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
“____” _____ 2021 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи автоматизації виробничих процесів створення сонячних батарей класу Tier 2”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-20М-1,4
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Свистунов В.В.
«____» _____ 2021 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Сергій СМІРНОВ
«____» _____ 2021 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 *123 "*Комп'ютерна інженерія*"*

Освітньо-професійна (освітньо-наукова) програма _*"*Комп'ютерна інженерія*"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 6 » вересня 2021 року

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

Свистунову Віктору Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи автоматизації виробничих процесів створення сонячних батарей класу Tier 2*

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

затверджені наказом вищого навчального закладу № 42-13 від 02.08.2021 року

3. Строк подання студентом роботи до захисту *10.12.2021 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи автоматизації виробничих процесів створення сонячних батарей класу Tier 2*

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

1. Призначення та область використання.	*7. Економічна ефективність розробленої*
2.Перегляд аналогічних існуючих систем.	*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. Наукова новизна	

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.	05.10.2021	14.11.2021
Охорона праці	Оришака О.В.	06.10.2021	16.11.2021

7. Дата видачі завдання « 6 » вересня 2021 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	10.10.2021 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2021 р.	
3.	Розробка моделі компонента	20.10.2021 р.	
4.	Розробка структур даних	25.10.2021 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2021 р.	
6.	Програмування алгоритмів	10.11.2021 р.	
7.	Розрахунок економічної ефективності	13.11.2021 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2021 р.	
9.	Оформлення ПЗ	17.11.2021 р.	
10.	Попередній захист роботи	10.12.2021 р.	

Дата видачі завдання
« 6 » вересня 2021 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2021 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Свистунов В.В. Дослідження та програмна реалізація системи автоматизації виробничих процесів створення сонячних батарей класу Tier 2. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2021.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи автоматизації виробничих процесів створення сонячних батарей класу Tier 2.

Метою розробки є дослідження та програмна реалізація системи автоматизації виробничих процесів створення сонячних батарей класу Tier 2.

Об'єктом дослідження є процес автоматизації виробничих процесів створення сонячних батарей класу Tier 2.

Предметом дослідження є методи автоматизації виробничих процесів створення сонячних батарей класу Tier 2.

Методи дослідження базуються на методах теорії автоматизованого управління, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи автоматизації виробничих процесів створення сонячних батарей класу Tier 2.

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 XP/Vista/7/8/10.

Програму розроблено в середовищі Delphi 10.4 Sydney.

Ключові слова: комп'ютерна інженерія, сонячні батареї, Tier 2

ABSTRACT

Svistunov V.V. Research and software implementation of the system of automation of production processes of creating Tier 2 solar panels. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2021

In this final qualification work on the second (master's) level of higher education the software which is intended for system of automation of production processes of creation of solar batteries of the Tier 2 class is developed.

The purpose of development is research and software implementation of the system of automation of production processes of creation of solar batteries of the Tier 2 class.

The object of research is the process of automation of production processes for the creation of Tier 2 class solar panels.

The subject of research is methods of automation of production processes of creation of solar batteries of the Tier 2 class.

Research methods are based on the methods of the theory of automated control, methods of mathematical statistics, methods of software development.

The result is a software implementation of the system of automation of production processes for the creation of Tier 2 class solar panels.

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

Developed user-friendly interface. Instructions for working with software are given.

The program can be used on an IBM PC with Windows XP / Vista / 7/8/ 10.

The program is developed in the environment of Delphi 10.4 Sydney.

Keywords: computer engineering, solar panels, Tier 2

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 3
ВСТУП 4
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 7
1.1 Призначення системи 7
1.2 Область застосування 9
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 16
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти 16
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування 20
2.3 Розгорнута постановка завдання 26
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 28
3.1 Опис функціонування системи 28
3.2 Розробка структурної схеми 36
3.3 Розробка функціональної схеми 39
3.4 Розробка діаграми процесів 41
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ 43
4.1 Розробка блок-схем та опис алгоритмів функціонування системи 43
4.2 Захист розробленого програмного забезпечення 57
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ..... 60
6 НАУКОВА НОВИЗНА 64

7 ЕКОНОМІЧНА ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ 65
7.1 Техніко економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. 65
7.2 Розрахунок трудомісткості розробки програмної продукції 67
7.3 Визначення чисельності виконавців і планового фонду зарплати 69
7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника 74
7.5 Визначення собівартості розробки та ціни програмної продукції. 78
7.6 Визначення об’єму капітальних вкладень та експлуатаційних витрат у споживача програмної продукції..... 81
7.7 Визначення експлуатаційних витрат..... 81
7.8 Визначення економічної ефективності програмної продукції..... 83
7.9 Висновок. 85
8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ 86
8.1 Вступ 86
8.2 Шкідливі і небезпечні фактори при роботі з комп’ютером 87
8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста .. 88
8.4 Розробка заходів з умов поліпшення охорони праці..... 91
8.5 Розрахункова частина 92
8.6 Висновки до розділу..... 93
9 ОСНОВНІ ВИСНОВКИ..... 94
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 96

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

АЕС – атомні електростанції

АКБ – акумуляторна батарея

БГУ – бензогенераторні установки

ГФП – гетерофотоперетворювач

ДГУ – дизельгенераторні установки

ЕОМ – електронно-обчислювальна машина

ЕРС – електрорушійна сила

ККД – коефіцієнт корисної дії

СЕС – сонячна електростанція

СМ – сонячний модуль

ФЕП – фотоелектричний перетворювач

ФЕС – фотоелектрична станція

ІТ – інформаційні технології

API – прикладний програмний інтерфейс

AppWizard – засоби автоматизованого створення додатків

ISO/IEC – стандарт

OLE – технологія зв'язування й вбудовування об'єктів

USB – universal serial bus

ВСТУП

Актуальність теми. Загальновизнано, що основним фактором розвитку цивілізації є використання джерел енергії. В основному ми використовуємо традиційні енергоресурси, такі як – нафта, вугілля, природний газ. При цьому наноситься колосальний збиток екології нашого загального будинку за назвою ЗЕМЛЯ. Сотні тисяч барелів нафти зливаються в океан, мільйони тонн окису вуглецю викидаються в атмосферу, чотири сотні АЕС виробляють десятки тонн радіоактивних відходів [1-5].

Але справа не тільки в цьому, запаси цих традиційних джерел далеко не нескінченні. Тому їх відносять до непоновлюваних джерел енергії.

Наприклад, у рік у світі споживається стільки нафти, скільки її утвориться за 2 млн. років [2]. У зв'язку із цим останнім часом велика увага приділяється так званим поновлюваним джерелам енергії, таким як енергія вітру, сонця, припливу й т.д. У цьому ряді сонячна енергетика посідає не останнє місце.

Повна кількість сонячної енергії, що надходить на поверхню Землі за тиждень перевищує енергію всіх світових запасів нафти, газу, вугілля й урану [6].

Перетворення сонячної енергії в електричну здійснюється нині в тому числі й за допомогою фотоелектричних перетворювачів – ФЕП. Матеріалом для них служить один з найпоширеніших у земній корі елементів – кремній, а "паливом" – безкоштовні сонячні промені. Як стаціонарні джерела електрики, фотоелектричні станції привабливі для районів, не забезпечених електрикою від централізованої енергосистеми. Установка сонячних модулів вигідна там, де витрата енергії незначна, а проводка електромереж вимагає чималих витрат [2-8].

Для забезпечення потреб у сонячних батареях, необхідно виробництво по їх виробленню. Для підвищення якості виробництва, запроваджують автоматизовані системи управління.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи автоматизації виробничих процесів створення сонячних батарей класу Tier 2.

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

– Огляд існуючих систем автоматизації виробничих процесів створення сонячних батарей класу Tier 2.

– Дослідження системи автоматизації виробничих процесів створення сонячних батарей класу Tier 2.

– Програмна реалізація системи автоматизації виробничих процесів створення сонячних батарей класу Tier 2.

Об'єктом дослідження є процес автоматизації виробничих процесів створення сонячних батарей класу Tier 2.

Предметом дослідження є методи автоматизації виробничих процесів створення сонячних батарей класу Tier 2.

Методи дослідження базуються на методах теорії автоматизованого управління, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод автоматизації виробничих процесів створення сонячних батарей класу Tier 2.

– Розроблено вітчизняний продукт автоматизації виробничих процесів створення сонячних батарей класу Tier 2, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі автоматизації виробничих процесів створення сонячних батарей класу Tier 2.

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.

Робота апробована на LV Науково-технічна конференція здобувачів вищої освіти «Наука – виробництву», 2021, основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №12.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи автоматизації виробничих процесів створення сонячних батарей класу Tier 2, 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 системи

Виробництво сонячних елементів у світі сьогодні перевищує 500 Мвт щорічно. Якщо у використанні сонячної енергії в промислових масштабах ще багато проблем, то в повсякденний побут багатьох і багатьох мільйонів людей геліосистеми ввійшли міцно й назавжди. Звичайно, скептики можуть заявити, що сонячна енергія у зв'язку з її циклічністю (день-ніч) і, особливо, сезонністю для багатьох районів, досить екзотична. І це певною мірою вірно. У цьому випадку на допомогу приходить застосування ветрофотоенергетичних систем. Максимальні значення швидкості вітру спостерігаються в осінньо-зимовий період, коли надходження сонячної енергії зменшується, а влітку відсутність вітру цілком компенсується сонячною енергією.

Фотоелектричні станції (ФЕС) ідеальні для подорожей, у варіантах мобільного використання, маючи ФЕС, ви можете стати енергетично незалежним і насолоджуватися комфортом усюди, де є сонячне світло. При цьому абсолютно безшумно й нешкідливо для навколишнього середовища, без шкідливих відходів або викидів. Місця відпочинку обладнані сонячними елементами вільні від шуму й заходу дизелів, які доводиться включати, щоб мати електрику. Фотоелектричні станції можуть бути застосовані для живлення релейних радіокомунікацій. Фотоелектричні модулі можуть забезпечити катодний захист металоконструкцій, забезпечити роботу знаків водної навігації, водопідйомних установок, побутової радіоапаратури, а також здійснити заряд акумуляторних батарей для яких би те не було інших цілей. Як цікаве використання можна привести приклад електроогорожі призначеної для випасу худоби. Джерелом енергії для генератора імпульсної напруги в електроогорожі служить сонячний модуль потужністю 3 Вт, розміром 200 * 240 мм. Його потужності вистачає на забезпечення нормальної

роботи огорожі, що покриває площу 4 га, а також зарядки акумулятора для роботи огорожі в нічний час.

Сонячні електростанції можуть бути використані не тільки для рішення локальних завдань, але також і глобальних проблем енергетики.

У США, наприклад, існує декілька експериментальних ФЕС потужністю від 0,3 Мвт до 6,5 Мвт, що працюють на енергосистему. У Європі, зокрема, у Німеччині діє урядова програма, що надає податкові пільги виробникам сонячних батарей, монтуємих на дахах будинків. У цей час 350 тис. будинків у Китаї, Зімбабве, Індонезії й Мексиці електрифіковано за допомогою фотоелектричних систем, в Індії використовується 300000 сонячних світильників. В 2001 році повна встановлена потужність фотоелектричних систем у світі перевищила 1 ГВт. Лідери "великої вісімки" заявили на саміті в Генуї в липні 2001 року: "Ми будемо передбачати розвиток сонячної енергії в наших національних планах і підтримувати дослідження й інвестиції в нові технології". Наприклад, у Німеччині 1%, на Україні 0,075% тарифу на електроенергію збирається в спеціальний державний фонд, з якого через уповноважений банк на 20-30% знижується для покупця ціна енергетичної установки й установлюється на 10 років підвищений у два рази тариф на продавану в мережу електроенергію. Пільговий тариф знижується на 5% щороку для того, щоб через 10 років він зрівнявся зі звичайним тарифом на енергоносії для екологічно "брудних" електростанцій. Після введення цього закону німецькі громадяни-власники будинків взяли гроші з банків і купили в 2007 році на свої заощадження 50 Мвт фотоелектричних систем для власних дахів загальною вартістю 500 мільйонів євро. Фотоелектричні системи потужністю 50 Мвт у Німеччині зробили за рік 50 мільйонів кВт/год електроенергії вартістю 50 мільйонів євро. Власники фотоелектричних систем одержали 10% річних на вкладений капітал, що у два рази вище, ніж у банку. Німецька держава не витратила ні однієї марки бюджетних засобів.

Фотоелектричні перетворювачі мають значні потенційні переваги:

– не мають частин, що рухаються, що значно знижує вартість обслуговування;

– термін служби буде досягати, імовірно, 100 років при незначному зниженні експлутаційних характеристик (проблема не в самих перетворювачах, а в герметизуючих матеріалах);

– не вимагають високої кваліфікації персоналу;

– ефективно використовують як пряме так і розсіяне (дифузійне) випромінювання;

– придатні для створення установок практично будь-якої потужності.

Так що все людство, а не тільки дачники й власники кишенькових калькуляторів, стоять на порозі важливої події – зміни енергетичної бази.

1.2 Область застосування

Автономні й резервні сонячні електростанції

Всі фотоелектричні системи (ФЕС) можна розділити на два типи: автономні й резервні (з'єднані з електричною мережею). Станції другого типу віддають надлишки енергії в мережу, що служить резервом у випадку виникнення внутрішнього дефіциту енергії.«Сонячна» енергосистема складається з набору сонячних модулів (СМ), розміщених на опорній конструкції або на даху, акумуляторної батареї (АКБ), контролера розряду – заряду акумулятора, сполучних кабелів. Якщо споживачеві необхідно мати змінну напругу, то до цього комплекту додається інвертор-перетворювач постійної напруги в змінну.

При розрахунку ємності АКБ в автономному режимі необхідно мати на увазі й наявність у природі похмурих днів, під час яких акумулятор повинен забезпечувати роботу споживачів.

Завдяки досить високому ККД для монокристалічного кремнію – 16%, сонячні модулі здатні виробляти електроенергію навіть у хмарну й похмуру погоду.

При використанні ФЕС рекомендується максимально знизити потужність споживачів. Наприклад, як освітлювачі використовувати (по можливості) тільки люмінесцентні економічні лампи. Для невеликих ФЕС доцільно встановлювати її модулі на поворотному кронштейні для оптимального розвороту відносно падаючих променів. Це дозволить збільшити потужність станції на 20-30 %.

Таблиця 1.1 – Порівняльні експлуатаційні витрати при використанні «сонячної» енергосистеми й дизель (бензо)- генераторної установки(ДГУ, БГУ)

енергоустановка	1-й рік	2 -й рік	3 -й рік	4 -й рік	5 -й рік
«сонячна»	0	0	0	0	0
ДГУ(БГУ)	3820 грн.	4775грн.	5968грн.	7460грн.	9325грн.

Енергія Сонця на 1000 років уперед гарантована, і прихід сонячної енергії на Землю не зміниться. При при розвитку нових технологій, вартість виробництва сонячних фотоелектричних систем буде поступово здешевлятися. Виробництво енергії на землі кожні 10 років подвоюється в 2 рази. Важко сказати, через 50 або 80 років кількість зробленої енергії зрівняється з величиною поглиненою Землею сонячної радіації, а це приведе до перегріву Землі, що спричинить танення льодовиків. Використання сонячної енергії й переробка її для потреб людини, скоротивши при цьому виробництво енергії на землі, збереже нашу екологію й баланс природи.

Перетворення енергії у фотоелектричному перетворювачі (ФЕП) засновано на фотовольтичному ефекті, що виникає в неоднорідних напівпровідникових структурах при впливі на них сонячного випромінювання.

Розглянемо природу цього явища.

Неоднорідність структури ФЕП може бути отримана легуванням того самого напівпровідника різними домішками (створення p-n-переходів) або

шляхом з'єднання різних напівпровідників з неоднаковою шириною забороненої зони-енергії відриву електрона з атома (створення гетеропереходів), або ж за рахунок зміни хімічного складу напівпровідника, що приводить до появи градієнта ширини забороненої зони (створення варизонних структур). Можливі також різні комбінації перерахованих способів.

Ефективність перетворення залежить від електрофізичних характеристик неоднорідної напівпровідникової структури, а також оптичних властивостей ФЕП, серед яких найбільш важливу роль грає фотопровідність , обумовлена явищами внутрішнього фотоефекта в напівпровідниках при опроміненні їхнім сонячним світлом.

Принцип роботи ФЕП можна пояснити на прикладі перетворювачів з p-n-переходом, які широко застосовуються в сучасній сонячній і космічній енергетиці.

Електроно-дирковий перехід створюється шляхом легування пластинки монокристалічного напівпровідникового матеріалу з певним типом провідності (тобто або p- або n- типу) домішкою, що забезпечує створення поверхневого шару із провідністю протилежного типу. Концентрація легуючої домішки в цьому шарі повинна бути значно вище, ніж концентрація домішки в базовому матеріалі (первісному монокристалі), щоб нейтралізувати наявні там основні вільні носії заряду й створити провідність протилежного знака. У границі n- і p-шарів у результаті перетічки зарядів утворяться збіднені зони з нескомпенсованим об'ємним позитивним зарядом в n-шарі й об'ємному негативному заряді в p-шарі. Ці зони в сукупності й утворять p-n-перехід.

Виниклий на переході потенційний бар'єр (контактна різниця потенціалів) перешкоджає проходженню основних носіїв заряду, тобто електронів з боку p-шару, але безперешкодно пропускають неосновні носії в протилежних напрямках. Це властивість p-n-переходів і визначає можливість одержання фото-ЕРС при опроміненні ФЕП сонячним світлом.

Створені світлом в обох шарах ФЕП нерівноважні носії заряду (електронно-дирочні пари) розділяються на p-n-переході: неосновні носії (тобто електрони) вільно проходять через перехід, а основні (дірки) затримуються. Таким чином, під дією сонячного випромінювання через p-n-перехід в обох напрямках буде протікати струм нерівновагих неосновних носіїв заряду- фотоелектронів і фотодірок, що саме й потрібно для роботи ФЕП. Якщо тепер замкнути зовнішній ланцюг, то електрони з n-шару, зробивши роботу на навантаженні, будуть вертатися в p-шар і там рекомбінувати (поєднуватися) з дірками, що рухаються усередині ФЕП у протилежному напрямку. Для збору й відводу електронів у зовнішній ланцюг на поверхні напівпровідникової структури ФЕП є контактна система.

На передній, освітленій поверхні перетворювача контакти виконуються у вигляді сітки або гребінки, а на тильній можуть бути суцільними.

Основні необоротні втрати енергії у ФЕП пов'язані з:

– відбиттям сонячного випромінювання від поверхні пеертворювача;

– проходженням частини випромінювання через ФЕП без поглинання в ньому;

– розсіюванням на теплових коливаннях ґрат надлишкової енергії фотонів;

– рекомбінацією фотопар, що утворилися, на поверхнях і в обсязі ФЕП;

– внутрішнім опором перетворювача;

– і деякими іншими фізичними процесами.

Для зменшення всіх видів втрат енергії у ФЕП розробляються й успішно застосовуються різні заходи.

До їхнього числа відносяться:

– використання напівпровідників з оптимальною для сонячного випромінювання шириною забороненої зони;

– спрямоване поліпшення властивостей напівпровідникової структури шляхом її оптимального легування й створення убудованих електричних полів;

– перехід від гомогенних до гетерогенного й варизоним напівпровідниковим структурам;

– оптимізація конструктивних параметрів ФЕП (глибини залягання p-n-переходу, товщини базового шару, частоти контактної сітки й ін.);

– застосування багатофункціональних оптичних покриттів, що забезпечують просвітління, терморегулювання й захист ФЕП від космічної радіації;

– розробка ФЕП, прозорих у довгохвильовій області сонячного спектра за краєм основної смуги поглинання;

– створення каскадних ФЕП зі спеціально підібраних по ширині забороненої зони напівпровідників, що дозволяють перетворювати в кожному каскаді випромінювання, що пройшло через попередній каскад, та ін.;

Також істотного підвищення ККД ФЕП удалося домогтися за рахунок створення перетворювачів із двосторонньою чутливістю (до +80 % до вже наявного ККД однієї сторони), застосування люмінесцентно перевипромінюючих структур, попереднього розкладання сонячного спектра на дві або більше спектральні області за допомогою багатошарових плівкових світлоділителів (дихроїчних дзеркал) с наступним перетворенням кожної ділянки спектра окремим ФЕП і т.д.

У системах перетворення енергії СЕС (сонячних електростанцій) у принципі можуть бути використані будь-які створені й розроблювальні в цей час типи ФЕП різної структури на базі різноманітних напівпровідникових матеріалів, однак не всі вони задовольняють комплексу вимог до цих систем:

– висока надійність при тривалому (десятки років!) ресурсі роботи;

– доступність вихідних матеріалів у достатній для виготовлення елементів системи перетворення кількості й можливість організації їхнього масового виробництва;

– прийнятні з погляду строків окупності енерговитрати на створення системи перетворення;

– мінімальні витрати енергії й маси, пов'язані з керуванням системою перетворення й передачі енергії (космос),включаючи орієнтацію й стабілізацію станції в цілому;

– зручність техобслуговування.

Так, наприклад, деякі перспективні матеріали важко одержати в необхідні для створення СЕС кількістях через обмеженість природних запасів вихідної сировини й складності його переробки. Окремі методи поліпшення енергетичних і експлутаційнних характеристик ФЕП, наприклад за рахунок створення складних структур, погано сумісні з можливостями організації їхнього масового виробництва при низької вартості й т.д.

Висока продуктивність може бути досягнута лише при організації повністю автоматизованого виробництва ФЕП, наприклад на основі стрічкової технології, і створенні розвитої мережі спеціалізованих підприємств відповідного профілю, тобто фактично цілої галузі промисловості, порівнянної по масштабах із сучасною радіоелектронною промисловістю. Виготовлення сонячних елементів і складання сонячних батарей на автоматизованих лініях забезпечить зниження собівартості модуля батареї в 2-2,5 рази.

Як найбільш ймовірні матеріали для фотоелектричних систем перетворення сонячної енергії СЕС у цей час розглядається кремній і арсенід галію (GaAs), причому в останньому випадку мова йде про гетерофотоперетворювач (ГФП) зі структурою AlGaAs-GaAs.

Таким чином, виходячи з усього вищеперерахованого, виробництво сонячних батарей є дуже актуальним. Але розроблені сонячні батареї повинні відповідати деяким вимогам, тобто повинні бути чітко визначені значення по наступним параметрам сонячних батарей:

- вологість;
- тиск на поверхні;
- щільність електричної енергії при 25°С
- максимальна система напруги;

– припустиме відхилення електричних даних;
– потужність ±5% (Вт) ;
– струм максимальної потужності I_{mp} (А);
– напруга максимальної потужності V_{mp} (В);
– номінальна напруга (В);
– струм короткого замикання I_{sc} (А);
– напруга холостого ходу V_{oc} (В);
– габаритні парметри повинні перебувати в рамках заданих;
– розмір сонячного елемента p/s –псевдоквадрат;
– кількість елементів (розташування);
– вага (кг) повинна бути не більш максимально припустимої.

Тобто необхідно розробити таке програмне забезпечення для автоматизованої лінії виробництва сонячних батарей (елементів сонячних батарей),

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи автоматизації виробничих процесів створення сонячних батарей класу Tier 2, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Фотоелектричні енергетичні установки являють собою сонячні батареї, що перетворюють сонячне світло в електричний струм. Вони звичайно виготовляються у вигляді тонких к ремнієвих пластин прямокутної або кругової форми розміром 5-10 сантиметрів.

Сонячні фотоелектричні системи прості в обігу й не мають механізмів, що рухаються, однак самі фотоелементи містять складні напівпровідникові пристрої, аналогічні використовуваним для виробництва інтегральних схем. В основі дії фотоелементів лежить фізичний принцип, при якому електричний струм виникає під впливом світла між двома напівпровідниками з різними електричними властивостями, що перебувають у контакті один з одним.

Принцип роботи сонячних батарей заснований на фотогальванічному ефекті. Потрапляючи на поверхню напівпровідника, фотони сонячного світла вивирають електрони з його атомів. Під впливом спеціальної хімічної речовини, що додається в структуру матеріалу, звільнені електрони переміщаються в певному напрямку, у результаті чого утвориться електричний струм.

Сукупність таких елементів утворить фотоелектричну панель, або модуль. Фотоелектричні модулі, завдяки своїм електричним властивостям, виробляють постійний, а не змінний струм. Він використовується в багатьох простих пристроях, що живляться від батарей. Змінний же струм, навпроти, міняє свій напрямок через регулярні проміжки часу. Саме цей тип електрики поставляють енерговиробники, він використовується для більшості сучасних приладів і електронних пристроїв. У найпростіших системах постійний струм

фотоелектричних модулів використовується прямо. Там же, де потрібний змінний струм, до системи необхідно додати інвертор, що перетворить постійний струм у змінний.

Сучасне виробництво фотоелементів практично повністю засновано на кремнії. Близько 80% всіх модулів виробляється з використанням полі- або монокристалічного кремнію, а інші 20% використовують аморфний кремній. Кристалічні фотоелементи – найпоширеніші, звичайно вони мають синій колір з відблиском. Аморфні, або некристалічні – гладкі на вид і міняють колір залежно від кута зору. Монокристалічний кремній має найкращу ефективність (близько 14%), але він дорожчий, ніж полікристалічний, ефективність якого в середньому становить 11%. Аморфний кремній широко застосовується в невеликих приладах, таких як годинники й калькулятори, але його ефективність і довгострокова стабільність значно нижче, тому він рідко застосовується в силових установках.

У дослідній розробці перебувають кілька типів альтернативних тонкоплівкових фотоелементів, які в майбутньому можуть завоювати ринок. Найбільш налагодженими з досліджуваних у цей час тонкоплівкових систем є фотоелементи з наступних матеріалів:

– аморфний кремній (a-Si: H);

– теллурид/сульфід кадмію (CTS);

– мідно-індієвий або мідно-галієвий диселенід (CIS or CIGS);

– тонкоплівковий кристалічний кремній(c-Si film);

– нанокристалічні сенсибілізовані барвником електрохімічні фотоелементи (nc-dye).

Фотоелемент являє собою "сендвич" із кремнію – другої по поширеності на Землі речовини. Дев'яносто дев'ять відсотків сучасних сонячних елементів виготовляють із кремнію (Si), а інші побудовані на тім же принципі, що й кремнієві сонячні елементи. На один шар кремнію наноситься певна речовина, завдяки якій утворюєся надлишок електронів. Виходить негативно заряджений ("N") шар. На іншому шарі створюється недолік електронів, він стає позитивно

зарядженим ("Р"). Зібрані разом із провідниками, ці дві поверхні утворять світлочутливий електроно-дирковий перехід. Він називається напівпровідником, тому що, на відміну від електропроводу, проводить струм тільки в одному напрямку – від негативного до позитивного. При впливі сонця або іншого інтенсивного джерела світла виникає постійний струм напругою приблизно в 0,5 Вольт. Сила струму (ампер) пропорційна світлової енергії (кількості фотонів). У будь-якій фотоелектричній системі напруга майже постійна, а струм пропорційний розміру фотоелементів і інтенсивності світла.

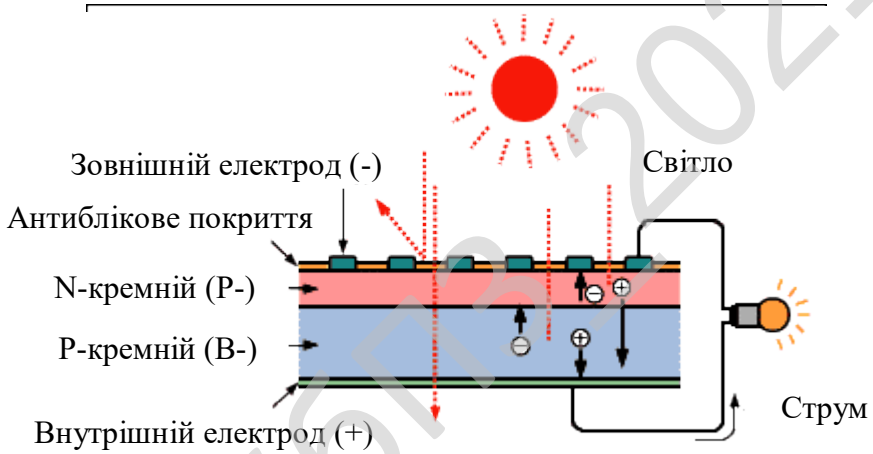

Рисунок 2.1 – Структура фотоелементу

Фотоелементи виробляються з надчистого кремнію, змішаного в точній пропорції з деякими іншими речовинами. Надчиста кремнієва подложка, з якої роблять фотоелементи, коштує дуже дорого. Кількості надчистого кремнію, необхідного для виготовлення одного фотоелектричного модуля потужністю 50 Вт, було б досить для інтегральних схем приблизно двох тисяч комп'ютерів. Крім того, у фотоелементах присутні алюміній, скло й пластмаса – недорогі й багаторазово використовувані матеріали.

Автоматизована технологія виробництва сонячних батарей

Процес виробництва батарей складається з декількох етапів:

1. Кремній ретельно очищають від домішок і розплавляють. Для очищення кремнію, здебільшого, усе ще застосовується недосконала, розроблена три десятиліття назад хлорисанова технологія виробництва.

2. З розплаву одержують твердий монолітний циліндр, називаний болванкою. Новітні методи виробництва дозволяють замість циліндричної болванки як заготівку використовувати кремнієві стрічки, блоки або плівки.

3. Від циліндра відрізаються найтонші пластини.

4. Після належної хімічної обробки вони формують фотоелектричні осередки, які часто прийнято називати сонячними батареями.

Для більшої ефективності й практичності безліч осередків паралельно або послідовно з'єднують у скляний корпус – модуль. Безліч модулів, у свою чергу, утворить масив. Додаванням масивів можна одержати стільки енергії, скільки необхідно.

Таким чином, виходячи з перерахованого вище, автоматизована система виробництва сонячних батарей, повинна складатися з наступних блоків:

– Блок автоматизованого управління процесу очищення від домішок, тобто блок, де по заданому алгоритмі, реалізуються кроки хлорисанової технології очищення.

– Блок автоматизованого управління процесу розплавлення, тобто блок включення печі на задану температуру й на встановлений час, достатнє для розплавлювання сировини з якого формується сонячна батарея (кремній (Si); аморфний кремній (a-Si: H); теллурид/сульфід кадмію (CTS); мідно-індієвий або мідно-галієвий диселенід (CIS or CIGS); тонкоплівковий кристалічний кремній(c-Si film); нанокристалічні сенсибілізовані барвником електрохімічні фотоелементи (nc-dye)).

– Блок автоматизованого керування процесом формування твердого монолітного циліндра, кремнієвих стрічок, блоків або плівок, який керує роботом, що заливає порцію розплавленого кремнію у відповідні форми.

– Блок автоматизованого керування процесом нарізки тонких пластин.

– Блок автоматизованого керування процесом хімічної обробки, результатом якої є формування фотоелектричних осередків.

– Блок автоматизованого керування процесом формування сонячних батарей, заданої розмірності й форми, з фотоелектричних осередків.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

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

Delphi 10.4 Sydney

Випущено 26 травня 2020 року. RAD Studio Delphi 10.4 забезпечує значно поліпшену високопродуктивну нативну підтримку Windows, кращу продуктивність розробки, миттєві підказки code completion, прискорення виконання коду із синтаксисом керованих записів, поліпшення виконання паралельних завдань на сучасних багатоядерних CPU, а також містить більш 1000 виправлень багів, поліпшення продуктивності середовища й бібліотек і багато чого крім того.

Основні можливості Delphi 10.4.1:

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

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

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

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

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

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

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

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

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

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

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

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

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

RAD Studio 10.4 Короткий огляд:

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

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

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

Істотне поліпшення Delphi Code Insight

Як найбільше й головне поліпшення інструментів програмування Delphi за багато років, в 10.4 Delphi Code Insight реалізований через Language Server Protocol (LSP). LSP – це технологія генерації результатів для code completion,

навігації й інших сервісів в окремому процесі. Це значить, що code completion і Code Insight одержать більш точні результати без блокування IDE. 10.4 забезпечує набагато більш високу продуктивність розроблювачів, які працюють із більшими проектами, що містять мільйони рядків коду.

Delphi Custom Managed Records

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

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

Єдине керування пам'яттю

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

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

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

Розширена підтримка бібліотек C++

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

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

Win 64-відладник і збирач для C++

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

Підвищення якості й швидкодії інструментів

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

Змінені стилі VCL для High DPI

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

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

Нові High DPI стилі й стилізація окремих VCL компонент

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

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

Поліпшена кроссплатформеність

– Додана підтримка Metal Driver GPU для macOS і iOS.

– Крім підтримки останнього iOS SDK, в RAD Studio 10.4 розроблювачі можуть задовольнити нові вимоги Apple до набору стартових екранів.

– Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

– Користувачам редакцій Enterprise або Architect доступна повна інтеграція Fmxlinux з IDE для створення клієнтських застосунків Linux з GUI.

– Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.

– Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

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

Універсальний інсталятор для установки Online і Offline

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

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

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи автоматизації виробничих процесів створення сонячних батарей класу Tier 2.

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 в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

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

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

д) розробити рекомендації по організаційних та методичних заходах, які

забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Для виробництва елементів сонячних батарей, використовується автоматизована система, елементом якої є ряд верстатів з ЧПУ, які, виходячи із завантаженого програмного забезпечення, виконують ту або іншу операцію, по формуванню елемента сонячної батареї.

Всі команди передаються по лініях зв'язку, які використовують інтерфейс передачі даних з керуючого пристрою на верстат з ЧПУ.

Один з можливих варіантів відкритої архітектури системи керування наведено на рисунку 3.1.

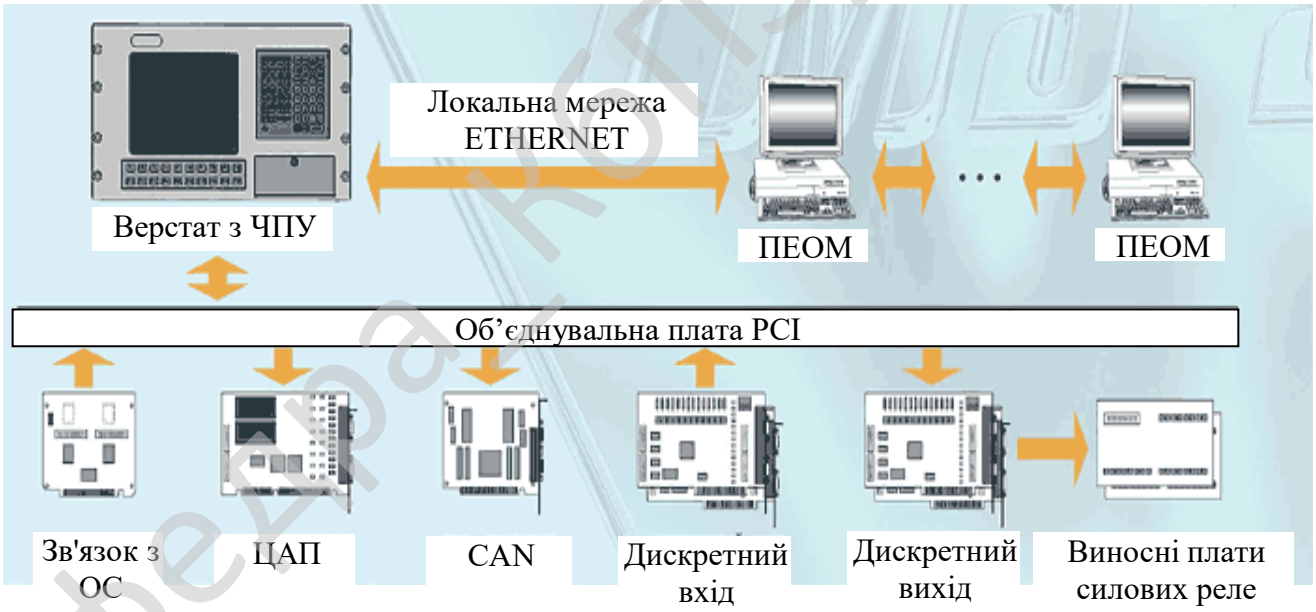

Рисунок 3.1 – Відкрита архітектура системи керування

Основа системи керування верстатом з ЧПУ для формування елемента сонячної батареї – персональний комп'ютер промислового виконання,

виготовлений по новітніх технологіях. Відкрита архітектура системи керування в комплексі з потужним програмним забезпеченням дозволяють легко інтегрувати в її состав комп'ютерні компоненти провідних виробників обчислювальної техніки.

Програмне забезпечення системи реалізоване на базі ядра твердого реального часу, що гарантує високу якість виконання таймерних завдань керування сервоприводами й циклами електроавтоматики. Завдяки новій технології Biesse WRT (Windows Real Time), що розширює функціональні можливості Windows для роботи в реальному часі, стійка ЧПУ верстата перебуває безпосередньо на ПК, що дозволяє сконцентрувати все необхідне апаратне забезпечення без використання додаткових плат керування. Відкрита структура системи дозволяє включати до складу системи програмні модулі користувача. При реалізації системи необхідно реалізувати наступні модулі:

Модуль графічної підтримки. Графіка дозволяє:

– переглядати реальний рух інструмента;

– перевіряти елементи траєкторії інструмента при відключеному верстаті;

– робити точне настроювання приводів у режимі осцилоскопування.

Модуль пооперaційного контролю. Система виготовляється з використанням пооперaційного контролю, починаючи від підготовки виробництва й до випуску готового виробу, включаючи температурний прогін з наступною функціональною перевіркою роботи всіх модулів. Технологічне програмування використовує:

– звичні G-функції;

– трибуквені коди, що дозволяють давати осмислені назви макрокомандам;

– вбудована мова опису графічних об'єктів, що дозволяє програмувати складні 2D контури безпосередньо із креслення без використання САПР. ЧПУ сам обчислює крапки перетинання й торкання геометричних елементів. Таке

рішення дозволяє легко орієнтуватися у виборі функцій для складання керуючих програм вручну.

Середовище статистичних даних, у якому записуються дані про роботу верстата й вироблених деталей, щоб здійснювати моніторинг надійності в часі й продуктивності верстата. Середовище може конфігуруватися під користувача, що дозволяє реєструвати можливих операцій, наприклад, таких як настроювання верстата, контроль обробки, задані паузи, цикли змащення елементів верстата й т.д.

Система телесервісу. Дозволяє миттєво одержувати прямий доступ через модем до стійки ЧПУ верстата. Таким чином, дозволяє перевірити дані верстата, програми користувача, сигнали вводу/виводу, змінні системи, а також установлювати оновлені версії ПЗ, що дозволяє:

- Проводити діагностику в реальному часі
- Швидко вирішувати проблеми
- Зменшити час простою
- Установлювати оновлені версії ПЗ в реальному часі

Обслуговування й робота з ЧПУ при формуванні елемента сонячної батареї не викликає ніяких труднощів у персоналу. Через повідомлення користувачеві надається повний контроль за виконуваними операціями, а також за станом комплексу ЧПУ – верстат.

Керування пристроєм ЧПУ повинне бути виконане в діалоговому режимі з використанням меню, що дозволяє легко орієнтуватися у виконуваних операціях.

Система підключення верстатів до ПЕОМ, що дозволяє інтегрувати існуючі верстати з ЧПУ, призначені для виробництва елементів сонячних батарей, (у тому числі й з морально й фізично застарілими системами ЧПУ) із сучасними CAD/CAM-системами повинна відповідати наступним вимогам:

– Висока надійність роботи.

– Використання наявного (найчастіше застарілого) устаткування. При цьому вартість модернізації не повинна перевищувати 1-5% вартості верстата.

– Підключення в мережу верстатів з будь-якими інтерфейсами введення інформації з єдиного протоколу. Дана вимога пов'язане із широким спектром наявного встаткування із всілякими інтерфейсами введення інформації – від магнітної стрічки й перфострічки до портів RS-232, RS-422 та USB. Єдиний протокол зв'язку з верстатами дозволяє значно спростити матзабезпечення на ПЕОМ. Можлива автоматична фіксація завантаження верстатів, включених у мережу, і передача відповідної інформації.

– Передача керуючих програм великого й особливо великого обсягу. Одноразова передача керуючих програм на верстати, а якщо керуюча програма перевищує мегабайт, то й організація режиму «підкачування». Отримані в ЧПУ програми можуть бути піддані корекції, тому керуюча програма повинна бути передана на верстат по можливості одноразово

– Відсутність доробок пристроїв ЧПУ верстата. Очевидно, що підключення верстата в мережу доцільно здійснювати через наявні у верстаті інтерфейси вводу-виводу.

– Максимально просте спілкування оператора верстата з ПЕОМ. У пристроях сполучення передбачені індикація й клавіатура, що дозволяють операторові верстата приймати/передавати необхідні керуючі програми/параметри від ПЕОМ і відслідковувати режими передачі.

– Технічні засоби підключення в комп'ютерну мережу повинні бути основою для розробки гнучкого програмного забезпечення (залежно від умов конкретного виробництва), що дозволяє значно скоротити строки впровадження керуючих програм для обробки різних деталей на верстатах з ЧПУ.

– Система передачі керуючої програми на верстати з ЧПУ повинна працювати як з архівом керуючої програми, так і з базою д аних керуючої програми, що найчастіше використовує СУБД працюючої на підприємстві АСУ.

Опис запропонованої модернізації існуючого програмного забезпечення

На існуючих верстатах з ЧПУ, механіка дозволяє, а застаріла електроніка верстата, а також можливості завантажувальних пристроїв і носія інформації не дозволяють повністю автоматизувати систему виробництва сонячних модулів. Тобто. для того, щоб верстати працювали в повністю автоматизованому режимі, необхідно до існуючого програмного забезпечення розробити ряд програмних модулів, які б дозволили прибрати існуючі недоліки.

В магістерській роботі, мною реалізовано програмне забезпечення, яке дозволяє ліквідувати деякі з недоліків, виявлених в результаті проведення дослідження, описаного у попередньому пункті. Зокрема при вирішені проблеми завантаження керуючих програм великого розміру в існуючі верстати з ЧПУ, для виробництва елементів сонячних батарей, з їхнім малим обсягом пам'яті, вихід полягає в реалізації наступного принципу: якщо в стійки не може бути цілком завантажена велика керуюча програма – її треба розбити на частини, а верстати нехай виконують програму, одержуючи її з одного центрального комп'ютера малими порціями. Керуюча програма великого обсягу, обробляється спеціальною програмою й одержує змінену керуючу програму (єдина велика керуюча програма усередині, розбита на шматки меншого розміру, що дозволяє завантажити шматок у стійку й оформлена образом, підходящим для стійки). Зі стійки запитується перетворений єдиний файл, що складається зі шматків. Перший раз завантажується перший шматок, потім він відпрацьовується. По завершенні шматка автоматично видаляється відпрацьований шматок з пам'яті й запитується наступний шматок єдиного файлу керуючої програми. Відпрацьовує – видаляє – запитує наступний шматок. Так доти, поки не будуть обрані всі шматки єдиного файлу керуючої програми.

Наступним моментом удосконалення стандартного програмного забезпечення є застосування передачі керуючої програми на верстат ЧПУ та даних, за допомогою USB порту.

Розглянемо пристрій USB всередині. Одна з головних концепцій USB полягає в тому, що в USB -системі може бути тільки один майстер. Їм є host - комп'ютер. USB – пристрою завжди відповідають на запити host-комп'ютера – вони ніколи не можу посилати інформацію самостійно. Є тільки одне виключення: після того, як хост перевів пристрій в suspend-режим, пристрій може надсилати запит remote wakeup. У всіх інших випадках хост формує запити, а пристрою відповідають на них. Хост завжди є майстром, а обмін даними повинен здійснюватися в обох напрямках:

OUT – відсилаючи пакет із прапором OUT, хост відсилає дані пристрою

IN – відсилаючи пакет із прапором IN, хост відправляє запит на прийом даних із пристрою.

До цьому потрібно просто звикнути. Щоб прийняти дані із пристрою, хост відсилає пакет із прапором IN. smile

По USB може передаватися кілька типів пакетів:

1. Token – запит, містить керуючу інформацію: напрямок операції (IN, OUT), номер endpoint

2. Data – пакет даних

3. Handshake – службові пакети, можуть містити підтвердження (ACK), повідомлення про помилку, відмова (NACK)

4. Special – службові пакети, такі як PING

Відсилання даних верстату ЧПУ, який застосовується для формування елемента сонячної батареї, виконуються в такий спосіб:

Щоб відіслати дані верстату, хост посилає пакет Token "OUT", потім пакет Data. Якщо верстат готовий обробити прийняті дані, він відсилає пакет Handshake "ACK", що підтверджує транзакцію. Якщо він зайнятий, то відсилає відмову – Handshake "NACK". Якщо відбулася якась помилка, то пристрій може не відсилати Handshake.

Для відсилання даних хосту виконуються наступні дії. Як уже говорилося, пристрій самостійно ніколи не відсилає дані. Тільки по запиту. Щоб прийняти

дані, хост посилає пакет Handshake "IN". Пристрій по запиту може відіслати пакет Data, а потім Handshake "ACK". Або може відіслати Handshake "NACK", не посилаючи Data.

Специфікація USB визначає 4 типи потоків даних:

1. bulk transfer – призначений для пакетної передачі даних з розміром пакетів 8, 16, 32, 64 для USB 1.1 і 512 для USB 2.0. Використовується алгоритм перепосилки (у випадку виникнення помилок), а керування потоком здійснюється з використанням handshake пакетів, тому даний тип є достовірним. Підтримуються обоє напрямку – IN і OUT.

2. control transfer – призначений для конфігурування й керування пристроєм. Також, як і в bulk, використовуються алгоритми підтвердження й перепосилки, тому цей тип забезпечує гарантований обмін даними. Напрямку – IN (status) і OUT(setup, control).

3. interrupt transfer – схожий на bulk. Розмір пакета – від 1 до 64 байт для USB 1.1 і до 1024 байт для USB 2.0. Цей тип гарантує, що пристрій буде опитуватися (тобто хост буде відсилати йому token "IN") хостом із заданим інтервалом. Напрямок – IN.

4. isochronous transfer – призначений для передачі даних без керування потоком (без підтверджень). Область застосування – аудіо-потоки, відео-потоки. Розмір пакета – до 1023 байт для USB 1.1 і до 1024 байт для USB 2.0. Передбачений контроль помилок (на прийомній стороні) по CRC16. Напрямки – IN і OUT.

Специфікація USB визначає endpoint (EP), як джерело або приймач даних. Пристрій може мати до 32 EP: 16 на прийом і 16 на передачу. Звертання до того або іншого endpoint'у відбувається по його адресі.

EP0 має особливе значення для USB. Це Control EP. Він повинен бути в кожному USB-пристрої. Цей EP використовує token "setup", щоб сигналізувати, що дані, що відправляються після нього, призначені для керування пристроєм. Використовуючи цей EP0, хост може передавати setup-пакет довжиною 8 байт і

дані, які випливають за цим пакетом. У багатьох випадках може вистачати передачі тільки setup-пакета. Однак пристрій може використовувати й передачу даних по EP0, наприклад для зміни прошивань компонентів пристрою, або одержання розширеної інформації про пристрій. Розглянемо побайтно setup-пакет. Байт № 0 – bmRequestType – Поле для вказівки типу запиту, напрямку, одержувача. Байт № 1 – bRequest -ідентифікатор запиту. 2 – wValue – 16-бітне значення wValue, залежить від запиту. 3 – wValue. 4 – wIndex 16-бітне значення wIndex, залежить від запиту. 5 – wIndex. 6 -wLength кількість байт, що відсилаються після setup-пакета. 7 – wLength. Як видно setup-пакет містить 5 полів. bmRequestType і bRequest визначають запит, а wValue, wIndex і wLength – його властивості. Специфікація USB резервує діапазон значень bRequest під стандартні запити. Кожний пристрій зобов'язаний відповідати на всі стандартні запити. Далі наведені тільки кілька стандартних запитів, з якими ми будемо зіштовхуватися далі.

– 0x05 Set Address установка унікальної адреси пристрою в системі.

– 0x06 Get Descriptor одержання інформації про пристрій. Тип інформації залежить від поля wValue.

Інший діапазон пристрій може використовувати за своїм розсудом.

Розпізнавання пристрою, що тільки що підключився до системи відбувається в такий спосіб. Уже згадувалося, що кожний пристрій зобов'язаний забезпечити доступ до EP0. Але крім цього, воно ще повинне відповідати на запити, зазначені в специфікації USB для EP0. Користуючись цими запитами й відбувається розпізнавання пристрою в системі.

Алгоритм детектування нового пристрою наступний:

1. хост відсилає setup-пакет "Get Descriptor" (wValue = "device").

2. хост одержує ідентифікуючу інформацію про пристрій

3. хост відсилає setup-пакет "Set address", після чого пристрій одержує унікальну адресу в системі

4. хост відсилає інші setup-пакети "Get Descriptor" і одержує додаткову інформацію про пристрій: кількість EP, вимоги до живлення, і т.п.

Розроблювальне програмне забезпечення повинне включати наступні програмні блоки:

– програму мікроконтролера для кожного типу стійки, що забезпечує інтерфейс зі стійкою ЧПУ;

– програму для комп'ютера по обслуговуванню верстатів;

– програму для комп'ютера по індикації якості мережі;

– програму для комп'ютера – драйвер для адаптера мережі.

3.2 Розробка структурної схеми

У поставленому завданні магістерської роботи – розробка програмного забезпечення системи автоматизації виробничих процесів створення сонячних батарей класу Tier 2 було розроблене додаткове програмне забезпечення для верстата LGZTT-500, що забезпечує його додатковими функціями – контролю параметрів шаблона майбутньої пластини сонячної батареї, її розміри та інші технічні характеристики. Програмне забезпечення розроблено на основі програмного забезпечення, яке поставляється з верстатом, та доповнює стандартне програмне забезпечення рядом функцій перерахованих вище. Пр цьому введена можливість передавати дані на верстат за допомогою додаткового USB порту.

Верстат LGZTT-500 під який розроблене додаткове програмне забезпечення був закуплений товариством з обмеженою відповідальністю малим впроваджувальним підприємством «ОКХ Ярило» (далі ТОВ МВП «ОКХ Ярило»).

При експлуатації верстата виявилось ряд дефектів у програмному забезпеченні що поставляється з верстатом, а саме – відсутність можливості керування за допомогою ПЗ розмірами пластини та її технічними параметрами та отримувати кодів відповідей верстата при його роботі над пластиною. Всі ці дії

виконуються у ручному режимі при нагляді оператора що значно ускладнює процес виготовлення через можливий людський фактор, що може привести до збою верстату та його поломку. Приведена структура станка та коди взаємодії дозволили розробити додаткове програмне забезпечення що я і виконав у магістерській роботі.

Розглянемо структурну схему роботи системи, а саме повний цикл виробничого процесу сонячних панелей. Без короткого огляду процесу виготовлення неможливо зрозуміти роль розробленого магістерського програмного забезпечення.

На рисунку 3.2 представлена розроблена структурна схема роботи системи, розглянемо її детально (зліва на право) для розуміння дій над верстатом та роботу програми.

Після закупки полікристалічного кремнію (покупної сировини) проводиться виробництво технічних злитків які будуть завантажуватися у верстат. Отримуються замовлення батарей (пластин) від покупців, докладні параметри батареї (пластин). Після цього інженери проводять аналіз цих даних та корегують виробничий процес якщо це необхідно, задають параметри у верстат та проводять виробництво батарей (пластин). Далі виконується перевірка технічних параметрів батарей (пластин) та поставка пластин замовникові.

У цей час у світі більше 90% фотоелементів виробляється на основі полікристалічного кремнію за технологіями, заснованим на використанні трихлорсилана.

Рисунок 3.2 – Структурна схема системи

«ОКХ Ярило» закупає полікристалічний кремній за високою ціною так як процес його виготовлення громіздкий та дорогий.

Через це кожний брак панелей являє собою значні матеріальні втрати товариства та розроблена магістерська програма дозволяє у значній мірі виправити це становище.

Процес виготовлення полі кремнію зображен на рисунку 3.3.

Трихлорсилан (ТХС) – SiHCl – кремнезмістовний неорганічне з'єднання, з використанням якого виробляється близько 90% світового обсягу полікристалічного кремнію (полікремнію). На основі трихлорсилана одержують моносилан і дихлорсилан, які також використаються у виробництві полікремнію. Трихлорсилан є сировиною в синтезі основного ряду кремніорганического мономеру. Існують і інші області застосування трихлорсилана, як, наприклад, мікроелектроніка, де ТХС використається для эпитаксиального осадження плівок монокрісталического кремнію.

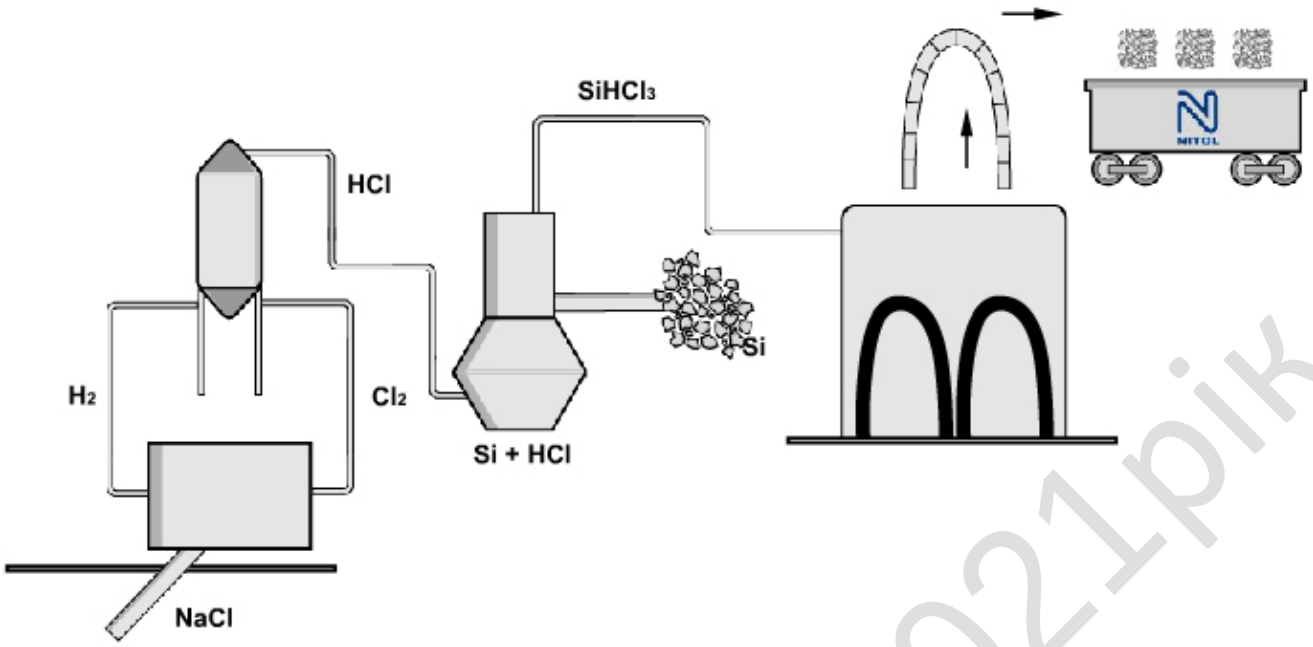

Рисунок 3.3 – Технологія виробництва полікремнію

3.3 Розробка функціональної схеми

На рисунку 3.4 зображена розроблена функціональна схема апаратної частини. З неї можна побачити як працює верстат та як їм керують. При подачі технічних злитків полікристалічного кремнію у верстат LGZTT-500 відбувається виробництво батарей (пластин) шляхом проведення хімічної взаємодії та подальшого заливання полікристалічного кремнію у комірки виготовленої форми.

Керування верстатом проходить у двох режимах: Ручне керування виконується оператором дії якого обмежені; Програмне керування яке в свою чергу підрозділяється на керування температурних характеристик та процесів плавлення технічних злитків основним програмним забезпеченням та керування розробленою магістерською програмою параметрами створюваної форми пластини та її стану.

Параметри комірок, слою заливання та інше встановлює розроблена у магістерській роботі програма.

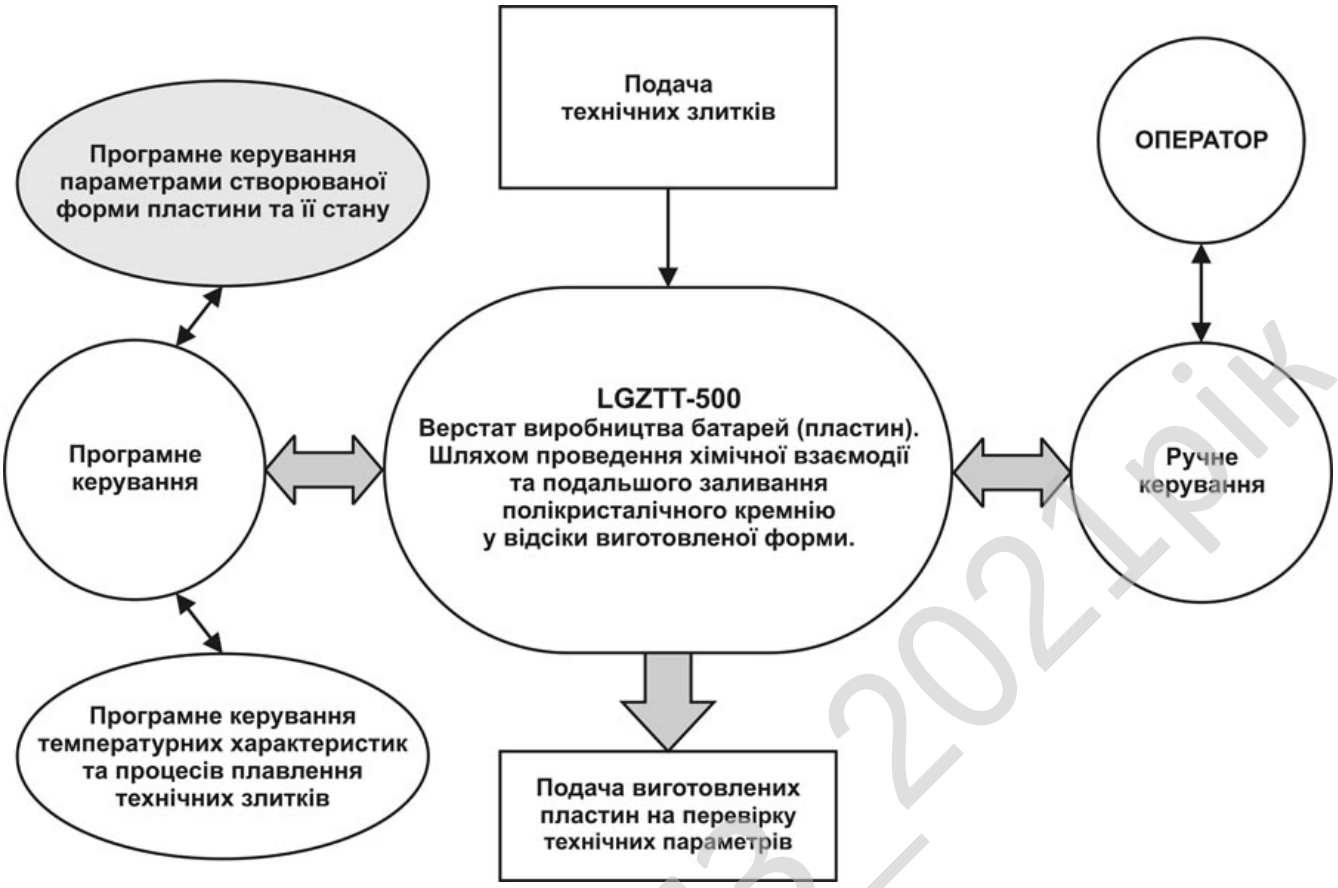

Рисунок 3.4 – Функціональна схема апаратної частини

У подальшому при виконанні всіх операцій виготовляється пластина та далі ця пластина подається на перевірку технічних параметрів.

На рисунку 3.5 представлена функціональна схема розробленої програми. Вона розподіляється на персональний комп'ютер оператора верстата з розробленою магістерською програмою та верстат виробництва батарей (пластин) LGZTT-500.

Взаємодія з верстатом проходить по USB – каналу хост контролера далі в операційній системі розроблене програмне забезпечення яке складається з інтерфейсу ПЗ, функції взаємодії з LGZTT-500, модулю захисту ПЗ та налагодження ПЗ проводить контроль та опит верстата.

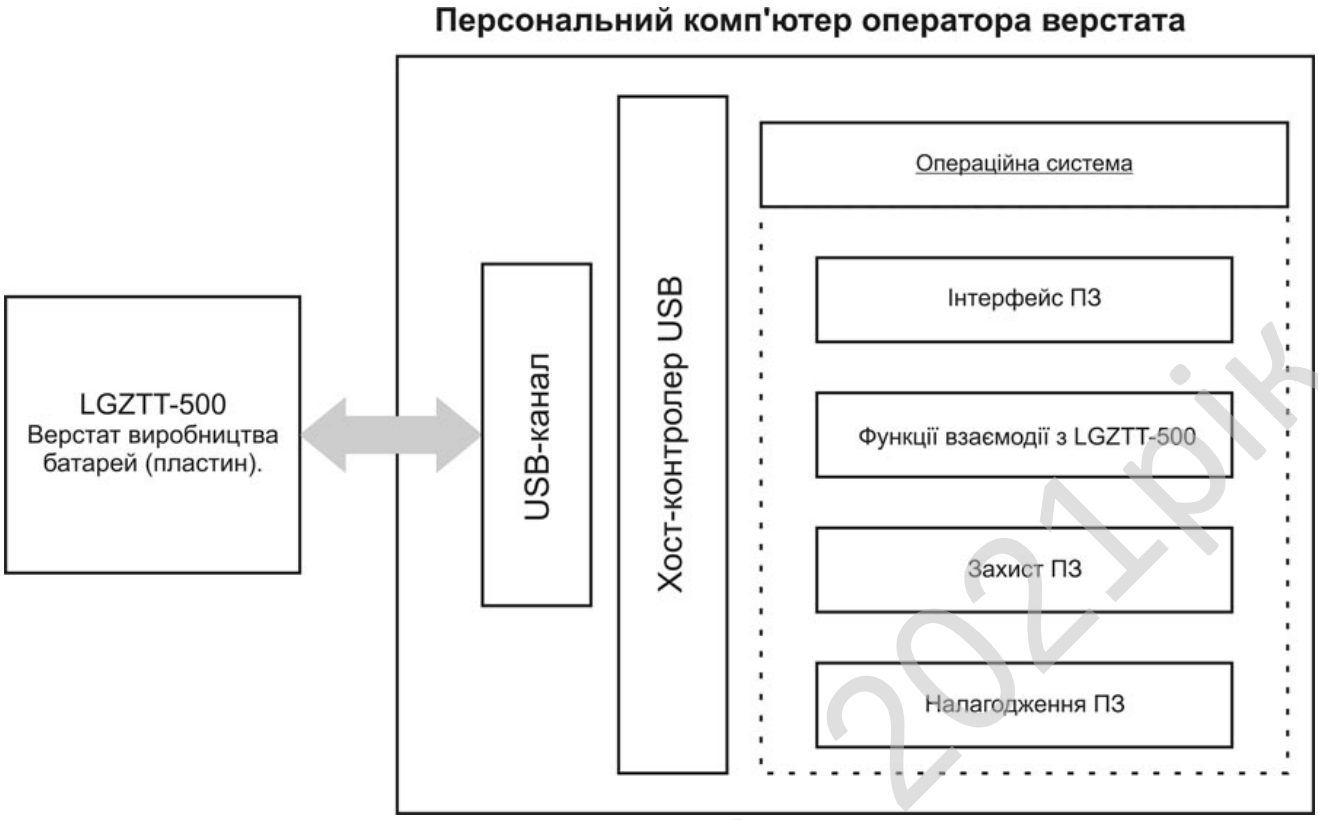

Рисунок 3.5 – Функціональна схема програмної частини

3.4 Розробка діаграми процесів

Розглянувши діаграму процесів зображену на рисунку 3.6 можна точно зрозуміти як працює програма та взаємодіє з верстатом.

Починається і закінчується програма в першому блоці це є основною точкою відрахунку діаграми. При переміщенні по стрілках можна побачити загальну схему взаємодії блоків і їх входження один в одного, та як вона робить.

Як можна побачити з основного блоку програми управляння переходить спочатку до інтерфейсу ОС та потім до інтерфейсу розробленого ПЗ який з'єднує інші модулі, такі як – модуль захисту ПЗ, налагодження ПЗ, бібліотеки налагодження та взаємодії з USB та хост – контролер USB. В свою чергу модуль хост – контролер USB проводить взаємодію з LGZTT-500.

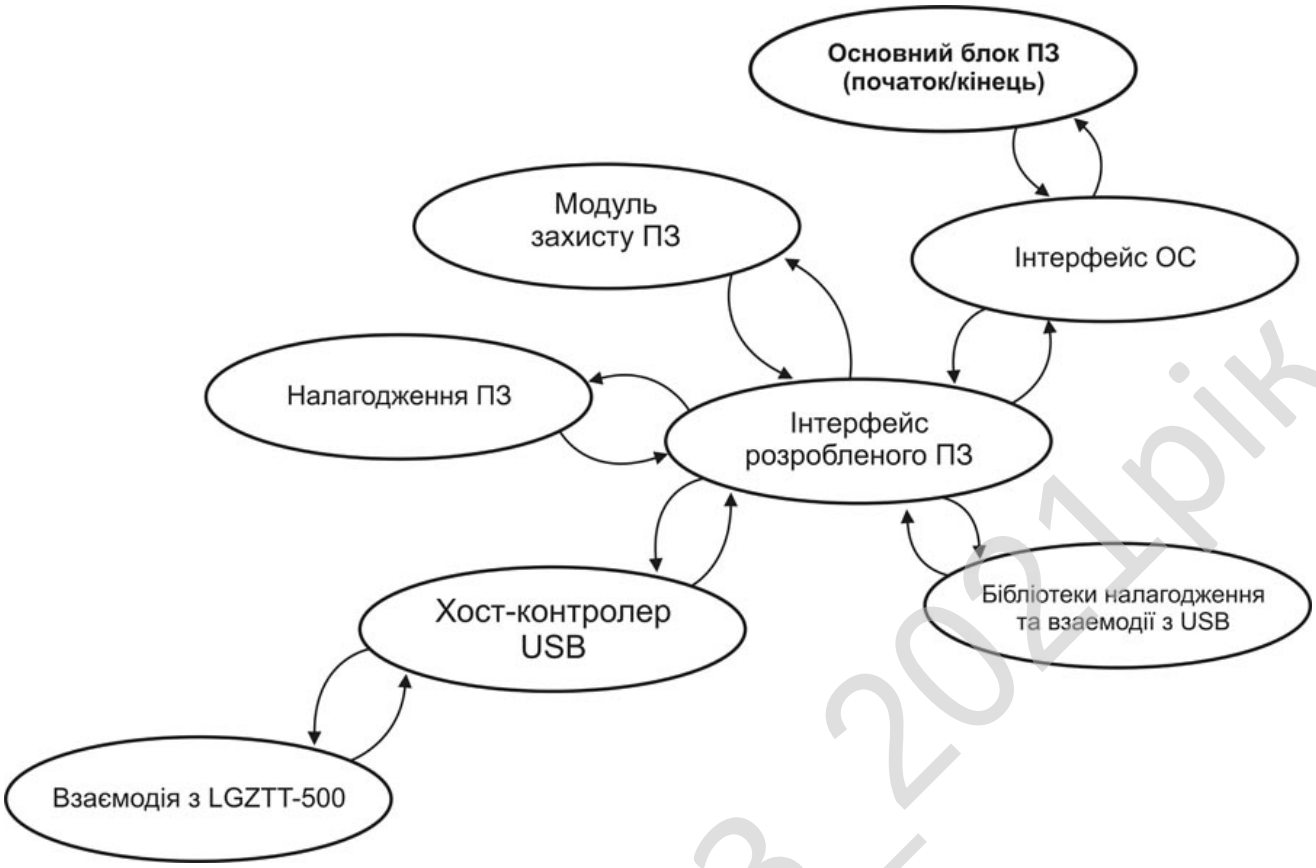

Рисунок 3.6 – Діаграма процесів

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Розглянемо блок–схеми (рисунки 4.1, 4.2, 4.3) розробленої магістерської програми. Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно є розробка блок-схем. З них виходить програма та все що з нею зв’язує.

З основного алгоритму програми (рисунок 4.1) винесені дві основних підпрограми це підпрограма встановлення розмірів пластини (рисунок 4.2) та блок-схема підпрограми обробки коду попередження. Виділимо у розробленій блок-схемі основні блоки.

Початок роботи та перевірки основної блок-схеми:

– Ініціалізація змінних ПЗ.

– Підключення модулів ПЗ.

– Підключення модулів взаємодії з USB.

– Тестове з'єднання з USB.

– Перевірка з'єднання.

Робота основної блок-схеми:

– Виведення головного вікна програми.

– Очікування дії користувача.

– Запит встановлення розмірів пластини – підпрограма встановлення розмірів пластини.

Початок роботи підпрограми встановлення розмірів пластини:

– Приховання головного вікна програми.

– Виведення вікна встановлення розмірів пластини.

– Введення розмірів користувача.

– Перевірка введених розмірів.

– Обробка даних.

– Перевірка доступу до USB.

– Встановлення розмірів пластини.

– Виведення повідомлення.

– Приховання вікна встановлення розмірів пластини.

– Виведення головного вікна програми.

Кінець роботи підпрограми встановлення розмірів пластини.

– Запит коду попередження – підпрограма обробки коду попередження.

Початок роботи підпрограми встановлення розмірів пластини:

– Завантаження коду попередження з LGZTT-500.

– Перевірка завантаження коду попередження.

– Перевірка коду відсутності шаблону та проведення обробки коду.

– Перевірка коду браку пластини та проведення обробки коду.

– Перевірка коду поломки верстата та проведення обробки коду.

Кінець роботи підпрограми встановлення розмірів пластини.

Завершення основної блок-схеми:

– Звільнення виділених ресурсів програми.

Як видно з блок-схеми підпрограми обробки коду попередження взаємодія з верстатом відбувається за допомогою кодів, розглянемо декілька з них (всього 356 кодів):

– Код № 0 Код попередження. Очікування наступної програмної дії.

– Код № 1 Код попередження. Повернення діагностичних даних.

– Код № 2-4 Службові зарезервовані коди.

– Код № 5 Код попередження. Подача неякісних брусків полікристалічного кремнія.

– Код № 6 Код попередження. Подача нормальних брусків полікристалічного кремнія.

– Код №7 Код попередження. Подача високоякісного брусків полікристалічного кремнія.

– Код № 8 Код попередження. Вихід за максимальні межі пластини.

– Код № 9 Код попередження. Межі пластини прийняті.

– Код № 10 Код попередження. Недостача енергії верстату.

– Код № 11 Код попередження. Недостача брусків полікристалічного кремнія.

– Код № 12 Код аварії! Недеагнастований код. Внутрішня помилка.

– Код № 13 Код аварії! Автоматична зупинка верстата!.

– Коди № 13-230 Службові коди.

– Коди № 230-356 Зарезервовані коди.

При складанні блок-схем програмного забезпечення і напрацювання алгоритмів взаємодії я зіткнувся з масою проблем, які вимагали напрацювання процедур і функцій над основною проблематикою. Для чого були створені додаткові процедури та функції що забезпечили вирішення проблем. Розглянемо ряд розроблених процедур та функцій, основні функції взаємодії з верстатом:

– Функція відкриття взаємодії з верстатом

```
UsbOpen(DevNum : Byte) : boolean.
```

– Функція закриття взаємодії з верстатом

```
UsbClose(DevNum : Byte) : boolean.
```

– Функція ініціалізації взаємодії

```
function UsbInit(DevNum : Byte) : Boolean.
```

– Функція ініціалізації верстата

```
UsbDeviceInitialized(DevNum : Byte) : Boolean.
```

– Функція перевірки наявності зв’язку з верстатом

```
UsbDevicePresent(DevNum : Byte) : Boolean.
```

– Функція повернення коду помилки

```
function UsbGetError:Byte.
```

– Функція повернення повторного тексту помилки

```
function UsbGetErrorStringByNr(ErrNr : Byte. PString : PChar):boolean.
```

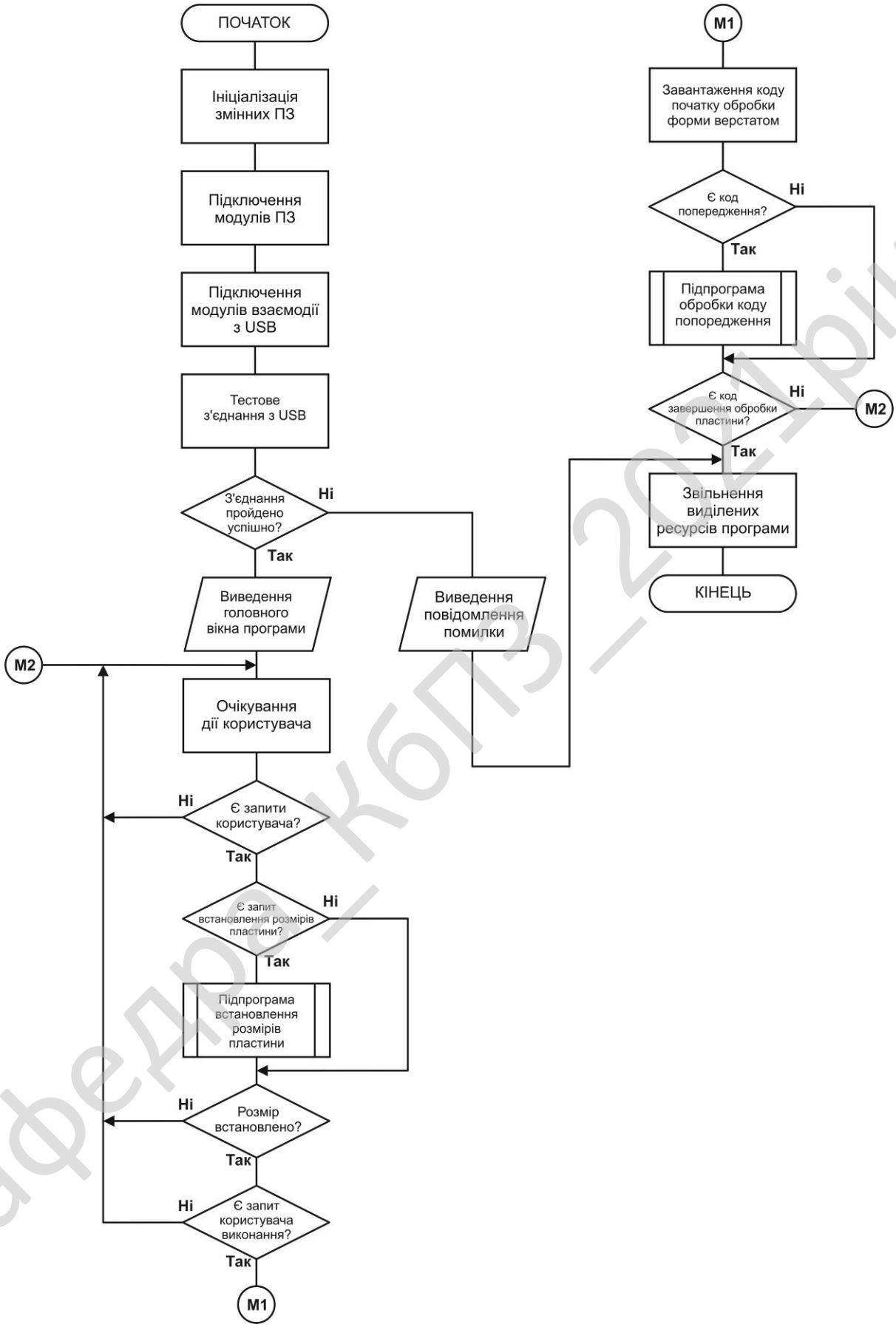

Рисунок 4.1 – Блок схема основної частини ПЗ

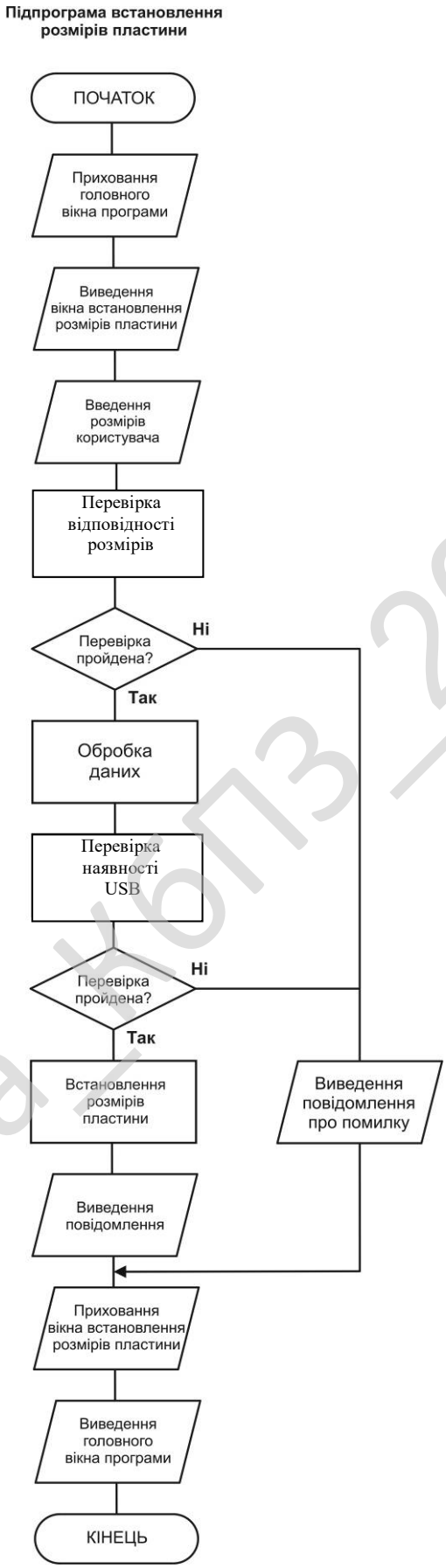

Рисунок 4.2 – Блок-схема підпрограми встановлення розмірів пластини

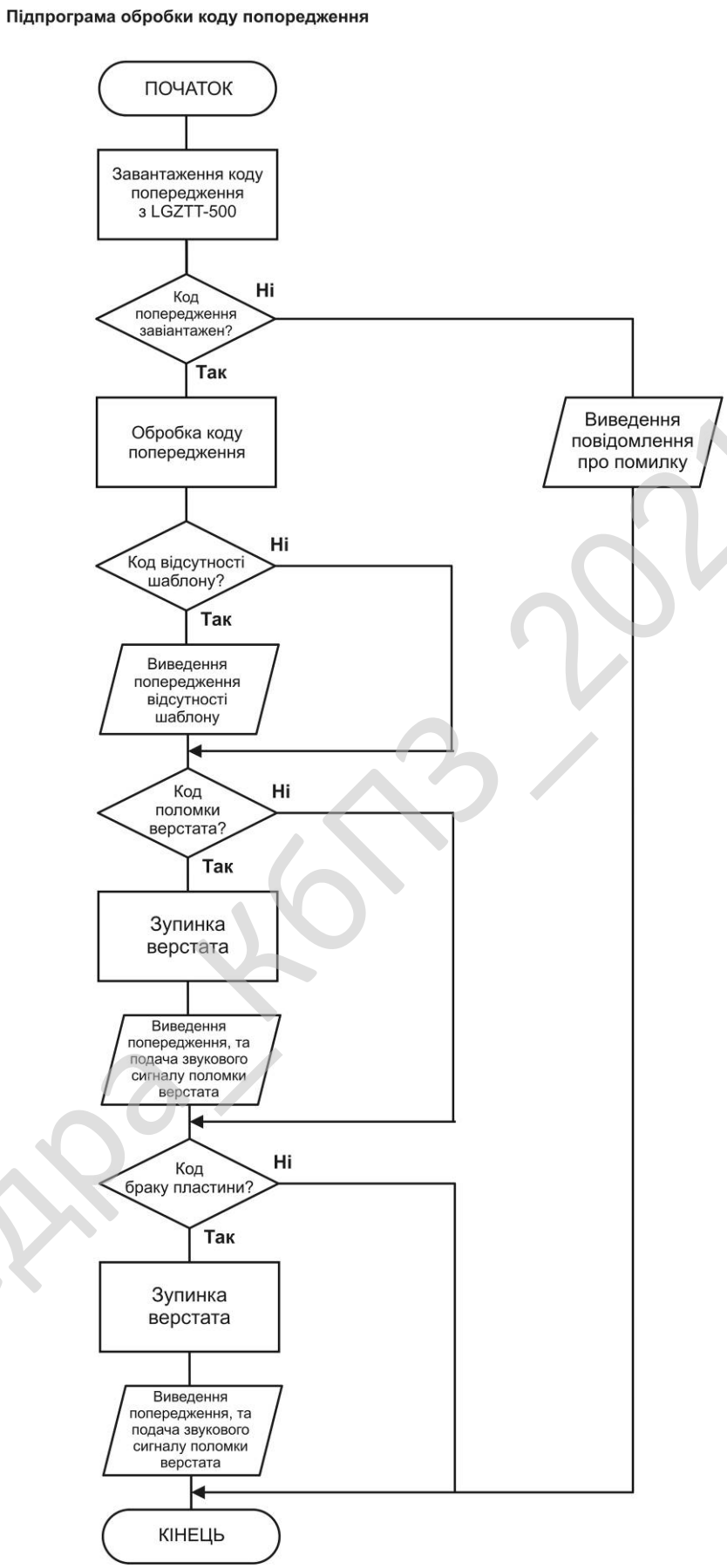

Рисунок 4.3 – Блок-схема підпрограми обробки коду попередження

– Функція встановлення часу відклику пристрою

```
function UsbSetTimeout(DevNum : Byte; timeout : Word) : boolean;
```

– Функція повернення поточної версії

```
UsbGetVersion(VersionStr : PChar);
```

– Функція повернення поточного статуса

```
function UsbSetIOState(DevNum, LineNum, State : Byte) : boolean;
```

– Функція передачі даних з вказаною кількістю байт (25,512,1500)

```
UsbParOut(0, @Buffer, 25);
UsbParOut(0, @Buffer, 512);
UsbParOut(0, @Buffer, 1500);
```

Розрахунки та розробка модуля взаємодії з верстатом

Розглянемо розроблений модуль взаємодії з верстатом. За допомогою цього модулю відбувається обмін даними та забезпечення відсутності конфліктів з основною програмною.

```
unit SetLGZTT-500;
//Модуль обміну информацією
interface
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, ExtCtrls, UsbTypes;
type
  TForm1 = class(TForm)
  LogListBox: TListBox;
  Panel1: TPanel;
  Button1: TButton;
  LogListBox: TListBox;
  Memo1: TMemo;
  Button2: TButton;
  procedure Button1Click(Sender: TObject);
  procedure Button2Click(Sender: TObject);
  private
   procedure Log(S : String);
  public
   procedure EnumerateHostDevices(hHost : THandle);
   procedure ShowRootHubInfo(RootHubName : String);
   procedure ShowRootHubDetail(hRoot : THandle);
procedure ShowHubPortDetail(hRoot:THandle; iPort:Integer);
procedure ShowDeviceDetail(hRoot:THandle; iPort:Integer);
procedure DisplayDescriptorInfo(Data:Array of byte;var IsHid:Boolean);
```

```
    procedure DisplayREPORTDescriptorInfo(Data:Array of byte);
    end;
```

Хост USB взаємодіє з верстатом через USB хост контролер. Хост відповідальний за наступне:

– виявлення приєднання і видалення пристроїв USB;

– керування потоком керування між хостом та верстатом USB;

– керування потоком даних між хостом і верстатом USB;

– збір статистики про стан і активність;

– забезпечення подачі обмеженої кількості потужності на приєднані пристрої USB.

```
var
  Form1: TForm1;
implementation
{$R *.dfm}
// Додавання рядка в LogListBox
procedure TForm1.Log(S : String);
var i:integer;
begin
for i:=0 to LogListBox.Items.Count do
Begin
U:=U+S;
end;
LogListBox.Items.Add(S);
memo1.Text:= LogListBox.Items.Text;  IntToStr(LogListBox.Items.Count);
LogListBox.ItemIndex:= LogListBox.Items.Count-1;
end;
// Нумерація хостів
procedure TForm1.EnumerateHostDevices(hHost : THandle);
var Success : LongBool; DeviceName : TDeviceName;
BytesReturned : Cardinal;
begin
// Одержуємо ім'я хост контролера (IOCTL_GET_HCD_DRIVERKEY_NAME)
 Success:= DeviceIoControl(hHost, GetUSBCtlCode(10),
             @DeviceName, SizeOf(DeviceName),
             @DeviceName, sizeof(DeviceName),
             BytesReturned, nil);
 If not(Success) then begin
  Log(' Помилка одержання ім'я контролера');
```

```
  Exit;
 End;
Log(Format('Ім'я контролера:%s', [WideCharToString(@DeviceName.Name)]));
 // Одержуємо ім'я кореневого хаба (IOCTL_USB_GET_ROOT_HUB_NAME)
 Success:= DeviceIoControl(hHost, GetUSBCtlCode(3),
             @DeviceName, SizeOf(DeviceName),
             @DeviceName, sizeof(DeviceName),
             BytesReturned, nil
           );
 If not(Success) then begin
  Log(' Помилка одержання ім'я хаба');
  Exit;
 End;
 Log(Format(' Ім'я кореневого хаба: %s',[WideCharToString(@DeviceName.Name)]));
 // відображення інформації про кореневий хабе
 ShowRootHubInfo(WideCharToString(@DeviceName.Name));
end;
```

Програмне забезпечення системи USB на хосту керує взаємодіями між пристроями USB і хост-основаним програмним забезпеченням пристрою. Маються п'ять областей взаємодії між програмним забезпеченням системи USB і програмним забезпеченням пристрою, вони наступні:

– перенумерація пристроїв і конфігурація;

– ізохронні передачі даних;

– асинхронні передачі даних;

– керування живленням;

– інформація про пристрій і керування шини.

```
procedure TForm1.ShowRootHubInfo(RootHubName : String);
var sRoot:String; hRoot:THandle; SA:SECURITY_ATTRIBUTES;
begin
 // Формування SA
 SA.nLength:= SizeOf(SECURITY_ATTRIBUTES);
 SA.lpSecurityDescriptor:= nil;
 SA.bInheritHandle:= False;
 // Одержуємо повне ім'я кореневого хаба з Pn-ім'я
 sRoot:= Format('\\.\ %s', [RootHubName]);
 // Пробуємо відкрити
 hRoot:= CreateFile(PChar(@sRoot[1]),
 GENERICWRITE,FILE_SHARE_WRITE,@SA,OPEN_EXISTING,0,0);
```

```
 If (hRoot <> INVALID_HANDLE_VALUE) then begin
 // Відображення інформації про кореневий хаб
  ShowRootHubDetail(hRoot);
  CloseHandle(hRoot);
 End else
begin
  Log(' Помилка одержання інформації від хаба');
 End;
end;
// Відображення інформації про кореневий хаб
procedure TForm1.ShowRootHubDetail(hRoot : THandle);
var Success : LongBool; NodeInformation : TNodeInformation;
    BytesReturned : Cardinal; iPort : Byte;
begin
 // Одержання інформації про кореневий хаб (IOCTL_USB_GET_NODE_INFORMATION)
 Success:= DeviceIoControl(hRoot, GetUSBCtlCode(3),
            @NodeInformation, sizeof(NodeInformation),
            @NodeInformation, sizeof(NodeInformation),
            BytesReturned, nil);
 If not(Success) then begin
  Log('Помилка одержання інформації про хаб');
  Exit;
 End;
 // Відображення інфромації про порти хаба
 Log(Format('  Число портів: %d',
[NodeInformation.HubDescriptor.bNumberOfPorts]));
 For iPort:= 0 to NodeInformation.HubDescriptor.bNumberOfPorts-1 do begin
// Відображення інформації про порт
ShowHubPortDetail(hRoot, iPort);
End;
end;
// Відображення інформації про порт
procedure TForm1.ShowHubPortDetail(hRoot : THandle; iPort : Integer);
var Success : LongBool; NodeConnInfo : TNodeConnectionInformation;
    BytesReturned : Cardinal; PortStatus : Byte;
begin
 NodeConnInfo.ConnectionIndex:= iPort+1; // нумерація з 1!
 // IOCTL_USB_GET_NODE_CONNECTION_INFORMATION
 Success:= DeviceIoControl(hRoot, GetUSBCtlCode(4),
            @NodeConnInfo, sizeof(NodeConnInfo),
            @NodeConnInfo, sizeof(NodeConnInfo),
            BytesReturned, nil
```

```
          );
 If not(Success) then begin
  Log('   Помилка одержання інформації про порт');
  Exit;
 End;
 PortStatus:= NodeConnInfo.ConnectionStatus[1];
 If (PortStatus <> Byte(DeviceConnected)) then begin
  If (PortStatus = Byte(NoDeviceConnected)) then begin
End else begin
   Log(Format('   Стан порту %d = %d',[iPort, Byte(PortStatus)]));
  End;
 End else begin
  If not(NodeConnInfo.DeviceIsHub) then begin
   Log(Format('   До порту %d підключений пристрій', [iPort]));
   ShowDeviceDetail(hRoot, iPort);
  End else begin
   Log(Format('   До порту %d підключений хаб', [iPort]));
   // послати IOCTL_USB_GET_NODE_CONNECTION_NAME
   // одержати дескриптор хаба й повторити операцію для
   // кожного порту цього хаба
   // рекурсія: ShowRootHubDetail(hRoot : THandle);
  End;
 End;
end;
// Інформація про підключений пристрій (верстаті)
procedure TForm1.ShowDeviceDetail(hRoot : THandle; iPort : Integer);
Var Success : LongBool; Packet : TDescriptorRequest;
    BytesReturned : Cardinal; IsHID : Boolean;
begin
// Одержання стандартного дескриптора пристрою (верстата)
 ZeroMemory(@Packet, SizeOf(Packet));
 Packet.ConnectionIndex       := iPort+1;
 Packet.SetupPacket.bmRequest := $80;
 Packet.SetupPacket.bRequest  := USB_REQUEST_GET_DESCRIPTOR;
 Packet.SetupPacket.wValue [2]:=USB_DEVICE_DESCRIPTOR_TYPE;
// Використовувати буфер 2 Кб
 Packet.SetupPacket.wLength[2]:= 1;
 // IOCTL_USB_GET_DESCRIPTOR_FROM_NODE_CONNECTION
 Success:= DeviceIoControl(hRoot, GetUSBCtlCode(5),
            @Packet, sizeof(Packet),
            @Packet, sizeof(Packet),
            BytesReturned, nil);
```

```
 If not(Success) then begin
Log(Format('Помилка одержання інформації про пристрій %s',
[SysErrorMessage(GetLastError())]));
  Exit;
 End;
 IsHid:= False;
 // відображення дескриптора пристрою (верстата)
 DisplayDescriptorInfo(Packet.Data, IsHid);
// Одержання дескрипторів конфігурації
 ZeroMemory(@Packet, SizeOf(Packet));
 Packet.ConnectionIndex:= iPort+1;
Packet.SetupPacket.bmRequest := $80;
Packet.SetupPacket.bRequest  :=USB_REQUEST_GET_DESCRIPTOR;
Packet.SetupPacket.wValue[2]:=USB_CONFIGURATION_DESCRIPTOR_TYPE;
// Використовувати буфер 2 Кб
Packet.SetupPacket.wLength[2]:= 1;
// IOCTL_USB_GET_DESCRIPTOR_FROM_NODE_CONNECTION
Success:= DeviceIoControl(hRoot, GetUSBCtlCode(5),
            @Packet, sizeof(Packet),
            @Packet, sizeof(Packet),
            BytesReturned, nil
           );
 If not(Success) then begin
Log(Format('Помилка одержання інформації про пристрій %s', [
SysErrorMessage(GetLastError())]));
  Exit;
 End;
 // відображення інформації про дескриптори конфігурації
 DisplayDescriptorInfo(Packet.Data, IsHid);
 If IsHid then begin
    // Одержання дескрипторів REPORT
     ZeroMemory(@Packet, SizeOf(Packet));
     Packet.ConnectionIndex       := iPort+1;
     Packet.SetupPacket.bmRequest := $80;
Packet.SetupPacket.bRequest  :=USB_REQUEST_GET_DESCRIPTOR;
Packet.SetupPacket.wValue [2]:=USB_REPORT_DESCRIPTOR_TYPE;
Packet.SetupPacket.wLength[2]:=1;// Використовувати буфер 2 Кб
     // IOCTL_USB_GET_DESCRIPTOR_FROM_NODE_CONNECTION
     Success:= DeviceIoControl(hRoot, GetUSBCtlCode(5),
                @Packet, sizeof(Packet),
                @Packet, sizeof(Packet),
                BytesReturned, nil);
```

```
     If not(Success) then begin
Log(Format('Помилка одержання REPORT-Дескриптора %s',[
SysErrorMessage(GetLastError())]));
      Exit;
     End;
    // відображення інформації про дескриптори конфігурації
    DisplayREPORTDescriptorInfo(Packet.Data);
 End;
end;
```

Усі передачі транзакций шини включають до трьох пакетів. Кожна транзакция починається, коли хост контролер, за розкладом, посилає USB пакет, що описує тип і напрямок транзакции, адресу пристрою USB, і номер кінцевої точки. Цей пакет згадується як Маркерний Пакет (Token Paket). Пристрій USB, до якого адресоване повідомлення самостійно ідентифікується, декодуя відповідні поля адреси. У даній транзакції, дані передані або з хоста на пристрій або із пристрою на хост. Напрямок передачі даних визначено в маркерному пакеті. Джерело транзакції потім посилає Пакет Даних (Data Paket) чи повідомляє, що більше немає ніяких даних для передачі. Адресат у загальному випадку посилає Пакет Квитирування (Handshake Paket), що показує, чи була передача успішною.

```
procedure TForm1.Button1Click(Sender: TObject);
var sHost : String; iHost : Integer; hHost : THandle;
    SA : SECURITY_ATTRIBUTES; EndOfSearch : Boolean;
begin
// Формування SECURITY_ATTRIBUTES
 SA.nLength:= SizeOf(SECURITY_ATTRIBUTES);
 SA.lpSecurityDescriptor:= nil;
 SA.bInheritHandle:= False;
 // Цикл по всім хостам
 iHost:= 0;
 EndOfSearch:= False;
 Repeat
  // Формування ім'я хоста
  sHost:= Format('\\.\HCD%d', [iHost]);
   Log(Format('Перевірка хоста <%s>...',[sHost]));
  // Пробуємо відкрити хост-контролери
hHost:= CreateFile(PChar(@sHost[1]),
GENERIC_WRITE,FILE_SHARE_WRITE,@SA,OPEN_EXISTING,0,0);
  // Хост успішно відкритий
```

```
  If (hHost <> INVALID_HANDLE_VALUE) then begin
Log(Format(' Хост-Контролер %d успішно відкритий',[iHost]));
// Пошук пристроїв, підключених до хосту
   EnumerateHostDevices(hHost);
   CloseHandle(hHost);
  End else begin
   Log(Format(' Хост-контролер <%s> не знайдений',[sHost]));
   EndOfSearch:= True;
  End;
  Inc(iHost);
 Until EndOfSearch;
end;
procedure TForm1.DisplayREPORTDescriptorInfo(Data : Array of byte);
begin
 Log('Дескриптор REPORT');
end;
procedure TForm1.Button2Click(Sender: TObject);
 var Success : LongBool; DeviceName : TDeviceName;
    BytesReturned : Cardinal; hHost : THandle;
begin
 // Одержуємо ім'я хост контролера (IOCTL_GET_HCD_DRIVERKEY_NAME)
 Success:= DeviceIoControl(hHost, GetUSBCtlCode(10),
             @DeviceName, SizeOf(DeviceName),
             @DeviceName, sizeof(DeviceName),
             BytesReturned, nil);
 If not(Success) then begin
  Log('Помилка одержання ім'я контролера');
  Exit;
 End;
Log(Format('Ім'я контролера:%s', [WideCharToString(@DeviceName.Name)]));
 // Одержуємо ім'я кореневого хаба
Success:= DeviceIoControl(hHost, GetUSBCtlCode(3),
             @DeviceName, SizeOf(DeviceName),
             @DeviceName, sizeof(DeviceName),
             BytesReturned, nil);
 If not(Success) then begin
  Log(' Помилка одержання ім'я хаба');
  Exit;
 End;
// відображення інформації про кореневий хаб
 ShowRootHubInfo(WideCharToString(@DeviceName.Name));
end;
end.
```

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм SHACAL-2, який шифрує дані 256-бітними блоками з використанням 512-бітного ключа. Допускається використання ключів менших розмірів (не менш 128 біт), які доповнюються бітовими нулями до 512 біт.

Шифруємий блок даних ділиться на 8 фрагментів по 32 біта (які позначені буквами *A*...*H*). Алгоритм виконує 64 раунду перетворень, у кожному з яких дані фрагменти обробляються в такий спосіб:

$$T = H_i + S_1(E_i) + Ch(E_i, F_i, G_i) + M_i + K_i,$$

$$H_{i+1} = G_i,$$

$$G_{i+1} = F_i,$$

$$F_{i+1} = E_i,$$

$$E_{i+1} = D_i + T,$$

$$D_{i+1} = C_i,$$

$$C_{i+1} = B_i,$$

$$B_{i+1} = A_i,$$

$$A_{i+1} = T + S_0(A_i) + Maj(A_i, B_i, C_i).$$

де T – тимчасова змінна.

Використовувані функції визначені в такий спосіб:

$$S_0(x) = (x >>> 2) \oplus (x >>> 13) \oplus (x >>> 22),$$

$$S_1(x) = (x >>> 6) \oplus (x >>> 11) \oplus (x >>> 25),$$

$$Ch(x, y, z) = (x \,\&\, y) \oplus (x' \,\&\, z),$$

$$Maj(x, y, z) = (x \,\&\, y) \oplus (x \,\&\, z) \oplus (y \,\&\, z),$$

де >>> – операція побітового циклічного зрушення вправо.

Константи, що модифікують, M_i ($i = 0 \ldots 63$) наведено нижче (одна за одною від M_0 до M_{63}):

428A2F98	71374491	B5C0FBCF	E9B5DBA5
3956C25B	59F111F1	923F82A4	AB1C5ED5

D807AA98	12835B01	243185BE	550C7DC3
72BE5D74	80DEB1FE	9BDC06A7	C19BF174
E49B69C1	EFBE4786	0FC19DC6	240CA1CC
2DE92C6F	4A7484AA	5CB0A9DC	76F988DA
983E5152	A831C66D	B00327C8	BF597FC7
C6E00BF3	D5A79147	06CA6351	14292967
27B70A85	2E1B2138	4D2C6DFC	53380D13
650A7354	766A0ABB	81C2C92E	92722C85
A2BFE8A1	A81A664B	C24B8B70	C76C51A3
D192E819	D6990624	F40E3585	106AA070
19A4C116	1E376C08	2748774C	34B0BCB5
391C0CB3	4ED8AA4A	5B9CCA4F	682E6FF3
748F82EE	78A5636F	84C87814	8CC70208
90BEFFFA	A4506CEB	BEF9A3F7	C67178F2

Фрагменти розширеного ключа $K_0 \ldots K_{63}$ обчислюються в процесі процедури розширення ключа, що виконується в такий спосіб:

Етап 1. 512-бітний вихідний ключ шифрування ділиться на 16 фрагментів по 32 біта $K_0 \ldots K_{15}$...

Етап 2. Інші фрагменти розширеного ключа $K_{16} \ldots K_{63}$ обчислюються з перших 16 фрагментів у такий спосіб:

$$K_i = O_1(K_{i-2}) + K_{i-7} + O_0(K_{i-15}) + K_{i-16},$$

де функції O_0 і O_1 визначені так:

$$O_0(x) = (x >>> 7) \oplus (x >>> 18) \oplus (x >> 3),$$

$$O_1(x) = (x >>> 17) \oplus (x >>> 19) \oplus (x >> 10),$$

де >> – операція побітового зрушення (не циклічного) вправо.

Раунди розшифрування алгоритму виконуються у зворотній послідовності:

$$T = A_{i+1} + S_0'(B_{i+1}) + Maj'(B_{i+1}, C_{i+1}, D_{i+1}) + 2,$$

$$H_i = T + S_1'(F_{i+1}) + Ch'(F_{i+1}, G_{i+1}, H_{i+1}) + M_i' + K_i' + 4,$$

$$G_i = H_{i+1},$$

$$F_i = G_{i+1},$$

$$E_i = F_{i+1},$$

$$D_i = E_{i+1} + T' + 1,$$

$$C_i = D_{i+1},$$

$$B_i = C_{i+1},$$

$$A_i = B_{i+1}.$$

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 наведено вигляд головної форми програми розробленої у результаті виконання магістерської роботи.

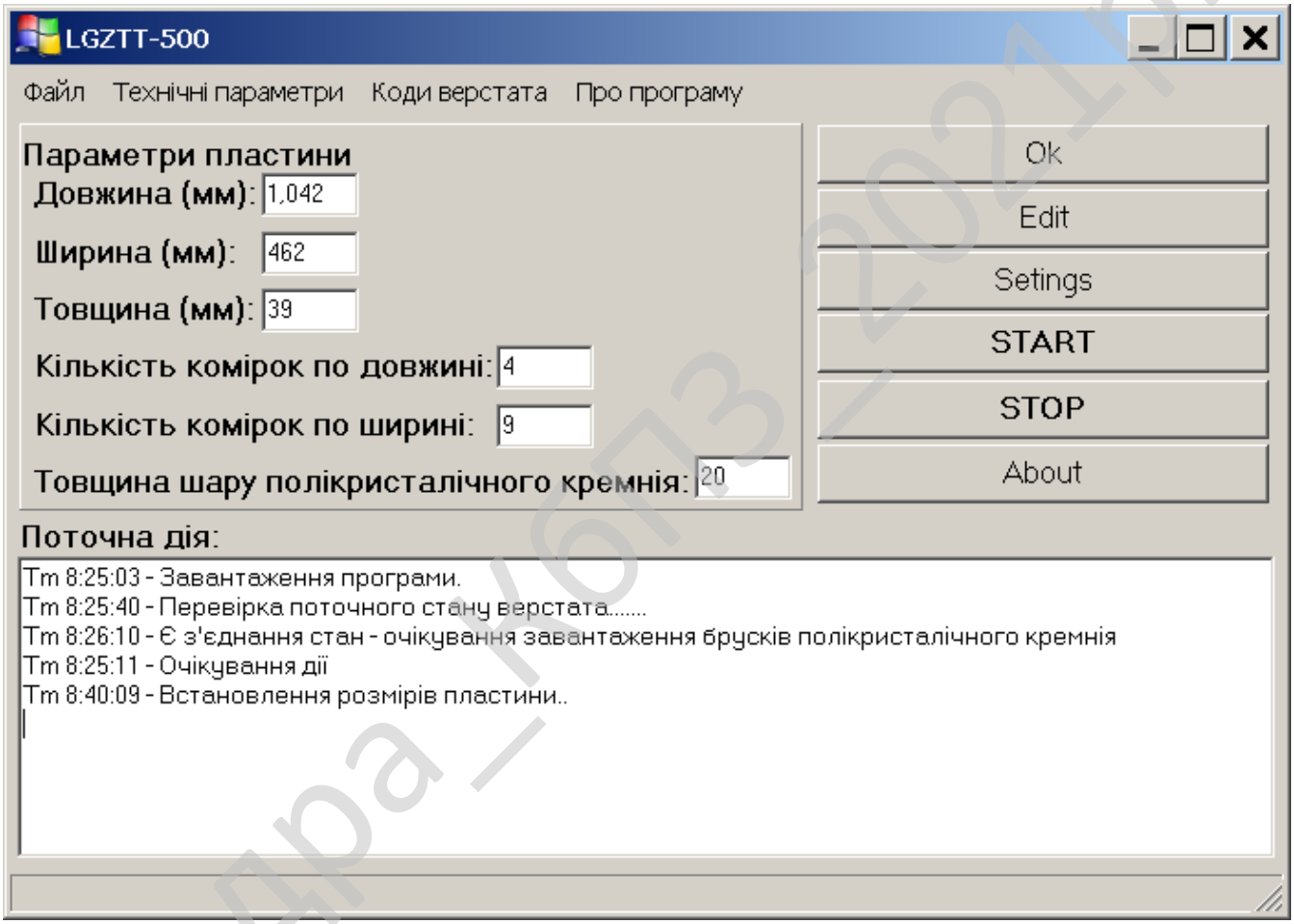

Рисунок 5.1 – Головне вікно програми

Після завантаження основного вікна програми, відбувається тестування усі блоків. Після цього при наявності усього справного обладнання у вікні вводяться параметри елемента сонячної батареї. Після їх введення починається технологічний процес перетворення сировини у вихідну продукцію.

На рисунках 5.2 – 5.6 наведена реакція розробленої автоматизованої системи на різні типи помилок.

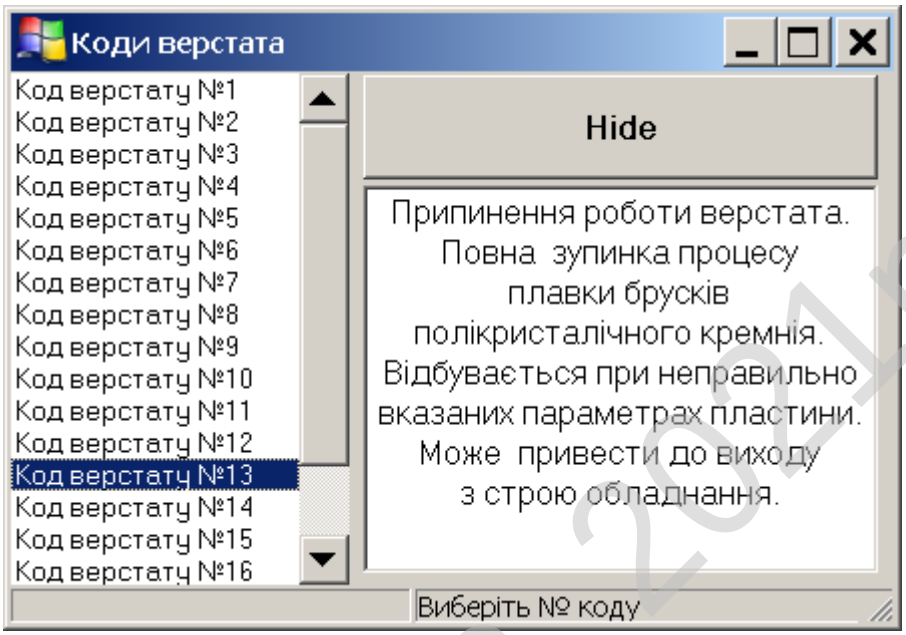

Рисунок 5.2 – Вікно кодів верстата

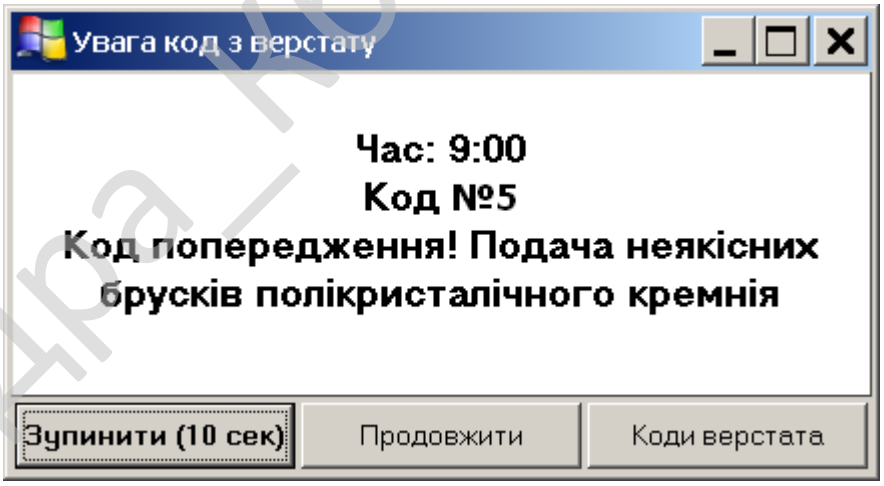

Рисунок 5.3 – Вікно попередження, код №5

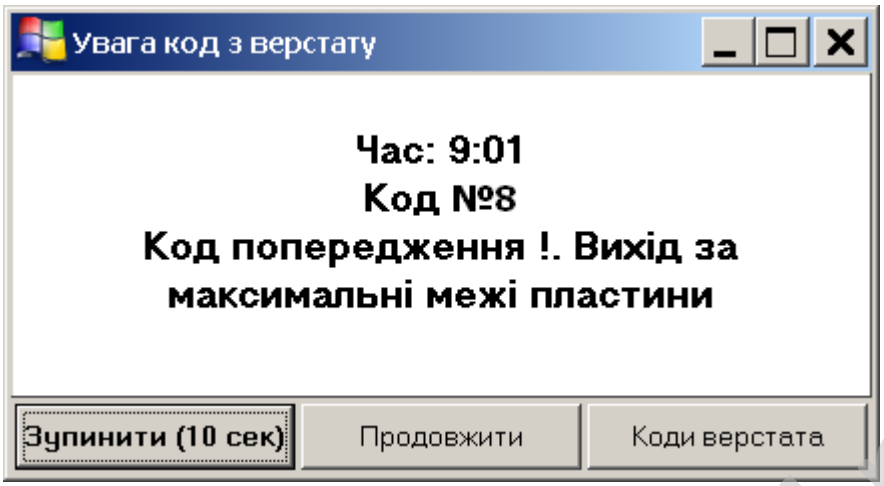

Рисунок 5.4 – Вікно попередження, код №8

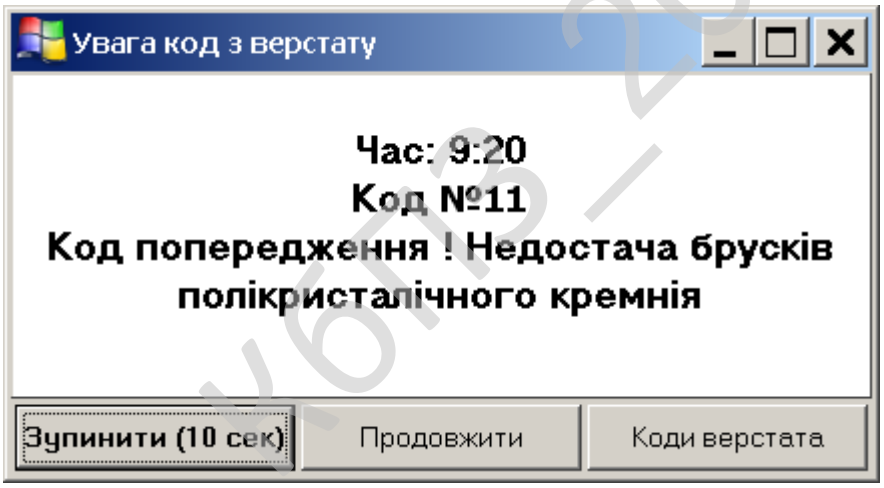

Рисунок 5.5 – Вікно попередження, код №11

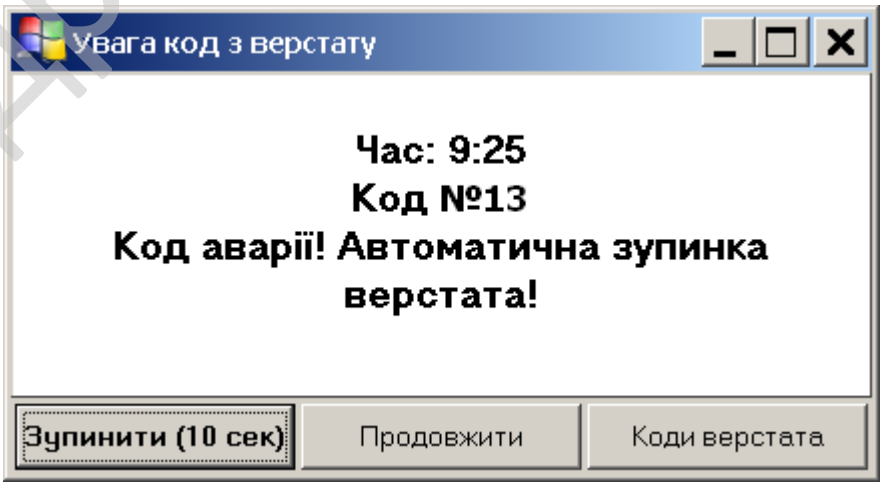

Рисунок 5.6 – Вікно попередження, код №13

На рисунку 5.7 зображено вікно інформації, про розроблювача системи, наукового керівника та місце впровадження автоматизованої системи управління.

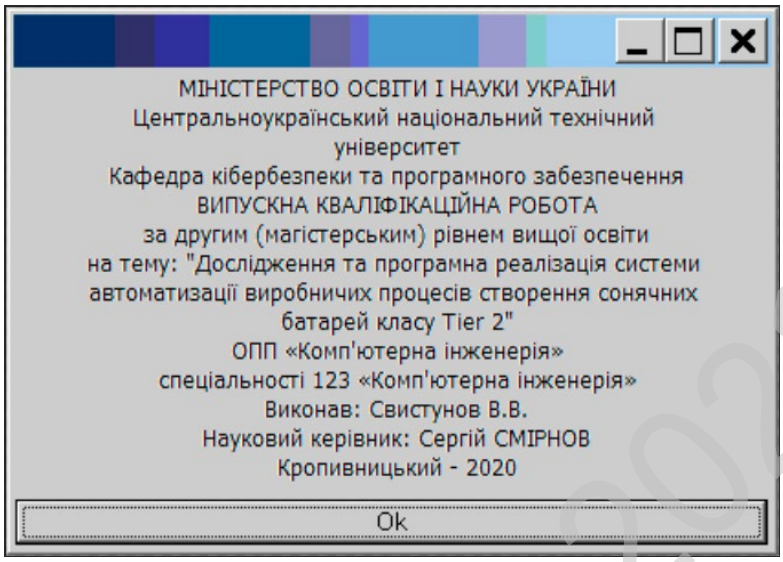

Рисунок 5.7 – Дані розробника

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи автоматизації виробничих процесів створення сонячних батарей класу Tier 2.

Метою розробки є дослідження та програмна реалізація системи автоматизації виробничих процесів створення сонячних батарей класу Tier 2.

Об'єктом дослідження є процес автоматизації виробничих процесів створення сонячних батарей класу Tier 2.

Предметом дослідження є методи автоматизації виробничих процесів створення сонячних батарей класу Tier 2.

Методи дослідження базуються на методах теорії автоматизованого управління, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод автоматизації виробничих процесів створення сонячних батарей класу Tier 2.

– Розроблено вітчизняний продукт автоматизації виробничих процесів створення сонячних батарей класу Tier 2, який має більш широкі можливості, на відміну від існуючих аналогів.

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 було проведено дослідження та виконана програмна реалізація системи автоматизації виробничих процесів створення сонячних батарей класу Tier 2. Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір;

б) невеликі системні потреби;

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

г) 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	17 (2 ост. цифри № зал)
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

Продовження таблиці 7.1

1	2	3
7. Кількість макетів вхідної інформації	–	7
8. Кількість форм вихідної інформації.	–	8
9. Мова програмування (1-6)	–	1
10. Попередній досвід (1-6)	–	2
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	–	2
13. Рівень спрацьованості колективу (1-6)	–	1
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)	–	3
17. Складність кінцевого програмного продукту (1-6)	–	2
18. Необхідний рівень забезпечення повторного використання (1-6)	–	1
19. Документованість відповідно до планованого життєвого циклу (1-6)	–	1
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)	–	1
23. Професійний рівень аналітиків (1-6)	–	2
24. Професійний рівень програмістів (1-6)	–	3
25. Постійність складу команди розробників (1-6)	–	2
26. Досвід розробки додатків (1-6)	–	3
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)	–	1
29. Досвід роботи з програмними інструментами розробки (1-6)	–	2
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	3
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	17000
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 визначає мо 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.:

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

де: A – коефіцієнт Боема, A = 2,45;

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

B – показник ступеня, що визначається співвідношенням:

$$B = 1{,}01 + 0{,}001\sum W_i, \qquad (7.2)$$

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

$$B = 1{,}01+0{,}001(3{,}24+3{,}64+3{,}38+4{,}94+2{,}73) = 1{,}028.$$

$$T_{ном} = 2{,}45 \cdot 2{,}7^{1{,}028} = 6{,}8 \text{ люд-міс.}$$

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

$$T_{уточн} = T_{ном}\, ПВ_j, \qquad (7.3)$$

де: $ПВ_j$ – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн} = 6{,}8 \cdot (0{,}88 \cdot 1 \cdot 0{,}88 \cdot 0{,}91 \cdot 0{,}89 \cdot 1 \cdot 1 \cdot 0{,}87 \cdot 1{,}22 \cdot 1 \cdot 1{,}1 \cdot 1 \cdot$$

$$\cdot 1{,}12 \cdot 1{,}22 \cdot 1{,}12 \cdot 1 \cdot 1{,}1) = 8{,}38 \text{ люд-міс.}$$

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:

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

де: *C* – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4). Для мови програмування Builder C++ – 3,23;

S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПЗ згідно встановленим вимогам. Вибираємо в межах (25…350)%. Приймаємо *S* = 120 %

$$T_{РП} = 0{,}3 \cdot 3{,}23 \cdot 8{,}38^{0{,}33+0{,}2(1{,}028-1{,}01)} \cdot 100 = 197 \text{ люд/день.}$$

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	15	Д7
Робочий проект	197	Ф 7.1-7.4
Впровадження	17	Д13
Всього	248	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

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

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

де: $F_{рq}$ – плановий фонд робочого часу одного спеціаліста, днів;

$T_{пз}$ – трудомісткість розробки програмного забезпечення люд-дні.

$$Ч = \frac{248 \cdot 1}{48\text{-}5} = 5,7 \text{ ставки.}$$

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	385	12	4620	77
Монітор	160	12	1920	32
Клавіатура	140	12	1680	28
Маніпулятор «мишка»	30	12	360	6
Принтер матричний	185	1	185	3
Принтер лазерний	355	2	710	12
Принтер струминний	300	1	300	5
Сканер	155	2	310	5
Концентратор-маршрутизатор	155	2	310	5
Кабельні господарства ЛОМ на 1 м. п.	2,5	100	250	4
Кабельне господарство електромережі	48	50	2400	40
Копіювальний апарат	285	2	570	10
Усього за рік:			$З_ч$	227

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

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

$$Ф_{др}^{с} = \frac{227 \cdot 2}{1,2} = 378,3 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

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

$Ч_{ел}$ = 378,3/(48·8) = 1 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 2019, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN, PPPoE, Frame Relay, Wi-Fi	2	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

Посада	Вид роботи	Час	К-ть штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	1	0,25
	Підтримка постійних клієнтів	0,5	
	Оформлення договорів, ведення тендерів	0,25	
	Контроль взаєморозрахунків з постачальниками	0,25	
Всього		2	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	1	0,25
	Створення графічних і стилістичних елементів сайту	0,5	
	Оформлення банерів і промо-сторінок	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,25
	Верстка друкованих видань	0,5	
	Додрукова підготовка макетів	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	1	13000	26000
Продакт-менеджер	0,25	12000	6000
Інженер-програміст	5,7	12000	136800
Інженер-електронщик	1	7500	15000
Інженер-системотехнік	0,25	6500	3250
Адміністратор мережі	0,5	6500	6500
Системний програміст	0,25	6500	3250
Дизайнер WEB	0,25	7000	3500
Інженер-верстальник	0,25	6700	3350
Бухгалтер-економіст	0,25	12500	6250
Всього за період розробки	$R_{сп} = 9,7$	-	$Ф_{роб} = 209900$

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рд}}, \quad (7.8)$$

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

$$З_{сд} = \frac{209900}{9,7 \cdot 48} = 451 \, грн.$$

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 ТОВ науково-дослідницького консалтингового підприємства «Пекторaль» (м. Кіровоград, вул. Глинки 16) ціна одного квадратного метра площі новобудови, вік якої не перевищує 25 років, по місту складає 800…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 25 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:

$$Б_{уд} = 9 \cdot 8 \cdot 20000 = 1440000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 144000 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: $Ц_м$ – ціна меблів для одного робочого місця, грн.

$$І_{нв} = 9 \cdot 3500 = 31500 \text{ грн.}$$

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 supercomp.kiev.ua за 29.10.21 – джерело http: //supercomp.kiev.ua.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		12721
Системний блок		7721
Процесор	AMD AM4 A8-9600 (AD9600AGM44AB box, TDP процесора складає 15 Вт, він працює з відеокартою Radeon R7 (384 шейдерних ядер / 6 обчислювальних ядер) двоканальним контролером пам'яті DDR4 18600	2000
Системна плата	GIGABYTE B450M S2H V2, 1 x PCI-E 2. x1, 1 x PCI -E 3.0 x16, 4 x USB 3.1 G en1, 1 VGA, 1 x RJ45, 2 x USB 2.0, 2 x PS/2, 1 x DVI-D, 1 x HDMI, 1 x optical SPDIF out, 3 x Audio, M.2 2280, 4 x Sata 6.0 Gb/s	1350
Відеокарта	Інтегрована AMD Radeon R7	-
Жорсткий диск	HDD 500 Gb SAMSUNG Barracud HD502HJ (3.5", 500ГБ, 16МБ)	1490
Оперативна пам’ять	DDR4 8GB 2666 MHZ KINGSTO (KVR26N19S8/8)	834
DVD-привод	DVD±RW ASUS DRW-24B5ST Black Bulk	416
Корпус	Logicpower 8702 - 550w 12cm	1411
Кардрідер внутрішній	Transcend TS-RDF8K USB 3.0	220
інше	Клавіатура, мишка	Подаруно

Продовження таблиці 7.5

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Монітор	Монітор BenQ GL2450HM Black	3600
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 безперебійного 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	9	12721	11448,9	125937,9
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	–	–	–	147569,4

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	1440000	-	-
2. Передавальні пристрої	144000	-	-
Всього по групі	1584000	5	79200
Група 4			
3. Обчислювальна техніка	147570	-	-
Всього по групі	147570	40	59028
Нематеріальні активи			
4. Нематеріальні активи	17000	10	1700
Група 5, 6			
5. Вимірювальні пристрої	9031	25	2257,75
6. Транспортні засоби	121875	20	24375
7. Господарський інвентар	31500	25	7875
Всього по групі	162406	-	34507,75
Разом	$К_р$ = 1910976		$А_р$ = 174435,75

Примітка: вартість автомобіля ГАЗ Газель взята по даним електронного ресурсу http://www.avtopoisk.ua, що враховуючи курс 25 складає 121875 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.

$$З_о = 451 \cdot 248/17 = 6579 \text{ грн.}$$

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, %.

$$З_д = 6579 \cdot 10 \cdot 0{,}01 = 658 \text{ грн.}$$

Відрахування на соціальні потреби за нормативом $Н_с = 37\%$ 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, %.

$$С_{оц} = 0{,}01 \cdot 37(6579+658) = 2678 \text{ грн.}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом $Н_г = 15\%$ від основної зарплати:

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

де: $Н_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 6579 \cdot 15 \cdot 0{,}01 = 987 \text{ грн.}$$

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 виданих викладачем норм приймаємо 0,5 пачки паперу на місяць розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п = 70$ грн., визначаємо вартість паперу за період розробки $N_м = 2$ міс:

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

$$З_{М1} = 0{,}5 \cdot 70 \cdot 2 = 70 \text{ грн.}$$

Згідно виданих викладачем норм до вартості запам'ятовуючих пристроїв входить вартість CD дисків в кількості, що дорівнює кількості екземплярів програм та одного DVD диска для збереження резервної копії програми:

$$З_{М2} = \sum Ц_д\ , \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR TDK 700Mb, 80Min, 52x Cake box – 2 грн./шт., DVD-R LG 4,7Gb, 16x speed Cake box – 2 грн./шт.

$$З_{М2} = 17 \cdot 2 + 2 = 36 \text{ грн.}$$

Згідно виданих викладачем 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.

$$З_{М3} = 500 + 508 + 1155 = 2163 \text{ грн.}$$

$$З_М = (70 + 36 + 2163)/17 = 133 \text{ грн.}$$

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, %.

$$О_п = 6579 \cdot 15 \cdot 0{,}01 = 987 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е = 17$ прим.):

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

де: $A_р$ – загальна річна сума амортизаційних відрахувань, грн.

$$A_м = 174436 \cdot 2/(17 \cdot 12) = 1710 \text{ грн.}$$

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	$З_о$	6579
2. Додаткова зарплата виконавців	$З_д$	658
3. Відрахування на соціальні потреби	$С_{оц}$	2678
4. Загальногосподарські витрати	$Г_{осп}$	987
5. Витрати на матеріали	$З_М$	133
6. Освоєння нових операційних систем, мов програмування	$О_п$	987
7. Амортизація основних фондів	$А_м$	1710
8. Повна собівартість програмного забезпечення	$С_п$	13732
9. Плановий прибуток	$П_р$	6866
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	20598
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	4119,6
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	24717,6

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції:

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

$$C_п = 6579+658+2678+987+133+987+1710 = 13732 \text{ грн.}$$

Визначимо плановий прибуток за рівнем рентабельності ($P_п$) 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: $P_п$ – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 50 \cdot 13732 = 6866 \text{ грн.}$$

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.

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

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	24718
Всього капітальних витрат	–	24718

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	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на обслуговування	$З_р$	173923	57974
2. Витрати на електроенергію	$З_{ел}$	562	389
3. Витрати на амортизацію	$З_{ам}$	0	6180
Всього витрат за рік	$І$	174485	64543

Витрати на обслуговування системи:

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

де: $Т_р$ – кількість годин обслуговування за рік, *год.*;

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

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 240 годин на рік до 80 годин на рік, тому витрати на технічне обслуговування зменшилися з:

$$З_{р\,баз} = 240 \cdot 60 \cdot 1{,}1 \cdot 1{,}22 \cdot 9 = 173923 \text{ грн},$$

до:

$$З_{р\,нов} = 80 \cdot 60 \cdot 1{,}1 \cdot 1{,}22 \cdot 9 = 57974 \text{ грн}.$$

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

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

$$З_{ел\,баз} = 9 \cdot 0{,}15 \cdot 130 \cdot 3{,}2 = 562 \text{ грн}.$$

$$З_{ел\,нов} = 9 \cdot 0{,}15 \cdot 90 \cdot 3{,}2 = 389 \text{ грн}.$$

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	25	–	24718	–	6179,5
Всього відрахувань	-	–	24718	–	6179,5

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:

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{p_m} \cdot K_{p_m}, \quad (7.24)$$

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

$$Ев = (20598 - 13732) \cdot 17 - (0,05 \cdot 1584000 + 0,4 \cdot 147570 + \\ + 0,2 \cdot 121875 + 0,25 \cdot 40531 + 0,1 \cdot 17000) \cdot 2/12 = 87649 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_p^*}{(Ц_п - C_п) \cdot N_е}, \quad (7.25)$$

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.

$$T_в = \frac{326976}{(20598 - 13732) \cdot 17 \cdot 12 / 2} = 0{,}5 \ років.$$

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.	17
2. Повна собівартість розробленої програми	Грн.	13732
3. Ціна розробленої програми	Грн.	20589
4. Плановий прибуток від реалізації розробленої програми	Грн.	6857
5. Рентабельність програмної продукції	%	50
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1910976
7. Загальний прибуток від реалізації програмної продукції	Грн.	116569
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	87649
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	0,5
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	24718
11.Величина економічного ефекту у користувача програмної продукції	Грн.	103763
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,2

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.

$$E_{сп} = (174485\text{-}64543)\text{-}0{,}25 \cdot 24718 = 103763 \text{ грн.}$$

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

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

$$T_{сп} = \frac{24718}{174485 - 64543} = 0{,}2 \ року.$$

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

Протягом усієї історії людство приділяє прискіпливу увагу безпеці життя. Охорона праці є складовою частиною безпеки життя [1].

Законом України “Про охорону праці” [2] 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», НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин».

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

Ці шкідливі фактори можуть привести до професійних захворювань.

При розгляді шкідливих чинників роботи програмістів та інших спеціалістів ІТ будемо керуватись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98, та

«Правила охорони праці під час експлуатації електронно-обчислювальних машин» НПАОП 0.00-1.28-10,

Умови праці програміста включають наступні фактори:

– параметри повітряного середовища в приміщенні;

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

Щоб запропонувати заходи щодо зменшення впливу комп'ютера на організм програміста визначимо фактори, які можуть викликати професійне захворювання і впливають на працездатність програміста,

8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

Програміст працює з електронно-обчислювальною машиною (ЕОМ) та іншим обладнанням, яке є джерелом небезпеки ураження електричним струмом. Так як робота програміста характеризується істотним зоровим навантаженням, то вимагає належного освітлення. Так як програміст постійно перебуває в приміщенні, тому для комфортних умов праці в цьому приміщенні необхідно створити належний мікроклімат.

При роботі з використанням ЕОМ відзначають наступні небезпечні та шкідливі фактори:

– ризик виникнення надзвичайних ситуацій природного або штучного характеру на об'єкті або території.

– ризик виникнення пожежі;

– негативний вплив на органи зору людини;

– ризики ураження електричним струмом;

– недостатня, або надмірна освітленість робочого місця;

– монотонність праці;

– електромагнітні (у т.ч. високочастотні) випромінювання (коливання);

– несприятливі мікрокліматичні умови;

– нервово-емоційна напруженість праці;

– інтелектуальні навантаження;

– невідповідність ергономічних показників робочого місця діючим вимогам;

– шуми;

– статичні навантаження на кістково-м’язовий апарат;

8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

Розглянемо умови праці у приміщенні, в якому працюють програмісти. Геометричні розміри приміщення наведено у таблиці 8.1.

Таблиця 8.1 – Розміри приміщення*

Найменування	Значення, м
Ширина	9
Довжина	10
Висота	2,9

* у таблиці вказано загальні розміри поєднаного приміщення, де загалом працюють 13 осіб, а фактично у наявності є дві кімнати, розділених перестінком.

Таблиця 8.2 – Площа та обсяг приміщення, на одного працюючого*

Геометрична характеристика	Одиниця виміру	Нормативне значення*	Фактичне значення
Площа, S	$м^2$	не менше 6.0	6,92
Обсяг, V	$м^3$	не менше 20.0	20

* Згідно ДСанПіН 3.3.2.007-98 (Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин).

У зазначеному приміщенні працює 13 осіб. За даними, які наведено у табл. 8.1 та табл. 8.2, можна зробити висновок, що площа та об'єм приміщення у розрахунку на одно робоче місце програміста відповідають нормативним вимогам (Наказу Міністерства соціальної політики України № 207, від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями», ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» та НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин»).

Температура повітря в приміщенні визначається впливом температури зовнішнього повітря і тепловою енергією, яка виділяється всередині приміщення. Джерелами виділення теплоти в даному приміщенні є електроустаткування, освітлювальні прилади, а також люди. У світлий час доби джерелом надлишкового тепла є сонячна радіація. Згідно Постанови № 42 від 01.12.1999 Головного державного санітарного лікаря України, робота, яка виконується в даному приміщенні, відноситься до категорії Іа. В цьому випадку людина витрачає енергії до 120 ккал у годину. Вологість повітря у приміщенні визначається впливом багатьох факторів, серед яких: вологість атмосферного повітря, виділення вологи людьми (при диханні та випарами з поверхні шкіри).

Мікроклімат повітряного середовища в приміщенні характеризується запиленістю та загазованістю повітря. Мікроклімат приміщення визначається діючим на організм людини поєднанням, вологості, температури, швидкості руху повітря та інтенсивності теплового випромінювання. Аналіз мікроклімату складається з визначення зазначених вище факторів і порівняння результатів із встановленими нормами.

У таблиці 8.3 наведено оптимальні та фактичні значення параметрів мікроклімату як для категорії ваги робіт Іа, так і розглянутого приміщення. У

приміщеннях, де встановлено ЕОМ, рекомендується застосування тільки оптимальних значень показників мікроклімату.

Таблиця 8.3 – Оптимальні і фактичні значення параметрів мікроклімату

Пора року	Оптимальні для Іа			Фактичні		
	Температура, °С	Воло-гість,%	Швидкість повітря, м/с	Температура, °С	Воло-гість%	Швидкість повітря, м/с
Холодна	22-24	40-60	0,1	22-24	44-55	0,1
Тепла	23-25	50-70	0,1	23-24	40-78	0,13

Проведений аналіз показує, що показники мікроклімату в приміщенні відповідають установленим нормам. Штучне опалення застосовується у холодний період року.

В літню пору застосовується кондиціонер.

Для боротьби з пилом робляться регулярні провітрювання та вологі прибирання приміщенні.

У приміщенні знаходяться наступні джерела шуму: електродвигуни вентиляторів ЕОМ, принтер *Canon PIXMA G2411*.

Одним з найважливіших факторів, які впливають на ефективність трудової діяльності людини, та попереджають травматизм і професійні захворювання програмістів є освітлення на робочому місці.

Працю працівника, який постійно працює за комп'ютером, згідно ДБН В.2.5 – 28 – 2006 р 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 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 поєднаному, спільному

освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 лк. Крім того все поле зору повинне бути освітлено достатньо рівномірно – ця основна гігієнічна вимога. Так як яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими.

8.4 Розробка заходів з умов поліпшення охорони праці

Згідно аналізу умов праці в розглянутому приміщенні, ми одержали наступні результати:

– розмірі приміщення, у розрахунку на одному працюючого, відповідають нормативам;

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значень;

Таким чином можна припустити, що основною причиною можливого зниження працездатності програміста є психофізіологічний фактор, тому основна пропозиція буде така: дотримання позитивної психологічної атмосфери в колективі та регламентованого режиму праці та відпочинку, організація робочого місця з урахуванням ергономічних вимог.

Рекомендовані заходи: регулярні періодичні наочні огляди персоналом шляхів для евакуації людей із приміщення, відповідно до плану евакуації (який повинен розташовуватись на видному місці у приміщенні), включення до колективного договору мінімально можливого вмісту аптечок з обов’язковою наявністю масок-клапанів, або іншого спорядження для штучного дихання. Регулярна періодична перевірка параметрів заземлення та занулення (вимірювання електричного опору ланцюга).

8.5 Розрахункова частина

Проведемо розрахунок штучного освітлення за методом коефіцієнта використання світлового потоку для приміщення ширина якого складає 9 м, довжина 10 м, висота 2,9 м.

Для того, щоб визначити потрібну кількість світильників, які повинні забезпечити нормований рівень освітленості, визначимо світловий потік, що падає на робочу поверхню за формулою:

$$F=ESKZ/n,$$

де:

F – світловий потік, що розраховується, Лм;

E – нормована мінімальна освітленість, Лк; E = 300 Лк;

S – площа освітлюваного приміщення (у нашому випадку S= 9 х 10 = 90 м^2);

K – коефіцієнт запасу, що враховує зменшення світлового потоку лампи в результаті забруднення світильників в процесі експлуатації (його значення залежить від типу приміщення і характеру робіт, що проводяться в ньому, в нашому випадку K = 1,5);

Z – відношення середньої освітленості до мінімальної (зазвичай приймається рівним 1.1... 1.2, в нашому випадку Z = 1,1);

n – коефіцієнт використання світлового потоку, (відношення світлового потоку, що падає на розрахункову поверхню, до сумарного потоку всіх ламп і обчислюється в долях одиниці; залежить від характеристик світильника, розмірів приміщення, забарвлення стін і стелі, що характеризуються коефіцієнтами відбиття від стін ($\rho_{стін.}$) і стелі ($\rho_{стелі}$), значення коефіцієнтів дорівнюють $\rho_{стін}$ = 50% і $\rho_{стелі}$ = 50%.

Обчислимо індекс приміщення за формулою:

$$i=S/(h(A+B)),$$

де:

S – площа приміщення, S = 90 м^2;

h – розрахункова висота підвісу, h = 2,9 м (співпадає з висотою стелі, т.я.

світильники освітлення закріплюються на стелі);

A – ширина приміщення, A = 9 м;

B – довжина приміщення, B = 10 м.

Підставимо всі значення у формулу та визначимо індекс приміщення:

$$i=1{,}63.$$

Знаючи індекс приміщення (i), за знаходимо n = 0,55 (з таб личних даних коефіцієнтів використання світлового потоку (n) світильників [3]). Підставимо всі значення у формулу, визначемо світловий потік: F=81000 Лм.

Для штучного освітлення приміщення використовуються *LED* світильники *Videx 54W-5000K*, світловий потік яких $F_л$ = 5940 Лм.

Число світильників визначається по формулі:

$$N=F/F_л$$

де:

F – світловий потік,

$F_л$ – світловий потік однієї лампи.

Підставимо всі значення у формулу та визначимо індекс приміщення:

$$N= 81000/ 5940=13{,}6 \text{ шт.}$$

Для забезпечення нормованої мінімальної освітленості приймаємо необхідну кількість світильників 14 шт.

8.6 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. Виконано розрахунок штучного освітлення, як одного з ключових факторів впливу на працездатність та здоров'я програміста. Розроблено заходи з охорони праці.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи автоматизації виробничих процесів створення сонячних батарей класу Tier 2.

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

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів автоматизації виробничих процесів створення сонячних батарей класу Tier 2.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем автоматизації виробничих процесів створення сонячних батарей класу Tier 2.

– Досліджена система автоматизації виробничих процесів створення сонячних батарей класу Tier 2.

– На основі отриманих результатів досліджень створена програмна реалізація системи автоматизації виробничих процесів створення сонячних батарей класу Tier 2.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання автоматизації виробничих процесів створення сонячних батарей класу Tier 2.

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, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

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

Програма реалізована на мові високого рівня Delphi 10.4 Sydney. Дана мова програмування дозволяє найбільш ефективно обробляти дані. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows XP/Vista/7/8/10.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм SHACAL-2.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 103763 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,2 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Свистунов В.В. Дослідження та програмна реалізація системи автоматизації виробничих процесів створення сонячних батарей класу Tier 2 // Збірник праць молодих науковців ЦНТУ. – Вип. 12. – Кропивницький: ЦНТУ, 2022.

2. Коваленко А.С. Разработка структуры базы данных интегрированной информационной системы / А.С. Коваленко, А.В. Коваленко // Информационные технологии и защита информации в информационно-коммуникационных системах: монографія / Под редакцией профессора В.С. Пономаренко. – Х.: Вид-во ТОВ «Щедра садиба плюс», 2015. – С. 54-64.

3. Кожанова А.С. Обґрунтування необхідності створення систем технічної діагностики інтегрованих інформаційних систем / О.А. Смірнов, А.С. Кожанова, О.В. Коваленко // Системи обробки інформації. – Х.: ХУПС, 2013. – Вип. 6(113). – С. 255-257.

4. Коваленко А.С. Задачи распознавания ситуаций в ERP системах / А.В. Коваленко ., А.А. Смірнов, А.С. Коваленко // Системи обробки інформації. – Х.: ХУПС, 2014. – Вип. 4(120). – С. 161-164.

5. Коваленко А.С. Підсистема технічної діагностики для автоматизації процесів керування в інтегрованих інформаційних системах / А.С. Коваленко , О.А.Смірнов, О.В. Коваленко // Системи озброєння і військова техніка.– Х.: ХУПС, 2014. – № 1(37). – С. 126-129.

6. Коваленко А.С. Анализ эффективности использования экспертной системы технической диагностики с традиционной структурой / А.С. Коваленко, А.А. Смирнов, А.В. Коваленко // Системи озброєння і військова техніка. – Х.: ХУПС, 2014. – № 2(38). – С. 106-108.

7. Коваленко А.С. Разработка структуры экспертной системы технической диагностики интегрированной информационной системы / А.С. Коваленко,

А.А. Смирнов, А.В. Коваленко // Наука і техніка Повітряних Сил Збройних Сил України. – Харків: ХУПС, 2014. – № 2(15). – С.154-157.

8. Коваленко А.С. Разработка структуры экспертной системы технической диагностики интегрированной информационной системы / А.С. Коваленко, А.А. Смирнов, А.В. Коваленко // Наука і техніка Повітряних Сил Збройних Сил України. – Харків: ХУПС, 2014. – № 2(15). – С.154-157.

9. Коваленко А.С. Структура системи технічної діагностики інтегрованих інформаційних систем / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Збірник наукових праць Кіровоградського національного технічного університету / техніка в сільськогосподарському виробництві, галузеве машинобудування, автоматизація. – Кіровоград: Вид-во КНТУ, 2014. – Вип. 27. – С. 245-251.

10. Коваленко А.С. Дослідження будови інтегрованої інформаційної системи та її елементів / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Системи озброєння і військова техніка. – Х.: ХУПС, 2014. – № 4(40). – С. 85-88.

11. Коваленко А.С. Розробка структури бази даних для обліку технічного стану елементів інтегрованої інформаційної системи з урахуванням вимог споживачів інформації / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Системи обробки інформації. – Х.: ХУПС, 2015. – Вип. 1(126). – С. 75-79.

12. Коваленко А.С. Обґрунтування набору даних для оцінки технічного стану інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Збірник наукових праць Харківського університету Повітряних Сил. – Харків: ХУПС, 2015. – Вип. 1(42). – С.39-41.

13. Коваленко А.С. Експертна система технічного діагностування інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Системи озброєння і військова техніка. – Х.: ХУПС, 2015. – № 1(41). – С. 106-111.

14. Коваленко А.С. Удосконалення методу технічного обслуговування об’єктів інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов,

О.В. Коваленко, О.П. Доренський // Системи озброєння і військова техніка. – Х.: ХУПС, 2016. – № 2(46). – С. 109-114.

15. Коваленко А.С. Метод визначення оптимального комплексу робіт з відновлення працездатності інтегрованої системи технічної діагностики в умовах ресурсних обмежень / А.С. Коваленко // Системи обробки інформації. – Х.: ХУПС, 2016. – Вип. 3(140). – С. 69-72.

16. Kovalenko A.S. Information model and its element for displaying information on technical condition of objects of integrated information system / A.S. Kovalenko, A.A. Smirnov, A.V. Kovalenko, A.P. Dorensky // International Journal of Computational Engineering Research (IJCER). – India: Delhi, 2016. – Volume 6, Issue 1. – P. 21-27.

17. Кожанова А.С. Система технічної діагностики інтегрованих інформаційних систем – обґрунтування необхідності створення, визначення понятійного апарату та напрямів досліджень / А.С. Кожанова, О.А. Смірнов, М.П. Савченко, Д.М. Ізосімов, В.В. Мороз // Створення та модернізація озброєння і військової техніки в сучасних умовах: Тринадцята наук.-техн. конф., 5-6 вер. 2013 р., м. Феодосія: тези доп. – Феодосія: ДНВЦ, 2013. – С. 187-188.

18. Кожанова А.С. Визначення основних напрямків досліджень щодо створення системи технічної діагностики інтегрованих інформаційних систем / А.С. Кожанова, О.А. Смірнов, А.В. Челпанов // Проблемні питання розвитку озброєння та військової техніки Збройних Сил України: IV наук.-техн. конф., 16-20 груд. 2013 р., м. Київ: зб. тез. – Київ: ЦНДІ ОВТ ЗСУ, 2013. – С. 293.

19. Коваленко А.С. Обґрунтування необхідності створення систем технічної діагностики інтегрованих інформаційних систем / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Інформатика та системні науки : V Всеукр. наук.-практ. конф., 13–15 бер. 2014 р., м. Полтава : зб. тез. – Полтава: ПУЕТ, 2014. – С. 292-294.

20. Коваленко А.С. Задачи распознавания ситуаций в системах организационной стратегии интеграции производства и операций

/ А.С. Коваленко, А.В. Коваленко // Комбінаторні конфігурації та їх застосування: XVI міжнар. наук.-практ. сем., 11-12 квіт. 2014 р., м. Кіровоград: зб. тез. – Кіровоград: КНТУ, 2014. – С. 53-55.

21. Коваленко А.С. Створення систем технічної діагностики для автоматизації процесів керування в інтегрованих інформаційних системах / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Проблеми і перспективи розвитку ІТ-індустрії: VI між нар. наук.-практ. конф., 17-18 квіт. 2014 р., м. Харків: зб. тез. – Харків: ХНЕУ, 2014. – С. 241.

22. Коваленко А.С. Визначення понятійного апарату та напрямів досліджень для синтезу систем технічної діагностики інтегрованих інформаційних систем / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Комп'ютерне моделювання у наукоємних технологіях (КМНТ-2014): наук.-техн. конф. з міжнар. участю, 28 -31 трав. 2014 р., м. Харків: зб. наук. праць. – Харків: ХНУ, 2014. – С. 190-193.

23. Коваленко А.С. Основні складові та функції системи технічної діагностики інтегрованих інформаційних систем / Коваленко А.С. // Інформаційні технології та комп'ютерна інженерія: наук.-практ. конф., 4 груд. 2014 р., м. Кіровоград: зб. тез доп. – Кіровоград: КНТУ, 2014. – С. 236.

24. Коваленко А.С. Розробка структури бази даних інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Проблеми і перспективи розвитку ІТ-індустрії: VII міжнар. наук.-практ. конф., 17-18 квіт. 2015 р., м. Харків: зб. тез. – Харків: ХНЕУ, 2015. – С. 15.

25. Коваленко А.С. Дослідження елементів інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Комбінаторні конфігурації та їх застосування: XVII між нар. наук.-практ. сем., 17-18 квіт. 2015 р., м. Кіровоград: зб. тез – Кіровоград: КНТУ, 2015. – С. 5.

26. Коваленко А.С. Метод автоматизованої перевірки результатів вимірювання параметрів об'єкті в інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Стратегія якості у

промисловості і освіті: XI міжнар. конф., 1 – 5 черв. 2015 р., м. Варна, Болгарія.: зб. матер. – Варна: ТУВ, 2015. – С. 423-426.

27. Коваленко А.С. Обґрунтування необхідності створення розподіленої бази даних для забезпечення захисту рухомих повітряних об`єктів / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Перспективні напрями захисту інформації: I всеукр. наук.-практ. конф., 07 вер. 2015 р., м. Одеса: зб. тез доп. – Одеса: ОНАЗ, 2015. – С. 35-39.

28. Коваленко А.С. Розробка інформаційної моделі автоматизованої оцінки технічного стану інтегральної інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Інформаційні технології та взаємодії (IT & I): II між нар. наук.-практ. конф., 3-5 лист. 2015 р., м. Київ: тези доп. – Київ: КНУ ім. Т. Шевченка, 2015. – С. 41-42.

29. Коваленко А.С. Разработка метода усовершенствования технического обслуживания интегрированной информационной системы / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Информационные и телекоммуникационные технологии: образование, наука, практика: II междунар. научн.-практ. конф., 3-4 дек. 2015 г., г. Алматы, Казахстан: сб. труд. – Алматы: КазНИТУ им. К.И. Сатпаева, 2015. – Т.2. – С. 423-427.

30. Королюк Н.А. Оценка временных интервалов работы лица, принимающего решение, на автоматизированном командном пункте / Н.А. Королюк, А.И. Тимочко // Системи обробки інформації. – Х.: ХУПС, 2005. – Вип. 8 (48). – С. 51-54.

31. Костерев В.В. Надёжность технических систем и управление риском: учебн. пособ. / В.В. Костерев. – М.: МИФИ, 2008. – 280 с.

32. Костюков А.В. Підвищення операційної ефективності підприємств на основі моніторингу в реальному часі. / А.В. Костюков, В.М. Костюков. – М.: Машинобудування, 2009. – 192 с.

33. Лазарев А.А. Выбор показателя затрат для анализа сравнительной экономической эффективности техники конечного потребления / А.А. Лазарев,

М.В. Бейлин // Сборник научных трудов ХГПУ.– Х.: ХГПУ, 1999. – Вып. 74. – С. 27-29.

34. Ланецкий Б.Н. Основы теории надежности, технического обслуживания и ремонта вооружения и военной техники: Справочные материалы, часть 1. / Б.Н. Ланецкий, А.А. Посудевский. – Харьков: ХВУ, 1993. – 308 с.

35. Ланецкий Б.Н. Основы теории надежности, технического обслуживания и ремонта вооружения и военной техники: Справочные материалы, часть 2. / Б.Н. Ланецкий, А.А. Посудевский. – Харьков: ХВУ, 1993. – 208 с.

36. Лапсарь А.П. Метод оценки состояния сложных технических объектов для синтеза быстродействующих прогнозирующих систем / А.П. Лапсарь // Измерительная техника. – 2004. – № 2. – С. 7-10.

37. Линейные задачи оптимизации: Учеб. пособие / С.В. Лутманов. – Пермь: ЛИТЕР-А, 2004. – Ч.1. – Линейное программирование. – 128 с.

38. Литвак Б.Г. Экспертные технологии в управлении. Учебное пособие / Б.Г. Литвак. – М.: Дело, 2014. – 318 с.

39. Локазюк В.М. Надійність, контроль, діагностика і модернізація ПК: Посібн. / В.М. Локазюк, Ю.Г. Савченко. – К.: Видавничий центр «Академія», 2004. – 376 с.

40. Лопатников Л. И. Экономико-математический словарь: Словарь современной экономической науки / Л.И. Лопатников. – М.: Дело, 2003. – 520 с.

41. Манухина С.Ю. Инженерная психология и эргономика: хрестоматия / С.Ю Манухина. – М.: Изд. центр ЕАОИ, 2009. –224 с.

42. Мартыненко М.В. Человекомашинные процедуры поддержки организационно–управленческих решений: учеб. пособие СПбГЭТУ / М.В. Мартыненко, О.И. Шеховцов. – СПб, 2012. – 250 с.

43. Мунипов О.В. Эргономика: человекоориентированное проектирование техники, программных средств и среды: Учебник / О.В. Мунипов, В.П. Зинченко. – М.: Логос, 2001. – 356 с.

44. Надеев А.И. Математическая модель эксплуатационной надежности интеллектуальных датчиков / А.И. Надеев, Р.А. Юсупов, Ю.К. Свечников, Д.Р. Юсупов // Измерительная техника. – М: Стандартинформ, 2004. – № 1. – С. 8-11.

45. Надійність техніки. Аналіз надійності. Основні положення: ДСТУ 2861-94 – [Чинний від 1997–01–01]. – Київ: Держстандарт України, 1995. – 33 с. – (Національний стандарт України).

46. Надійність техніки. Терміни та визначення: ДСТУ 2860-94 – [Чинний від 1996–01–01]. – Київ: Держстандарт України, 1994. – 36 с. – (Національний стандарт України).

47. Нейлор К. Как построить свою экспертную систему / К. Нейлор. – М.: Энегроатомиздат, 2007. – 242 с.

48. Николаева И. П. Экономический словарь / И.П. Николаева. – Проспект, 2015. – 399 с.

49. Зеркалов Д. В. Охорона праці в Галузі: Загальні вимоги: навч. посіб. Київ: Основа. 2011. 551 с.

50. Про охорону праці: Закон України від 14.10.1992 р. № 2694-XII. Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2694-12#Text

51. Сакулин В.П., Шептовицкий В.М. Безопасность труда при монтаже и эксплуатации электроустановок / В.П.Сакулин, В.М.Шептовицкий. – Л. : “Колос”, 1973. – 238 с.

52. 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 до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/4358

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування..... 2
2 Підстава для розробки..... 2
3 Мета та призначення розробки..... 2
4 Джерела розробки..... 2
5 Технічні вимоги..... 2
5.1 Вміст проекту..... 2
5.2 Показники призначення..... 3
5.3 Вимоги до функціональних характеристик..... 3
5.4 Вимоги до архітектури..... 3
5.5 Вимоги до надійності..... 3
5.6 Умови експлуатації..... 4
5.7 Вимоги до складу та параметрів технічних засобів..... 4
5.8 Вимоги до інформаційної і програмної сумісності..... 4
5.8.1 Обладнання..... 4
5.8.2 Мова програмування..... 4
5.8.3 Вхідні дані..... 5
5.8.4 Вихідні дані..... 5
6 Вимоги до програмної документації..... 5
7 Економічні вимоги..... 5
8 Вимоги щодо охорони праці..... 5
9 Перелік документів, що розробляються..... 6
10 Етапи розробки..... 6
11 Порядок контролю та приймання..... 6

					ВКРМ-123.21.0017.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Свистунов В.В.			Дослідження та програмна реалізація системи автоматизації виробничих процесів створення сонячних батарей класу Tier 2	Літ. М	Аркуш 1 / Аркушів 6
Перевірив		Смірнов С.А.					
Н. Контр.		Гермак В.С.				ЦНТУ КІ-20М-1,4	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи автоматизації виробничих процесів створення сонячних батарей класу Tier 2.

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. № 42-13 від 02.08.2021 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи автоматизації виробничих процесів створення сонячних батарей класу Tier 2.

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 автоматизації виробничих процесів створення сонячних батарей класу Tier 2;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows XP/Vista/7/8/10 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 XP/Vista/7/8/10.

5.8.1 Обладнання

Комп'ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14" 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Delphi 10.4 Sydney.

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 3 вересня 2021 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинні бути розглянуті шкідливі і небезпечні фактори при роботі з комп’ютером.

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 – 102 аркуші.

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 10.12.2021 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 22.12.2021 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 С.А.

Дослідження та програмна реалізація системи автоматизації виробничих процесів створення сонячних батарей класу Tier 2

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 44

Літера: РП

Кропивницький – 2021 року

Файл проекту програми ProjectSN.dpr

```
program SN;

{2021, ЦНТУ
Керівник Смірнов С.А.
Виконав Свистунов Віктор Вікторович
Дослідження та програмна реалізація системи автоматизації виробничих процесів створення сонячних батарей класу Tier 2
}


uses
  Forms,Dialogs,
  SysUtils,
  StanokLGZTT in 'StanokLGZTT_SN.pas' {Form1},
  SetLGZTT-500 'SetLGZTT-500.pas' {Form3},
  FormSTwindow 'FormSTwindow.pas' {Form4},
  StandardStanokData 'StandardStanokData.pas' {Form2},
  StanokWinData 'StanokWinData.pas' {Form5},
  Box in 'AboutBox.pas' {AboutBox};

{$R *.res}
var
i:integer;

begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(TForm3, Form3);
  Application.CreateForm(TForm4, Form4);
  Application.CreateForm(TForm2, Form2);
  Application.CreateForm(TForm5, Form5);
  Application.CreateForm(TAboutBox, AboutBox);
  Application.Run;
end.
```

Файл форми StanokLGZTT_SN.pas

```
unit StanokLGZTT_SN;

{2021, ЦНТУ
Керівник Смірнов С.А.
Виконав Свистунов Віктор Вікторович
Дослідження та програмна реалізація системи автоматизації виробничих процесів створення сонячних батарей класу Tier 2
}

interface

uses Windows, ActiveX, SysUtils, Classes, VirtIntf, AxCtrls;

type
  { StanokLGZTTAdapter }
TSNStanokLGZTTAdapter = class(TStanokLGZTTAdapter, IStanokLGZTTModifyTime)
  protected
    FModifyTime: Longint;
  public
constructor Create(StanokLGZTT: TStanokLGZTT; Ownership: TStanokLGZTTOwnership = soReference);
function Write(pv: Pointer; cb: Longint; pcbWritten: PLongint): HResult; override;
function Stat(out statstg: TStatStg; grfStatFlag: Longint): HResult; override;
    function GetModifyTime: Longint; virtual; stdcall;
    procedure SetModifyTime(Value: Longint); virtual; stdcall;
  end;

  TSNMemoryStanokLGZTT = class(TSNStanokLGZTTAdapter)
  private
    function GetMemoryStanokLGZTT: TMemoryStanokLGZTT;
  public
    constructor Create(StanokLGZTT: TMemoryStanokLGZTT; Ownership: TStanokLGZTTOwnership = soReference);
    property MemoryStanokLGZTT: TMemoryStanokLGZTT read GetMemoryStanokLGZTT;
  end;

  TSNFileStanokLGZTT = class(TStanokLGZTTAdapter, IStanokLGZTTModifyTime)
  private
    function GetFileStanokLGZTT: TFileStanokLGZTT;
  public
    constructor Create(const FileName: string; Mode: Word);
    function Commit(grfCommitFlags: Longint): HResult; override;
    function Stat(out statstg: TStatStg; grfStatFlag: Longint): HResult; override;
    function GetModifyTime: Longint; stdcall;
    procedure SetModifyTime(Time: Longint); stdcall;
    property FileStanokLGZTT: TFileStanokLGZTT read GetFileStanokLGZTT;
  end;

  TVirtualStanokLGZTT = class(TOleStanokLGZTT)
  private
    FStanokLGZTTModifyTime: IStanokLGZTTModifyTime;
  public
```

```
    constructor Create(AStanokLGZTT: IStanokLGZTT);
    function GetModifyTime: Longint;
    procedure SetModifyTime(Time: Longint);
  end;

  TExceptionHandler = procedure;

const
  ExceptionHandler: TExceptionHandler = nil;

implementation

constructor TSNStanokLGZTTAdapter.Create(StanokLGZTT: TStanokLGZTT;
  Ownership: TStanokLGZTTOwnership);
begin
  inherited Create(StanokLGZTT, Ownership);
  FModifyTime:=DateTimeToFileDate(Now);
end;

function TSNStanokLGZTTAdapter.Write(pv: Pointer; cb: Longint;
  pcbWritten: PLongint): HResult;
begin
  Result:=inherited Write(pv, cb, pcbWritten);
  FModifyTime:=DateTimeToFileDate(Now);
end;

function TSNStanokLGZTTAdapter.Stat(out statstg: TStatStg; grfStatFlag:
Longint): HResult;
var
  DosFileTime: Longint;
  LocalFileTime: TFileTime;
begin
  Result:=inherited Stat(statstg, grfStatFlag);
  if Result <> 0 then Exit;
  DosFileTime:=GetModifyTime;
  DosDateTimeToFileTime(LongRec(DosFileTime).Hi, LongRec(DosFileTime).Lo,
LocalFileTime);
  LocalFileTimeToFileTime(LocalFileTime, statstg.mtime);
end;

function TSNStanokLGZTTAdapter.GetModifyTime: Longint;
begin
  Result:=FModifyTime;
end;

procedure TSNStanokLGZTTAdapter.SetModifyTime(Value: Longint);
begin
  FModifyTime:=Value;
end;

constructor TSNMemoryStanokLGZTT.Create(StanokLGZTT: TMemoryStanokLGZTT;
  Ownership: TStanokLGZTTOwnership);
begin
  if StanokLGZTT = nil then
  begin
    Ownership:=soOwned;
    StanokLGZTT:=TMemoryStanokLGZTT.Create;
```

```
  end;
  inherited Create(StanokLGZTT, Ownership);
end;

function TSNMemoryStanokLGZTT.GetMemoryStanokLGZTT: TMemoryStanokLGZTT;
begin
  Result:=TMemoryStanokLGZTT(StanokLGZTT);
end;

constructor TSNFileStanokLGZTT.Create(const FileName: string; Mode: Word);
begin
  if Mode = fmCreate then
    unlink(PChar(FileName));
  inherited Create(TFileStanokLGZTT.Create(FileName, Mode), soOwned);
end;

function TSNFileStanokLGZTT.GetFileStanokLGZTT: TFileStanokLGZTT;
begin
  Result:=TFileStanokLGZTT(StanokLGZTT);
end;

function TSNFileStanokLGZTT.Stat(out statstg: TStatStg; grfStatFlag: Longint):
HResult;
begin
  Result:=inherited Stat(statstg, grfStatFlag);
  if Result <> 0 then Exit;
  GetFileTime(TFileStanokLGZTT(StanokLGZTT).Handle, @statstg.ctime,
@statstg.atime, @statstg.mtime);
end;

function TSNFileStanokLGZTT.GetModifyTime: Longint;
begin
  Result:=FileGetDate(FileStanokLGZTT.Handle);
end;

procedure TSNFileStanokLGZTT.SetModifyTime(Time: Longint);
begin
  FileSetDate(FileStanokLGZTT.Handle, Time);
end;

function TSNFileStanokLGZTT.Commit(grfCommitFlags: Longint): HResult;
begin
  FlushFileBuffers(FileStanokLGZTT.Handle);
  Result:=inherited Commit(grfCommitFlags);
end;

constructor TVirtualStanokLGZTT.Create(AStanokLGZTT: IStanokLGZTT);
begin
  inherited Create(AStanokLGZTT);
  if AStanokLGZTT.QueryInterface(IStanokLGZTTModifyTime, FStanokLGZTTModifyTime)
<> 0 then
    FStanokLGZTTModifyTime:=nil;
end;

function TVirtualStanokLGZTT.GetModifyTime: Longint;
begin
  if FStanokLGZTTModifyTime <> nil then
```

```
    Result:=FStanokLGZTTModifyTime.GetModifyTime
  else
    Result:=0;
end;

procedure TVirtualStanokLGZTT.SetModifyTime(Time: Longint);
begin
  if FStanokLGZTTModifyTime <> nil then
    FStanokLGZTTModifyTime.SetModifyTime(Time);
end;
end.
```

Файл програми SetLGZTT-500

```
unit SetLGZTT-500;

{2021, ЦНТУ
Керівник Смірнов С.А.
Виконав Свистунов Віктор Вікторович
Дослідження та програмна реалізація системи автоматизації виробничих процесів створення сонячних батарей класу Tier 2
}

interface
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, ExtCtrls, UsbTypes;

type
  TForm1 = class(TForm)
  LogListBox: TListBox;
  Panel1: TPanel;
  Button1: TButton;
  LogListBox: TListBox;
  Memo1: TMemo;
  Button2: TButton;
  procedure Button1Click(Sender: TObject);
  procedure Button2Click(Sender: TObject);
  private
    procedure Log(S : String);
  public
   procedure EnumerateHostDevices(hHost : THandle);
   procedure ShowRootHubInfo(RootHubName : String);
   procedure ShowRootHubDetail(hRoot : THandle);
procedure ShowHubPortDetail(hRoot:THandle; iPort:Integer);
procedure ShowDeviceDetail(hRoot:THandle; iPort:Integer);
procedure DisplayDescriptorInfo(Data:Array of byte;var IsHid:Boolean);
procedure DisplayREPORTDescriptorInfo(Data:Array of byte);
end;
var
  Form1: TForm1;
implementation

{$R *.dfm}

procedure TForm1.Log(S : String);
var i:integer;
U:string;
begin
for i:=0 to LogListBox.Items.Count do
Begin
U:=U+S;
end;
LogListBox.Items.Add(S);
memo1.Text:= LogListBox.Items.Text;  IntToStr(LogListBox.Items.Count);
LogListBox.ItemIndex:= LogListBox.Items.Count-1;
end;
```

```
procedure TForm1.EnumerateHostDevices(hHost : THandle);
var Success : LongBool; DeviceName : TDeviceName;
BytesReturned : Cardinal;
begin
Success:= DeviceIoControl(hHost, GetUSBCtlCode(10),
             @DeviceName, SizeOf(DeviceName),
             @DeviceName, sizeof(DeviceName),
             BytesReturned, nil
           );
 If not(Success) then begin
  Log(' Ошибка получения имени контроллера');
  Exit;
 End;
Log(Format('Имя контроллера:%s', [WideCharToString(@DeviceName.Name)]));
 (IOCTL_USB_GET_ROOT_HUB_NAME)
 Success:= DeviceIoControl(hHost, GetUSBCtlCode(3),
             @DeviceName, SizeOf(DeviceName),
             @DeviceName, sizeof(DeviceName),
             BytesReturned, nil
           );
 If not(Success) then begin
  Log(' Ошибка получения имени хаба');
  Exit;
 End;
Log(Format(' Имя корнев. хаба: %s',[WideCharToString(@DeviceName.Name)]));
 ShowRootHubInfo(WideCharToString(@DeviceName.Name));
end;

procedure TForm1.ShowRootHubInfo(RootHubName : String);
var sRoot:String; hRoot:THandle; SA:SECURITY_ATTRIBUTES;
begin
SA.nLength:= SizeOf(SECURITY_ATTRIBUTES);
 SA.lpSecurityDescriptor:= nil;
 SA.bInheritHandle:= False;

 sRoot:= Format('\\.\%s', [RootHubName]);

 hRoot:= CreateFile(PChar(@sRoot[1]),
 GENERICWRITE,FILE_SHARE_WRITE,@SA,OPEN_EXISTING,0,0);
 If (hRoot <> INVALID_HANDLE_VALUE) then begin
  ShowRootHubDetail(hRoot);
  CloseHandle(hRoot);
 End else begin
  Log(' Ошибка получения информации от хаба');
 End;
end;

procedure TForm1.ShowRootHubDetail(hRoot : THandle);
var Success : LongBool; NodeInformation : TNodeInformation;
    BytesReturned : Cardinal; iPort : Byte;
begin
 (IOCTL_USB_GET_NODE_INFORMATION)
 Success:= DeviceIoControl(hRoot, GetUSBCtlCode(3),
            @NodeInformation, sizeof(NodeInformation),
            @NodeInformation, sizeof(NodeInformation),
            BytesReturned, nil
```

```
          );
 If not(Success) then begin
  Log('Ошибка получения информации о хабе');
  Exit;
 End;
Log(Format('  Число портов: %d',
[NodeInformation.HubDescriptor.bNumberOfPorts]));
 For iPort:= 0 to NodeInformation.HubDescriptor.bNumberOfPorts-1 do begin
    ShowHubPortDetail(hRoot, iPort);
   End;
end;

procedure TForm1.ShowHubPortDetail(hRoot : THandle; iPort : Integer);
var Success : LongBool; NodeConnInfo : TNodeConnectionInformation;
    BytesReturned : Cardinal; PortStatus : Byte;
begin
 NodeConnInfo.ConnectionIndex:= iPort+1;
Success:= DeviceIoControl(hRoot, GetUSBCtlCode(4),
            @NodeConnInfo, sizeof(NodeConnInfo),
            @NodeConnInfo, sizeof(NodeConnInfo),
            BytesReturned, nil
           );
 If not(Success) then begin
  Log('Ошибка получения информации о порте');
  Exit;
 End;
 PortStatus:= NodeConnInfo.ConnectionStatus[1];
 If (PortStatus <> Byte(DeviceConnected)) then begin
  If (PortStatus = Byte(NoDeviceConnected)) then begin
End else begin
Log(Format('Состояние порта %d = %d',[iPort, Byte(PortStatus)]));
  End;
 End else begin
  If not(NodeConnInfo.DeviceIsHub) then begin
   Log(Format('   К порту %d подключено устройство', [iPort]));
   ShowDeviceDetail(hRoot, iPort);
  End else begin
   Log(Format('   К порту %d подключен хаб', [iPort]));
  End;
 End;
end;

procedure TForm1.ShowDeviceDetail(hRoot : THandle; iPort : Integer);
Var Success : LongBool; Packet : TDescriptorRequest;
    BytesReturned : Cardinal; IsHID : Boolean;
begin
ZeroMemory(@Packet, SizeOf(Packet));
 Packet.ConnectionIndex       := iPort+1;
 Packet.SetupPacket.bmRequest := $80;
 Packet.SetupPacket.bRequest  := USB_REQUEST_GET_DESCRIPTOR;
 Packet.SetupPacket.wValue [2]:=USB_DEVICE_DESCRIPTOR_TYPE;
Packet.SetupPacket.wLength[2]:= 1;
Success:= DeviceIoControl(hRoot, GetUSBCtlCode(5),
            @Packet, sizeof(Packet),
            @Packet, sizeof(Packet),
            BytesReturned, nil
           );
```

```
If not(Success) then begin
Log(Format('Ошибка получения информации об устройстве %s',
[SysErrorMessage(GetLastError())]));
  Exit;
 End;
 IsHid:= False;
DisplayDescriptorInfo(Packet.Data, IsHid);
 ZeroMemory(@Packet, SizeOf(Packet));
 Packet.ConnectionIndex       := iPort+1;
Packet.SetupPacket.bmRequest := $80;
Packet.SetupPacket.bRequest:=USB_REQUEST_GET_DESCRIPTOR;
Packet.SetupPacket.wValue[2]:=USB_CONFIGURATION_DESCRIPTOR_TYPE;

Packet.SetupPacket.wLength[2]:= 1;

Success:= DeviceIoControl(hRoot, GetUSBCtlCode(5),
            @Packet, sizeof(Packet),
            @Packet, sizeof(Packet),
            BytesReturned, nil
           );
If not(Success) then begin
Log(Format('Ошибка получения информации об устройстве %s', [
SysErrorMessage(GetLastError())]));
Exit;
End;

 DisplayDescriptorInfo(Packet.Data, IsHid);
 If IsHid then begin
     ZeroMemory(@Packet, SizeOf(Packet));
     Packet.ConnectionIndex    := iPort+1;
     Packet.SetupPacket.bmRequest := $80;
Packet.SetupPacket.bRequest  :=USB_REQUEST_GET_DESCRIPTOR;
Packet.SetupPacket.wValue [2]:=USB_REPORT_DESCRIPTOR_TYPE;
Packet.SetupPacket.wLength[2]:=1;

     Success:= DeviceIoControl(hRoot, GetUSBCtlCode(5),
                @Packet, sizeof(Packet),
                @Packet, sizeof(Packet),
                BytesReturned, nil
               );
     If not(Success) then begin
Log(Format('Ошибка получения REPORT-дескриптора %s',[
SysErrorMessage(GetLastError())]));
     Exit;
     End;
    DisplayREPORTDescriptorInfo(Packet.Data);
 End;
end;

procedure TForm1.Button1Click(Sender: TObject);
var sHost : String; iHost : Integer; hHost : THandle;
    SA : SECURITY_ATTRIBUTES; EndOfSearch : Boolean;
begin
SA.nLength:= SizeOf(SECURITY_ATTRIBUTES);
 SA.lpSecurityDescriptor:= nil;
 SA.bInheritHandle:= False;
iHost:= 0;
```

```
 EndOfSearch:= False;
 Repeat
  sHost:= Format('\\.\HCD%d', [iHost]);
  Log(Format('Проверка хоста <%s>...',[sHost]));
hHost:= CreateFile(PChar(@sHost[1]),
GENERIC_WRITE,FILE_SHARE_WRITE,@SA,OPEN_EXISTING,0,0);
  If (hHost <> INVALID_HANDLE_VALUE) then begin
Log(Format('Хост-контроллер %d успешно открыт',[iHost]));
   EnumerateHostDevices(hHost);
   CloseHandle(hHost);
  End else begin
   Log(Format('Хост-контроллер <%s> не найден',[sHost]));
   EndOfSearch:= True;
  End;
  Inc(iHost);
 Until EndOfSearch;
end;

procedure TForm1.DisplayREPORTDescriptorInfo(Data : Array of byte);
begin
 Log('Дескриптор REPORT');
end;

procedure TForm1.Button2Click(Sender: TObject);
 var Success : LongBool; DeviceName : TDeviceName;
    BytesReturned : Cardinal; hHost : THandle;
begin
 (IOCTL_GET_HCD_DRIVERKEY_NAME)
 Success:= DeviceIoControl(hHost, GetUSBCtlCode(10),
             @DeviceName, SizeOf(DeviceName),
             @DeviceName, sizeof(DeviceName),
             BytesReturned, nil);
 If not(Success) then begin
  Log('Ошибка получения имени контроллера');
  Exit;
 End;
Log(Format('Имя контроллера:%s', [WideCharToString(@DeviceName.Name)]));
Success:= DeviceIoControl(hHost, GetUSBCtlCode(3),
           @DeviceName, SizeOf(DeviceName),
           @DeviceName, sizeof(DeviceName),
           BytesReturned, nil);
If not(Success) then begin
  Log(' Ошибка получения имени хаба');
  Exit;
 End;
ShowRootHubInfo(WideCharToString(@DeviceName.Name));
end;
end.
```

Файл форми FormSTwindow.pas

```
unit FormSTwindow;

interface
{2021, ЦНТУ
Керівник Смірнов С.А.
Виконав Свистунов Віктор Вікторович
Дослідження та програмна реалізація системи автоматизації виробничих процесів створення сонячних батарей класу Tier 2
}

uses Messages, SysUtils, Classes, Controls, Forms, DesignIntf,
ComponentDesigner;

type
  TStanokWindow = class(TForm, IUnknown, IDesignWindow, IDesignNotification,
    IEditHandler, IActivatable)
  private
    FSelection: IDesignerSelections;
    FOwner: TComponent;
    FDesigner: IDesigner;
    FComponentDesigner: IComponentDesigner;
    FActive: Boolean;
    procedure ComponentRead(Component: TComponent);
    procedure ReaderSetName(Reader: TReader; Component: TComponent;
      var Name: string);
    procedure WMActivate(var Msg: TWMActivate); message WM_ACTIVATE;
  protected
    procedure Activated; dynamic;
    procedure ActivateInspector(Ch: Char);
    function ClipboardComponents: Boolean;
    procedure CopyComponents(Root: TComponent;
      const Components: IDesignerSelections);
    procedure PasteComponents(AOwner, AParent: TComponent;
      const Components: IDesignerSelections);
    procedure SetSelection(const Components: IDesignerSelections);
    function UniqueName(Component: TComponent): string; virtual;
  public
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
    function GetEditState: TEditState; virtual;
    function EditAction(Action: TEditAction): Boolean; virtual;
    procedure ItemDeleted(const ADesigner: IDesigner; Item: TPersistent);
virtual;
    procedure ItemInserted(const ADesigner: IDesigner; Item: TPersistent);
virtual;
    procedure SelectionChanged(const ADesigner: IDesigner; const ASelection:
IDesignerSelections); virtual;
    procedure DesignerOpened(const Designer: IDesigner; AResurrecting: Boolean);
virtual;
    procedure DesignerClosed(const Designer: IDesigner; AGoingDormant: Boolean);
virtual;
    procedure ItemsModified(const Designer: IDesigner); virtual;
    procedure WindowHide; virtual;
    procedure WindowShow; virtual;
```

```
    property Active: Boolean read FActive;
    property Designer: IDesigner read FDesigner write FDesigner;
    property ComponentDesigner: IComponentDesigner read FComponentDesigner;
  end;

implementation

uses Windows, Clipbrd, DesignEditors, VCLEditors;

constructor TStanokWindow.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  FComponentDesigner:=ActiveDesigner;
  RegisterDesignNotification(Self);
  SetBounds(200, ComponentDesigner.Environment.GetMainWindowSize.Bottom + 2,
Width, Height);
end;

destructor TStanokWindow.Destroy;
begin
  FComponentDesigner:=nil;
  UnregisterDesignNotification(Self);
  inherited Destroy;
end;

procedure TStanokWindow.Activated;
begin
end;

procedure TStanokWindow.WMActivate(var Msg: TWMActivate);
begin
  inherited;
  FActive:=Msg.Active <> 0;
  if FActive then
    Activated;
end;

procedure TStanokWindow.ActivateInspector(Ch: Char);
begin
  Designer.ModalEdit(Ch, Self);
end;

function TStanokWindow.ClipboardComponents: Boolean;
begin
  try
    Result:=Clipboard.HasFormat(CF_COMPONENTS) or
      (Clipboard.HasFormat(CF_TEXT) and PossibleStanokLGZTT(Clipboard.AsText));
  except
    Result:=False;
  end;
end;

procedure TStanokWindow.CopyComponents(Root: TComponent;
  const Components: IDesignerSelections);
var
  S: TMemoryStanokLGZTT;
  W: TWriter;
```

```
  I: Integer;
begin
  S:=TMemoryStanokLGZTT.Create;
  try
    W:=TWriter.Create(S, 1024);
    try
      W.Root:=Root;
      for I:=0 to Components.Count - 1 do
      begin
        W.WriteSignature;
        W.WriteComponent(TComponent(Components[I]));
      end;
      W.WriteListEnd;
    finally
      W.Free;
    end;
    CopyStanokLGZTTToClipboard(S);
  finally
    S.Free;
  end;
end;

function TStanokWindow.GetEditState: TEditState;
begin
  Result:=[];
end;

function TStanokWindow.EditAction(Action: TEditAction): Boolean;
begin
  Result:=False;
end;

procedure TStanokWindow.WindowHide;
begin
  if Visible then
    ShowWindow(Handle, SW_HIDE);
end;

procedure TStanokWindow.WindowShow;
const
ShowCommands: array[TWindowState] of Word =
 (SW_SHOWNOACTIVATE, SW_SHOWMINNOACTIVE, SW_SHOWMAXIMIZED);
begin
  if Visible then
    ShowWindow(Handle, ShowCommands[WindowState]);
end;

procedure TStanokWindow.ComponentRead(Component: TComponent);
begin
  FSelection.Add(Component);
end;

procedure TStanokWindow.ReaderSetName(Reader: TReader; Component: TComponent;
var Name: string);
begin
  if (Reader.Root = FOwner) and (FOwner.FindComponent(Name) <> nil) then
    Name:=UniqueName(Component);
```

```
end;

procedure TStanokWindow.PasteComponents(AOwner, AParent: TComponent;
  const Components: IDesignerSelections);
var
  S: TStanokLGZTT;
  R: TReader;
begin
  S:=GetClipboardStanokLGZTT;
  try
    R:=TReader.Create(S, 1024);
    try
      R.OnSetName:=ReaderSetName;
      FOwner:=AOwner;
      FSelection:=Components;
      R.ReadComponents(AOwner, AParent, ComponentRead);
    finally
      R.Free;
    end;
  finally
    S.Free;
  end;
end;

procedure TStanokWindow.SelectionChanged(const ADesigner: IDesigner;
  const ASelection: IDesignerSelections);
begin
end;

procedure TStanokWindow.SetSelection(const Components: IDesignerSelections);
begin
  ComponentDesigner.SetSelection(Designer, Self, Components);
end;

function TStanokWindow.UniqueName(Component: TComponent): string;
begin
  Result:=Designer.UniqueName(Component.ClassName);
end;
end.
```

Файл форми StandardStanokData.pas

```
unit StandardStanokData;

{2021, ЦНТУ
Керівник Смірнов С.А.
Виконав Свистунов Віктор Вікторович
Дослідження та програмна реалізація системи автоматизації виробничих процесів створення сонячних батарей класу Tier 2
}

interface

uses Classes, Graphics, ActnMan;

type

  TStandardStanokData = class(TCustomActionBarStanokData)
  protected
    procedure SetColor(const Value: TColor); override;
  public
    procedure UpdateColors; override;
  published
    property HighlightColor;
    property UnusedColor;
    property BtnFrameColor default clBtnFace;
    property BtnSelectedColor default clBtnFace;
    property BtnSelectedFont default clWindowText;
    property Color default clBtnFace;
    property DisabledFontColor default clGrayText;
    property DisabledFontShadow default clBtnHighlight;
    property FontColor default clWindowText;
    property HotColor default clDefault;
    property HotFontColor default clDefault;
    property FrameTopLeftInner default clBtnHighlight;
    property FrameTopLeftOuter default clBtnFace;
    property FrameBottomRightInner default clBtnShadow;
    property FrameBottomRightOuter default clBlack;
    property DisabledColor default clDefault;
    property SelectedColor default clMenuHighlight;
    property SelectedFontColor default clHighlightText;
    property ShadowColor default clBtnShadow;
    property OnColorChange;
  end;

  TStanokData = class(TCustomActionBarStanokData)
  public
    procedure UpdateColors; override;
  published
    property ShadowColor default cl3DDkShadow;
    property Color default clBtnFace;
    property DisabledColor default clGray;
    property DisabledFontColor default clGrayText;
    property DisabledFontShadow default clBtnHighlight;
    property FontColor default clWindowText;
    property HighlightColor;
```

```
    property HotColor default clDefault;
    property HotFontColor default clDefault;
    property MenuColor default clWindow;
    property FrameTopLeftInner default clWhite;
    property FrameTopLeftOuter default $007A868A;
    property FrameBottomRightInner default clWhite;
    property FrameBottomRightOuter default $007A868A;
    property BtnFrameColor default $00C66931;
    property BtnSelectedColor default clWhite;
    property SelectedColor default $00EFD3C6;
    property SelectedFontColor default clBlack;
    property UnusedColor;
    property OnColorChange;
  end;

  TTwilightStanokData = class(TCustomActionBarStanokData)
  public
    procedure UpdateColors; override;
  published
    property Color default clBlack;
    property DisabledFontColor default clGrayText;
    property DisabledFontShadow default clBlack;
    property FontColor default clWhite;
    property HighlightColor;
    property HotColor default clDefault;
    property HotFontColor default clWhite;
    property FrameTopLeftInner default clBlack;
    property FrameTopLeftOuter default cl3DDkShadow;
    property FrameBottomRightInner default clBlack;
    property FrameBottomRightOuter default cl3DDkShadow;
    property BtnFrameColor default cl3DDkShadow;
    property BtnSelectedColor default cl3DDkShadow;
    property MenuColor default clBlack;
    property DisabledColor default clDefault;
    property SelectedColor default cl3dDkShadow;
    property SelectedFontColor default clBlack;
    property ShadowColor default clBlack;
    property UnusedColor default clBlack;
    property OnColorChange;
  end;

implementation

uses GraphUtil;

procedure TStandardStanokData.SetColor(const Value: TColor);
begin
  if Value <> Color then
  begin
    HighlightColor := GetHighlightColor(Value);
    UnusedColor := GetHighlightColor(Value, 10);
  end;
  inherited;
end;

procedure TStandardStanokData.UpdateColors;
begin
```

```
  inherited;
  BtnFrameColor := clBtnFace;
  BtnSelectedColor := clBtnFace;
  BtnSelectedFont := clWindowText;
  Color := clBtnFace;
  MenuColor := clBtnFace;
  DisabledFontColor := clGrayText;
  DisabledFontShadow := clBtnHighlight;
  DisabledColor := clDefault;
  FontColor := clWindowText;
  FrameTopLeftInner := clBtnHighlight;
  FrameTopLeftOuter := clBtnFace;
  FrameBottomRightInner := clBtnShadow;
  FrameBottomRightOuter := clBlack;
  HighlightColor := GetHighlightColor(Color);
  HotColor := clDefault;
  HotFontColor := clDefault;
  SelectedColor := clHighlight;
  SelectedFontColor := clHighlightText;
  ShadowColor := clBtnShadow;
  UnusedColor := GetHighlightColor(Color, 18);
end;

procedure TStanokData.UpdateColors;
begin
  inherited;
  Color := clBtnFace;
  MenuColor := clWindow;
  BtnFrameColor := $00C66931;
  BtnSelectedColor := clBtnFace;
  DisabledFontColor := clGrayText;
  DisabledFontShadow := clBtnHighlight;
  DisabledColor := clGray;
  FontColor := clWindowText;
  FrameTopLeftInner := clWhite;
  FrameTopLeftOuter := $007A868A;
  FrameBottomRightInner := clWhite;
  FrameBottomRightOuter := $007A868A;
  HighlightColor := GetHighLightColor(clBtnFace, 15);
  HotColor := clDefault;
  HotFontColor := clDefault;
  SelectedColor := $00EFD3C6;
  SelectedFontColor := clBlack;
  ShadowColor := cl3DDkShadow;
  UnusedColor := GetHighLightColor(clBtnFace, 15);
end;

procedure TTwilightStanokData.UpdateColors;
begin
  inherited;
  Color := clBlack;
  MenuColor := clBlack;
  DisabledFontColor := clGrayText;
  DisabledFontShadow := clBlack;
  DisabledColor := cl3DDkShadow;
  FontColor := clWhite;
  FrameTopLeftInner := clBlack;
```

```
  FrameTopLeftOuter := cl3DDkShadow;
  FrameBottomRightInner := clBlack;
  FrameBottomRightOuter := cl3DDkShadow;
  HighlightColor := clBlack;
  HotColor := clDefault;
  HotFontColor := clWhite;
  BtnSelectedColor := cl3DDkShadow;
  SelectedColor := cl3DDkShadow;
  SelectedFontColor := clBlack;
  ShadowColor := clBlack;
  UnusedColor := clBlack;
end;

end.
```

Файл форми StanokWinData.pas

```
unit StanokWinData;

{2021, ЦНТУ
Керівник Смірнов С.А.
Виконав Свистунов Віктор Вікторович
Дослідження та програмна реалізація системи автоматизації виробничих процесів створення сонячних батарей класу Tier 2
}

interface

uses Windows, Messages, Classes, Controls, Forms, Graphics, StdCtrls,
     ExtCtrls, CommCtrl;

type
  TstButton = class;

  TstButtonActionLink = class(TControlActionLink)
  protected
    FClient: TstButton;
    procedure AssignClient(AClient: TObject); override;
    function IsCheckedLinked: Boolean; override;
    function IsGroupIndexLinked: Boolean; override;
    procedure SetGroupIndex(Value: Integer); override;
    procedure SetChecked(Value: Boolean); override;
  end;

  TstButton = class(TGraphicControl)
  private
    FGroupIndex: Integer;
    FGlyph: Pointer;
    FDown: Boolean;
    FDragging: Boolean;
    FAllowAllUp: Boolean;
    FLayout: TButtonLayout;
    FSpacing: Integer;
    FTransparent: Boolean;
    FMargin: Integer;
    FFlat: Boolean;
    FMouseInControl: Boolean;
    procedure GlyphChanged(Sender: TObject);
    procedure UpdateExclusive;
    function GetGlyph: TBitmap;
    procedure SetGlyph(Value: TBitmap);
    function GetNumGlyphs: TNumGlyphs;
    procedure SetNumGlyphs(Value: TNumGlyphs);
    procedure SetDown(Value: Boolean);
    procedure SetLayout(Value: TButtonLayout);
    procedure SetSpacing(Value: Integer);
    procedure SetTransparent(Value: Boolean);
    procedure SetMargin(Value: Integer);
    procedure UpdateTracking;
  protected
    FState: TButtonState;
```

```
    procedure ActionChange(Sender: TObject; CheckDefaults: Boolean); override;
    function GetActionLinkClass: TControlActionLinkClass; override;
    function GetPalette: HPALETTE; override;
    procedure Loaded; override;
    procedure Paint; override;
    property MouseInControl: Boolean read FMouseInControl;
  public
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
    procedure Click; override;
  published
    property Action;
    property AllowAllUp: Boolean read FAllowAllUp write SetAllowAllUp default
False;
    property Anchors;
    property BiDiMode;
    property Constraints;
    property GroupIndex: Integer read FGroupIndex write SetGroupIndex default 0;
    property Down: Boolean read FDown write SetDown default False;
    property Caption;
    property Enabled;
    property Flat: Boolean read FFlat write SetFlat default False;
    property Font;
    property Glyph: TBitmap read GetGlyph write SetGlyph;
    property ParentFont;
    property ParentShowHint;
    property ParentBiDiMode;
    property PopupMenu;
    property ShowHint;
property Spacing: Integer read FSpacing write SetSpacing default 4;
property Transparent: Boolean read FTransparent write SetTransparent default
True;
    property Visible;
    property OnClick;
    property OnDblClick;
    property OnMouseDown;
    property OnMouseMove;
    property OnMouseUp;
  end;

TBitBtnKind = (bkCustom, bkOK, bkCancel, bkHelp, bkYes, bkNo, bkClose,
    bkAbort, bkRetry, bkIgnore, bkAll);

  TBitBtn = class(TButton)
  private
    FCanvas: TCanvas;
    FGlyph: Pointer;
    FStyle: TButtonStyle;
    FKind: TBitBtnKind;
    FLayout: TButtonLayout;
    FSpacing: Integer;
    FMargin: Integer;
    IsFocused: Boolean;
    FModifiedGlyph: Boolean;
    FMouseInControl: Boolean;
    procedure DrawItem(const DrawItemStruct: TDrawItemStruct);
    procedure SetGlyph(Value: TBitmap);
```

```
    function GetGlyph: TBitmap;
    function GetNumGlyphs: TNumGlyphs;
    procedure SetNumGlyphs(Value: TNumGlyphs);
    procedure GlyphChanged(Sender: TObject);
    function IsCustom: Boolean;
    function IsCustomCaption: Boolean;
    procedure SetStyle(Value: TButtonStyle);
    procedure SetKind(Value: TBitBtnKind);
    function GetKind: TBitBtnKind;
    procedure SetLayout(Value: TButtonLayout);
    procedure SetSpacing(Value: Integer);
    procedure SetMargin(Value: Integer);
  protected
    procedure ActionChange(Sender: TObject; CheckDefaults: Boolean); override;
    procedure CreateHandle; override;
    procedure CreateParams(var Params: TCreateParams); override;
    function GetPalette: HPALETTE; override;
    procedure SetButtonStyle(ADefault: Boolean); override;
  public
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
    procedure Click; override;
  published
    property Action;
    property Anchors;
    property BiDiMode;
    property Cancel stored IsCustom;
    property Caption stored IsCustomCaption;
    property Constraints;
    property Default stored IsCustom;
    property Enabled;
    property Glyph: TBitmap read GetGlyph write SetGlyph stored IsCustom;
    property Kind: TBitBtnKind read GetKind write SetKind default bkCustom;
    property Layout: TButtonLayout read FLayout write SetLayout default
blGlyphLeft;
    property Margin: Integer read FMargin write SetMargin default -1;
    property ModalResult stored IsCustom;
    property NumGlyphs: TNumGlyphs read GetNumGlyphs write SetNumGlyphs stored
IsCustom default 1;
    property ParentShowHint;
    property ParentBiDiMode;
    property ShowHint;
    property Style: TButtonStyle read FStyle write SetStyle default
bsAutoDetect;
    property Spacing: Integer read FSpacing write SetSpacing default 4;
    property TabOrder;
    property TabStop;
    property Visible;
    property OnEnter;
    property OnExit;
  end;

function DrawButtonFace(Canvas: TCanvas; const Client: TRect;
  BevelWidth: Integer; Style: TButtonStyle; IsRounded, IsDown,
  IsFocused: Boolean): TRect;

implementation
```

```
uses Consts, SysUtils, ActnList, ImgList, Themes;


var
  BitBtnGlyphs: array[TBitBtnKind] of TBitmap;


function GetBitBtnGlyph(Kind: TBitBtnKind): TBitmap;
begin
  if BitBtnGlyphs[Kind] = nil then
  begin
    BitBtnGlyphs[Kind]:=TBitmap.Create;
    BitBtnGlyphs[Kind].LoadFromResourceName(HInstance, BitBtnResNames[Kind]);
  end;
  Result:=BitBtnGlyphs[Kind];
end;

type
  TGlyphList = class(TImageList)
  private
    Used: TBits;
    FCount: Integer;
    function AllocateIndex: Integer;
  public
    constructor CreateSize(AWidth, AHeight: Integer);
    destructor Destroy; override;
    function AddMasked(Image: TBitmap; MaskColor: TColor): Integer;
    procedure Delete(Index: Integer);
    property Count: Integer read FCount;
  end;

  TGlyphCache = class
  private
    GlyphLists: TList;
  public
    constructor Create;
    destructor Destroy; override;
    function GetList(AWidth, AHeight: Integer): TGlyphList;
    procedure ReturnList(List: TGlyphList);
    function Empty: Boolean;
  end;

  TButtonGlyph = class
  private
    FOriginal: TBitmap;
    FGlyphList: TGlyphList;
    FIndexs: array[TButtonState] of Integer;
    FTransparentColor: TColor;
    FNumGlyphs: TNumGlyphs;
    FOnChange: TNotifyEvent;
    procedure GlyphChanged(Sender: TObject);
    procedure SetGlyph(Value: TBitmap);
    procedure SetNumGlyphs(Value: TNumGlyphs);
    procedure Invalidate;
    function CreateButtonGlyph(State: TButtonState): Integer;
    procedure DrawButtonGlyph(Canvas: TCanvas; const GlyphPos: TPoint;
```

```
      State: TButtonState; Transparent: Boolean);
    procedure DrawButtonText(Canvas: TCanvas; const Caption: string;
      TextBounds: TRect; State: TButtonState; BiDiFlags: Longint);
    procedure CalcButtonLayout(Canvas: TCanvas; const Client: TRect;
      const Offset: TPoint; const Caption: string; Layout: TButtonLayout;
      Margin, Spacing: Integer; var GlyphPos: TPoint; var TextBounds: TRect;
      BiDiFlags: Longint);
  public
    constructor Create;
    destructor Destroy; override;
    function Draw(Canvas: TCanvas; const Client: TRect; const Offset: TPoint;
      const Caption: string; Layout: TButtonLayout; Margin, Spacing: Integer;
      State: TButtonState; Transparent: Boolean; BiDiFlags: Longint): TRect;
    property Glyph: TBitmap read FOriginal write SetGlyph;
    property NumGlyphs: TNumGlyphs read FNumGlyphs write SetNumGlyphs;
    property OnChange: TNotifyEvent read FOnChange write FOnChange;
  end;

constructor TGlyphList.CreateSize(AWidth, AHeight: Integer);
begin
  inherited CreateSize(AWidth, AHeight);
  Used:=TBits.Create;
end;

destructor TGlyphList.Destroy;
begin
  Used.Free;
  inherited Destroy;
end;

function TGlyphList.AllocateIndex: Integer;
begin
  Result:=Used.OpenBit;
  if Result >= Used.Size then
  begin
    Result:=inherited Add(nil, nil);
    Used.Size:=Result + 1;
  end;
  Used[Result]:=True;
end;

function TGlyphList.AddMasked(Image: TBitmap; MaskColor: TColor): Integer;
begin
  Result:=AllocateIndex;
  ReplaceMasked(Result, Image, MaskColor);
  Inc(FCount);
end;

procedure TGlyphList.Delete(Index: Integer);
begin
  if Used[Index] then
  begin
    Dec(FCount);
    Used[Index]:=False;
  end;
end;
```

```
constructor TGlyphCache.Create;
begin
  inherited Create;
  GlyphLists:=TList.Create;
end;

destructor TGlyphCache.Destroy;
begin
  GlyphLists.Free;
  inherited Destroy;
end;

function TGlyphCache.GetList(AWidth, AHeight: Integer): TGlyphList;
var
  I: Integer;
begin
  for I:=GlyphLists.Count - 1 downto 0 do
  begin
    Result:=GlyphLists[I];
    with Result do
      if (AWidth = Width) and (AHeight = Height) then Exit;
  end;
  Result:=TGlyphList.CreateSize(AWidth, AHeight);
  GlyphLists.Add(Result);
end;

procedure TGlyphCache.ReturnList(List: TGlyphList);
begin
  if List = nil then Exit;
  if List.Count = 0 then
  begin
    GlyphLists.Remove(List);
    List.Free;
  end;
end;

function TGlyphCache.Empty: Boolean;
begin
  Result:=GlyphLists.Count = 0;
end;

var
  GlyphCache: TGlyphCache = nil;
  ButtonCount: Integer = 0;

constructor TButtonGlyph.Create;
var
  I: TButtonState;
begin
  inherited Create;
  FOriginal:=TBitmap.Create;
  FOriginal.OnChange:=GlyphChanged;
  FTransparentColor:=clOlive;
  FNumGlyphs:=1;
  for I:=Low(I) to High(I) do
    FIndexs[I]:=-1;
  if GlyphCache = nil then GlyphCache:=TGlyphCache.Create;
```

```
end;

destructor TButtonGlyph.Destroy;
begin
  FOriginal.Free;
  Invalidate;
  if Assigned(GlyphCache) and GlyphCache.Empty then
  begin
    GlyphCache.Free;
    GlyphCache:=nil;
  end;
  inherited Destroy;
end;

procedure TButtonGlyph.Invalidate;
var
  I: TButtonState;
begin
  for I:=Low(I) to High(I) do
  begin
    if FIndexs[I] <> -1 then FGlyphList.Delete(FIndexs[I]);
    FIndexs[I]:=-1;
  end;
  GlyphCache.ReturnList(FGlyphList);
  FGlyphList:=nil;
end;

procedure TButtonGlyph.GlyphChanged(Sender: TObject);
begin
  if Sender = FOriginal then
  begin
    FTransparentColor:=FOriginal.TransparentColor;
    Invalidate;
    if Assigned(FOnChange) then FOnChange(Self);
  end;
end;

procedure TButtonGlyph.SetGlyph(Value: TBitmap);
var
  Glyphs: Integer;
begin
  Invalidate;
  FOriginal.Assign(Value);
  if (Value <> nil) and (Value.Height > 0) then
  begin
    FTransparentColor:=Value.TransparentColor;
    if Value.Width mod Value.Height = 0 then
    begin
      Glyphs:=Value.Width div Value.Height;
      if Glyphs > 4 then Glyphs:=1;
      SetNumGlyphs(Glyphs);
    end;
  end;
end;

procedure TButtonGlyph.SetNumGlyphs(Value: TNumGlyphs);
begin
```

```
  if (Value <> FNumGlyphs) and (Value > 0) then
  begin
    Invalidate;
    FNumGlyphs:=Value;
    GlyphChanged(Glyph);
  end;
end;

function TButtonGlyph.CreateButtonGlyph(State: TButtonState): Integer;
const
  ROP_DSPDxax = $00E20746;
var
  TmpImage, DDB, MonoBmp: TBitmap;
  IWidth, IHeight: Integer;
  IRect, ORect: TRect;
  I: TButtonState;
  DestDC: HDC;
begin
  if (State = bsDown) and (NumGlyphs < 3) then State:=bsUp;
  Result:=FIndexs[State];
  if Result <> -1 then Exit;
  if (FOriginal.Width or FOriginal.Height) = 0 then Exit;
  IWidth:=FOriginal.Width div FNumGlyphs;
  IHeight:=FOriginal.Height;
  if FGlyphList = nil then
  begin
    if GlyphCache = nil then GlyphCache:=TGlyphCache.Create;
    FGlyphList:=GlyphCache.GetList(IWidth, IHeight);
  end;
  TmpImage:=TBitmap.Create;
  try
TmpImage.Width:=IWidth;
TmpImage.Height:=IHeight;
IRect:=Rect(0, 0, IWidth, IHeight);
TmpImage.Canvas.Brush.Color:=clBtnFace;
TmpImage.Palette:=CopyPalette(FOriginal.Palette);
I:=State;
if Ord(I) >= NumGlyphs then I:=bsUp;
ORect:=Rect(Ord(I) * IWidth, 0, (Ord(I) + 1) * IWidth, IHeight);
case State of
bsUp, bsDown,
bsExclusive:
begin
TmpImage.Canvas.CopyRect(IRect, FOriginal.Canvas, ORect);
if FOriginal.TransparentMode = tmFixed then
FIndexs[State]:=FGlyphList.AddMasked(TmpImage, FTransparentColor)
else
FIndexs[State]:=FGlyphList.AddMasked(TmpImage, clDefault);
end;
bsDisabled:
begin
MonoBmp:=nil;
DDB:=nil;
try
MonoBmp:=TBitmap.Create;
DDB:=TBitmap.Create;
DDB.Assign(FOriginal);
```

```
DDB.HandleType:=bmDDB;
if NumGlyphs > 1 then
with TmpImage.Canvas do
begin
CopyRect(IRect, DDB.Canvas, ORect);
MonoBmp.Monochrome:=True;
MonoBmp.Width:=IWidth;
MonoBmp.Height:=IHeight;
DDB.Canvas.Brush.Color:=clWhite;
MonoBmp.Canvas.CopyRect(IRect, DDB.Canvas, ORect);
Brush.Color:=clBtnHighlight;
DestDC:=Handle;
SetTextColor(DestDC, clBlack);
SetBkColor(DestDC, clWhite);
BitBlt(DestDC, 0, 0, IWidth, IHeight,
       MonoBmp.Canvas.Handle, 0, 0, ROP_DSPDxax);
DDB.Canvas.Brush.Color:=clGray;
MonoBmp.Canvas.CopyRect(IRect, DDB.Canvas, ORect);
Brush.Color:=clBtnShadow;
DestDC:=Handle;
SetTextColor(DestDC, clBlack);
SetBkColor(DestDC, clWhite);
BitBlt(DestDC, 0, 0, IWidth, IHeight,
       MonoBmp.Canvas.Handle, 0, 0, ROP_DSPDxax);
DDB.Canvas.Brush.Color:=ColorToRGB(FTransparentColor);
MonoBmp.Canvas.CopyRect(IRect, DDB.Canvas, ORect);
Brush.Color:=clBtnFace;
DestDC:=Handle;
SetTextColor(DestDC, clBlack);
SetBkColor(DestDC, clWhite);
BitBlt(DestDC, 0, 0, IWidth, IHeight,
       MonoBmp.Canvas.Handle, 0, 0, ROP_DSPDxax);
end
else
begin
with MonoBmp do
begin
  Assign(FOriginal);
  HandleType:=bmDDB;
  Canvas.Brush.Color:=clBlack;
  Width:=IWidth;
  if Monochrome then
  begin
   Canvas.Font.Color:=clWhite;
   Monochrome:=False;
   Canvas.Brush.Color:=clWhite;
  end;
  Monochrome:=True;
end;
with TmpImage.Canvas do
begin
  Brush.Color:=clBtnFace;
  FillRect(IRect);
  Brush.Color:=clBtnHighlight;
  SetTextColor(Handle, clBlack);
  SetBkColor(Handle, clWhite);
  BitBlt(Handle, 1, 1, IWidth, IHeight,
```

```
    MonoBmp.Canvas.Handle, 0, 0, ROP_DSPDxax);
  Brush.Color:=clBtnShadow;
  SetTextColor(Handle, clBlack);
  SetBkColor(Handle, clWhite);
  BitBlt(Handle, 0, 0, IWidth, IHeight,
    MonoBmp.Canvas.Handle, 0, 0, ROP_DSPDxax);
end;
end;
finally
DDB.Free;
MonoBmp.Free;
end;
FIndexs[State]:=FGlyphList.AddMasked(TmpImage, clDefault);
end;
end;
finally
TmpImage.Free;
end;
Result:=FIndexs[State];
end;

procedure TButtonGlyph.DrawButtonGlyph(Canvas: TCanvas; const GlyphPos: TPoint;
  State: TButtonState; Transparent: Boolean);
var
  Index: Integer;
  Details: TThemedElementDetails;
  R: TRect;
  Button: TThemedButton;
begin
  if FOriginal = nil then Exit;
  if (FOriginal.Width = 0) or (FOriginal.Height = 0) then Exit;
  Index:=CreateButtonGlyph(State);
  with GlyphPos do
  begin
    if ThemeServices.ThemesEnabled then
    begin
      R.TopLeft:=GlyphPos;
      R.Right:=R.Left + FOriginal.Width div FNumGlyphs;
      R.Bottom:=R.Top + FOriginal.Height;
      case State of
        bsDisabled:
          Button:=tbPushButtonDisabled;
        bsDown,
        bsExclusive:
          Button:=tbPushButtonPressed;
      else
        // bsUp
        Button:=tbPushButtonNormal;
      end;
      Details:=ThemeServices.GetElementDetails(Button);
      ThemeServices.DrawIcon(Canvas.Handle, Details, R, FGlyphList.Handle,
Index);
    end
    else
      if Transparent or (State = bsExclusive) then
      begin
        ImageList_DrawEx(FGlyphList.Handle, Index, Canvas.Handle, X, Y, 0, 0,
```

```
          clNone, clNone, ILD_Transparent)
      end
      else
        ImageList_DrawEx(FGlyphList.Handle, Index, Canvas.Handle, X, Y, 0, 0,
          ColorToRGB(clBtnFace), clNone, ILD_Normal);
  end;
end;

procedure TButtonGlyph.DrawButtonText(Canvas: TCanvas; const Caption: string;
  TextBounds: TRect; State: TButtonState; BiDiFlags: LongInt);
begin
  with Canvas do
  begin
    Brush.Style:=bsClear;
    if State = bsDisabled then
    begin
      OffsetRect(TextBounds, 1, 1);
      Font.Color:=clBtnHighlight;
      DrawText(Handle, PChar(Caption), Length(Caption), TextBounds,
        DT_CENTER or DT_VCENTER or BiDiFlags);
      OffsetRect(TextBounds, -1, -1);
      Font.Color:=clBtnShadow;
      DrawText(Handle, PChar(Caption), Length(Caption), TextBounds,
        DT_CENTER or DT_VCENTER or BiDiFlags);
    end else
      DrawText(Handle, PChar(Caption), Length(Caption), TextBounds,
        DT_CENTER or DT_VCENTER or BiDiFlags);
  end;
end;

function TButtonGlyph.Draw(Canvas: TCanvas; const Client: TRect;
  const Offset: TPoint; const Caption: string; Layout: TButtonLayout;
  Margin, Spacing: Integer; State: TButtonState; Transparent: Boolean;
  BiDiFlags: LongInt): TRect;
var
  GlyphPos: TPoint;
begin
  CalcButtonLayout(Canvas, Client, Offset, Caption, Layout, Margin, Spacing,
    GlyphPos, Result, BiDiFlags);
  DrawButtonGlyph(Canvas, GlyphPos, State, Transparent);
  DrawButtonText(Canvas, Caption, Result, State, BiDiFlags);
end;

procedure TstButtonActionLink.AssignClient(AClient: TObject);
begin
  inherited AssignClient(AClient);
  FClient:=AClient as TstButton;
end;

function TstButtonActionLink.IsCheckedLinked: Boolean;
begin
  Result:=inherited IsCheckedLinked and (FClient.GroupIndex <> 0) and
    FClient.AllowAllUp and (FClient.Down = (Action as TCustomAction).Checked);
end;

function TstButtonActionLink.IsGroupIndexLinked: Boolean;
begin
```

```
  Result:=(FClient is TstButton) and
    (TstButton(FClient).GroupIndex = (Action as TCustomAction).GroupIndex);
end;

procedure TstButtonActionLink.SetChecked(Value: Boolean);
begin
  if IsCheckedLinked then TstButton(FClient).Down:=Value;
end;

procedure TstButtonActionLink.SetGroupIndex(Value: Integer);
begin
  if IsGroupIndexLinked then TstButton(FClient).GroupIndex:=Value;
end;

constructor TstButton.Create(AOwner: TComponent);
begin
  FGlyph:=TButtonGlyph.Create;
  TButtonGlyph(FGlyph).OnChange:=GlyphChanged;
  inherited Create(AOwner);
  SetBounds(0, 0, 23, 22);
  ControlStyle:=[csCaptureMouse, csDoubleClicks];
  ParentFont:=True;
  Color:=clBtnFace;
  FSpacing:=4;
  FMargin:=-1;
  FLayout:=blGlyphLeft;
  FTransparent:=True;
  Inc(ButtonCount);
end;

destructor TstButton.Destroy;
begin
  Dec(ButtonCount);
  inherited Destroy;
  TButtonGlyph(FGlyph).Free;
end;

procedure TstButton.Paint;
const
  DownStyles: array[Boolean] of Integer = (BDR_RAISEDINNER, BDR_SUNKENOUTER);
  FillStyles: array[Boolean] of Integer = (BF_MIDDLE, 0);
var
  PaintRect: TRect;
  DrawFlags: Integer;
  Offset: TPoint;
  Button: TThemedButton;
  ToolButton: TThemedToolBar;
  Details: TThemedElementDetails;
begin
  if not Enabled then
  begin
    FState:=bsDisabled;
    FDragging:=False;
  end
  else if FState = bsDisabled then
    if FDown and (GroupIndex <> 0) then
      FState:=bsExclusive
```

```
  else
    FState:=bsUp;
Canvas.Font:=Self.Font;

if ThemeServices.ThemesEnabled then
begin
  PerformEraseBackground(Self, Canvas.Handle);

  if not Enabled then
    Button:=tbPushButtonDisabled
  else
    if FState in [bsDown, bsExclusive] then
      Button:=tbPushButtonPressed
    else
      if MouseInControl then
        Button:=tbPushButtonHot
      else
        Button:=tbPushButtonNormal;

  ToolButton:=ttbToolbarDontCare;
  if FFlat then
  begin
    case Button of
      tbPushButtonDisabled:
        Toolbutton:=ttbButtonDisabled;
      tbPushButtonPressed:
        Toolbutton:=ttbButtonPressed;
      tbPushButtonHot:
        Toolbutton:=ttbButtonHot;
      tbPushButtonNormal:
        Toolbutton:=ttbButtonNormal;
    end;
  end;

  PaintRect:=ClientRect;
  if ToolButton = ttbToolbarDontCare then
  begin
    Details:=ThemeServices.GetElementDetails(Button);
    ThemeServices.DrawElement(Canvas.Handle, Details, PaintRect);
    PaintRect:=ThemeServices.ContentRect(Canvas.Handle, Details, PaintRect);
  end
  else
  begin
    Details:=ThemeServices.GetElementDetails(ToolButton);
    ThemeServices.DrawElement(Canvas.Handle, Details, PaintRect);
    PaintRect:=ThemeServices.ContentRect(Canvas.Handle, Details, PaintRect);
  end;

  if Button = tbPushButtonPressed then
  begin
    if ToolButton <> ttbToolbarDontCare then
      Canvas.Font.Color:=clHighlightText;
    Offset:=Point(1, 0);
  end
  else
    Offset:=Point(0, 0);
```

```
    TButtonGlyph(FGlyph).Draw(Canvas, PaintRect, Offset, Caption, FLayout,
FMargin, FSpacing, FState, Transparent,
      DrawTextBiDiModeFlags(0));
  end
  else
  begin
    PaintRect:=Rect(0, 0, Width, Height);
    if not FFlat then
    begin
      DrawFlags:=DFCS_BUTTONPUSH or DFCS_ADJUSTRECT;
      if FState in [bsDown, bsExclusive] then
        DrawFlags:=DrawFlags or DFCS_PUSHED;
      DrawFrameControl(Canvas.Handle, PaintRect, DFC_BUTTON, DrawFlags);
    end
    else
    begin
      if (FState in [bsDown, bsExclusive]) or
        (FMouseInControl and (FState <> bsDisabled)) or
        (csDesigning in ComponentState) then
        DrawEdge(Canvas.Handle, PaintRect, DownStyles[FState in [bsDown,
bsExclusive]],
          FillStyles[Transparent] or BF_RECT)
      else if not Transparent then
      begin
        Canvas.Brush.Color:=Color;
        Canvas.FillRect(PaintRect);
      end;
      InflateRect(PaintRect, -1, -1);
    end;
    if FState in [bsDown, bsExclusive] then
    begin
      if (FState = bsExclusive) and (not FFlat or not FMouseInControl) then
      begin
        Canvas.Brush.Bitmap:=AllocPatternBitmap(clBtnFace, clBtnHighlight);
        Canvas.FillRect(PaintRect);
      end;
      Offset.X:=1;
      Offset.Y:=1;
    end
    else
    begin
      Offset.X:=0;
      Offset.Y:=0;
    end;
    TButtonGlyph(FGlyph).Draw(Canvas, PaintRect, Offset, Caption, FLayout,
FMargin,
      FSpacing, FState, Transparent, DrawTextBiDiModeFlags(0));
  end;
end;

procedure TstButton.UpdateTracking;
var
  P: TPoint;
begin
  if FFlat then
  begin
    if Enabled then
```

```
    begin
      GetCursorPos(P);
      FMouseInControl:=not (FindDragTarget(P, True) = Self);
      if FMouseInControl then
        Perform(CM_MOUSELEAVE, 0, 0)
      else
        Perform(CM_MOUSEENTER, 0, 0);
    end;
  end;
end;

procedure TstButton.Loaded;
var
  State: TButtonState;
begin
  inherited Loaded;
  if Enabled then
    State:=bsUp
  else
    State:=bsDisabled;
  TButtonGlyph(FGlyph).CreateButtonGlyph(State);
end;

procedure TstButton.MouseDown(Button: TMouseButton; Shift: TShiftState;
  X, Y: Integer);
begin
  inherited MouseDown(Button, Shift, X, Y);
  if (Button = mbLeft) and Enabled then
  begin
    if not FDown then
    begin
      FState:=bsDown;
      Invalidate;
    end;
    FDragging:=True;
  end;
end;

procedure TstButton.MouseMove(Shift: TShiftState; X, Y: Integer);
var
  NewState: TButtonState;
begin
  inherited MouseMove(Shift, X, Y);
  if FDragging then
  begin
    if not FDown then NewState:=bsUp
    else NewState:=bsExclusive;
    if (X >= 0) and (X < ClientWidth) and (Y >= 0) and (Y <= ClientHeight) then
      if FDown then NewState:=bsExclusive else NewState:=bsDown;
    if NewState <> FState then
    begin
      FState:=NewState;
      Invalidate;
    end;
  end
  else if not FMouseInControl then
    UpdateTracking;
```

```
end;

procedure TstButton.MouseUp(Button: TMouseButton; Shift: TShiftState;
  X, Y: Integer);
var
  DoClick: Boolean;
begin
inherited MouseUp(Button, Shift, X, Y);
if FDragging then
begin
  FDragging:=False;
  DoClick:=(X >= 0) and (X < ClientWidth) and (Y >= 0) and (Y <= ClientHeight);
  if FGroupIndex = 0 then
  begin
    FState:=bsUp;
    FMouseInControl:=False;
    if DoClick and not (FState in [bsExclusive, bsDown]) then
      Invalidate;
  end
  else
    if DoClick then
    begin
      SetDown(not FDown);
      if FDown then Repaint;
    end
    else
    begin
      if FDown then FState:=bsExclusive;
      Repaint;
    end;
  if DoClick then Click;
  UpdateTracking;
end;
end;

procedure TstButton.Click;
begin
  inherited Click;
end;

function TstButton.GetPalette: HPALETTE;
begin
  Result:=Glyph.Palette;
end;

function TstButton.GetActionLinkClass: TControlActionLinkClass;
begin
  Result:=TstButtonActionLink;
end;

function TstButton.GetGlyph: TBitmap;
begin
  Result:=TButtonGlyph(FGlyph).Glyph;
end;

procedure TstButton.SetGlyph(Value: TBitmap);
begin
```

```
  TButtonGlyph(FGlyph).Glyph:=Value;
  Invalidate;
end;

function TstButton.GetNumGlyphs: TNumGlyphs;
begin
  Result:=TButtonGlyph(FGlyph).NumGlyphs;
end;

procedure TstButton.SetNumGlyphs(Value: TNumGlyphs);
begin
  if Value < 0 then Value:=1
  else if Value > 4 then Value:=4;
  if Value <> TButtonGlyph(FGlyph).NumGlyphs then
  begin
    TButtonGlyph(FGlyph).NumGlyphs:=Value;
    Invalidate;
  end;
end;

procedure TstButton.GlyphChanged(Sender: TObject);
begin
  Invalidate;
end;

procedure TstButton.UpdateExclusive;
var
  Msg: TMessage;
begin
  if (FGroupIndex <> 0) and (Parent <> nil) then
  begin
    Msg.Msg:=CM_BUTTONPRESSED;
    Msg.WParam:=FGroupIndex;
    Msg.LParam:=Longint(Self);
    Msg.Result:=0;
    Parent.Broadcast(Msg);
  end;
end;

procedure TstButton.SetDown(Value: Boolean);
begin
  if FGroupIndex = 0 then Value:=False;
  if Value <> FDown then
  begin
    if FDown and (not FAllowAllUp) then Exit;
    FDown:=Value;
    if Value then
    begin
      if FState = bsUp then Invalidate;
      FState:=bsExclusive
    end
    else
    begin
      FState:=bsUp;
      Repaint;
    end;
    if Value then UpdateExclusive;
```

```
  end;
end;

procedure TstButton.SetFlat(Value: Boolean);
begin
  if Value <> FFlat then
  begin
    FFlat:=Value;
    Invalidate;
  end;
end;

procedure TstButton.SetGroupIndex(Value: Integer);
begin
  if FGroupIndex <> Value then
  begin
    FGroupIndex:=Value;
    UpdateExclusive;
  end;
end;

procedure TstButton.SetLayout(Value: TButtonLayout);
begin
  if FLayout <> Value then
  begin
    FLayout:=Value;
    Invalidate;
  end;
end;

procedure TstButton.SetMargin(Value: Integer);
begin
  if (Value <> FMargin) and (Value >= -1) then
  begin
    FMargin:=Value;
    Invalidate;
  end;
end;

procedure TstButton.SetSpacing(Value: Integer);
begin
  if Value <> FSpacing then
  begin
    FSpacing:=Value;
    Invalidate;
  end;
end;

procedure TstButton.SetTransparent(Value: Boolean);
begin
  if Value <> FTransparent then
  begin
    FTransparent:=Value;
    if Value then
      ControlStyle:=ControlStyle - [csOpaque] else
      ControlStyle:=ControlStyle + [csOpaque];
    Invalidate;
```

```
  end;
end;

procedure TstButton.SetAllowAllUp(Value: Boolean);
begin
  if FAllowAllUp <> Value then
  begin
    FAllowAllUp:=Value;
    UpdateExclusive;
  end;
end;

procedure TstButton.WMLButtonDblClk(var Message: TWMLButtonDown);
begin
  inherited;
  if FDown then DblClick;
end;

procedure TstButton.CMEnabledChanged(var Message: TMessage);
const
  NewState: array[Boolean] of TButtonState = (bsDisabled, bsUp);
begin
  TButtonGlyph(FGlyph).CreateButtonGlyph(NewState[Enabled]);
  UpdateTracking;
  Repaint;
end;

procedure TstButton.CMButtonPressed(var Message: TMessage);
var
  Sender: TstButton;
begin
  if Message.WParam = FGroupIndex then
  begin
    Sender:=TstButton(Message.LParam);
    if Sender <> Self then
    begin
      if Sender.Down and FDown then
      begin
        FDown:=False;
        FState:=bsUp;
        if (Action is TCustomAction) then
         TCustomAction(Action).Checked:=False;
        Invalidate;
      end;
      FAllowAllUp:=Sender.AllowAllUp;
    end;
  end;
end;

procedure TstButton.CMDialogChar(var Message: TCMDialogChar);
begin
  with Message do
    if IsAccel(CharCode, Caption) and Enabled and Visible and
      (Parent <> nil) and Parent.Showing then
    begin
      Click;
      Result:=1;
```

```
    end else
      inherited;
end;

procedure TstButton.CMFontChanged(var Message: TMessage);
begin
  Invalidate;
end;

procedure TstButton.CMTextChanged(var Message: TMessage);
begin
  Invalidate;
end;

procedure TstButton.CMSysColorChange(var Message: TMessage);
begin
  with TButtonGlyph(FGlyph) do
  begin
    Invalidate;
    CreateButtonGlyph(FState);
  end;
end;

procedure TstButton.CMMouseLeave(var Message: TMessage);
var
  NeedRepaint: Boolean;
begin
  inherited;
  NeedRepaint:=FFlat and FMouseInControl and Enabled and not FDragging;
  if NeedRepaint or ThemeServices.ThemesEnabled then
  begin
    FMouseInControl:=False;
    if Enabled then
      Repaint;
  end;
end;

procedure TstButton.ActionChange(Sender: TObject; CheckDefaults: Boolean);

  procedure CopyImage(ImageList: TCustomImageList; Index: Integer);
  begin
    with Glyph do
    begin
      Width:=ImageList.Width;
      Height:=ImageList.Height;
      Canvas.Brush.Color:=clFuchsia;
      Canvas.FillRect(Rect(0,0, Width, Height));
      ImageList.Draw(Canvas, 0, 0, Index);
    end;
  end;

begin
  inherited ActionChange(Sender, CheckDefaults);
  if Sender is TCustomAction then
    with TCustomAction(Sender) do
    begin
      if CheckDefaults or (Self.GroupIndex = 0) then
```

```
        Self.GroupIndex:=GroupIndex;
      if (Glyph.Empty) and (ActionList <> nil) and (ActionList.Images <> nil)
and
        (ImageIndex >= 0) and (ImageIndex < ActionList.Images.Count) then
        CopyImage(ActionList.Images, ImageIndex);
    end;
end;

constructor TBitBtn.Create(AOwner: TComponent);
begin
  FGlyph:=TButtonGlyph.Create;
  TButtonGlyph(FGlyph).OnChange:=GlyphChanged;
  inherited Create(AOwner);
  FCanvas:=TCanvas.Create;
  FStyle:=bsAutoDetect;
  FKind:=bkCustom;
  FLayout:=blGlyphLeft;
  FSpacing:=4;
  FMargin:=-1;
  ControlStyle:=ControlStyle + [csReflector];
  DoubleBuffered:=True;
end;

destructor TBitBtn.Destroy;
begin
  inherited Destroy;
  TButtonGlyph(FGlyph).Free;
  FCanvas.Free;
end;

procedure TBitBtn.CreateHandle;
var
  State: TButtonState;
begin
  if Enabled then
    State:=bsUp
  else
    State:=bsDisabled;
  inherited CreateHandle;
  TButtonGlyph(FGlyph).CreateButtonGlyph(State);
end;

procedure TBitBtn.CreateParams(var Params: TCreateParams);
begin
  inherited CreateParams(Params);
  with Params do Style:=Style or BS_OWNERDRAW;
end;

procedure TBitBtn.SetButtonStyle(ADefault: Boolean);
begin
  if ADefault <> IsFocused then
  begin
    IsFocused:=ADefault;
    Refresh;
  end;
end;
```

```
procedure TBitBtn.Click;
var
  Form: TCustomForm;
  Control: TWinControl;
begin
  case FKind of
    bkClose:
      begin
        Form:=GetParentForm(Self);
        if Form <> nil then Form.Close
        else inherited Click;
      end;
    bkHelp:
      begin
        Control:=Self;
        while (Control <> nil) and (Control.HelpContext = 0) do
          Control:=Control.Parent;
        if Control <> nil then Application.HelpContext(Control.HelpContext)
        else inherited Click;
      end;
    else
      inherited Click;
  end;
end;

procedure TBitBtn.CNMeasureItem(var Message: TWMMeasureItem);
begin
  with Message.MeasureItemStruct^ do
  begin
    itemWidth:=Width;
    itemHeight:=Height;
  end;
end;

procedure TBitBtn.CNDrawItem(var Message: TWMDrawItem);
begin
  DrawItem(Message.DrawItemStruct^);
end;

procedure TBitBtn.CMFontChanged(var Message: TMessage);
begin
  inherited;
  Invalidate;
end;

procedure TBitBtn.CMEnabledChanged(var Message: TMessage);
begin
  inherited;
  Invalidate;
end;

procedure TBitBtn.WMLButtonDblClk(var Message: TWMLButtonDblClk);
begin
  Perform(WM_LBUTTONDOWN, Message.Keys, Longint(Message.Pos));
end;

function TBitBtn.GetPalette: HPALETTE;
```

```
begin
  Result:=Glyph.Palette;
end;

procedure TBitBtn.SetGlyph(Value: TBitmap);
begin
  TButtonGlyph(FGlyph).Glyph:=Value as TBitmap;
  FModifiedGlyph:=True;
  Invalidate;
end;

function TBitBtn.GetGlyph: TBitmap;
begin
  Result:=TButtonGlyph(FGlyph).Glyph;
end;

procedure TBitBtn.GlyphChanged(Sender: TObject);
begin
  Invalidate;
end;

function TBitBtn.IsCustom: Boolean;
begin
  Result:=Kind = bkCustom;
end;

procedure TBitBtn.SetStyle(Value: TButtonStyle);
begin
  if Value <> FStyle then
  begin
    FStyle:=Value;
    Invalidate;
  end;
end;

procedure TBitBtn.SetKind(Value: TBitBtnKind);
begin
  if Value <> FKind then
  begin
    if Value <> bkCustom then
    begin
      Default:=Value in [bkOK, bkYes];
      Cancel:=Value in [bkCancel, bkNo];

      if ((csLoading in ComponentState) and (Caption = '')) or
        (not (csLoading in ComponentState)) then
      begin
        if BitBtnCaptions[Value] <> nil then
          Caption:=LoadResString(BitBtnCaptions[Value]);
      end;

      ModalResult:=BitBtnModalResults[Value];
      TButtonGlyph(FGlyph).Glyph:=GetBitBtnGlyph(Value);
      NumGlyphs:=2;
      FModifiedGlyph:=False;
    end;
    FKind:=Value;
```

```
    Invalidate;
  end;
end;

function TBitBtn.IsCustomCaption: Boolean;
begin
  Result:=AnsiCompareStr(Caption, LoadResString(BitBtnCaptions[FKind])) <> 0;
end;

function TBitBtn.GetKind: TBitBtnKind;
begin
  if FKind <> bkCustom then
    if ((FKind in [bkOK, bkYes]) xor Default) or
      ((FKind in [bkCancel, bkNo]) xor Cancel) or
      (ModalResult <> BitBtnModalResults[FKind]) or
      FModifiedGlyph then
      FKind:=bkCustom;
  Result:=FKind;
end;

procedure TBitBtn.SetLayout(Value: TButtonLayout);
begin
  if FLayout <> Value then
  begin
    FLayout:=Value;
    Invalidate;
  end;
end;

function TBitBtn.GetNumGlyphs: TNumGlyphs;
begin
  Result:=TButtonGlyph(FGlyph).NumGlyphs;
end;

procedure TBitBtn.SetNumGlyphs(Value: TNumGlyphs);
begin
  if Value < 0 then Value:=1
  else if Value > 4 then Value:=4;
  if Value <> TButtonGlyph(FGlyph).NumGlyphs then
  begin
    TButtonGlyph(FGlyph).NumGlyphs:=Value;
    Invalidate;
  end;
end;

procedure TBitBtn.SetSpacing(Value: Integer);
begin
  if FSpacing <> Value then
  begin
    FSpacing:=Value;
    Invalidate;
  end;
end;

procedure TBitBtn.SetMargin(Value: Integer);
begin
  if (Value <> FMargin) and (Value >= - 1) then
```

```
  begin
    FMargin:=Value;
    Invalidate;
  end;
end;

procedure DestroyLocals; far;
var
  I: TBitBtnKind;
begin
  for I:=Low(TBitBtnKind) to High(TBitBtnKind) do
    BitBtnGlyphs[I].Free;
end;

procedure TBitBtn.CMMouseEnter(var Message: TMessage);
begin
  inherited;
  if ThemeServices.ThemesEnabled and not FMouseInControl and not (csDesigning in
ComponentState) then
  begin
    FMouseInControl:=True;
    Repaint;
  end;
end;

procedure TBitBtn.CMMouseLeave(var Message: TMessage);
begin
  inherited;
  if ThemeServices.ThemesEnabled and FMouseInControl then
  begin
    FMouseInControl:=False;
    Repaint;
  end;
end;

end.
```