементи:

– список, що містить список заблокованих захистів;

– «Блокування» – можливість встановлення/зняття захисного блокування. Він активний тільки при установці ключа доступу.

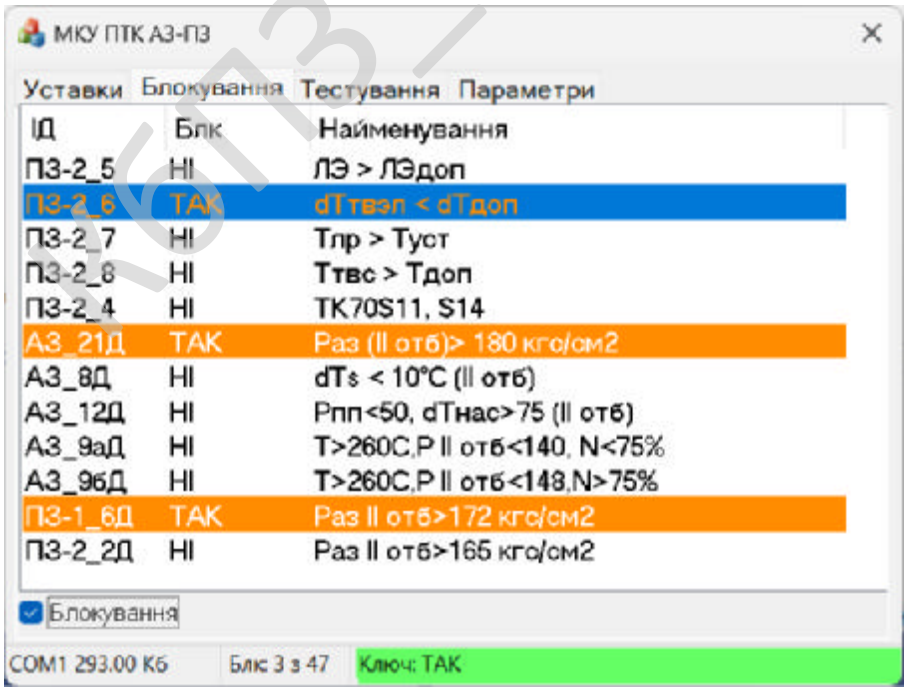

Рисунок 5.6 – Зміна значень блокувань

Тестування вихідних сигналів захисту

Випробування захистів полягає у формуванні блоком БФЗ спрацьовуючих сигналів. У вікні випробування захисту (рисунок 5.7) є кнопки для тестування захистів (активні при установці ключа доступу).

Тестування можна проводити лише для одного захисту одночасно.

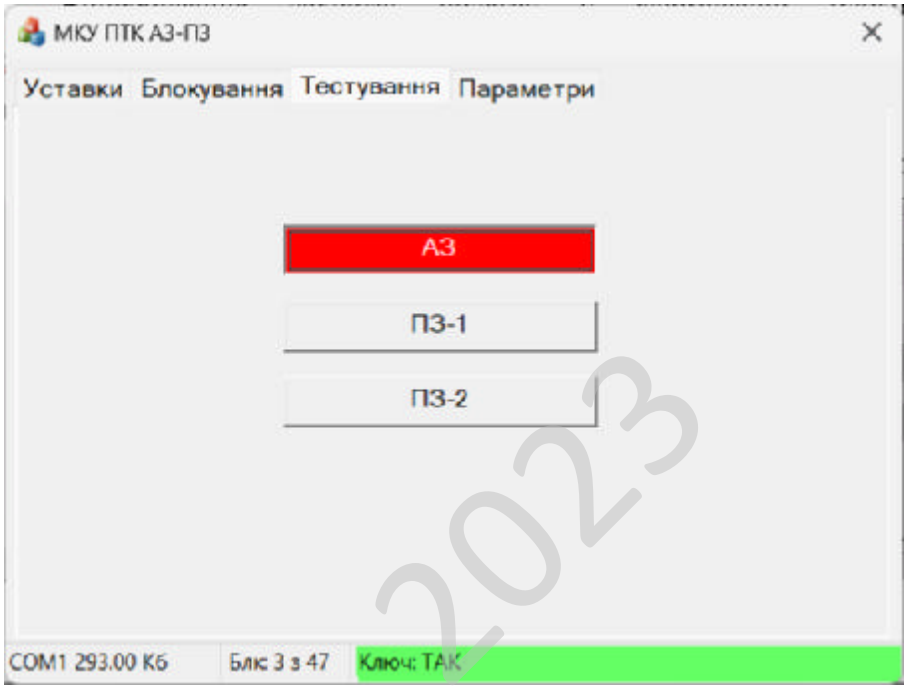

Рисунок 5.7 – Тестування вихідних сигналів

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено використання в комп'ютерній системі керування АЕС, важливої для безпеки.

Метою розробки є дослідження вимог до програмного забезпечення, яке використовується в комп'ютерних системах контроля і управління технологічними процесами на атомних електростанціях, і викладення досвіду розробки і інтеграції в ПТК цього програмного забезпечення.

Об'єктом дослідження є розробка програмного забезпечення, яке використовується в комп'ютерних системах контроля і управління технологічними процесами на атомних електростанціях.

Предметом дослідження є методи та галузеві стандарти розробки програмного забезпечення критичного застосування та їх практичне застосування під час реалізації програмного забезпечення, що входить до складу ПТК управління технологічними процесами на АЕС, і який направлений зміну умов спрацювання керуючої системи безпеки.

Методи дослідження базуються на вимогах нормативних документів, що стосуються до розробки програмного забезпечення, важливого для безпеки, та методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– досліджені нормативні документи за якими класифікують функції, що виконуються комп'ютерними системами контролю і управління АЕС;

– визначені категорії і вимоги, що висуваються до ПЗ;

– окреслена специфікація вимог до проектування ПЗ, важливого для безпеки;

– розроблена методика керування конфігурацією;

– програмне забезпечення реалізоване відповідно до нормативних вимог.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 48 днів (два місяці).

В магістерській роботі проведене дослідження та виконана програмна реалізація системи управління АЕС критичного застосування.

Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір та системні потреби;

б) незалежність від встановлених на комп'ютері баз даних;

в) зручність у користуванні та надійність

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт	N	1
2. Кількість екземплярів програм, шт	Ne	60
3. Запланований термін розробки, днів	Fpq	48 (2 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2
7. Кількість макетів вхідної інформації	–	3

Продовження таблиці 7.1

1	2	3
8. Кількість форм вихідної інформації.	–	4
9. Мова програмування (1-6)	–	1
10. Попередній досвід (1-6)	–	3
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	_	2
13. Рівень спрацьованості колективу (1-6)	_	2
14. Ступінь вимірності процесів (1-6)	_	3
15. Необхідна надійність програмного забезпечення (1-6)	_	2
16. Розмір бази даних (порівняно з розміром програми) (1-6)	_	2
17. Складність кінцевого програмного продукту (1-6)	_	2
18. Необхідний рівень забезпечення повторного використання (1-6)	_	2
19. Документованість відповідно до планованого життєвого циклу (1-6)	_	2
20. Вимоги до швидкодії ПП (1-6)	_	2
21. Обмеження на розміри основного сховища даних (1-6)	_	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	_	2
23. Професійний рівень аналітиків (1-6)	_	2
24. Професійний рівень програмістів (1-6)	_	2
25. Постійність складу команди розробників (1-6)	_	2
26. Досвід розробки додатків (1-6)	_	2
27. Досвід роботи з обчислювальною платформою (1-6)	_	2

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПО для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн	–	60000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	50
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначаємо по типовим нормам часу приведеним в додатках МВ. Стадія РП є найбільш тривалою і трудомісткою, що робить значний вплив на інші стадії проекту.

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

$$T_{ном} = A\ \text{Size}^B \qquad (7.1)$$

де А – коефіцієнт Боема, А=2,45;

Size – загальний об'єм відлагодженого програмного коду, тис. рядків;

B – показник ступеня, що визначається співвідношенням

$$B = 1{,}01 + 0{,}001\sum W_i \tag{7.2}$$

де Wi – сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

$$B=1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}026$$

$$T_{ном} = 2{,}45 \cdot 2{,}7^{1{,}026} = 6{,}78 \text{ люд-міс.}$$

Визначаємо уточнені (з урахуванням приведених в МВ додатку 3 сімнадцяти додаткових коефіцієнтів) трудовитрати, люд-міс.:

$$T_{уточн}=T_{ном}\, ПВ_j, \tag{7.3}$$

де $ПВ$j – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн}= 6{,}78\cdot(0{,}88\cdot0{,}93\cdot0{,}88\cdot0{,}91\cdot0{,}95\cdot1\cdot1\cdot0{,}87\cdot1{,}22\cdot1{,}16\cdot1{,}1\cdot1{,}1\cdot$$

$$\cdot1{,}12\cdot1{,}1\cdot1\cdot1{,}1\cdot1{,}1)=9{,}37 \text{ люд-міс.}$$

Ці коефіцієнти дозволяють диференційовано оцінювати результати роботи програмістів, беручи до уваги швидкодію програми, використання різноманітних обчислювальних платформ і інструментів розробки, взаємодію декількох серверів, вимоги до об'ємів баз даних і ін.

Визначаємо підсумкові трудовитрати по стадії робочий проект, люд-дні:

$$T_{РП} = 0{,}3CT_{уточн}^{0{,}33+0{,}2(B-1{,}01)}S, \tag{7.4}$$

де C – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4); S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПО згідно встановленим вимогам. Вибираємо в межах (25…350)%

$$T_{РП}=0{,}3\cdot3{,}23\cdot9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)}\cdot130=265 \text{ люд/день}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.2.

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	9	Д7
Робочий проект	265	Ф 7.1-7.4
Впровадження	13	Д13
Всього	306	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Чисельність ставок інженерів-програмістів для розробки програмного забезпечення визначається за формулою

$$Ч = \frac{T_{пз} N}{F_{рq} - H_{ев}}, \tag{7.5}$$

де $F_{рq}$ – плановий фонд робочого часу одного спеціаліста, днів, $T_{пз}$ – трудомісткість розробки програмного забезпечення люд-дні,

$$Ч = \frac{306 \cdot 1}{48\text{-}5} = 7{,}1 \text{ ставки}$$

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року. Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3

Таблиця 7.3 – Затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за розрахунковий період

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	11	990	16,5
Монітор	60	11	660	11
Клавіатура	30	11	330	5,5
Маніпулятор «мишка»	30	11	330	5,5
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	2	240	4
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор-маршрутизатор	30	2	60	1
Кабельні господарства ЛОМ на 1 м.п.	2,5	200	500	8,33
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	55,49

Час на профілактику обладнання в загальному балансі робочого часу інженерів-електронщиків неповинен складати більше 10%

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф_{др}^{с} = \frac{З_ч \cdot n_{міс}}{1,2} \quad (7.6)$$

$$Ф_{др}^{с} = \frac{55 \cdot 2}{1,2} = 92 \text{ год}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф^{с}_{др}}{F_{др} \cdot T_{зм}} \quad (7.7)$$

$Ч_{ел}$=92/(48·8)=0,24 ставки

Для забезпечення нормального технічного обслуговування засобів ТО та мереж, необхідно прийняти найбільше ціле значення розрахункової чисельності інженерів–електронщиків.

Чисельність інженерів-системотехніків, адміністраторів мережі, дизайнерів WEB вузлів, системних програмістів (аналітиків), бухгалтерів-економістів визначається за потребою в залежності від функціональних обов'язків. Після визначення чисельності персоналу складається штатний розклад.

Таблиця 7.4 – Розрахунок чисельності штатного персоналу сектору системного та адміністративного обслуговування засобів ОТ та комп'ютерних мереж

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 2016, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	0,8	0,2
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,2	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,2	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	0,4	
Всього		1,6	

Продовження таблиці 7.4

Посада	Вид роботи	Час	Кількість штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	2	0,5
	Підтримка постійних клієнтів	1	
	Оформлення договорів, ведення тендерів	0,5	
	Контроль взаєморозрахунків з постачальниками	0,5	
Всього		4	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	0,5	0,2
	Створення графічних і стилістичних елементів сайту	0,5	
	Оформлення банерів і промо-сторінок	0,3	
	Розміщення графіки і контенту на Інтернет сторінках	0,3	
Всього		1,6	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,2
	Верстка друкованих видань	0,2	
	Додрукова підготовка макетів	0,2	
	Розміщення графіки і контенту на Інтернет сторінках	0,2	
Всього		1,6	

Складемо штатний розклад виконавців у таблицю 7.5.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньо-місячний оклад, грн.	Всього за період роз-робки, грн.
Керівник (IT-менеджер)	1	17000	34000
Продакт-менеджер	0,5	14965	14965
Інженер-програміст	7,1	16200	230040
Інженер-електронщик	0,24	10000	4800
Інженер-системотехнік	0,2	10000	4000
Адміністратор мережі	0,2	11000	4400
Системний програміст	0,2	10000	4000
Дизайнер WEB	0,2	11000	4400
Інженер-верстальник	0,2	11000	4400
Бухгалтер-економіст	0,2	10000	4000
Всього за період розробки	$R_{сп}$=10,04	-	$Ф_{роб}$=309005

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рq}}, \tag{7.8}$$

де $Ф_{роб}$ – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{309005}{10,4 \cdot 48} = 619 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість будівель визначається з урахуванням кількості робочих місць виконавців, питомої площі на одне робоче місце, та вартості одного квадратного метра виробничої площі

$$Б_{уд} = R^1_{сп} S_у Ц_{пл}, \quad (7.9)$$

де $R^1_{сп}$ – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць.

$S_у$ – питома площа на одне робоче місце, $м^2$,

$Ц_{пл}$ – вартість одного квадратного метра площі, грн.

Згідно даних інтернет ресурсу DOM.RIA (https://dom.ria.com) ціна одного квадратного метра площі, вік якої не перевищує 30 років, по місту складає 500…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 38 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно 8 $м^2$. З урахуванням цього:

$$Б_{уд} = 8 \cdot 8 \cdot 20000 = 1280000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 128000 грн.

Балансова вартість інвентарю розраховується за нормою 3500 грн на одне робоче місце. Тобто

$$І_{нв} = R^1_{сп} \cdot Ц_м, \quad (7.10)$$

де Цм – ціна меблів для одного робочого місця, грн.

$$І_{нв} = 8 \cdot 3500 = 28000 \text{ грн.}$$

Балансова вартість обчислювальної техніки визначається по оптовим цінам постачальника з врахуванням витрат на транспортування.

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу фірми Комп'ютерторг за 20.10.23 – джерело http://computorg.ua/ru/price.html

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		11771
Системний блок		7771
Процесор	Intel Core i5-6400 (4 ядра по 2.7 - 3.3 GHz), 6 MB Smart Cache	-
Системна плата	Intel Q150 Chipset, 6x USB 2.0, 4x USB 3.0, 1x VGA, 1x COM-порт, 1x DVI, 2x DisplayPort, 2x PS/2, 5x Audio, 1x LAN (RJ-45)	-
Жорсткий диск	SSD 128 Gb + HDD 500 Gb SAMSUNG (3.5", 500ГБ, 32МБ, SATA III)	-
Оперативна пам’ять	8 GB DDR4	-
DVD-привод	DVD±RW ASUS DRW-24B5ST Black Bulk	-
Корпус	Fujitsu Esprimo P756 E90+ Tower	-
Кардрідер внутрішній	Transcend TS-RDF8K USB 3.0	-
інше	Клавіатура, мишка	-
Монітор	Монітор BenQ GL2450HM Black	2600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Сканер	Epson Perfection V37	2800
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965
Пристрій живлення	Powercom BNT-600AP USB	1400

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межах до 10% від оптової ціни.

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп'ютери	8	11771	9416,8	103584,8
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	1	2800	280	3080
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	125216,3

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400

Продовження таблиці 7.8

1	2	3	4
Група 4			
3. Обчислювальна техніка	125216	-	-
Всього по групі	125216	50	62608
Група 5,6			
4. Вимірювальні пристрої	5190	-	-
5. Транспортні засоби	143000	-	
6. Господарський інвентар	28000	-	-
Всього по групі	176190	20	35238
7. Нематеріальні активи	60000	10	6000
Разом	$К_р$= 1769406		$А_р$= 174246

Примітка: вартість автомобіля Dacia Logan LS 2010 взята по даним автобазару «Авто-РІА», джерело https://auto.ria.com/uk/auto_dacia_logan_30226404.html, складає 143000 грн.

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

$$З_о = \frac{З_{сд} \cdot Т_{пз}}{N_е}, \quad (7.11)$$

де $N_е$ – Кількість екземплярів програм, шт.

$$З_о = 619 \cdot 306/60 = 3158 \text{ грн}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних та суспільних обов'язків) на рівні 10%

$$З_д = З_о \cdot Н_q \cdot 0{,}01, \quad (7.12)$$

де $Н_q$ – норматив додаткової зарплати, %

$$З_д = 3158 \cdot 10 \cdot 0{,}01 = 316 \text{ грн.}$$

Відрахування на соціальні потреби за нормативом Нс=22% від суми основної та додаткової зарплати

$$С_{оц} = 0{,}01 \cdot Н_с (З_о + З_д), \quad (7.13)$$

де $Н_с$ – відрахування на соціальні потреби, %

$$С_{оц} = 0{,}01 \cdot 22(3158+316) = 764 \text{ грн}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом *Нг*=15% від основної зарплати

$$Г_{осп} = З_о \cdot Н_г \cdot 0{,}01, \quad (7.14)$$

де $Н_г$ – загальногосподарські витрати, %

$$Г_{осп} = 3158 \cdot 15 \cdot 0{,}01 = 474 \text{ грн.}$$

Визначимо витрати на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.15)$$

де $З_{М1}$ – вартість паперу, грн., $З_{М2}$ – вартість запам'ятовуючих пристроїв, грн., $З_{М3}$ – вартість фарби, картріджей, тонеру, грн., $N_е$ – кількість екземплярів програм, шт.

Згідно прийнятих норм на підприємстві $n_{вит}$ приймаємо 0,4 пачки паперу на період розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=200 грн., визначаємо вартість паперу за період розробки:

$$З_{М1} = Ц_п \cdot N_м. \quad (7.16)$$

$$З_{М1} = 200 \cdot 0{,}4 = 80 \text{ грн.}$$

Згідно прийнятих норм по комплектації до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків. Їх кількість дорівнює кількості коробочних версій запропонованого продукту (приймаємо 10):

$$З_{М2} = \sum Ц_д, \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR box – 23,6 грн./шт., DVD-R box – 49 грн./шт.

$$З_{М2} = 49 \cdot 10 = 490 \text{ грн.}$$

Згідно норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_з., \quad (7.18)$$

де: $Ц_з$ – вартість розхідних матеріалів друкуючих пристроїв: відновлення та заправка картриджу для Canon i-SENSYS LBP6030W – 574 грн.; картридж для Epson Stylus Photo P50 – 558 грн.; відновлення картриджу для MF217W – 570 грн.

$$З_{М3} = 574 + 558 + 570 = 1702 \text{ грн.}$$

$$З_М = (80 + 490 + 1702)/60 = 38 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційних систем за нормативом ($Н_п$ = 15%) від основної зарплати виконавців

$$О_п = З_о \cdot Н_п \cdot 0{,}01, \quad (7.19)$$

де $Н_п$ – норматив витрат на освоєння нових мов програмування, %

$$О_п = 3158 \cdot 15 \cdot 0{,}01 = 474 \text{ грн}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 60 прим.)

$$А_м = \frac{А_р \cdot N_{міс}}{N_е \cdot 12}, \quad (7.20)$$

де $А_р$ – загальна річна сума амортизаційних відрахувань, грн.

$$А_м = 174246 \cdot 2/(60 \cdot 12) = 484 \text{ грн}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції

$$С_п = З_о + З_д + С_{оц} + Г_{осп} + З_м + О_п + А_м. \quad (7.21)$$

$$С_п = 3158 + 316 + 764 + 474 + 38 + 474 + 484 = 5708 \text{ грн.}$$

Визначимо плановий прибуток за рівнем рентабельності (Рп) програмної продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 50%

$$П_р = 0{,}01 \cdot P_п \cdot C_п, \quad (7.22)$$

де $P_с$ – рівень рентабельності, %

$$П_р = 0{,}01 \cdot 50 \cdot 5708 = 2854 \text{ грн.}$$

Величини ціна підприємства, податок на додану вартість, відпускна ціна програмної продукції визначаються за формулами, приведеними в таблиці 7.9

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн.
1	2	3
1. Основна зарплата виконавців	$З_о$	3158
2. Додаткова зарплата виконавців	$З_д$	316
3. Відрахування на соціальні потреби	$С_{оц}$	764
4. Загальногосподарські витрати	$Г_{осп}$	474
5. Витрати на матеріали	$З_М$	38
6. Освоєння нових операційних систем, мов програмування	$О_п$	474
7. Амортизація основних фондів	$А_м$	484
8. Повна собівартість програмного забезпечення	$С_п$	5708
9. Плановий прибуток	$П_р$	2854
10.Ціна підприємства $Ц_п=С_п+П_р$	$Ц_п$	8562
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	1712,4
12.Відпускна ціна програмної продукції Ц = Цп + ПДВ	$Ц$	10274,4

7.6 Визначення об’єму капітальних вкладень у споживача програмної продукції

Об’єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах. Результати розрахунків зводимо у таблицю 7.10.

Таблиця 7.10 – Розрахунок об’єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн	
	Базовий	Новий
Вартість програмної продукції	–	10274
Всього капітальних витрат	–	10274

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаємо при умові роботи підсистеми на протязі року. Результати зводимо до таблиці 7.11.

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позна-чення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на технічне обслуговування	$З_р$	64416	24156
2. Витрати на електроенергію	$З_{ел}$	0	0
3. Витрати на амортизацію	$З_{ам}$	0	5137
Всього витрат за рік	$І$	64416	29293

Витрати на технічне обслуговування:

$$З_р = Т_р \cdot З_г \cdot (1 + 0{,}01 \cdot Н_q) \cdot (1 + 0{,}01 \cdot Н_с), \quad (7.23)$$

де $Т_р$ – кількість годин обслуговування системи за рік, *год*,

$З_г$ – заробітна плата обслуговуючого персоналу, *грн / год*

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 320 годин на рік до 120 годин на рік, тому витрати на технічне обслуговування зменшилися з

$$З_{р\,баз} = 320 \cdot 150 \cdot 1{,}1 \cdot 1{,}22 = 64416\ грн.$$

до

$$З_{р\,нов} = 120 \cdot 150 \cdot 1{,}1 \cdot 1{,}22 = 24156\ грн.$$

Витрати на електроенергію визначаються з урахуванням спожитої потужності ($П_{ел}$) в кіловатах, часу експлуатації технічних засобів ($Т_р$) в годинах та ціни однієї кіловат-години ($Ц_{ел}$).

$$З_{ел} = П_{ел} \cdot Т_р \cdot Ц_{ел}. \quad (7.24)$$

Зел баз= Зел нов

Витрати на електроенергію не змінились.

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.12.

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн., за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	50	–	10274	–	5137
Всього відрахувань	-	–	10274	–	5137

7.8 Визначення економічної ефективності програмної продукції

Економічна ефективність програмного забезпечення визначається для виготовлювача і споживача за такими показниками.

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{p_m} \cdot K_{p_m}, \quad (7.25)$$

де: $K_р$ – балансова вартість основних фондів розробника, грн.; $E_р$ – розрахунковий коефіцієнт капіталовкладень.

$$Ев = (8562 - 5708) \cdot 60 - (0,05 \cdot 1408000 + 0,5 \cdot 125216 +$$
$$+0,2 \cdot 176190 + 0,1 \cdot 60000) \cdot 2/12 = 142200 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_p^*}{(Ц_п - C_п) \cdot N_е}, \quad (7.26)$$

де: K_p^* – балансова вартість основних фондів розробника без врахування вартості ОФ третьої групи, так як їх строк служби на порядок більший ніж період розробки ПЗ.

$$T_в = \frac{361406}{(8562 - 5708) \cdot 60 \cdot 12 / 2} = 0,4 \text{ років}$$

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б - I_н) - E_н(K_н - K_б), \quad (7.27)$$

де $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно, $K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються

$$E_{сп} = (64416\text{-}29293)\text{-}0,5 \cdot 10274 = 29986 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н} \quad (7.28)$$

$$T_{сп} = \frac{10274}{64416 - 29293} = 0,3 \ року$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	60
2. Повна собівартість розробленої програми	Грн.	5708
3. Ціна розробленої програми	Грн.	8562
4. Плановий прибуток від реалізації розробленої програми	Грн.	2854
5. Рентабельність програмної продукції	%	50
6. Об’єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1769406
7. Загальний прибуток від реалізації програмної продукції	Грн.	171240
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	142200
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	0,4
10.Об’єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	10274
11.Величина економічного ефекту у користувача програмної продукції	Грн.	29986
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,3

7.9 Висновки

Розроблена програма економічно вигідна. За рахунок впровадження програмного забезпечення досягається скорочення часу обробки інформації, підвищується культура праці, підвищення якості приймаючих управлінських рішень.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

В охорону праці включають санітарно-гігієнічні, лікувально-профілактичні та організаційно-технічні системи правових і соціально-економічних заходів.

В кожній ІТ-компанії є трудові відносини з працівниками. Згідно закону України “Про охорону праці” [42], кожна компанія впроваджує заходи з охорони праці. Реалізується трудові відносини з вживанням необхідних засобів з охорони праці та розробки відповідних документів:

– інструкції з охорони праці по кожній професії і загальні;

– положення про охорону праці;

– накази з охорони праці;

– журнали реєстрації та інструктажу.

Роботодавець створює відділ, який працює відповідно до типового положення, який затверджується центральним органом виконавчої влади і забезпечує виконання вимог державної політики у сфері охорони праці.

За недотримання вимог, керівники ІТ-компаній можуть бути притягнуті до відповідальності, яка полягає у накладанні штрафу. Якщо в результаті порушення умов охорони праці є постраждалі працівники, то керівні особи ІТ-компаній притягуються до кримінальної відповідальності.

Законом України “Про охорону праці” регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями», яким затверджено нормативно-правовий акт з охорони праці НПАОП 0.00-7.15-18 «Правила охорони праці під час експлуатації

електронно-обчислювальних машин» [44], та «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-о

б Програмісти у процесі роботи мають негативний вплив на органи зору, а також мають значну розумову напругу і нервово-емоційне навантаження. Руки (суглоби пальців та м'язи рук) при роботі з клавіатурою мають теж істотне навантаження. До шкідливих факторів, які впливають на робітників галузі інформаційних технологій, відносять високочастотні електромагнітні коливання (випромінювання) роботи апаратної частини ЕОМ.

в Ці шкідливі фактори можуть привести до професійних захворювань.

а Розглянемо шкідливі чинники роботи програмістів, керуючись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98, та «Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» НПАОП 0.00-7.15-18.

х Умови праці програміста включають наступні фактори:

– параметри повітряного середовища в приміщенні;

м – рівень шуму в приміщенні;

а – освітлення приміщення.

ш Щоб запропонувати заходи щодо зменшення негативного впливу комп’ютера на організм людини, визначимо фактори, які можуть викликати професійне захворювання і впливають на працездатність програміста.

»

8.2 Аналіз умов праці на робочому місці програміста

Д Згідно НПАОП 0.00 – 1.28 – 10 «Правила охорони праці під час С експлуатації електронно-обчислювальних машин» [46], площа робочого а приміщення повинна задовольняти умові – не менш 6 м^2 на одне робоче місце. В Виконання даних вимог забезпечить підтримку в приміщенні оптимального П значення вологості й складу повітря.

і

Н

Відповідно ДБН В.2.5 – 28 – 2018 [47] роботу програміста можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення, де працює програміст, можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює, на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при сполученому висвітленні), повинен становити 0,5%, освітленість при штучному висвітленні повинна становити 300 лк.

За результатами виміру освітленості відділом охорони праці величина освітленості від системи загального штучного висвітлення лежить у межах 200-250 лк, що не відповідає вимогам, які пред'являються до приміщення.

Відповідно ДСанПіН 3.3.2.007 – 98, рівні звукового тиску в робочому приміщенні не повинні перевищувати в октавних смугах із середньо-геометричними частотами наступних значень, наведених у таблиці 8.1.

Таблиця 8.1 – Допустимі спектри рівнів звукового тиску

Робоче місце	Рівень звукового тиску, дБ, в октавних смугах із середньогеометричними частотами, Гц								Рівень звуку і еквівалентний рівень звуку, дБА
	63	125	250	500	1000	2000	4000	8000	
Приміщення конструкторських бюро, розташування обчислювальних машин, лабораторій для теоретичних робіт і опрацювання експериментальних даних	71	61	54	49	45	42	40	38	50

У приміщенні перебувають наступні джерела шуму: електродвигуни вентиляторів ЕОМ; працюючі принтери; працюючі дисководи. Шум, вироблений вентилятором, можна класифікувати як постійний, всі інші джерела шуму, як імпульсні. Відповідно паспорта на приміщення рівень звуку, Дб(А), обмірюваний за шкалою (А) шумоміра рівень шуму досяг величини 28,3 Дб(А) при роботі всього устаткування, включаючи й копіювальний апарат. Це дозволяє зробити висновок про відповідність рівня звуку в приміщенні вимогам нормативних актів.

Ергономічні вимоги до робочого місця працюючого з ВДТ ЕОМ і ПЕОМ нормуються НПАОП 0.00 – 1.28 – 10**.** Оптимальне положення тіла того, що працює забезпечується відповідною конструкцією робочого місця, а також регуляцією висоти робочої поверхні, сидіння, простору й підставки для ніг. Відмінності реальних параметрів робочого місця від параметрів відповідно вимог нормативного акту приведені в таблиці 8.2.

Таблиця 8.2 – Відмінності реальних параметрів робочого місця від параметрів відповідно вимог нормативного акту

Ріст людини, см	Висота робочої поверхні мм,	Висота простору для ніг, мм	Висота робочого сидіння, мм
175	765 (740)	655 (600)	450 (440)

У дужках зазначені реальні значення параметрів робочого місця; всі вони не відповідають параметрам, зазначеним у стандарті.

Параметри мікроклімату можуть мінятися в широких межах, тоді як необхідною умовою життєдіяльності людини є підтримка сталості температури тіла завдяки властивості терморегуляції, тобто здатності організму регулювати віддачу тепла в навколишнє середовище.

У приміщеннях, де встановлені комп'ютери, повинні дотримуватися певні параметри мікроклімату. У санітарних нормах встановлені величини параметрів мікроклімату, що створюють комфортні умови. Ці норми встановлюються в залежності від пори року, характеру трудового процесу і характеру виробничого приміщення (див. табл. 8.3).

Таблиця 8.3 – Параметри мікроклімату для приміщень, де встановлені комп'ютери

Період року	Параметр мікроклімату	Величина
Холодний	Температура повітря в приміщенні Відносна вологість Швидкість руху повітря	22 – 24°С 40 – 60% до 0,1 м/с
Теплий	Температура повітря в приміщенні Відносна вологість Швидкість руху повітря	23 – 25°С 40 ... 60% 0,1 ... 0,2 м / с

8.3 Розробка заходів з умов поліпшення охорони праці

Перерахуємо проведені заходи щодо забезпечення умов праці на робочому місці програміста.

З точки зору забезпечення електробезпеки до цих заходів можна віднести: обладнання розподільних щитів спеціальними розетками з заземлюючими контактами; організація заземлення всіх приладів і пристроїв; періодична перевірка всіх приладів і пристроїв; щорічна здача іспитів з охорони праці.

З точки зору забезпечення оптимальних умов мікроклімату, рівня звуку і освітленості до цих заходів можна віднести: організацію природної вентиляції для забезпечення необхідного повітрообміну в виробничому приміщенні; організацію системи центрального опалювання для підтримки оптимальної температури в холодний період року; організацію штучного загального освітлення для забезпечення необхідних умов зорової роботи, оформлення паспорта на виробниче приміщення з занесенням в нього вимірювань освітленості і рівня звуку, проведених відділом охорони праці.

З точки зору забезпечення пожежної безпеки до цих заходів можна віднести наявність схеми евакуації з виробничого приміщення у випадку пожежі, вказану на вхідних дверях.

Аналіз умов праці на робочому місці інженера-програміста показав, що на робочому місці не виконуються вимоги ергономіки. Для виконання їх можна запропонувати заміну не регульованого сидіння на крісло з регульованими ергономічними параметрами, а також заміну використовуваного столу на робоче місце оператора ЕОМ.

Різними фірмами в сукупності розроблено понад 11 схем регулювань параметрів робочого крісла, які забезпечують плавне переміщення сидіння по висоті за допомогою газової пружини; плавна зміна нахилу спинки і сидіння; регулювання пружинного протитиску спинки крісла на спину оператора; перестановку спинки по висоті; зміна глибини сидіння шляхом зміни вигину краю сидіння; синхронне повторення рухів оператора сидінням і спинкою в

правильному кутовому співвідношенні; синхронне повторення спинкою крісла рухів верхньої частини тулуба того, що сидить; амортизацію сидіння.

Найбільш популярними моделями комп'ютерних крісел, які володіють чотирма основними регулюваннями (висоти сидіння, висоти, глибини і нахилу спинки), є італійські, фінські моделі «Senior», «Volos», «Ergo», «Toronto», «Bini», «Metro», «NewStar», «Capris», «Fenix», «Xenus», «Quintus» і т.д. Подібні крісла, відрегульовані відповідно до зростання і ваги оператора, а також характеру виконуваної роботи, дозволяють понизити навантаження на опорно-руховий апарат людини, що працює за комп'ютером.

8.4 Розрахункова частина

Для захисного штучного заземлення будемо застосовувати вертикальні електроди з сталевого прокату круглого перерізу діаметром *48 мм.*, довжиною *L=2 м.,* та горизонтальний електрод – металева полоса з перетином *45·5* мм. Напруга – *220/380 В*. Розрахункова схема розташування заземлюючих електродів – по контуру (прямокутником).

Розрахунок проводиться за допустимим опором розтіканню струму заземлювача.

Початкові дані для розрахунку захисного заземлення: тип верхнього шару грунта – чорнозем, нижнього шару грунта – глина (питомий опір $\rho_2 = 40$ *Ом·м)*. Умовна товщина верхнього шару грунта: *H=0,6 м*. Відстань між вертикальними заземлювачами (електродами) *A=2,5 м*. Глибина закладення горизонтального контура заземлення *t=0,85 м*. Опір заземлювача, який нормується: $R_{ЗН} = 4$ *Ом*. Необхідно визначити необхідну кількість вертикальних заземлювачів та довжину полоси (горизонтального заземлювача).

Розрахунок

Відстань від центра вертикального заземлювача до поверхні землі:

$$T=t+L/2=0,85+2/2=1,85 \text{ м.}$$

Розрахунковий питомий опір грунта (з врахуванням того, що фактично вся конструкція заземлювача розташовується у нижньому шарі грунта):

$$\rho = \psi\rho = 1,36 \cdot 40 = 54,5 \text{ Ом·м.}$$

де $\psi = 1,36$ – табличне значення коефіцієнта сезонності для відповідної кліматичної зони у багатошаровому грунті [11];

$\rho_2 = 40$ *Ом·м.* – табличне значення питомого опору нижнього шару грунта (глина) [11].

Діаметр вертикального електрода (заданий):

$$Dв = 48 \text{ мм.} = 0,048 \text{ м.}$$

Відношення $A/L=2,5/2=1,25$.

Опір розтіканню електричного струму одного електрода вертикального заземлювача з урахуванням заглиблення заземлювача [11]:

$$R_O=0,366(\rho/L)[\lg(2L/Dв)+(1/2)\lg((4T+L)/(4T-L))]=$$

$$=0,366(54,5/2)[(\lg(2\cdot 2/0,048)+(1/2)\lg((4\cdot 1,75+2)/(4\cdot 1,75-2))]=$$

$$=20,2 \text{ Ом.}$$

Визначаємо коефіцієнт екранування вертикальних електродів $Кев=0,53$ при орієнтовній кількості вертикальних електродів, яке дорівнює 5 [11].

Визначаємо необхідну кількість вертикальних електродів заземлювача (без врахування горизонтального заземлювача), при $R_{ЗН} = 4$ *Ом*:

$$N=R_O/(Кев\ R_{ЗН})= 20,2/(0,53\cdot 4)= 9,85 \approx 10 \text{ шт.}$$

Визначаємо довжину з'єднуючої полоси:

$$L_П = 1,05\cdot A\cdot N = 1,05\cdot 2,5\cdot 10 = 25,5 \approx 26 \text{ м.}$$

Опір розтіканню електричного струму з'єднуючої полоси з урахуванням кліматичного коефіцієнта питомого опору грунта $K_П$ [11]:

$$R_П = 0,366(\rho\cdot K_П/L_П)\lg(2(L_П\cdot L_П)/(B\cdot t))=$$

$$=0,366(40\cdot 5/26)\cdot\lg((2\cdot 26^2)/(0,048\cdot 0,65))=20,3 \text{ Ом.}$$

де $K_П=5$ – табличне значення кліматичного коефіцієнта питомого опору грунта для відповідної кліматичної зони для з'єднуючої полоси [11];

$B = 45$ *мм.* $= 0,045$ *м.*- ширина з'єднуючої полоси (задана).

Загальний опір розтіканню електричного струму заземлювача [11]:

$$R =(R_O \cdot R_П)/(R_O \cdot \eta_П + N \cdot R_П \cdot Кев)=$$

$$=(20,6 \cdot 20,2)/(20,6 \cdot 0,55 + 10 \cdot 20,2 \cdot 0,53)=3,36 \text{ Ом.}$$

де $\eta_П = 0,55$ – табличне значення коефіцієнта екранування з'єднуючої полоси [11].

Умова $R \le R_{ЗН}$ виконується ($3,36 \le 4$).

Так як R суттєво більше $R_{ЗН}$, зменшимо кількість вертикальних електродів до 9 і виконаємо перерахунок. У результаті остаточно отримали: кількість вертикальних електродів дорівнює 9 при $R = 3,75$ *Ом.*

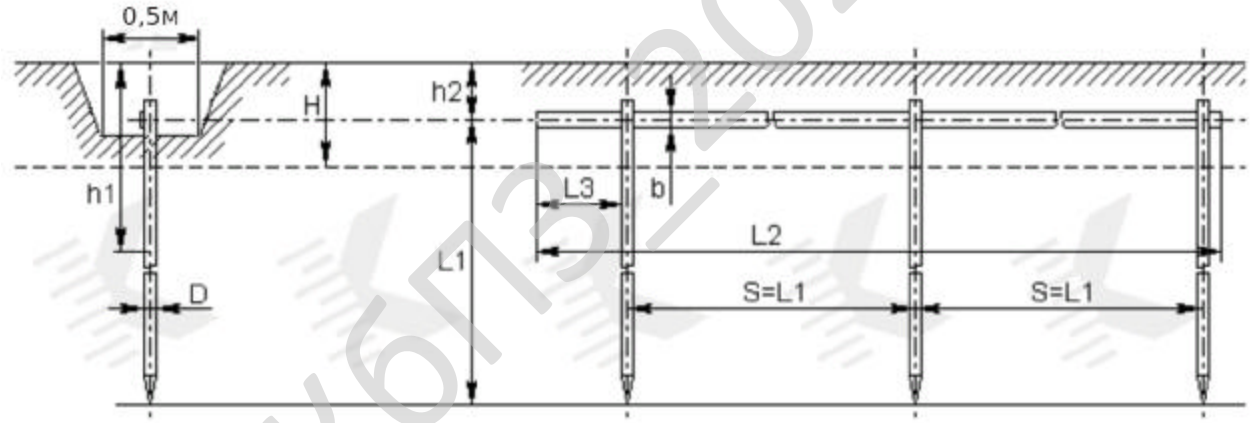

Рисунок 8.1 – Схема штучного заземлення

8.5 Висновки до розділу

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров'я працівників, а також підвищує ефективність виробництва в цілому.

З цих міркувань було здійснено аналіз приміщення, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи.

Тільки повна усвідомленість працівника про можливі небезпеки, що можуть підстерігати його на робочому місці та дотримання вимог нормативних актів з питань охорони праці та відповідних рекомендацій фахівців, дозволять значною мірою знизити негативний вплив шкідливих та небезпечних факторів при роботі з комп'ютером на організм людини.

Виконано розрахунок захисного штучного заземлення, як одного з ключових факторів безпеки програміста.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для функціонування у складі системи аварійного і попереджувального захисту, важливої для безпеки АЕС.

Ця розробка є частиною «Державної Комплексної зведеної програми підвищення безпеки».

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання розробки програмного забезпечення для комп'ютерних систем контроля і управління систем, важливих для безпеки.

При рішенні даного завдання були вирішенні наступні задачі:

– визначене функціональне призначення програмного забезпечення;

– оглянуті існуючі системи-аналоги, технології, архітектури, програмних рішень;

– визначені категорії, класу безпеки і вимоги до програмного забезпечення;

– пройшов ознайомлення з комп'ютерною системою контроля і управління, в складі якої запроваджене функціонування програмного забезпечення;

– визначенні вимоги до інтерфейсу користувача, каналів комунікації з технічним обладнанням, кіберзахисту і модифікованості;

– розроблене програмного забезпечення, що реалізує, згідно вимог, наступні функції:

– зміни значень контрольованих параметрів, при яких відбувається спрацювання захисту (зміни уставок);

– блокування спрацювання вибраних умов захисту;

– виконав інтеграцію програмного забезпечення у ПТК.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти технічні рішення дозволяють успішно функціонувати у складі комп'ютерних систем контроля і управління.

Під час валідаційних іспитів всієї системи отримані підтвердження, що:

– програма «МКУ» виконує покладені на неї функції у повному обсязі;

– програма «МКУ» відповідає пред'явленим до неї вимогам безпеки.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

Розроблене програмне забезпечення має простий та зручний інтерфейс користувача.

При створенні проекту було проведено дослідження вимог та методів розробки програмного забезпечення критичного застосування, зокрема таких етапів життєвого циклу, як проектування, розробка, верифікація, валідація та впровадження.

При створенні проекту програмного забезпечення було проведено детальний аналіз вимог та створена специфікація вимог відповідно до стандарту розробки IEC62138.

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

Програма реалізована на мові високого рівня C++ в середовищах розробки Microsoft Visual Studio та Microsoft Visual Studio Embedded. Це дозволило забезпечити гарну структурованість та швидкодію програми.

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

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 галузях критичного застосування.

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 29986 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,3 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Конвенція про ядерну безпеку (укр/рос). Офіційний вебпортал парламенту України. Режим доступу: https://zakon.rada.gov.ua/go/995_023.

2. НП 306.2.141–2008. Загальні положення безпеки атомних станцій. Держатомрегулювання України, Київ, 2007.

3. НП 306.2.202-2015. Вимоги з ядерної та радіаційної безпеки до інформаційних та керуючих систем, важливих для безпеки атомних станцій. Держатомрегулювання України, Київ, 2015.

4. IEC61508-2010: Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 1: General requirements. Geneva, International Electrotechnical Commission (IEC), 2010.

5. IEC61513-2011: Nuclear power plants – Instrumentation and control important to safety – General requirements for systems requirements. Geneva, International Electrotechnical Commission (IEC), 2011.

6. IEC61226-2009: Nuclear power plants – Instrumentation and control important to safety – Classification of instrumentation and control functions. Geneva, International Electrotechnical Commission (IEC), 2009.

7. IEC60880-2006: Nuclear power plants – Instrumentation and control systems important to safety – Software aspects for computer-based systems performing category A functions. Geneva, International Electrotechnical Commission (IEC), 2006.

8. IEC62138-2004: Nuclear power plants –Instrumentation and control important for safety –Software aspects for computer-based systems performing category B or C functions. Geneva, International Electrotechnical Commission (IEC), 2004.

9. IEC61772:2009: Nuclear power plants — Control rooms — Application of visual display units (VDUs). Geneva, International Electrotechnical Commission (IEC), 2009.

10. IEC62646-2019: Nuclear power plants – Control rooms – Computer based procedures. Geneva, International Electrotechnical Commission (IEC), 2012.

11. ISO/IEC 27000-2018: Information technology — Security techniques — Information security management systems — Overview and vocabulary. 2018.

12. Keith Stouffer, Victoria Pillitteri, Suzanne Lightman,Marshall Abrams, Adam Hahn. NIST SP 800-82 Revision 2: Guide to Industrial Control Systems (ICS) Security. National Institute of Standards and Technology, 2015.

13. Design of Instrumentation and Control Systems for Nuclear Power Plants: Specific Safety Guide No. SSG-39. Vienna, IAEA, 2016.

14. Safety Classification of Structures, Systems and Components in Nuclear Power Plants: Specific Safety Guide No. SSG-30. Vienna, IAEA, 2014.

15. Application of the Safety Classification of Structures, Systems, and components in Nuclear Power Plants: IAEA-TECDOC-1787. Vienna, IAEA, 2016.

16. ASME NQA-1–2008. Quality Assurance Requirements for Nuclear Facility Applications. An American National Standard. American Society of Mechanical Engineers, 2008.

17. NUREG/CR-6463. Review Guidelines on Software Languages for Use in Nuclear Power Plant Safety Systems. U.S. Nuclear Regulatory Commission, 1996.

18. NUREG-0700 Revision 3. Human-System Interface Design Review Guidelines. U.S. Nuclear Regulatory Commission, 2020.

19. СОУ НАЕК 100:2022. Інформаційні та керуючі системи, важливі для безпеки атомних електричних станцій: загальні технічні вимоги. Стандарт державного підприємства «Національна атомна енергогенеруюча компанія «Енергоатом». ДП НАЕК «Енергоатом», Київ, 2022.

20. Клєвцов А. Л., Ястребенецький М. А., Трубчанінов С. А. Комп'ютерна безпека інформаційних та керуючих систем АЕС: нормативна база. Ядерна та радіаційна безпека 4(68), 2015.

21. 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. Режим доступу: https://doi.org/10.26906/SUNZ.2023.2.170.

22. Вінтенко Б.Ю., Смірнов О.А., Коваленко А.С., Смірнов С.А., Буравченко К.О. «Дослідження вимог міжнародних стандартів IEC60880 та IEC62138 з розробки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Системи управління, навігації та зв'язку*, 2023, вип. 3(73), С. 155-166.Режим доступу: https://doi.org/10.26906/SUNZ.2023.3.155.

23. Daivd J. Smith, Kenneth G. L. Simpson. The Safety Critical Systems Handbook. A Straightforward Guide to Functional Safety IEC 61508 (2010 Edition), IEC 61511 (2015 Edition) and Related Guidance: Forth Edition. Elsevier Ltd, 2016.

24. Hobbs C. Embedded Software Development for Safety-Critical Systems: Second Edition. CRC Press, 2020.

25. McConnel S. Code Complete. Second Edition. Microsoft Press, 2010.

26. Бородкіна І.Л., Бородкін Г.О. Інженерія програмного забезпечення: посіб. для студ. вищ. навч. закладів, Київ: Центр навчальної літератури, 2018.

27. Лавріщева К. М. Програмна інженерія. Підручник. Інститут програмних систем НАН України, Київ, 2008.

28. Грицюк Ю.І. Аналіз вимог до програмного забезпечення: Львів: Львівська політехніка, 2018.

29. Шилдт Г., C++: базовий курс, 3-е видання. Київ: Науковий світ, 2022.

30. Страуструп Б. Програмування. Принципи та практика з використанням C++.: Вільямс, 2018. 1328 с.

31. Knuth D. The Art of Computer Programming, Vol. 1: Fundamental Algorithms, 3rd Edition 3rd Edition. – Addison-Wesley Professional, 2019.

32. Knuth D. The Art of Computer Programming: Vol. 3: Sorting and Searching 2nd Edition, Kindle Edition. – Addison-Wesley Professional, 2019.

33. Knuth D. Art of Computer Programming, Vol. 2: Seminumerical Algorithms 3rd Edition, Kindle Edition. – Addison-Wesley Professional, 2019.

34. Бєлов Ю. А., Карнаух Т. О., Коваль Ю. В., Ставровський А. Б. Вступ до програмування мовою С++. К.: Видавничо-поліграфічний центр «Київський університет», 2012.

35. Кравець П. Об'єктно-орієнтоване програмування: навч. посібник / П. О. Кравець. – Львів: Видавництво Львівської політехніки, 2012.

36. Роберт С. Мартін. Чистий код. Створення, аналіз і рефакторинг, 2019.

37. Robert C. Martin. Agile Software Development, Principles, Patterns and Practices. – Prentice Hall. 2018.

38. Офіційна документація Visual Studio. – [Електронний ресурс] - 2022 Режим доступу: https://docs.microsoft.com/en-us/visualstudio/ide/?view=vs-2022.

39. Довідник з мови С++. – [Електронний ресурс] - Режим доступу: https://en.cppreference.com.

40. Guckkenheimer S., Peter J. Software Engineering With Microsoft Visual Studio Team System. – Adison Wesley, 2006.

41. Мелешко Є.В., Якименко М.С., Поліщук Л.І. Алгоритми та структури даних: Навчальний посібник для студентів технічних спеціальностей денної та заочної форми навчання. – Кропивницький: Видавець – Лисенко В.Ф., 2019.

42. Oleksii Smirnov, Oleksandr Kovalenko, Jamil Al-Azzeh, Anna Kovalenko, Serhii Smirnov. Qualitative risk analysis of software development. Asian Journal of Information Technology. – Volume 17(3). – Medwell Journals. – 2018. – P. 218-230.

43. Закон України «Про охорону праці» від 14.10.1992 р. № 2694-XII. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2694-12.

44. Наказ Міністерства соціальної політики України 14.02.2018 № 207 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z0508.

45. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин: ДСанПІН 3.3.2-007-98. – Режим доступу: https://zakon.rada.gov.ua/rada/show/v0007282-98.

46. «Про затвердження Правил охорони праці під час експлуатації електронно-обчислювальних машин», Офіційний вебпортал парламенту України. – Режим доступу: https://zakon.rada.gov.ua/go/z0293-10.

47. Державні будівельні норми України: ДБН В.2.5-28:2018. – Режим доступу до ресурсу: https://goo.su/9AkQ.

48. Зеркалов Д. В. Охорона праці в галузі: Загальні вимоги: навч. посіб. Київ: Основа. 2011.

49. Охорона праці. Ч. 1. Захисне заземлення: метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ сумісного типу / Кіровоград. ін-т с.-г. машинобуд.; [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака]. – Кіровоград: КІСМ, 1997. – 20 с. Режим доступу: http://dspace.kntu.kr.ua/jspui/handle/123456789/4358.

50. Постанова № 42 від 01.12.1999 Головного державного санітарного лікаря України «Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99. – Режим доступу: https://zakon.rada.gov.ua/rada/show/va042282-99.

51. Сакулін В.П., Шептовицький В.М. Безпека праці під час монтажу та експлуатації електроустановок / В.П.Сакулін, В.М.Шептовицький. – Л. : “Колос”, 1973.

52. Оришака, О. В. Основи охорони праці: навч. посіб. / О. В. Оришака, Г. П. Горбачова, К. М. Марченко; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. – Кропивницький: ЦНТУ, 2022. Режим доступу: http://dspace.kntu.kr.ua/jspui/handle/123456789/12161.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію комп’ютерної системи управління АЕС критичного застосування.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 33-13 від 04.08.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація комп’ютерної системи управління АЕС критичного застосування.

4 Джерела розробки

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

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

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

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи комп'ютерної системи управління АЕС критичного застосування;

– відповідність реалізації програми галузевим вимогам;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows 10/11 та Windows CE 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 та Windows CE.

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

Microsoft Visual C++ 2022/Microsoft Visual Studio Embedded.

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 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 20 жовтня 2023 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинен бути розглянутий аналіз умов праці на робочому місці програміста.

9 Перелік документів, що розробляються

- Наукова новизна – 1 аркуш.
- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Показники економічної ефективності – 1 аркуш.
- Пояснювальна записка – 113 аркушів.

10 Етапи розробки

10.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.

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

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 10.12.2023 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 22.12.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

_____ Смірнов О. А.

Дослідження та програмна реалізація комп’ютерної системи управління АЕС критичного застосування

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 45

Літера: РП

Кропивницький – 2023

Основна програма

MKY2023.cpp – головний клас програми

```
// MKY2023.cpp : Defines the class behaviors for the application.
//
#include "pch.h"
#include "MKY2023.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif

// CMKYApp

BEGIN_MESSAGE_MAP(CMKYApp, CWinApp)
	ON_COMMAND(ID_HELP, &CWinApp::OnHelp)
END_MESSAGE_MAP()


// CMKYApp construction

CMKYApp::CMKYApp() :
	m_LogFile(_T("MKYLog.txt")),
	m_Security(m_ParamStore),
	m_CommThread(m_DataManager, m_Security, m_LogFile)
{
	// support Restart Manager
	m_dwRestartManagerSupportFlags = AFX_RESTART_MANAGER_SUPPORT_RESTART;
}

//-----------------------------------------------------------------------------
CMKYApp::~CMKYApp()
{
}

const CLogFile& CMKYApp::LogFile() const
{
	return m_LogFile;
}

CLogFile& CMKYApp::LogFile()
{
	return m_LogFile;
}

const CParamStore& CMKYApp::ParamStore() const
{
	return m_ParamStore;
}

CParamStore& CMKYApp::ParamStore()
{
	return m_ParamStore;
}

const CProjectSettings& CMKYApp::ProjectSettings() const
{
	return m_ProjectSettings;
}

CProjectSettings& CMKYApp::ProjectSettings()
{
	return m_ProjectSettings;
}
```

```
const CSecurity& CMKYApp::Security() const
{
	return m_Security;
}

CSecurity& CMKYApp::Security()
{
	return m_Security;
}

const CCommThread& CMKYApp::CommThread() const
{
	return m_CommThread;
}

CCommThread& CMKYApp::CommThread()
{
	return m_CommThread;
}

const CDataManager& CMKYApp::DataManager() const
{
	return m_DataManager;
}

CDataManager& CMKYApp::DataManager()
{
	return m_DataManager;
}

// The one and only CMKYApp object

CMKYApp theApp;


// CMKYApp initialization

BOOL CMKYApp::InitInstance()
{
	// InitCommonControlsEx() is required on Windows XP if an application
	// manifest specifies use of ComCtl32.dll version 6 or later to enable
	// visual styles.  Otherwise, any window creation will fail.
	INITCOMMONCONTROLSEX InitCtrls;
	InitCtrls.dwSize = sizeof(InitCtrls);
	// Set this to include all the common control classes you want to use
	// in your application.
	InitCtrls.dwICC = ICC_WIN95_CLASSES;
	InitCommonControlsEx(&InitCtrls);

	CWinApp::InitInstance();

	AfxEnableControlContainer();

	// Create the shell manager, in case the dialog contains
	// any shell tree view or shell list view controls.
	CShellManager* pShellManager = new CShellManager;

	// Activate "Windows Native" visual manager for enabling themes in MFC
controls
	CMFCVisualManager::SetDefaultManager(RUNTIME_CLASS(CMFCVisualManagerWindow
s));
	SetRegistryKey(_T("MKY2023"));


	//
	BOOL ConfigurationOk = m_ParamStore.Load();
	if (ConfigurationOk == TRUE)
	{
```

```
            ConfigurationOk = m_ProjectSettings.Load(m_ParamStore.ProjectFile(),
m_ParamStore.ShowReserved(), m_DataManager);

            if (ConfigurationOk == FALSE)
            {
                  CString str;
                  str.Format(_T("Помилка              проекта              [%s]!"),
m_ParamStore.ProjectFile());
                  AfxMessageBox(str, MB_OK | MB_ICONERROR);
            }
      }

      m_CommThread.Execute(m_ParamStore.Port());

      BOOL ConnectedOK = m_CommThread.IsConnected();

      if (ConnectedOK == FALSE)
      {
            AfxMessageBox(_T("Помилка порта ") + m_ParamStore.Port(), MB_OK |
MB_ICONERROR);
      }

      //if (/*ConfigurationOk == TRUE && */ConnectedOK == TRUE)
      {
            CMKYDlg dlg(m_ParamStore, m_LogFile, m_DataManager, m_Security,
m_CommThread, m_ProjectSettings);
            m_pMainWnd = &dlg;
            INT_PTR nResponse = dlg.DoModal();
            if (nResponse == IDOK)
            {
                  // TODO: Place code here to handle when the dialog is
                  //  dismissed with OK
            }
            else if (nResponse == IDCANCEL)
            {
                  // TODO: Place code here to handle when the dialog is
                  //  dismissed with Cancel
            }
            else if (nResponse == -1)
            {
                  TRACE(traceAppMsg, 0, "Warning: dialog creation failed, so
application is terminating unexpectedly.\n");
                  TRACE(traceAppMsg, 0, "Warning: if you are using MFC controls
on the dialog, you cannot #define _AFX_NO_MFC_CONTROLS_IN_DIALOGS.\n");
            }
      }

      // Delete the shell manager created above.
      if (pShellManager != nullptr)
      {
            delete pShellManager;
      }

#if !defined(_AFXDLL) && !defined(_AFX_NO_MFC_CONTROLS_IN_DIALOGS)
      ControlBarCleanUp();
#endif

      // Since the dialog has been closed, return FALSE so that we exit the
      //  application, rather than start the application's message pump.
      return FALSE;
}
```

MKY2023.h – головний клас програми

```
// MKY2023.h : main header file for the PROJECT_NAME application
//

#pragma once

#ifndef __AFXWIN_H__
	#error "include 'pch.h' before including this file for PCH"
#endif

#include "resource.h"		// main symbols

#include "MKYDlg.h"

#include "ParamStore.h"
#include "ProjectSettings.h"

#include "CommThread.h"
#include "LogFile.h"
#include "DataManager.h"
#include "Security.h"

class CMKYApp : public CWinApp
{
public:
	CMKYApp();
	~CMKYApp();

	const CLogFile& LogFile() const;
	CLogFile& LogFile();

	const CParamStore& ParamStore() const;
	CParamStore& ParamStore();

	const CProjectSettings& ProjectSettings() const;
	CProjectSettings& ProjectSettings();

	const CSecurity& Security() const;
	CSecurity& Security();

	const CCommThread& CommThread() const;
	CCommThread& CommThread();

	const CDataManager& DataManager() const;
	CDataManager& DataManager();

	// Overrides
protected:
	virtual BOOL InitInstance();
	DECLARE_MESSAGE_MAP()

private:

	// Base objects
	//
	CParamStore m_ParamStore;
	CLogFile m_LogFile;
	CDataManager m_DataManager;
	CSecurity m_Security;
	CCommThread m_CommThread;
	CProjectSettings m_ProjectSettings;
};

extern CMKYApp theApp;
```

MKYDlg.cpp – головний діалог програми

```
// MKYDlg.cpp : implementation file
//
#include "pch.h"
#include "MKYDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

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

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

//-----------------------------------------------------------------------------
--------------------------
void CMKYStatusBarCtrl::DrawItem(LPDRAWITEMSTRUCT lpDS)
{

	CDC dc;
	dc.Attach(lpDS->hDC);

	//элемент 0 - состояние порта
	if (lpDS->itemID == 0)
	{
		if (m_CommThread.IsTimeout())
			str = m_Port + _T(" - таймаут!");
		else
			if (m_CommThread.IsCRCError())
				str = m_Port + _T(" - Помилка CRC!");
			else

				if (m_CommThread.IsBlocksCRCError())
					str = _T("Помилка КС БЛОК");
				else
					if (m_CommThread.IsSettingsCRCError())
						str = _T("Помилка КС УСТ");
					else
					{
						double TotalTraffic =
(m_CommThread.PacketReceived + m_CommThread.PacketSent) * (sizeof(OutPacket_t) +
sizeof(InPacket_t)) / 1024.0;

						if (TotalTraffic > 1024.0)
						{
							TotalTraffic /= 1024.0;
							str.Format(m_Port + _T(" %.3f Мб"),
TotalTraffic);

						}
						else
							str.Format(m_Port + _T(" %.2f Кб"),
TotalTraffic);
						//str = _T("COM1:");
					}

		if ((m_CommThread.IsTimeout() || m_CommThread.IsBlocksCRCError() ||
m_CommThread.IsCRCError() || m_CommThread.IsSettingsCRCError()) && Blink)
		{
			dc.SetBkColor(RGB(255, 0, 0));
			dc.SetTextColor(RGB(255, 255, 255));
		}
		else
			dc.SetBkColor(GetSysColor(COLOR_BTNFACE));
```

```
	}

	//элемент 1 - число блокувань
	if (lpDS->itemID == 1)
	{

		int BlockOnCount = m_DataManager.GetBlockOnCount();
		if (BlockOnCount > 0)
			str.Format(_T("Блк: %d из %d"), BlockOnCount,
m_DataManager.BlocksCount());
		else
			str.Empty();

		if (BlockOnCount > 0 && Blink)
		{
			dc.SetBkColor(RGB(255, 140, 0));
			dc.SetTextColor(RGB(255, 255, 255));
		}
		else
			dc.SetBkColor(GetSysColor(COLOR_BTNFACE));
	}

	//элемент 2 - стан ключа
	if (lpDS->itemID == 2)
	{

		if (m_CommThread.IsKey())
		{
			dc.SetBkColor(RGB(100, 255, 100));
			str = _T("Ключ: ТАК");
		}
		else
		{
			dc.SetBkColor(GetSysColor(COLOR_BTNFACE));
			str = _T("Ключ: НІ");
		}
	}



	lpDS->rcItem.left += 2;
	dc.ExtTextOut(lpDS->rcItem.left, lpDS->rcItem.top, ETO_OPAQUE, &lpDS-
>rcItem, str, NULL);
	dc.Detach();
}


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

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

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

CMKYDlg::CMKYDlg(CParamStore& ParamStore,
	CLogFile& LogFile,
	CDataManager& DataManager,
	CSecurity& Security,
	CCommThread& CommThread,
	CProjectSettings ProjectSettings,
	CWnd* pParent /*=NULL*/) : CDialog(CMKYDlg::IDD, pParent),
	m_ParamStore(ParamStore),
	m_LogFile(LogFile),
	m_DataManager(DataManager),
	m_Security(Security),
	m_CommThread(CommThread),
	m_ProjectSettings(ProjectSettings),
```

```
      DialogSettings(m_DataManager, m_CommThread, m_Security),
      DialogBlocks(m_DataManager, m_CommThread, m_Security),
      DialogTryout(m_DataManager, m_CommThread, m_Security, m_ProjectSettings),
      DialogOptions(m_CommThread, m_DataManager, m_ParamStore,
m_ProjectSettings, m_Security),
      m_StatusBar(m_CommThread, m_DataManager, m_ParamStore.Port())

{
      //{{AFX_DATA_INIT(CMKYDlg)
            // NOTE: the ClassWizard will add member initialization here
      //}}AFX_DATA_INIT
      // Note that LoadIcon does not require a subsequent DestroyIcon in Win32
      m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CMKYDlg::DoDataExchange(CDataExchange* pDX)
{
      CDialog::DoDataExchange(pDX);
      //{{AFX_DATA_MAP(CMKYDlg)
      DDX_Control(pDX, IDC_TAB, m_Tab);
      //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CMKYDlg, CDialog)
      //{{AFX_MSG_MAP(CMKYDlg)
      ON_WM_DESTROY()
      ON_WM_ACTIVATE()
      ON_NOTIFY(TCN_SELCHANGE, IDC_TAB, OnSelchangeTab)
      ON_WM_TIMER()
      ON_NOTIFY(TCN_SELCHANGING, IDC_TAB, OnSelchangingTab)
      //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CMKYDlg message handlers

//-----------------------------------------------------------------------------
-----------------------------
BOOL CMKYDlg::OnInitDialog()
{
      CDialog::OnInitDialog();

      SetIcon(m_hIcon, TRUE);            // Set big icon
      SetIcon(m_hIcon, FALSE);           // Set small icon

      //HideTaskbar();
      CRect r;
      SystemParametersInfo(SPI_GETWORKAREA, 0, &r, 0);

      //if (r.Width() > 320)
      r.right = 640 + r.left;

      //if (r.Height() > 240)
      r.bottom = 480 + r.top;

      SetWindowPos(NULL, 0, 0, r.Width(), r.Height(), SWP_NOZORDER);
      CenterWindow(GetDesktopWindow());

      int Ident = 5;

      const int nParts = 3;
      int nPartWidths[nParts] = { 150, 240, -1 };
      m_StatusBar.Create(WS_CHILD | WS_VISIBLE | CCS_BOTTOM, CRect(0, 0, 0, 0),
this, IDC_STATUSBAR);
      m_StatusBar.SetParts(nParts, nPartWidths);

      CRect stRect;
      m_StatusBar.GetWindowRect(&stRect);
```

```
	//таб
	GetClientRect(&r);
	m_Tab.SetWindowPos(NULL, Ident, Ident, r.Width() - Ident * 2, r.Height() -
Ident - stRect.Height(), SWP_NOZORDER);



	m_Tab.GetClientRect(&r);

	int TabIndex = 0;

	//уставки
	DialogSettings.Create(IDD_SETTINGS, &m_Tab);
	DialogSettings.SetWindowPos(NULL, Ident, Ident * 5, r.Width() - Ident * 2,
r.Height() - Ident * 6, SWP_NOZORDER);
	if (m_DataManager.SettingsCount() > 0)
	{
		m_Tab.InsertItem(TabIndex, _T("Уставки"));
		Index_Settings = TabIndex;
		TabIndex++;
	}

	//Блокування
	DialogBlocks.Create(IDD_BLOCKS, &m_Tab);
	DialogBlocks.SetWindowPos(NULL, Ident, Ident * 5, r.Width() - Ident * 2,
r.Height() - Ident * 6, SWP_NOZORDER);
	if (m_DataManager.BlocksCount() > 0)
	{
		m_Tab.InsertItem(TabIndex, _T("Блокування "));
		Index_Blocks = TabIndex;
		TabIndex++;
	}

	//Опробование
	DialogTryout.Create(IDD_TRYOUT, &m_Tab);
	DialogTryout.SetWindowPos(NULL, Ident, Ident * 5, r.Width() - Ident * 2,
r.Height() - Ident * 6, SWP_NOZORDER);
	m_Tab.InsertItem(TabIndex, _T("Тестування"));
	Index_Tryout = TabIndex;
	TabIndex++;


	//Параметры
	DialogOptions.Create(IDD_OPTIONS, &m_Tab);
	DialogOptions.SetWindowPos(NULL, Ident, Ident * 5, r.Width() - Ident * 2,
r.Height() - Ident * 6, SWP_NOZORDER);
	m_Tab.InsertItem(TabIndex, _T("Параметри"));
	Index_Options = TabIndex;
	TabIndex++;

	if (Index_Settings != -1)
	{
		DialogSettings.ShowWindow(SW_SHOWNORMAL);
	}
	else
	{
		if (Index_Blocks != -1)
		{
			DialogBlocks.ShowWindow(SW_SHOWNORMAL);
		}
		else
		{
			DialogTryout.ShowWindow(SW_SHOWNORMAL);
		}
	}

	//
	TimerID = SetTimer(1024, 200, NULL);
```

```
	return TRUE;  // return TRUE  unless you set the focus to a control
}


//-----------------------------------------------------------------------------
-----------------------------
void CMKYDlg::ShowTaskbar()
{
	CWnd* pBar = FindWindow(_T("HHTaskBar"), NULL);
	CWnd* pDesktop = FindWindow(_T("DesktopExplorerWindow"), NULL);
	if (!pDesktop || !pBar)
		return;

	pBar->ShowWindow(TRUE);
	SystemParametersInfo(SPI_SETWORKAREA, 0, &DesktopRect, 0);
	pDesktop->MoveWindow(&DesktopRect, TRUE);
}

//-----------------------------------------------------------------------------
-----------------------------
void CMKYDlg::HideTaskbar()
{
	CWnd* pBar = FindWindow(_T("HHTaskBar"), NULL);
	CWnd* pDesktop = FindWindow(_T("DesktopExplorerWindow"), NULL);

	if (!pDesktop || !pBar)
		return;

	CRect BarRect;
	pBar->GetWindowRect(&BarRect);
	pBar->ShowWindow(FALSE);

	SystemParametersInfo(SPI_GETWORKAREA, 0, &DesktopRect, 0);

	CRect NewR;
	NewR.left = 0;
	NewR.top = 0;
	NewR.right = DesktopRect.right;
	NewR.bottom = BarRect.bottom;
	SystemParametersInfo(SPI_SETWORKAREA, 0, &NewR, 0);

	pDesktop->MoveWindow(&NewR, TRUE);
}


//-----------------------------------------------------------------------------
-----------------------------
void CMKYDlg::OnDestroy()
{
	KillTimer(TimerID);
	//ShowTaskbar();
	CDialog::OnDestroy();
}

//-----------------------------------------------------------------------------
-----------------------------
void CMKYDlg::OnActivate(UINT nState, CWnd* pWndOther, BOOL bMinimized)
{
	CDialog::OnActivate(nState, pWndOther, bMinimized);
	if (nState != WA_INACTIVE || !pWndOther)
		return;

	CString str;
	pWndOther->GetWindowText(str);

	CWnd* pWnd = pWndOther->GetWindow(GW_CHILD);
	while (pWnd)
	{
```

```
            pWnd->GetWindowText(str);

            if (str == _T("&Yes"))
                  pWnd->SetWindowText(_T("Да"));
            if (str == _T("&No"))
                  pWnd->SetWindowText(_T("Нет"));
            if (str == _T("Cancel"))
                  pWnd->SetWindowText(_T("Скасувати"));

            pWnd = pWnd->GetWindow(GW_HWNDNEXT);
      }

}

//-------------------------------------------------------------------------------
-----------------------------
void CMKYDlg::OnSelchangeTab(NMHDR* pNMHDR, LRESULT* pResult)
{
      int nItem = m_Tab.GetCurSel();

      DialogSettings.ShowWindow(nItem == Index_Settings ? SW_SHOWNORMAL :
SW_HIDE);
      DialogBlocks.ShowWindow(nItem == Index_Blocks ? SW_SHOWNORMAL : SW_HIDE);
      DialogTryout.ShowWindow(nItem == Index_Tryout ? SW_SHOWNORMAL : SW_HIDE);
      DialogOptions.ShowWindow(nItem == Index_Options ? SW_SHOWNORMAL :
SW_HIDE);

      if (nItem == Index_Blocks)
            m_CommThread.InitiateBlocksRead();

      if (nItem == Index_Settings)
            m_CommThread.InitiateSettingsRead();

      if (nItem == Index_Tryout)
            m_CommThread.InitiateTryoutMode();

      *pResult = 0;
}

//-------------------------------------------------------------------------------
-----------------------------
void CMKYDlg::OnTimer(UINT_PTR nIDEvent)
{
      CDialog::OnTimer(nIDEvent);

      //статусбар

      static int  BlinkCounter = 0;
      if (BlinkCounter++ == 2)
      {
            BlinkCounter = 0;
            m_StatusBar.Blink = !m_StatusBar.Blink;

            //m_StatusBar.Invalidate();

            //AfxMessageBox(_T("X"));

            m_StatusBar.SetText(0, 0, SBT_OWNERDRAW);
            m_StatusBar.SetText(0, 1, SBT_OWNERDRAW);
            m_StatusBar.SetText(0, 2, SBT_OWNERDRAW);
      }


      //таймеры диалогов
      if (DialogBlocks.IsWindowVisible())
            DialogBlocks.On_Timer();

      if (DialogSettings.IsWindowVisible())
            DialogSettings.On_Timer();
```

```
	if (DialogTryout.IsWindowVisible())
		DialogTryout.On_Timer();

	//показ диалога прогресса

	if (!m_CommThread.IsIdle() &&
		!CDialogProgress::pDialogProgress &&
		!m_CommThread.IsTimeout() &&
		m_CommThread.IsConnected())
	{
		CDialogProgress dlg(m_DataManager, m_CommThread);
		CDialogProgress::pDialogProgress = &dlg;
		dlg.DoModal();
	}

	if (CDialogProgress::pDialogProgress)
		CDialogProgress::pDialogProgress->On_Timer();

	int CurrentMode = 0;
	if (m_CommThread.GetInitializeMode(CurrentMode))
	{
		int nItem = 0;
		switch (CurrentMode)
		{
		case STATE_COMMAND_SETTING:
			nItem = Index_Settings;
			break;
		case STATE_COMMAND_BLOCK:
			nItem = Index_Blocks;
			break;
		case STATE_COMMAND_TEST:
			nItem = Index_Tryout;
			break;
		default:
			ASSERT(0);
			nItem = Index_Options;
		}

		m_Tab.SetCurSel(nItem);
		DialogSettings.ShowWindow(nItem == Index_Settings ? SW_SHOWNORMAL :
SW_HIDE);
		DialogBlocks.ShowWindow(nItem == Index_Blocks ? SW_SHOWNORMAL :
SW_HIDE);
		DialogTryout.ShowWindow(nItem == Index_Tryout ? SW_SHOWNORMAL :
SW_HIDE);
		DialogOptions.ShowWindow(nItem == Index_Options ? SW_SHOWNORMAL :
SW_HIDE);
	}

}

//-------------------------------------------------------------------------------
-----------------------------
void CMKYDlg::OnSelchangingTab(NMHDR* pNMHDR, LRESULT* pResult)
{
	if (m_CommThread.IsRunningTryout())
	{
		AfxMessageBox(_T("Для зміни режиму завершіть тестування!"), MB_OK |
MB_ICONERROR);
		*pResult = 1;
		return;
	}

	if (m_CommThread.IsKey())
	{
		AfxMessageBox(_T("Для зміни режиму вийміть ключ доступу!"), MB_OK |
MB_ICONERROR);
		*pResult = 1;
```

```
            return;
      }

      *pResult = 0;
}
```

MKYDlg.h – головний діалог програми

```
#pragma once
#include "resource.h"

#include "DialogSettings.h"
#include "DialogBlocks.h"
#include "DialogTryout.h"
#include "DialogOptions.h"
#include "DialogProgress.h"

/////////////////////////////////////////////////////////////////////////////
// CMKYDlg dialog

class CMKYStatusBarCtrl :public CStatusBarCtrl
{
      CString str;
      virtual void DrawItem(LPDRAWITEMSTRUCT lpDrawItemStruct);

      CCommThread& m_CommThread;
      CDataManager& m_DataManager;
      CString m_Port;


public:

      BOOL Blink;
      CMKYStatusBarCtrl(CCommThread& CommThread, CDataManager& DataManager,
CString Port):
            m_CommThread(CommThread),
            m_DataManager(DataManager),
            m_Port(Port)
      {
            Blink = FALSE;
      }


};

class CMKYDlg : public CDialog
{
      // Construction
public:
      CMKYDlg(CParamStore& ParamStore,
            CLogFile& LogFile,
            CDataManager& DataManager,
            CSecurity& Security,
            CCommThread& CommThread,
            CProjectSettings ProjectSettings,
            CWnd* pParent = NULL);  // standard constructor

      // Dialog Data
            //{{AFX_DATA(CMKYDlg)
      enum { IDD = IDD_MKY2023_DIALOG };
      // NOTE: the ClassWizard will add data members here
//}}AFX_DATA

// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CMKYDlg)
protected:
      virtual void DoDataExchange(CDataExchange* pDX);     // DDX/DDV support
      //}}AFX_VIRTUAL
```

```
// Implementation
protected:
	HICON m_hIcon;

	UINT_PTR TimerID;
	CTabCtrl m_Tab;

	// Generated message map functions
	//{{AFX_MSG(CMKYDlg)
	virtual BOOL OnInitDialog();
	virtual void OnOK() {}
	virtual void OnCancel() {}
	afx_msg void OnDestroy();
	afx_msg void OnActivate(UINT nState, CWnd* pWndOther, BOOL bMinimized);
	afx_msg void OnSelchangeTab(NMHDR* pNMHDR, LRESULT* pResult);
	afx_msg void OnTimer(UINT_PTR nIDEvent);
	afx_msg void OnSelchangingTab(NMHDR* pNMHDR, LRESULT* pResult);
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()

private:
	void ShowTaskbar();
	void HideTaskbar();

private:
	CRect DesktopRect;

	int Index_Settings{-1};
	int Index_Blocks{-1};
	int Index_Tryout{-1};
	int Index_Options{-1};

private:
	CParamStore& m_ParamStore;

	CLogFile& m_LogFile;

	CDataManager& m_DataManager;

	CSecurity& m_Security;

	CCommThread& m_CommThread;

	CProjectSettings m_ProjectSettings;

	CDialogSettings DialogSettings;
	CDialogBlocks DialogBlocks;
	CDialogTryout DialogTryout;
	CDialogOptions DialogOptions;
	CMKYStatusBarCtrl m_StatusBar;
};

//{{AFX_INSERT_LOCATION}}
// Microsoft eMbedded Visual C++ will insert additional declarations immediately
before the previous line.
```

Block.cpp – опис класу блокування

```
#include "pch.h"
#include "Block.h"

CBlock::CBlock() :
	Blocked(FALSE),
	Valid(FALSE),
	Updated(FALSE)
{
}
```

Block.h – опис класу блокування

```
#pragma once

class CBlock
{
public:
	CBlock();

public:
	CString StrID;
	CString Description;

	BOOL Blocked{FALSE};
	BOOL Valid{FALSE};
	BOOL Updated{FALSE};
};
```

Setting.cpp – опис класу уставки

```
#include "pch.h"
#include "Setting.h"

CSetting::CSetting() :
	MinValue(0),
	MaxValue(10),
	Multiplier(1),
	ChangingStep(0),
	Value(0),
	Valid(FALSE),
	Updated(FALSE)
{
}


int CSetting::Round(double Value)
{
	return (int)(Value + ((Value < 0.0) ? -0.5 : 0.5));
}

void CSetting::SetValue(short ADCValue)
{
	Value = ADCValue * Multiplier;
}

short CSetting::GetValue()
{
	return Round(Value / Multiplier);
}

CString CSetting::GetStringValue()
{
	return GetStringValue(Value);
}

CString CSetting::GetStringValue(double Val)
{
```

```
	CString str;

	if (Valid)
	{
		if (Multiplier == 1)
		{
			str.Format(_T("%.0f"), Val);
		}
		else
		{
			if (Multiplier == 0.1)
			{
				str.Format(_T("%.1f"), Val);
			}
			else
			{
				str.Format(_T("%.2f"), Val);
			}
		}
	}
	else
	{
		str = _T("???");
	}
	return str;
}
```

Setting.h – опис класу уставки

```
#pragma once

class CSetting
{
public:
	CSetting();

public:
	CString StrID;
	CString Description;

	double MinValue;
	double MaxValue;

	double Multiplier;	//Коеффіцієнт уставки
	double ChangingStep;//Крок уставки


	double Value;
	BOOL Valid;
	BOOL Updated;

public:
	static int Round(double Value);

	void SetValue(short ADCValue);
	short GetValue();

	CString GetStringValue();
	CString GetStringValue(double Val);
};
```

DataManager.cpp – клас зберігання даних

```
#include "pch.h"
#include "DataManager.h"
#include "AutoCS.h"

void CDataManager::SetSettingArray(std::vector <CSetting> SettingArray)
{
```

```
	auto_cs acs(cs);
	m_SettingArray = SettingArray;
}

void CDataManager::SetBlockArray(std::vector<CBlock> BlockArray,
std::vector<int> BlockIndexArray)
{
	auto_cs acs(cs);
	m_BlockArray = BlockArray;
	m_BlockIndexArray = BlockIndexArray;
}

int CDataManager::GetBlockOnCount()
{
	auto_cs acs(cs);
	int BlockedCount = 0;
	size_t nCount = m_BlockArray.size();
	for (int i = 0; i < nCount; i++)
	{
		if (m_BlockArray[i].Blocked && m_BlockArray[i].Valid)
		{
			BlockedCount++;
		}
	}

	return BlockedCount;
}

BOOL CDataManager::IsFirstSettingValid()
{
	auto_cs acs(cs);
	return m_SettingArray.size() > 0 && m_SettingArray[0].Valid;
}

void CDataManager::InvalidateData()
{
	auto_cs acs(cs);
	size_t nCount = m_BlockArray.size();
	for (int i = 0; i < nCount; i++)
	{
		m_BlockArray[i].Valid = FALSE;
		m_BlockArray[i].Updated = TRUE;
	}

	nCount = m_SettingArray.size();
	for (size_t i = 0; i < nCount; i++)
	{
		m_SettingArray[i].Valid = FALSE;
		m_SettingArray[i].Updated = TRUE;
	}
}

int CDataManager::SettingsCount() const
{
	auto_cs acs(cs);
	return (int)m_SettingArray.size();
}

int CDataManager::BlocksCount() const
{
	auto_cs acs(cs);
	return (int)m_BlockArray.size();
}

CSetting CDataManager::Setting(int index) const
{
	auto_cs acs(cs);
	return m_SettingArray[index];
}
```

```
void CDataManager::SetSetting(int index, const CSetting& Setting)
{
	auto_cs acs(cs);
	m_SettingArray[index] = Setting;
}

CBlock CDataManager::Block(int index) const
{
	auto_cs acs(cs);
	return m_BlockArray[index];
}

void CDataManager::SetBlock(int index, const CBlock& Block)
{
	auto_cs acs(cs);
	m_BlockArray[index] = Block;
}

int CDataManager::BlockIndex(int index) const
{
	return m_BlockIndexArray[index];
}
```

DataManager.h – клас зберігання даних

```
#pragma once

#include "Block.h"
#include "Setting.h"

class CDataManager
{
public:
	void SetSettingArray(std::vector <CSetting> SettingArray);
	void SetBlockArray(std::vector<CBlock> BlockArray, std::vector<int>
BlockIndexArray);

	int GetBlockOnCount();
	BOOL IsFirstSettingValid();
	void InvalidateData();

	int SettingsCount() const;
	int BlocksCount() const;

	CSetting Setting(int index) const;
	void SetSetting(int index, const CSetting& Setting);

	CBlock Block(int index) const;
	void SetBlock(int index, const CBlock& Block);

	int BlockIndex(int index) const;

private:
	mutable CCriticalSection cs;

	// Масив уставок
	//
	std::vector <CSetting> m_SettingArray;

	// Масив блокувань
	//
	std::vector<CBlock> m_BlockArray;
	std::vector<int> m_BlockIndexArray;
};
```

Security.cpp – код діалога перевіркии пароля

```
#include "pch.h"
```

```
#include "Security.h"
#include "DialogPassword.h"

//---------------------------------------------------------------
CSecurity::CSecurity(CParamStore& ParamStore) :
	m_ParamStore(ParamStore)
{
	m_PasswordValid.store(FALSE);
}

//---------------------------------------------------------------
BOOL CSecurity::AskForPassword()
{
	//пароль введено
	if (m_PasswordValid.load())
	{
		return TRUE;
	}

	//пароль порожній
	if (m_ParamStore.Password().IsEmpty() == TRUE)
	{
		m_PasswordValid.store(TRUE);
		return TRUE;
	}

	//непустой - надо спросить
	CDialogPassword dlg;
	dlg.SetCaption(TEXT("Введите пароль:"));
	if (dlg.DoModal() != IDOK)
	{
		return FALSE;
	}

	CString sUserPassword = dlg.GetPassword();
	if (sUserPassword == m_ParamStore.ServicePassword())
	{
		m_PasswordValid.store(TRUE);
		return TRUE;
	}

	if (m_ParamStore.Password() != sUserPassword)
	{
		AfxMessageBox(TEXT("Невірний пароль!"));
		m_PasswordValid.store(FALSE);
		return FALSE;
	}

	m_PasswordValid.store(TRUE);
	return TRUE;
}

void CSecurity::ResetPassword()
{
	m_PasswordValid.store(FALSE);
}
```

Security.h – код діалога перевіркии пароля

```
#pragma once
#include "ParamStore.h"

class CSecurity
{
public:
	CSecurity(CParamStore& ParamStore);

	BOOL AskForPassword();
	void ResetPassword();
```

```
private:
	CParamStore& m_ParamStore;
	std::atomic<BOOL> m_PasswordValid;
};
```

DialogSettings.cpp – код діалога керування уставками

```
// DialogSettings.cpp : implementation file
//
#include "pch.h"
#include "DialogSettings.h"
#include "DialogSettingDetails.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CDialogSettings dialog


CDialogSettings::CDialogSettings(CDataManager& DataManager, CCommThread&
CommThread, CSecurity& Security, CWnd* pParent /*=NULL*/)
	: CDialog(CDialogSettings::IDD, pParent),
	m_DataManager(DataManager),
	m_CommThread(CommThread),
	m_Security(Security),
	m_List(m_DataManager)

{
	//{{AFX_DATA_INIT(CDialogSettings)
		// NOTE: the ClassWizard will add member initialization here
	//}}AFX_DATA_INIT
	EraseValueOnEdit = TRUE;
}


void CDialogSettings::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CDialogSettings)
		// NOTE: the ClassWizard will add DDX and DDV calls here
	DDX_Control (pDX, IDC_LIST, m_List);
	DDX_Control (pDX, IDC_APPLY, m_Apply);
	DDX_Control (pDX, IDC_VALUE, m_Value);
	//}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CDialogSettings, CDialog)
	//{{AFX_MSG_MAP(CDialogSettings)
	ON_NOTIFY(LVN_GETDISPINFO, IDC_LIST, OnGetdispinfoList)
	ON_NOTIFY(LVN_ITEMCHANGED, IDC_LIST, OnItemchangedList)
	ON_BN_CLICKED(IDC_KC, OnKc)
	ON_BN_CLICKED(IDC_KP, OnKp)
	ON_BN_CLICKED(IDC_K1, OnK1)
	ON_BN_CLICKED(IDC_K2, OnK2)
	ON_BN_CLICKED(IDC_K3, OnK3)
	ON_BN_CLICKED(IDC_K4, OnK4)
	ON_BN_CLICKED(IDC_K5, OnK5)
	ON_BN_CLICKED(IDC_K6, OnK6)
	ON_BN_CLICKED(IDC_K7, OnK7)
	ON_BN_CLICKED(IDC_K8, OnK8)
	ON_BN_CLICKED(IDC_K9, OnK9)
	ON_BN_CLICKED(IDC_K0, OnK0)
	ON_BN_CLICKED(IDC_APPLY, OnApply)
```

```
	ON_NOTIFY(NM_DBLCLK, IDC_LIST, OnDblclkList)
	ON_BN_CLICKED(IDC_KM, OnKm)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CDialogSettings message handlers


//-----------------------------------------------------------------------------
BOOL CDialogSettings::OnInitDialog()
{
	CDialog::OnInitDialog();

	m_List.InsertColumn ( 0, _T("ИД"), LVCFMT_LEFT, 100 );
	m_List.InsertColumn ( 1, _T("Значение"), LVCFMT_LEFT, 180 );
	m_List.InsertColumn ( 2, _T("Описание"), LVCFMT_LEFT, 300 );

	m_List.CountColumns();
	m_List.Type = TYPE_SETTING;

	int nCount = (int)m_DataManager.SettingsCount();

	m_List.SetItemCountEx (nCount);

	m_List.SetExtendedStyle ( LVS_EX_FULLROWSELECT );

	return TRUE;  // return TRUE unless you set the focus to a control
	              // EXCEPTION: OCX Property Pages should return FALSE
}

//-----------------------------------------------------------------------------
void CDialogSettings::OnGetdispinfoList(NMHDR* pNMHDR, LRESULT* pResult)
{

	*pResult = 0;
	LV_DISPINFO* pDispInfo = (LV_DISPINFO*)pNMHDR;
	LV_ITEM item = pDispInfo->item;

	if ((item.mask & LVIF_TEXT) == 0)
		return;

	CString str;

	CSetting Setting = m_DataManager.Setting(item.iItem);

	switch (item.iSubItem)
	{
	case 0:
		_tcscpy(item.pszText, Setting.StrID);
		break;
	case 1:
		str = Setting.GetStringValue();
		_tcscpy(item.pszText, str);
		break;
	case 2:
		_tcscpy(item.pszText, Setting.Description);
		break;
	}
}




//-----------------------------------------------------------------------------
void CDialogSettings::On_Timer()
{
	int SelectedIndex = -1;
	if (m_List.GetSelectedCount() != 0)
```

```
	{
		POSITION pos = m_List.GetFirstSelectedItemPosition();
		SelectedIndex = m_List.GetNextSelectedItem(pos);
	}
	CString str;

	int nCount = (int)m_DataManager.SettingsCount();

	for (int i = 0; i < nCount; i++)
	{
		CSetting Setting = m_DataManager.Setting(i);

		if (Setting.Updated)
		{
			Setting.Updated = FALSE;
			m_List.RedrawItems (i,i);

			if (SelectedIndex == i)
			{
				str = Setting.GetStringValue();
				m_Value.SetWindowText(str);
				EraseValueOnEdit = TRUE;
			}
		}
	}

	m_Value.EnableWindow (m_CommThread.IsKey() && (m_List.GetSelectedCount()
!= 0 && m_DataManager.Setting(SelectedIndex).Valid));
}

//-----------------------------------------------------------------------------
void CDialogSettings::OnItemchangedList(NMHDR* pNMHDR, LRESULT* pResult)
{
	NM_LISTVIEW* pNMListView = (NM_LISTVIEW*)pNMHDR;

	int nItem = pNMListView->iItem;
	if (pNMListView->uNewState & LVIS_SELECTED)
	{
		CSetting Setting = m_DataManager.Setting(nItem);

		CString str;
		str = Setting.GetStringValue();
		m_Value.SetWindowText(str);
		EraseValueOnEdit = TRUE;

		m_Value.EnableWindow (Setting.Valid && m_CommThread.IsKey());
	}

	*pResult = 0;
}

//-----------------------------------------------------------------------------
void CDialogSettings::OnKc()
{
	if (!m_Value.IsWindowEnabled())
		return;

	m_Value.SetWindowText(_T(""));
	m_Apply.EnableWindow(FALSE);

}

//-----------------------------------------------------------------------------
void CDialogSettings::OnKp()
{
	if (!m_Value.IsWindowEnabled())
		return;

	CString str;
```

```
	m_Value.GetWindowText(str);
	if (str.IsEmpty())
		return;

	if (str.Find('.') != -1)
		return;

	str = str + '.';
	m_Value.SetWindowText(str);
	m_Apply.EnableWindow(FALSE);
}

//--------------------------------------------------------------------------
void CDialogSettings::AddSymbol(char c)
{
	if (!m_Value.IsWindowEnabled())
		return;

	if (EraseValueOnEdit)
	{
		EraseValueOnEdit = FALSE;
		m_Value.SetWindowText(_T(""));
	}

	CString str;
	m_Value.GetWindowText(str);

	if (c == '-' && str.Find('-') != -1)
		return;

	str = str + c;
	m_Value.SetWindowText(str);

	if (str != '-')
		m_Apply.EnableWindow(TRUE);
}

//--------------------------------------------------------------------------
void CDialogSettings::OnK1()
{
	AddSymbol('1');
}

//--------------------------------------------------------------------------
void CDialogSettings::OnK2()
{
	AddSymbol('2');
}

//--------------------------------------------------------------------------
void CDialogSettings::OnK3()
{
	AddSymbol('3');
}

//--------------------------------------------------------------------------
void CDialogSettings::OnK4()
{
	AddSymbol('4');
}

//--------------------------------------------------------------------------
void CDialogSettings::OnK5()
{
	AddSymbol('5');
}

//--------------------------------------------------------------------------
void CDialogSettings::OnK6()
```

```
{
	AddSymbol('6');
}

//-----------------------------------------------------------------------------
void CDialogSettings::OnK7()
{
	AddSymbol('7');
}

//-----------------------------------------------------------------------------
void CDialogSettings::OnK8()
{
	AddSymbol('8');
}

//-----------------------------------------------------------------------------
void CDialogSettings::OnK9()
{
	AddSymbol('9');
}

//-----------------------------------------------------------------------------
void CDialogSettings::OnK0()
{
	AddSymbol('0');
}

//-----------------------------------------------------------------------------
void CDialogSettings::OnKm()
{
	AddSymbol('-');
}

//-----------------------------------------------------------------------------
void CDialogSettings::OnApply()
{
	if (!m_Security.AskForPassword())
		return;

	if (m_List.GetSelectedCount() == 0)
		return;

	POSITION pos = m_List.GetFirstSelectedItemPosition();
	int SelectedIndex = m_List.GetNextSelectedItem(pos);

	//создаем КОПИЮ уставки
	CSetting Setting = m_DataManager.Setting(SelectedIndex);

	//Вводим данные пользователя
	CString str;
	m_Value.GetWindowText(str);
	char c[128];

	WideCharToMultiByte(CP_ACP, 0, str.GetBuffer(128), -1, c, 128, 0, 0);
	double NewValue = atof(c);

	//корректируем диапазон
	if (NewValue < Setting.MinValue)
	{
		AfxMessageBox (_T("Введене значення менше допустимого! Проведена
корекція значення."), MB_OK|MB_ICONINFORMATION);
		NewValue = Setting.MinValue;
	}

	if (NewValue > Setting.MaxValue)
	{
		AfxMessageBox (_T("Введенное значение больше допустимого! Проведена
корекція значення."), MB_OK|MB_ICONINFORMATION);
```

```
            NewValue = Setting.MaxValue;
      }

      int ChangeStepCount = CSetting::Round((NewValue - Setting.Value) /
Setting.ChangingStep);
      if (ChangeStepCount == 0)
      {
            AfxMessageBox (_T("Введене значення уставки вже встановлено!"),
MB_OK|MB_ICONINFORMATION);
            return;
      }

      str.Format(_T("Ви дійсно хочете встановити значення %s уставки %s (%s)?"),
Setting.GetStringValue(NewValue), Setting.StrID, Setting.Description);
      if (AfxMessageBox (str, MB_YESNO|MB_ICONQUESTION) != IDYES)
            return;

      m_Apply.EnableWindow(FALSE);
      EraseValueOnEdit = TRUE;

      m_CommThread.InitiateSettingChange(SelectedIndex, NewValue,
ChangeStepCount);

}

//-----------------------------------------------------------------------------
void CDialogSettings::OnDblclkList(NMHDR* pNMHDR, LRESULT* pResult)
{
      POSITION pos = m_List.GetFirstSelectedItemPosition();
      if (!pos)
            return;
      int SelectedIndex = m_List.GetNextSelectedItem(pos);

      CDialogSettingDetails dlg(m_DataManager.Setting(SelectedIndex));
      dlg.DoModal();

      *pResult = 0;
}
```

DialogSettings.h – код діалогу керування уставками

```
#pragma once
#include "resource.h"
#include "DataManager.h"
#include "CommThread.h"
#include "Security.h"
#include "MKYListCtrl.h"

/////////////////////////////////////////////////////////////////////////////
// CDialogSettings dialog

class CDialogSettings : public CDialog
{
      // Construction
public:
      CDialogSettings(CDataManager& DataManager, CCommThread& CommThread,
CSecurity& Security, CWnd* pParent = NULL);   // standard constructor

      // Dialog Data
            //{{AFX_DATA(CDialogSettings)
      enum { IDD = IDD_SETTINGS };
      // NOTE: the ClassWizard will add data members here
//}}AFX_DATA


// Overrides
      // ClassWizard generated virtual function overrides
      //{{AFX_VIRTUAL(CDialogSettings)
```

```
protected:
      virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
      //}}AFX_VIRTUAL

// Implementation
protected:
      // Generated message map functions
      //{{AFX_MSG(CDialogSettings)
      virtual BOOL OnInitDialog();
      virtual void OnOK() {}
      virtual void OnCancel() {}
      afx_msg void OnGetdispinfoList(NMHDR* pNMHDR, LRESULT* pResult);
      afx_msg void OnItemchangedList(NMHDR* pNMHDR, LRESULT* pResult);
      afx_msg void OnKc();
      afx_msg void OnKp();
      afx_msg void OnK1();
      afx_msg void OnK2();
      afx_msg void OnK3();
      afx_msg void OnK4();
      afx_msg void OnK5();
      afx_msg void OnK6();
      afx_msg void OnK7();
      afx_msg void OnK8();
      afx_msg void OnK9();
      afx_msg void OnK0();
      afx_msg void OnApply();
      afx_msg void OnDblclkList(NMHDR* pNMHDR, LRESULT* pResult);
      afx_msg void OnKm();
      //}}AFX_MSG
      DECLARE_MESSAGE_MAP()

private:
      void AddSymbol(char c);

public:
      void On_Timer();

private:
      CMKYListCtrl m_List;
      CButton m_Apply;
      CEdit m_Value;
      BOOL EraseValueOnEdit{FALSE};

private:
      CDataManager& m_DataManager;
      CCommThread& m_CommThread;
      CSecurity& m_Security;
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before
the previous line.
```

DialogBlocks.cpp – код діалогу керування блокуваннями

```
// DialogBlocks.cpp : implementation file
//
#include "pch.h"
#include "DialogBlocks.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CDialogBlocks dialog
```

```
CDialogBlocks::CDialogBlocks(CDataManager& DataManager, CCommThread& CommThread,
CSecurity& Security, CWnd* pParent /*=NULL*/)
	: CDialog(CDialogBlocks::IDD, pParent),
	m_DataManager(DataManager),
	m_CommThread(CommThread),
	m_Security(Security),
	m_List(m_DataManager)
{
	//{{AFX_DATA_INIT(CDialogBlocks)
		// NOTE: the ClassWizard will add member initialization here
	//}}AFX_DATA_INIT
}


void CDialogBlocks::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CDialogBlocks)
		// NOTE: the ClassWizard will add DDX and DDV calls here
	DDX_Control (pDX, IDC_LIST, m_List);
	DDX_Control (pDX, IDC_BLOCK, m_Block);
	//}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CDialogBlocks, CDialog)
	//{{AFX_MSG_MAP(CDialogBlocks)
	ON_NOTIFY(LVN_GETDISPINFO, IDC_LIST, OnGetdispinfoList)
	ON_BN_CLICKED(IDC_BLOCK, OnBlock)
	ON_NOTIFY(LVN_ITEMCHANGED, IDC_LIST, OnItemchangedList)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CDialogBlocks message handlers

//-----------------------------------------------------------------------------
------------
BOOL CDialogBlocks::OnInitDialog()
{
	CDialog::OnInitDialog();

	m_List.InsertColumn ( 0, _T("ИД"), LVCFMT_LEFT, 100 );
	m_List.InsertColumn ( 1, _T("Блк"), LVCFMT_LEFT, 100 );
	m_List.InsertColumn ( 2, _T("Описание"), LVCFMT_LEFT, 380 );

	m_List.CountColumns();
	m_List.Type = TYPE_BLOCK;

	m_List.SetExtendedStyle ( LVS_EX_FULLROWSELECT/*|LVS_EX_CHECKBOXES*/);

	int nCount = (int)m_DataManager.BlocksCount();
	m_List.SetItemCountEx (nCount);

	return TRUE;  // return TRUE unless you set the focus to a control
		      // EXCEPTION: OCX Property Pages should return FALSE
}

//-----------------------------------------------------------------------------
------------
void CDialogBlocks::OnGetdispinfoList(NMHDR* pNMHDR, LRESULT* pResult)
{
	*pResult = 0;
	LV_DISPINFO* pDispInfo = (LV_DISPINFO*)pNMHDR;
	LV_ITEM item = pDispInfo->item;

	if ((item.mask & LVIF_TEXT) == 0)
		return;
```

```
	CBlock Block = m_DataManager.Block(m_DataManager.BlockIndex(item.iItem));

	switch (item.iSubItem)
	{
	case 0:
		_tcscpy (item.pszText, Block.StrID );
		break;
	case 1:
		if (Block.Valid)
			_tcscpy (item.pszText, Block.Blocked ? _T("ТАК") : _T("НІ") );
		else
			_tcscpy (item.pszText, _T("???") );
		break;
	case 2:
		_tcscpy (item.pszText, Block.Description );
		break;
	}
}

//---------------------------------------------------------------------------
void CDialogBlocks::On_Timer()
{
	int SelectedIndex = -1;
	if (m_List.GetSelectedCount() != 0)
	{
		POSITION pos = m_List.GetFirstSelectedItemPosition();
		SelectedIndex = m_List.GetNextSelectedItem(pos);
	}

	int nCount = m_DataManager.BlocksCount();
	for (int i = 0; i < nCount; i++)
	{
		CBlock Block = m_DataManager.Block(m_DataManager.BlockIndex(i));
		if (Block.Updated)
		{
			Block.Updated = FALSE;
			m_List.RedrawItems (i,i);

			if (SelectedIndex == i)
				m_Block.SetCheck ((Block.Valid && Block.Blocked) ?
BST_CHECKED : BST_UNCHECKED);
		}
	}

	//Разрешить/запретить чекбокс
	m_Block.EnableWindow (m_CommThread.IsKey() &&
		(m_List.GetSelectedCount() != 0 &&

	m_DataManager.Block(m_DataManager.BlockIndex(SelectedIndex)).Valid));
}

//---------------------------------------------------------------------------
void CDialogBlocks::OnItemchangedList(NMHDR* pNMHDR, LRESULT* pResult)
{
	NM_LISTVIEW* pNMListView = (NM_LISTVIEW*)pNMHDR;


	int nItem = pNMListView->iItem;
	if (pNMListView->uNewState & LVIS_SELECTED)
	{

		CBlock Block = m_DataManager.Block(m_DataManager.BlockIndex(nItem));

		m_Block.EnableWindow (Block.Valid && m_CommThread.IsKey());
		m_Block.SetCheck ((Block.Valid && Block.Blocked) ? BST_CHECKED :
BST_UNCHECKED);
	}
```

```
	*pResult = 0;
}

//-----------------------------------------------------------------------------
void CDialogBlocks::OnBlock()
{
	if (!m_Security.AskForPassword())
	{
		return;
	}

	if (m_List.GetSelectedCount() == 0)
		return;

	POSITION pos = m_List.GetFirstSelectedItemPosition();
	int SelectedIndex = m_List.GetNextSelectedItem(pos);


	int BlockIndex = m_DataManager.BlockIndex(SelectedIndex);
	CBlock Block = m_DataManager.Block(BlockIndex);

	CString str;
	str.Format (_T("Ви дійсно бажаєте %s блокування %s (%s)?"), Block.Blocked
? _T("вимкнути"):_T("увімкнути"), Block.StrID, Block.Description);
	if (AfxMessageBox (str, MB_YESNO|MB_ICONQUESTION) != IDYES)
		return;

	m_CommThread.InitiateBlockToggle(BlockIndex);
}
```

DialogBlocks.h – код діалогу керування блокуваннями

```
#pragma once
#include "resource.h"
#include "DataManager.h"
#include "CommThread.h"
#include "Security.h"
#include "MKYListCtrl.h"

/////////////////////////////////////////////////////////////////////////////
// CDialogBlocks dialog

class CDialogBlocks : public CDialog
{
// Construction
public:
	CDialogBlocks(CDataManager& DataManager, CCommThread& CommThread,
CSecurity& Security, CWnd* pParent = NULL);   // standard constructor

// Dialog Data
	//{{AFX_DATA(CDialogBlocks)
	enum { IDD = IDD_BLOCKS };
		// NOTE: the ClassWizard will add data members here
	//}}AFX_DATA

// Overrides
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CDialogBlocks)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
	//}}AFX_VIRTUAL

// Implementation
protected:
	// Generated message map functions
	//{{AFX_MSG(CDialogBlocks)
	virtual BOOL OnInitDialog();
	virtual void OnOK(){}
```

```
	virtual void OnCancel(){}
	afx_msg void OnGetdispinfoList(NMHDR* pNMHDR, LRESULT* pResult);
	afx_msg void OnBlock();
	afx_msg void OnItemchangedList(NMHDR* pNMHDR, LRESULT* pResult);
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()

public:
	void On_Timer();

private:
	CButton m_Block;

private:
	CDataManager& m_DataManager;
	CCommThread& m_CommThread;
	CSecurity& m_Security;
	CMKYListCtrl m_List;
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before
the previous line.
```

ComThread.cpp – реалізація комунікаційного потоку

```
// CommThread.cpp: implementation of the CCommThread class.
//
//////////////////////////////////////////////////////////////////////

#include "pch.h"
#include "CommThread.h"
#include "AutoCS.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif



unsigned char TableCRC8[256] = {
    0,  94, 188, 226,  97,  63, 221, 131, 194, 156,
  126,  32, 163, 253,  31,  65, 157, 195,  33, 127,
  252, 162,  64,  30,  95,   1, 227, 189,  62,  96,
  130, 220,  35, 125, 159, 193,  66,  28, 254, 160,
  225, 191,  93,   3, 128, 222,  60,  98, 190, 224,
    2,  92, 223, 129,  99,  61, 124,  34, 192, 158,
   29,  67, 161, 255,  70,  24, 250, 164,  39, 121,
  155, 197, 132, 218,  56, 102, 229, 187,  89,   7,
  219, 133, 103,  57, 186, 228,   6,  88,  25,  71,
  165, 251, 120,  38, 196, 154, 101,  59, 217, 135,
    4,  90, 184, 230, 167, 249,  27,  69, 198, 152,
  122,  36, 248, 166,  68,  26, 153, 199,  37, 123,
   58, 100, 134, 216,  91,   5, 231, 185, 140, 210,
   48, 110, 237, 179,  81,  15,  78,  16, 242, 172,
   47, 113, 147, 205,  17,  79, 173, 243, 112,  46,
  204, 146, 211, 141, 111,  49, 178, 236,  14,  80,
  175, 241,  19,  77, 206, 144, 114,  44, 109,  51,
  209, 143,  12,  82, 176, 238,  50, 108, 142, 208,
   83,  13, 239, 177, 240, 174,  76,  18, 145, 207,
   45, 115, 202, 148, 118,  40, 171, 245,  23,  73,
    8,  86, 180, 234, 105,  55, 213, 139,  87,   9,
  235, 181,  54, 104, 138, 212, 149, 203,  41, 119,
  244, 170,  72,  22, 233, 183,  85,  11, 136, 214,
   52, 106,  43, 117, 151, 201,  74,  20, 246, 168,
  116,  42, 200, 150,  21,  75, 169, 247, 182, 232,
   10,  84, 215, 137, 107,  53
```

```
};

//-------------------------------------------------------------------------------
-----------
BOOL CCommThread::IsError()
{
	auto_cs acs(cs);
	BOOL bRes = Error;
	Error = FALSE;
	return bRes;
}

//-------------------------------------------------------------------------------
-----------
BOOL CCommThread::IsIdle()
{
	auto_cs acs(cs);
	return State == Idle;
}

//-------------------------------------------------------------------------------
-----------
CString CCommThread::GetErrorMessage()
{
	auto_cs acs(cs);
	return ErrorMessage;
}

//-------------------------------------------------------------------------------
-----------
BOOL CCommThread::IsKey()
{
	auto_cs acs(cs);
	return Key;
}

//-------------------------------------------------------------------------------
-----------
BOOL CCommThread::IsRunningTryout()
{
	auto_cs acs(cs);
	return RunningTryout;
}

//------------------------подсчет КС-----------------------------------------
-------
BYTE CCommThread::CountCRC(BYTE* data, WORD length)
{
	BYTE CountCRC=0;
	BYTE x;

	for (WORD i=0; i< length; i++ )
	{
		x = * (data + i);
		CountCRC = TableCRC8[ CountCRC ^ x ];
	}
	return CountCRC;
}


//------------------------проверка КС-----------------------------------------
-------
BOOL CCommThread::CheckCRC(BYTE* data, WORD length)
{

	BYTE CountCRC=0;
	BYTE x;

	//КС считается без учета последнего байта (самой КС)
```

```
	length --;

	for (WORD i=0; i< length; i++ )
	{
		x = * (data + i);
		CountCRC = TableCRC8[ CountCRC ^ x ];
	}

	if (CountCRC == * ( data + length ) )
		return TRUE;
	else
		return FALSE;

}

//-----------------------------------------------------
CCommThread::CCommThread(CDataManager& DataManager, CSecurity& Security,
CLogFile& LogFile):
	m_DataManager(DataManager),
	m_Security(Security),
	m_LogFile(LogFile)
{
	PacketSent = 0;
	PacketReceived = 0;
	ErrorCount = 0;
	EnterCount = 0;
	EscapeCount = 0;

	Key = FALSE;
	PreviousKey = FALSE;

	hCom = NULL;

	Timeout = FALSE;
	TimeoutCounter = TIMEOUT_VALUE;

	m_Port = _T("\\.\\\\COM3");

	ZeroMemory (&InPacket, sizeof (InPacket));
	ZeroMemory (&OutPacket, sizeof (OutPacket));

	State = Initialize;
	CommandState = None;

	SearchingModeWanted = 0;
	ItemsReadingCount = 0;

	Error = FALSE;

	CRCError = FALSE;

	SearchingObjectIndex = 0;
	CurrentSearchingObjectIndex = 0;
	TotalSearchingObjectCount = 0;

	SettingNewValue = 0;
	SettingChangeStepCount = 0;
	SettingChangeStepCountTotal = 0;

	CurrentChangingSettingValue = 0;
	CurrentSettingMultiplier = 0;
	CurrentSettingStep = 0;

	SettingsCRCError = FALSE;;
	BlocksCRCError = FALSE;

	RunningTryout = FALSE;
	InitializeMode = -1;
```

```
	TimeoutWriteToFlash = FALSE;

}

//-----------------------------------------------------
CCommThread::~CCommThread()
{
	if (hCom != NULL)
	{
		bTerminateThread = TRUE;
		Sleep (1000);
	}

	Disconnect();

}


//-----------------------------------------------------
void CCommThread::Execute(const CString& Port)
{
	m_Port = _T("\\\\.\\") + Port;
	Connect();
	if (hCom != NULL)
		AfxBeginThread ( (AFX_THREADPROC)ThreadProc, this );
}


//-------------------------------------------------------------------------
-----------
void CCommThread::Connect()
{
	hCom = CreateFile( m_Port, GENERIC_READ | GENERIC_WRITE, 0, NULL,
OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL , NULL);
	if ( hCom == INVALID_HANDLE_VALUE )
	{
		hCom = NULL;
		return;
	}


	if ( ! SetupComm( hCom, 1024, 1024) )
	{
		if (GetLastError() != ERROR_NOT_SUPPORTED)
		{
			CloseHandle( hCom );
			hCom = NULL;
			return;
		}
	}

	if ( ! PurgeComm( hCom, PURGE_TXABORT | PURGE_RXABORT | PURGE_TXCLEAR |
PURGE_RXCLEAR ) )
	{
		CloseHandle( hCom );
		hCom = NULL;
		return;
	}


	PacketSent = 0;
	PacketReceived = 0;
	ErrorCount = 0;
	EnterCount = 0;
	EscapeCount = 0;

	TimeoutCounter = TIMEOUT_VALUE;
	TimeoutWriteToFlash = FALSE;
	Timeout = FALSE;
```

```
}


//-------------------------------------------------------------------------------
-----------
void CCommThread::Disconnect()
{
	if (hCom)
	{
		CloseHandle ( hCom );
		hCom = NULL;
	}
}


//-------------------------------------------------------------------------------
-----------
AFX_THREADPROC CCommThread::ThreadProc( LPVOID pParam )
{
	CCommThread* pThis = (CCommThread*)pParam;

	TRACE0("CCommThread started.\n");

	while (!pThis->bTerminateThread)
	{
		Sleep(50);

		pThis->Send();

		Sleep (100);

		do{
		}while (pThis->Receive());
	}

	TRACE0("CCommThread terminated.\n");

	return 0;
}

//-------------------------------------------------------------------------------
-----------
void CCommThread::Send()
{
	//---------------------------- готовим ответ для блока -------------------
-----
	double Value;

	switch (State)
	{

	case Idle:
		OutPacket.Command = COMMAND_CHOICE;
		break;

	case Initialize:
		OutPacket.Command = COMMAND_STATUS;
		break;

	case SearchingModeLevel:
		OutPacket.Command = COMMAND_ESCAPE;
		break;

	case SaveChanges:
		OutPacket.Command = COMMAND_CHOICE;
		break;
	}
```

```
	if (OutPacket.Command == COMMAND_ENTER)
		EnterCount++;

	if (OutPacket.Command == COMMAND_ESCAPE)
		EscapeCount++;

	OutPacket.Size = sizeof (OutPacket);
	OutPacket.Crc = CountCRC ((BYTE*)&OutPacket, sizeof (OutPacket));
	DWORD wr;
	WriteFile(hCom, &OutPacket, sizeof(OutPacket), &wr, NULL);

	if (wr != sizeof (OutPacket))
	{
		ErrorCount++;
		return;
	}

	PacketSent++;


}


//---------------------------------------------------------------------------
-----------
BOOL CCommThread::Receive()
{
	DWORD rd;
	if (ReadFile(hCom, &InPacket, sizeof(InPacket), &rd, NULL) == FALSE)
	{
		return FALSE;
	}

	if (rd != sizeof(InPacket))
	{

		if (!Timeout)
		{
			if (TimeoutCounter > 0)
				TimeoutCounter--;

			if (TimeoutCounter <= 0)
			{
				Timeout = TRUE;
				{
					m_DataManager.InvalidateData();
				}

				SettingsCRCError = FALSE;
				BlocksCRCError = FALSE;

				auto_cs acs(cs);
				State = Initialize;
				Key = FALSE;
				RunningTryout = FALSE;
			}
		}
		return FALSE;
	}

	if (Timeout)
	{
		Timeout = FALSE;
	}

	if (TimeoutCounter < TIMEOUT_VALUE)
		TimeoutCounter = TIMEOUT_VALUE;
```

```
	if (TimeoutWriteToFlash)
	{
		TRACE0("TIMEOUT_VALUE_FLASH Released\n");
		TimeoutWriteToFlash = FALSE;
	}

	if (!CheckCRC((BYTE*)&InPacket, sizeof(InPacket)))
	{
		ErrorCount++;
		SettingsCRCError = FALSE;;
		BlocksCRCError = FALSE;
		RunningTryout = FALSE;

		auto_cs acs(cs);
		Key = FALSE;

		CRCError = TRUE;

		return TRUE;
	}

	CRCError = FALSE;


	static BOOL PrevKey = FALSE;

	Lock();
	Key = (InPacket.Flags_Reserved & BIT_KEY) != 0;
	if (!Key)
	{
		m_Security.ResetPassword();
	}
	Unlock();

	if (PrevKey != Key)
	{
		m_LogFile.Write(Key ? _T("Ключ встановлено.") : _T("Ключ знято."));
		PrevKey = Key;
	}

	SettingsCRCError = (InPacket.Flags_Reserved & BIT_CRCERROR_SETTING) != 0;
	BlocksCRCError = (InPacket.Flags_Reserved & BIT_CRCERROR_BLOCK) != 0;


	PacketReceived++;
	//CString str;
	//str.Format(_T("Received: Size = %d, Command = %d, Param = %d,
Flags_Reserved = %d, Data = %d\n"), InPacket.Size, InPacket.Command,
InPacket.Param, InPacket.Flags_Reserved, InPacket.Data);
	//TRACE1("%s", str);


	int SettingsCount = m_DataManager.SettingsCount();
	int BlocksCount = m_DataManager.BlocksCount();

	auto_cs acs(cs);

	CSetting Setting;

	switch (State)    // for SettingChanging
	{


	case SearchingItem:
		CurrentSearchingObjectIndex = InPacket.Param;

		if (!IsKey())
		{
```

```
                  Error = TRUE;
                  State = Idle;
                  break;
            }

            if (CurrentSearchingObjectIndex == SearchingObjectIndex)
            {
                  if (CommandState == ToggleBlock)
                        State = BlockChanging;

                  if (CommandState == ChangeSetting)
                        State = SettingChanging;

                  if (CommandState == RunTryout)
                  {
                        RunningTryout = TRUE;
                        State = TryoutRunning;
                  }

            }
            break;

      case BlockChanging:
            if (!IsKey())
            {
                  Error = TRUE;
                  State = Idle;
                  break;
            }

            ASSERT (InPacket.Command == MESSAGE_STATE_BLOCK);
            State = SaveChanges;
            break;
      }

      return TRUE;

}

//-----------------------------------------------------------------------------
-----------
void CCommThread::InitiateSettingsRead()
{
      if (IsTimeout())
            return;
      if (!IsIdle())
      {
            ASSERT(0);
            return;
      }

      auto_cs acs(cs);

      SearchingModeWanted = STATE_COMMAND_SETTING;
      State = ChangeMode;
}


//-----------------------------------------------------------------------------
-----------
void CCommThread::CancelTryout()
{
      if (!IsRunningTryout())
      {
            ASSERT(0);
            return;
      }

      auto_cs acs(cs);
```

```
	CommandState = StopTryout;
}

//-------------------------------------------------------------------------------
-----------
BOOL CCommThread::GetInitializeMode(int& CurrentState)
{
	CurrentState = InitializeMode;
	if (InitializeMode == -1)
		return FALSE;

	InitializeMode = -1;
	return TRUE;
}
```

ComThread.h – реалізація комунікаційного потоку

```
#pragma once

#include "DataManager.h"
#include "Security.h"
#include "LogFile.h"

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

#define MESSAGE_STATE_MENU 0
#define MESSAGE_STATE_SETTING 1
#define MESSAGE_STATE_BLOCK 2
#define MESSAGE_STATE_TEST 3
#define MESSAGE_STATE_EDIT 4

#define STATE_COMMAND_SETTING 0
#define STATE_COMMAND_BLOCK 1
#define STATE_COMMAND_TEST 2

#define BIT_KEY 1
#define BIT_CRCERROR_SETTING 2
#define BIT_CRCERROR_BLOCK 4

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

struct InPacket_t
{
	BYTE Size;
	BYTE Command;
	BYTE Param;
	BYTE Flags_Reserved;
	short Data;
	BYTE Reserved;
	BYTE Crc;

};

//Команды, отправляемые в БФЗ
#define COMMAND_ESCAPE  1	//Керування - ESCAPE
#define COMMAND_ENTER   2	// Керування - ENTER
#define COMMAND_CHOICE  3	// Керування - Выбор
#define COMMAND_SETTING 4	//Запис уставки
#define COMMAND_STATUS  5	//Читання статусу

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

struct OutPacket_t
{
	BYTE Size;
	short Data;
	WORD Reserved;
	BYTE Command;
	BYTE Param_Reserved;
```

```
	BYTE Crc;

};

//---------------------------------------------------------------------------
class CCommThread
{
public:
	CCommThread(CDataManager& DataManager, CSecurity& Security, CLogFile&
LogFile);
	virtual ~CCommThread();

	void Lock() { cs.Lock(); }
	void Unlock() { cs.Unlock(); }

	void Execute(const CString& Port);

	DWORD PacketSent;
	DWORD PacketReceived;
	DWORD ErrorCount;
	DWORD EnterCount;
	DWORD EscapeCount;

	BOOL IsSettingsCRCError() { return SettingsCRCError; }
	BOOL IsBlocksCRCError() { return BlocksCRCError; }

	BOOL IsCRCError() { return CRCError; }

	BOOL IsTimeout() { return Timeout; }
	BOOL IsConnected () { return hCom != NULL; }

	BOOL IsKey();
	BOOL IsIdle();

	BOOL IsError();
	CString GetErrorMessage();

	//команды
	void InitiateBlocksRead();
	void InitiateSettingsRead();
	void InitiateBlockToggle(int Index);
	void InitiateSettingChange (int Index, double Value, int ChangeStepCount);
	void InitiateTryoutMode();
	void InitiateTryout(int Index, int MaxIndex);
	void CancelTryout();

	int GetItemsReadingCount() { return ItemsReadingCount; }

	BYTE GetCurrentSearchingObjectIndex() { return
CurrentSearchingObjectIndex; }	//номер текущего объекта при поиске
	BYTE GetSearchingObjectIndex() { return SearchingObjectIndex; }
	BYTE GetTotalSearchingObjectsCount() { return TotalSearchingObjectCount; }

	int GetSettingChangeStepCountTotal() { return SettingChangeStepCountTotal;
	int GetSettingChangeStepCount() { return SettingChangeStepCount; }
	double GetCurrentChangingSettingValue() { return
CurrentChangingSettingValue; }

	BOOL IsRunningTryout();
	int GetTryoutIndex() { return SearchingObjectIndex; }

	BOOL GetInitializeMode(int& CurrentState);

public:
	enum EState
	{
		Idle,
		Initialize,
```

```
            //

            SettingsReading,
            BlocksReading,

            SearchingModeLevel,
            SearchingMode,
            ChangeMode,
            EnteringMode,

            //
            SearchingItem,

            //
            BlockChanging,
            SettingChanging,

            EditConfirming,

            TryoutRunning
      }State;

      enum ECommand
      {
            None,
            ToggleBlock,
            ChangeSetting,
            RunTryout,
            StopTryout
      }CommandState;

private:
      CCriticalSection cs;

      InPacket_t InPacket;
      OutPacket_t OutPacket;

      CDataManager& m_DataManager;
      CSecurity& m_Security;
      CLogFile& m_LogFile;

private:
      BYTE CountCRC(BYTE* data, WORD length);
      BOOL CheckCRC(BYTE* data, WORD length);

      BOOL bTerminateThread;
      static AFX_THREADPROC ThreadProc( LPVOID pParam );

      void Send();
      BOOL Receive();

      void Connect();
      void Disconnect();

private:
      BOOL Key;
      BOOL PreviousKey;

      BOOL SettingsCRCError;
      BOOL BlocksCRCError;

      BOOL Error;
      CString ErrorMessage;

      BOOL CRCError;

      CString m_Port;
      HANDLE hCom;
```

```
	BOOL Timeout;
	WORD TimeoutCounter;

	BOOL TimeoutWriteToFlash;

	BYTE SearchingModeWanted;
	BYTE ItemsReadingCount;

	BYTE TotalSearchingObjectCount;
	BYTE CurrentSearchingObjectIndex;
	BYTE SearchingObjectIndex;

	double CurrentSettingStep;
	double CurrentSettingMultiplier;
	double CurrentChangingSettingValue;
	double SettingNewValue;

	int SettingChangeStepCountTotal;
	int SettingChangeStepCount;

	BOOL RunningTryout;
	int InitializeMode;
};
```

ColorButton.cpp – опис класу кнопки керування

```
// ColorButton.cpp : implementation file
//
#include "pch.h"
#include "ColorButton.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CColorButton
IMPLEMENT_DYNAMIC(CColorButton, CButton)

CColorButton::CColorButton()
{
	BlinkPhase = FALSE;
	bPushed = FALSE;

	ColorText_Alert = RGB (255, 0, 0);
	ColorBack_Alert = RGB (255, 0, 0);
	ColorText_NoAlert = RGB (0, 0, 0);
	ColorBack_NoAlert = GetSysColor (COLOR_BTNFACE);
	ColorText_Grayed = RGB (255, 255, 255);
	ColorBack_Grayed = RGB (255, 255, 128);
	ColorIfDisabled = FALSE;

	TextBlink = FALSE;
	BackBlink = FALSE;

	bAlerted = FALSE;
	bGrayed = FALSE;

	Data = 0;

	FontName = _T("Arial");
	FontSize = 12;
	FontBold = FALSE;
	FontItalic = FALSE;
	pFont = NULL;

}
```

```
CColorButton::~CColorButton()
{
	if (pFont)
	{
		delete pFont;
		pFont = NULL;
	}
}


BEGIN_MESSAGE_MAP(CColorButton, CButton)
	//{{AFX_MSG_MAP(CColorButton)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()



/////////////////////////////////////////////////////////////////////////////
// CColorButton message handlers
void CColorButton::DrawItem(LPDRAWITEMSTRUCT lpDS)
{
	CDC dc;
	dc.Attach ( lpDS->hDC );

	CString str;
	GetWindowText(str);

	CRect r;
	GetClientRect ( r );

	if ((GetState () & 0x0004) || bPushed)
		dc.DrawFrameControl ( r, DFC_BUTTON, DFCS_BUTTONPUSH|DFCS_PUSHED );
	else
		dc.DrawFrameControl ( r, DFC_BUTTON, DFCS_BUTTONPUSH );


	r.left += 2;
	r.top += 2;
	r.bottom -= 2;
	r.right -= 2;

	dc.SetBkMode ( TRANSPARENT );

	COLORREF TextColor = RGB(0, 0, 0);
	COLORREF BackColor = GetSysColor (COLOR_BTNFACE);


	//Выбор цвета
	if (bGrayed)
	{
		TextColor = ColorText_Grayed;
		BackColor = ColorBack_Grayed;
	}
	else
	{
		if (BlinkPhase  || bAlerted)
		{
			TextColor = TextBlink|| bAlerted ? ColorText_Alert :
ColorText_NoAlert;
			BackColor = BackBlink|| bAlerted ? ColorBack_Alert :
ColorBack_NoAlert;
		}
		else
		{
			TextColor = ColorText_NoAlert;
			BackColor = ColorBack_NoAlert;
		}
	}
```

```
  //Задизейбленное состояние
  if ( !IsWindowEnabled ())
  {
        TextColor =  RGB ( 100, 100, 100 );

        if (ColorIfDisabled && (bPushed || bGrayed || bAlerted) )
        {
              CBitmap bm;
              int iBits = dc.GetDeviceCaps(BITSPIXEL);
              bm.CreateBitmap(4, 4, 1, iBits, NULL);

              CDC BitmDC;
              BitmDC.CreateCompatibleDC( &dc );
              CBitmap* pOldMemBitmap = BitmDC.SelectObject(&bm);

              BitmDC.FillSolidRect(0, 0, 4, 4, RGB(0xFF, 0xFF, 0xFF) );
              BitmDC.SetPixel(0, 0, BackColor);
              BitmDC.SetPixel(0, 2, BackColor);
              BitmDC.SetPixel(2, 0, BackColor);
              BitmDC.SetPixel(2, 2, BackColor);
              BitmDC.SetPixel(1, 1, BackColor);
              BitmDC.SetPixel(1, 3, BackColor);
              BitmDC.SetPixel(3, 1, BackColor);
              BitmDC.SetPixel(3, 3, BackColor);

              BitmDC.SelectObject(pOldMemBitmap);
              CBrush ValueBrush;
              ValueBrush.CreatePatternBrush(&bm);
              dc.FillRect(r, &ValueBrush);
              dc.SetBkColor ( BackColor);
        }
        else
        {
              BackColor = GetSysColor (COLOR_BTNFACE);
              dc.FillSolidRect ( r, BackColor );
              dc.SetBkColor ( BackColor);
        }
  }
  else
  {
        dc.FillSolidRect ( r, BackColor );
        dc.SetBkColor ( BackColor);
  }

  dc.SetTextColor ( TextColor );


  CFont* pOldFont = NULL;

  if (pFont)
  {
        pOldFont = dc.SelectObject (pFont);
  }

  CRect rct = r;
  int h = dc.DrawText(str, rct, DT_CALCRECT | DT_WORDBREAK);
  int w = rct.Width();

rct.left = (r.Width() / 2 - rct.Width() / 2) + 1;
rct.right = rct.left + w;

  rct.top = (r.Height() / 2 - h / 2);
rct.bottom = r.bottom;
  dc.DrawText ( str, rct, DT_WORDBREAK | DT_CENTER);

  if (pOldFont)
  {
```

```
            dc.SelectObject (pOldFont);
      }

      if (GetFocus() == this)
      {
            r.DeflateRect(2, 2, 2, 2);
            dc.DrawFocusRect(r);
      }

      dc.Detach();

}


void CColorButton::SetAlert (BOOL bAlert)
{
      bAlerted = bAlert;
      Invalidate();
}


void CColorButton::SetPushed (BOOL bpushed)
{
      bPushed = bpushed;
      Invalidate();
}

void CColorButton::SetGrayed (BOOL Grayed)
{
      bGrayed = Grayed;
      Invalidate();
}

void CColorButton::Blink()
{
      if ( IsWindowEnabled())
      {
            if (BlinkPhase)
                  BlinkPhase = FALSE;
            else
                  BlinkPhase = TRUE;

            Invalidate();
      }
}
```

ColorButton.h – опис класу кнопки керування

```
#pragma once

/////////////////////////////////////////////////////////////////////////////
// CColorButton window

class CColorButton : public CButton
{
	DECLARE_DYNAMIC(CColorButton)
public:
	CColorButton();   //RSP Конструктор
public:
// Implementation
public:
	virtual ~CColorButton();

public:
	void SetFont(CString FontName, int Size, BOOL Bold, BOOL Italic);

	void Blink();
	void SetAlert(BOOL bAlert);

	void SetPushed(BOOL bPushed);
	void SetGrayed(BOOL Grayed);
	BOOL IsGrayed() { return bGrayed; }
	BOOL IsPushed() { return bPushed; }

	// Generated message map functions
protected:
	//{{AFX_MSG(CColorButton)
	afx_msg void OnEnable(BOOL bEnable);
	//}}AFX_MSG

	DECLARE_MESSAGE_MAP()

private:
	CString FontName;
	int FontSize;
	BOOL FontBold{FALSE};
	BOOL FontItalic{FALSE};
	CFont* pFont{NULL};

public:
	COLORREF ColorText_Alert;
	COLORREF ColorBack_Alert;
	COLORREF ColorText_NoAlert;
	COLORREF ColorBack_NoAlert;
	COLORREF ColorText_Grayed;
	COLORREF ColorBack_Grayed;

	BOOL TextBlink;
	BOOL BackBlink;

	BOOL ColorIfDisabled;

private:
	int Data;

	BOOL BlinkPhase;
	BOOL bAlerted;
	BOOL bPushed;
	BOOL bGrayed;
};
```