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

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

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи верстата ЧПК обробки електроізоляційного картону”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-22М-1
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Коваль В.В.
«____» _____ 2023 р.

Керівник проекту
доктор технічних наук, доцент
_____ Коваленко О.В.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _12 _____*"Інформаційні технології"*_____

Спеціальність 123 *"Комп'ютерна інженерія"*

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

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

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

Ковалю Владиславу Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи верстата ЧПК обробки електроізоляційного картону*

2. Керівник роботи *Коваленко Олександр Володимирович, докт. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 34-13 від 04.08.2023 року

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

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи верстата ЧПК обробки електроізоляційного картону*

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

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

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

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

6. Консультанти розділів роботи

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

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

КАЛЕНДАРНИЙ ПЛАН

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

Дата видачі завдання
« 6 » вересня 2023 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2023 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Коваль В.В. Дослідження та програмна реалізація системи верстата ЧПК обробки електроізоляційного картону. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи верстата ЧПК обробки електроізоляційного картону.

Метою розробки є дослідження та програмна реалізація системи верстата ЧПК обробки електроізоляційного картону.

Об'єктом дослідження є процес верстата ЧПК обробки електроізоляційного картону.

Предметом дослідження є методи верстата ЧПК обробки електроізоляційного картону.

Методи дослідження базуються на методах чисельно-програмного керування, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи верстата ЧПК обробки електроізоляційного картону.

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

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

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

Програму розроблено в середовищі Delphi 10, G-CODE.

Ключові слова: комп'ютерна інженерія, верстата ЧПК

ABSTRACT

Koval V.V. Research and software implementation of the CNC machine tool system for processing electrical insulating cardboard. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this final qualification work for the second (master's) level of higher education, software was developed, which is intended for the system of the CNC machine tool for processing electrical insulating cardboard.

The purpose of the development is the research and software implementation of the CNC machine tool system for processing electrical insulating cardboard.

The object of the study is the process of the electrical insulating cardboard processing machine.

The subject of the study is the methods of the CNC machine tool for processing electrical insulating cardboard.

Research methods are based on methods of numerical and software control, methods of mathematical statistics, methods of software development.

The result of the work is the software implementation of the CNC machine tool system for processing electrical insulating cardboard.

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

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

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

The program was developed in the Delphi 10 environment, G-CODE.

Keywords: computer engineering, CNC machine tool

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

БД	–	база даних
ГАВ	–	автоматична лінія (виробництва масштабу ділянки або цеху)
ГАД	–	гнучка автоматизована ділянка
ГВС	–	гнучка автоматизована виробнича система
ЄСКД	–	єдина система конструкторської документації
ОЗП	–	оперативний запам’ятовуючий пристрій
ПВП	–	пристрій введення програми
ПЗ	–	програмне забезпечення
ПЗП	–	постійний запам’ятовуючий пристрій
ПК	–	пульт корекції
ПО	–	пульт оператора
ППМК	–	пристрій пам'яті мікрокоманд
САМ	–	система автоматизованого програмування
САПР	–	система автоматизованого проектування
УП	–	управляюча програма
ЧПК	–	чисельне програмне керування
SQL	–	Structured Query Language – мова структурованих запросів

ВСТУП

Актуальність теми. Електрокартон електротехнічний – це один з найбільш популярних ізоляційних матеріалів. Найбільше поширення електрокартон одержав у сфері створення ізоляції для електроприладів і пристроїв (від зварювальних агрегатів до ізоляції електричних стендів) а також як матеріал для лекал.

Основне призначення електрокартону (електроізоляційний картон) – це виготовлення елементів електроізоляції. Але на практиці сфера його застосування набагато ширша. Завдяки високій зносостійкості електрокартон придбав величезну популярність у підприємств швейного виробництва й став звичним матеріалом для виготовлення лекал.

Найбільше часто електрокартон (картон електроізоляційний) застосовується разом із супутніми електротехнічними матеріалами: лаком ГФ 95, трансформаторним маслом, електробумагою і багато чим іншим.

Із всіх марок електрокартону (пресшпану), що виготовляються згідно зі стандартом ДСТ 4194 від 1983 року, найбільше поширення одержав електроізоляційний картон марок ЕВ, А, Б. Картон електротехнічний (електрокартон) марки ЕВ застосовується в різних галузях промисловості для електроізоляції при роботі в повітряних середовищах при температурі до 90 градусів Цельсія. Картон електроізоляційний марки Б (електротехнічний картон) використовується для деталей електроізоляції в трансформаторах, в апаратах, а також в іншому електроустаткуванні з масляним заповненням при робочій температурі до 150 °С. Картон електроізоляційний марки А (пресшпан) застосовується як електроізоляційний матеріал для різних агрегатів, трансформаторів і приладів, що працюють при напрузі до 750 кВольт.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи верстата ЧПК обробки електроізоляційного картону.

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

– Огляд існуючих систем верстата ЧПК обробки електроізоляційного картону.

– Дослідження системи верстата ЧПК обробки електроізоляційного картону.

– Програмна реалізація системи верстата ЧПК обробки електроізоляційного картону.

Об'єктом дослідження є процес верстата ЧПК обробки електроізоляційного картону.

Предметом дослідження є методи верстата ЧПК обробки електроізоляційного картону.

Методи дослідження базуються на методах чисельно-програмного керування, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод верстата ЧПК обробки електроізоляційного картону.

– Розроблено вітчизняний продукт верстата ЧПК обробки електроізоляційного картону, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі верстата ЧПК обробки електроізоляційного картону.

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 LVII Науково-технічній конференції здобувачів вищої освіти «Наука – виробництву», 2023, основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №14.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи верстата ЧПК обробки електроізоляційного картону, 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

Система призначена для програмного забезпечення верстата ЧПК обробки електроізоляційного картону**.**

Електрокартон (електротехнічний картон) марки ЕВ, виробляється відповідно ДО ДСТУ 2824 від 86 року виготовляється зі спеціальної небілої целюлози. На заводі-виготовлювачі електрокартон ЕВ обробляється просочувальними спеціальними матеріалами, через що електрокартон досягає високого значення щільності.

Состав і властивості аркуша електроізоляційного картону (електрокартон, синтофлекс) документально закріплені в ДСТ 4194-83 і в ДСТ 2824-86. При застосуванні електрокартону для лекал не відбувається "розлохмачування" матеріалу по краях, не відбувається розривів аркуша, що особливо важливо на швейних виробництвах, так як дані якості дозволяють використовувати лекала з електрокартону часто й довгий час. Більшу популярність електрокартон придбав як матеріал, що використовується для документів тривалого зберігання (архівних папок і т.д.).

Електроізоляційний картон марки ЕКС випускається аркушами з товщиною від 1 до 8 мм, різної гладкості. Сфера застосування електрокартону ЕКС: в електричних апаратах при робочій температурі до 105 градусів Цельсія.

Електрокартон використовується для виготовлення деталей головної ізоляції трансформаторів напругою до 220 кВ, для деталей зрівняльної і ярмової ізоляції й для виготовлення склеєних деталей трансформаторів всіх класів напруги, а також для ізоляції в іншому електроустаткуванні з масляним заповненням.

Картон електроізоляційний (електрокартон) марки Г використовується в трансформаторах, в апаратах, а також в іншому електроустаткуванні з масляним заповненням при робочій температурі до 105°С включно.

Картон електроізоляційний (електрокартон, пресшпан) марки ЕВС призначений для пазової ізоляції автомобільних стартерів і інших найважливіших деталей автотракторного встаткування. Картон електроізоляційний (електрокартон) марки Б призначений для високої електричної міцності деталей ізоляції напругою до 220 кВ. Картон електроізоляційний (електрокартон) марки А, У призначений для електроізоляційних цілей у трансформаторах і апаратах з масляним заповненням, для виготовлення деталей ізоляції в іншому електротехнічному встаткуванні.

Дуже широке поширення одержало сполучення електрокартону ЕВ, А, Б, Г у сполученні з різними ізоляційними матеріалами: кабельним папером, трансформаторними й мінеральними маслами й ін.

Основою електрокартону є целюлоза невибілена, також додається в основу бавовняна натуральна целюлоза.

Розглянувши поняття електрокар тону, перейдемо до визначення поняття станків з ЧПК.

Числове програмне керування (ЧПК) – комп'ютеризована система керування, що управляє приводами технологічного устаткування, включаючи верстатне оснащення. Устаткування зі ЧПК може бути представлено:

1. Верстатним парком, наприклад верстатами (верстати, обладнані числовим програмним керуванням, називаються верстатами з ЧПК):

– для обробки металів (наприклад, фрезерні або токарські), дерева, пластмас,

– для різання листових заготівель,

– для обробки тиском і т.д.

2. Приводами асинхронних електродвигунів, що використовують векторне керування;

3. Характерною системою керування сучасними промисловими роботами.

Кілька верстатів зі ЧПК можуть об'єднатися в гнучку автоматизовану виробничу систему (ГВС), що у свою чергу може бути доповнена гнучкою автоматизованою ділянкою (ГАД) і ввійти до складу автоматичної лінії (виробництва масштабу ділянки або цеху), ГАВ.

1.2 Область застосування

Винахідником першого верстата із числовим (програмним) керуванням (англ. Numerical Control, NC) є Джон Персонс (John T. Parsons), що працював інженером у компанії свого батька Parsons Inc, що випускала наприкінці Другої світової війни пропелери для вертольотів. Він уперше запропонував використовувати для обробки пропелерів верстат, що працює по програмі, що вводиться з перфокарт. В 1949 році ВВС США профінансували Parsons Inc розробку верстата для контурного фрезерування складних за формою деталей авіаційної техніки. Однак, компанія не змогла самостійно виконати роботи й звернулася по допомогу в лабораторію сервомеханіки Массачусетского технологічного інституту (MIT). Співробітництво Parsons Inc з MIT тривало до 1950 року. В 1950 році MIT придбав компанію по виробництву фрезерних станків Hydro-Tel і відмовився від співробітництва з Parsons Inc, уклавши самостійний контракт із ВВС США на створення фрезерного верстата із програмним керуванням. У вересні 1952 року верстат був уперше продемонстрований публіці – про нього була надрукована стаття в журналі Scientific American. Верстат управлявся за допомогою перфострічки. Перший верстат зі ЧПК відрізнявся особливою складністю й не міг бути використаний у виробничих умовах. Перший серійний пристрій ЧПК було створено компанією Bendix Corp. в 1954 році й з 1955 року стало встановлюватися на верстати. Широке впровадження верстатів зі ЧПК йшло повільно. Підприємці з недовірою ставилися до нової техніки. Міністерство оборони США змушено було за свої кошти виготовити 120 верстатів зі ЧПК, щоб передати їх в оренду приватним компаніям.

Першими вітчизняними верстатами зі ЧПК промислового застосування є токарно-гвинторізний верстат 1К62ПУ й токарно-карусельного 1541П. Ці верстати були створені в першій половині 1960-х років. Верстати працювали разом з керуючими системами типу ПРС-3К и іншими. Потім були розроблені вертикально-фрезерні верстати зі ЧПК 6Н13 із системою керування «Контур-3п». У наступні роки для токарських верстатів найбільше поширення одержали системи ЧПК вітчизняного виробництва 2Р22 і Електроніка НЦ-31.

Числове програмне керування також характерно для систем керування сучасними промисловими роботами.

Абревіатура ЧПК відповідає двом англомовним – NC і CNC, – які відображають еволюцію розвитку систем керування встаткуванням.

1. Системи типу NC (англ. Numerical control), що з'явилися першими, передбачали використання жорстко заданих схем керування обробкою – наприклад, завдання програми за допомогою штекерів або перемикачів, зберігання програм на зовнішніх носіях. Яких-небудь пристроїв оперативного зберігання даних, керуючих процесорів не передбачалося.

2. Більше сучасні системи ЧПК, називані CNC (англ. Computer numerical control) – системи керування що дозволяють використовувати для модифікації існуючих/написання нових програм програмні засоби. Базою для побудови CNC служать сучасний (мікро)контролер або (мікро)процесор:

– мікроконтролер;

– контролер із програмувальною логікою;

– керуючий комп'ютер на базі мікропроцесора.

Можлива реалізація моделі із централізованим автоматизованим робочим місцем (наприклад, ABB Robot Studio, Microsoft Robotics Developer Studio) з наступним завантаженням програми за допомогою передачі по промисловій мережі.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи верстата ЧПК обробки електроізоляційного картону, 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

Програма для фрезерного верстата JDPaint

Характеристики:

– Призначення: 3D, 2D моделювання, створення й редагування будь-яких складних поверхонь, машинна обробка.

– Інтерфейс: український.

– Тип вихідного файлу: G-code, легке ручне редагування постпроцесору.

– Розширення вихідного файлу: .nc і .eng.

– Імпортовані об'єкти: stl, igs, wrl, obj (3D Max, 3D Studio Max).

– Імпортовані вектора: eps, ai, plt, dxf, gbr.

– Імпортовані зображення: jpg, bmp, png, tif, pcx, убудований векторизатор фотозображень.

– Підтримка форматів: 3-х , 4-х осьова обробка, 5-ти осьова-опційно.

– Редактор NC траєкторій: убудований як для власних траєкторій, так і для сторонніх по форматі FANUC.

– Ексклюзивна функція згинання готової траєкторії (можливість накласти створену траєкторію по площині на будь-який 3D рельєф або 3D форму, тим самим одержавши амплітудно модульовану траєкторію фрези відповідно до рельєфу.

– Підтримка експорту: IGES для повнооб’ємних 3D об'єктів, і Z-растр – для однобічних 3D об'єктів.

– Гравірування: векторна й точкова з тінями (для фотозображень).

– Обробка: чорнове, фінішна, різання, розкрій, 2D і 3D, комбінована, з доробкою низів і боків рельєфу, точкова, зіркою й ін.

– Фрези: Торцеві, сферичні, торцеві з радіусом, конусні сфери, конусні торці, конусний радіус, звичайний конус.

– Режими: ручний скульптор, векторизація фото, створення однобічних і повнооб’ємних моделей, створення будь-яких повнооб’ємних поверхонь, сіток і т.д.

– Скульптор: наявність скульптора для створення й редагування рельєфів/барельєфів і 3D скульптора для створення й редагування повнооб’ємних 3D скульптур і об'єктів .

– Ексклюзивні функції: підсвічування висот рельєфу різним кольором, автообмеження висот рельєфу при ручний скульптуризації, усереднення висот рельєфу в різних точках по обраній висоті, протягання мишкою обраної висоти в необхідну сторону, автосглажування сторони рельєфу в місці підрізування, убудовані плагіни створення цегельної кладки, черепиці, і інші скульптурні 3D арт-эффекти. Можливість створення кольорового фотореалістичного рельєфу.

– Зручності: налаштовується користувачем меню під правою кнопкою мишки. Ця сервісна можливість програми дозволяє користувачеві самостійно створити своє меню з будь-яких часто використовуваних функцій, як наприклад: копіювати/вставити, згрупувати/розгрупувати, перетворити в лінію/дугу/криву безье, видалити/додати точку, розірвати, згладити, зростити й т.д. будь-які потрібні функції програми під правою кнопкою мишки. Крім того, програма запам'ятовує також будь-які дві останні використані функції (з векторного блоку, скульптора або блоку поверхонь) і виставляє їх автоматично в першу позицію спливаючого меню під правою кнопкою.

– ОС: Windows 10/11.

– Сумісність із англійськими версіями Windows: підтримується, але варто ретельно налаштувати підтримку української мови в системі, і якщо після всіх налаштувань якесь слово в програмі нечитаемо на українській, то перевірити укомплектованість англійської версії українськими шрифтами .

– Особливості: Це одна з маловідомих у нас програм для високопродуктивної роботи з верстатами ЧПК. У своєму комплекті містить масу рідких корисних функцій, які відсутні частково або повністю в інших подібних програмах. Коли ви Не маєте часу, щоб вивчити досконально безліч програм як Mastercam, Artcam, Rhino, 3Dmax, Type3, SolidWorks, AutoCAD, DeskProto, Cat3D – досить добре розібратися в JDPaint, і Ви з легкістю намалюєте все що завгодно й виготовите будь-який виріб на ЧПК верстаті, застосовуючи всього тільки одну програму для проектування, моделювання, візуалізації й створення траєкторій.

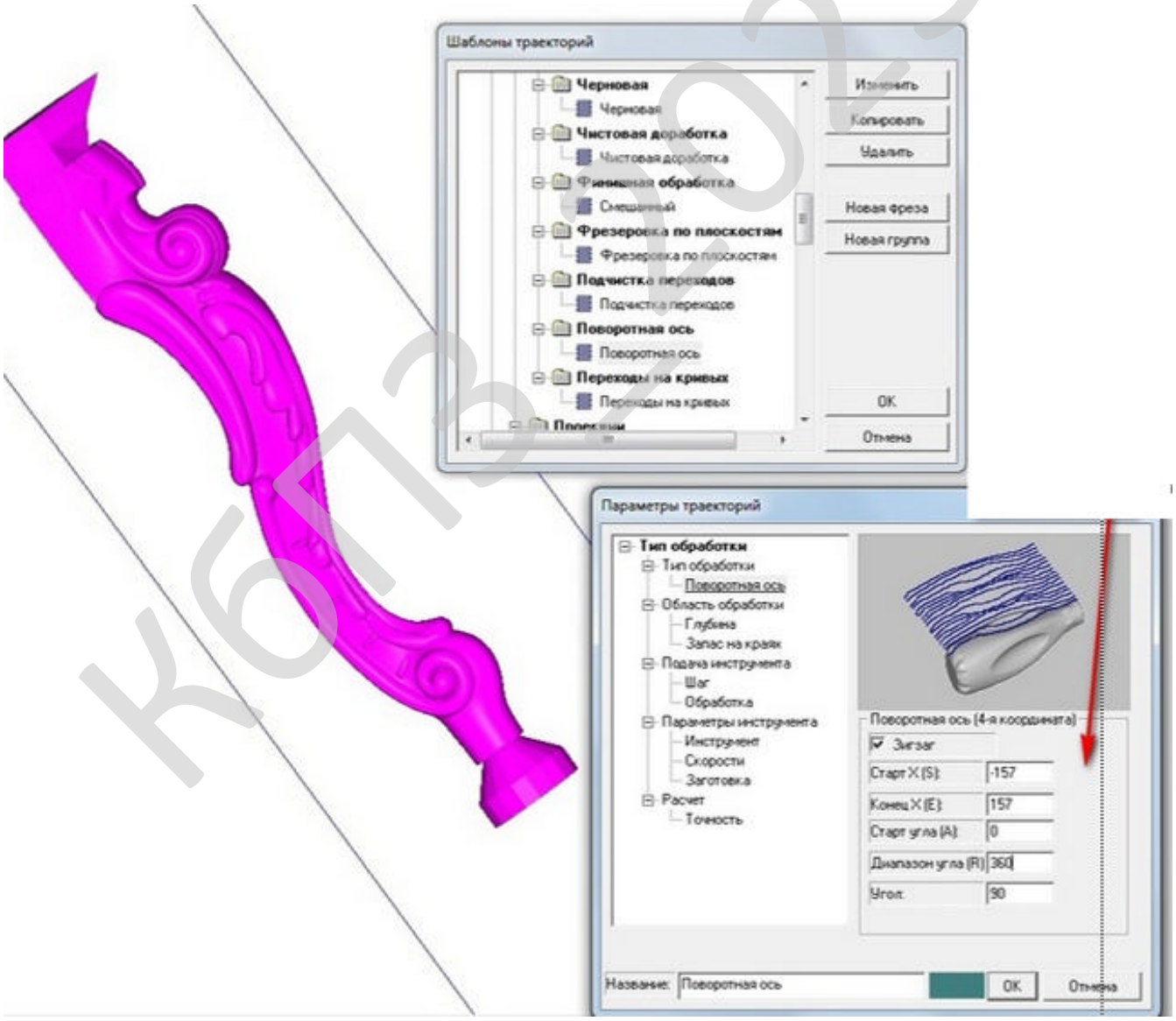

Рисунок 2.1 – Інтерфейс користувача JDPaint

MACH3

Дизайнер деталей використовує головним чином CAD/CAM програми. Видаваний цими програмами результат, називаний підпрограмою й найчастіше представлений у вигляді G-коду, передається (через мережу або на флоппі-диску) у контролер верстата. Контролер верстата відповідає за переклад підпрограми у вид, що підходить для керування різальним інструментом. Осі верстата рухаються за рахунок тросів або ременів, які пов'язані із серводвигунами або кроковими двигунами. Драйвери перевіряють, чи досить потужні й припустимі за часом сигнали, що надходять від контролера верстата, для того щоб оперувати двигунами.

Часто контролер верстата може управляти запуском і зупинкою обертання шпинделя (або навіть управляти його швидкістю), може включати й виключати охолодження а також перевіряти, чи не намагається підпрограма або оператор верстата вийти за межі осі. Контролер верстата також має такі керуючі компоненти як кнопки, клавіатура, ручне колесо генератора імпульсів (MPG), або джойстик, так що оператор може управляти верстатом вручну, і може запустити або зупинити виконання підпрограми. Контролер верстата має дисплей, так що оператор може бачити що відбувається в цей момент. У зв'язку з тим, що команда програми G-коду може запросити складний рух по координатах осей верстата, контролер верстата повинен уміти робити велику кількість обчислень у реальному часі (так вирізання еліпса вимагає великої кількості тригонометричних обчислень). Це зробило його досить дорогою частиною встаткування.

Mach3 це програмний пакет, що запускається на PC, і перетворює його в дуже потужний і економічний контролер верстата.

Для запуску Mach2 вам буде потрібно Windows 10/11. Ідеальна працездатність досягається на процесорі частотою 1GHz і екрані, що підтримує розширення 1024x768 точок. Настільний комп'ютер має більшу продуктивність ніж більшість мобільних аналогів, і звичайно має меншу вартість. Також ви зможете використовувати цей комп'ютер для рішення інших завдань, таких як

запуск і робота з CAD/CAM програмами, коли він не використовується для керування верстатом.

Mach2 працює через один (або за бажанням два) паралельні (принтерні) порти, або, за бажанням, через послідовний (COM) порт.

Драйвера ваших двигунів осей повинні розуміти крокові імпульси й сигнали напрямків.

Теоретично всі драйвери крокові двигуни повинні працювати за такою схемою, так само як це роблять сучасні DC і AC сервосистеми із цифровими пристроями кодування. Майте через, що якщо ви конвертуєте старий NC верстат, чиї серводвигуни використовують стару систему позиціонування осей, то вам доведеться ставити новий привод на кожну вісь.

Огляд програми Mach2

Вам не знадобиться верстат для того щоб ознайомитися із програмою.

Установка

Mach2 являє собою один файл для автоматичної установки (у даній версії це біля 4-х мегабайт). Демонстраційна версія не має терміну дії, але має деякі обмеження по швидкості, обсягу виконуваної роботи й деяких спеціальних можливостей. Після придбання ліцензії, обмеження вже встановленої й настроєної демоверсії будуть зняті.

Скачування

Скачати демоверсію можна із сайту www.artofcnc.ca, використовуючи праву кнопку миші й пункт меню, що випадає, «Зберегти як...». Після того як файл завантажиться, ви можете запустити установку відразу за допомогою кнопки «Відкрити», або пізніше, двічі кликнувши по завантаженому файлі лівою кнопкою миші.

Установка

Поки що не потрібно підключати верстат. Якщо ви ще навичок, краще буде відключити його, виключивши комп'ютер і вийнявши відповідний коннектор з роз’єму на задній панелі. Тепер включите комп'ютер. Після того як

ви запустите завантажений файл, вам запропонують пройти звичні для Windows програм кроки установки, такі як прийняття ліцензійної угоди й вибір папки для установки. Після установки вам прийдеться перезавантажити комп'ютер перш ніж запускати програму Mach2.

Обов'язкове перезавантаження

Це перезавантаження обов'язкове. Якщо ви її не зробите, то стикнетеся з великими труднощами, які можна здолати тільки вручну видаливши драйвер через Панель Керування Windows. Тому перезавантажитеся відразу.

Незважаючи на те, що Mach2 виглядає як одиночна програма при її використанні, насправді вона складається із трьох частин: драйвер, установлюваний як частина Windows, так само як драйвер принтера або мережний драйвер, графічний інтерфейс і OCX, що приймає повідомлення від і посилає відповіді в графічний інтерфейс. Існують певні причини для використання цих трьох частин (наприклад експерт може написати власну програму, що буде управляти Mach2 без використання графічного інтерфейсу), але драйвер найважливіша й необхідна частина.

Mach2 повинна мати можливість посилати дуже точно вивірені за часом сигнали для того щоб управляти осями верстата. Windows же у свою чергу любить бути за головного, і запускати користувальницькі додатки в той час коли їй нема чого робити. Так що Mach2 не можна запускати як звичайний додаток. Вона повинна бути на нижчому рівні в Windows (у такий спосіб вона одержує можливість контролювати переривання). Далі для того щоб робити це на максимально можливій швидкості яка може знадобитися (до кожної осі можливо 45000 звернень до секунди), драйвер повинен мати можливість змінювати свій власний код. Windows цього не схвалює (так іноді поводяться віруси), тому треба подати запит на одержання спеціального дозволу. Цей процес вимагає перезавантаження. Тому якщо ви не перезавантажилися, то Windows видасть «Синій Екран Смерті», і драйвер буде ушкоджений. Єдиний вихід у такій ситуації, це вручну видалити драйвер. Також потрібно сказати що

перезавантаження потрібно тільки коли драйвер установлюється вперше. Якщо ви обновляєте систему за допомогою нової версії, то перезавантаження не є обов'язковою. Але установник усе-же попросить вас неї зробити. Так як Windows 10/11 завантажується досить швидко, те перезавантаження при кожному відновленні не зажадає багато зусиль.

Іконки робочого стола

Отже ви перезавантажилися. Майстер установки створить на робочому столі іконки для головних програм. Mach2.exe це програма вибору користувальницького інтерфейсу. При запуску, вона запитає який профіль ви хочете використовувати. Mach2Mill, Mach2Turn і т.д. це іконки, що запускають певний профіль. Ними можна користуватися для запуску необхідної системи.

Тепер варто створити іконки для деяких інших програм Mach2. Використовуючи праву кнопку миші, перетягніть потрібні .exe файли на робочий стіл, і в меню, що з'явився, виберіть «Створити ярлики». Зробіть ярлики для програм OCXDriverTest.exe і KeyGrabber.exe.

Тестування установки

Тепер строго рекомендується протестувати систему. Mach2 не проста для освоєння. Вона використовує високі права доступу в Windows для своєї роботи. Це значить що вона може не працювати на деяких системах у зв'язку з багатьма факторами. Наприклад, системний монітор Quick Time (qtask.exe), що працює у фоновому режимі, може завершити виконання Mach2. Також є й інші програми які можуть зробити те-же саме. Windows може запустити й запускає безліч процесів у фоновому режимі. Деякі відображаються як іконки в панелі завдань, деякі взагалі ніяк не проявляють свою присутність. Інші можливі причини аварійного припинення роботи, це з'єднання по локальній мережі, які можуть бути настроєні на автоматичне визначення швидкості. Їх варто налаштувати на реальну швидкість вашої мережі, 10 або 100 Мбіт/с. Також на комп'ютері, що має доступ до Інтернету, можуть перебувати шпигунські модулі й програми, які стежать за вашими діями й відсилають цю інформацію з мережі своїм творцям.

Цей трафік може перетинатися з Mach2 і викликати різні збої в роботі. Використовуйте пошукові системи в Інтернеті щоб знайти програми, що очищають ваш комп'ютер від шпигунських модулів.

Беручи до уваги дані фактори, важливо, хоча й не обов'язково, перевірити вашу у випадку якщо вам здається що щось не так або ви просто хочете переконатися що установка пройшла успішно. Двічі кликніть лівою кнопкою миші по іконці OCXDriverTest яку ви створили на робочому столі. Можете проігнорувати всі діалогові вікна із заголовком Pulse Frequency. За замовчуванням його значення повинне перебувати в межах 24,600Hz, але воно може варіюватися, на деяких системах навіть у дуже широких межах. Це не означає що таймер імпульсу можливо нестабільний, це може означати що комп'ютер завантажений іншими завданнями або спочатку працює повільніше. Так як Mach2 має найвищий пріоритет у системі, таймер може бути загальмований досить сильно, щоб мати більшу погрішність. Так як підрахунок імпульсу базується на одній секунді часу Windows, зміни в часі можуть спотворити підрахунок імпульсу, так що він він буде виглядати сильно плаваючим, навіть якщо насправді він гранично точний. За замовчуванням, якщо ви бачите вікно, схоже на рисунок 2.1, значить все працює нормально, так що можете закрити програму OCXDriverTest і перейти до розділу екрани.

Люди добре розбираються в Windows, можуть зацікавитися наступної. Біле прямокутне вікно це щось начебто аналізатора часових вимірів. Коли запущено, воно показує лінію з невеликими зафіксованими відхиленнями. Ці відхилення – це зміни в часових вимірах від одного циклу переривання до іншого. Не повинне бути ліній, більш довгої ¼ дюйма, або біля того на 17-ти дюймових моніторах більшості систем. Навіть якщо спостерігаються розходження, можливо вони менше тих, які можуть створити неточності в часових замірах. Тому коли ви підключите ваш верстат, варто провести тестування руху, щоб перевірити чи є рухи прогону G0/G1 гладкими.

При запуску тестування ви можете зштовхнутися із двома проблемами, які вказують на неполадку.

1. «Драйвер не знайдений або не встановлений. Зв'яжіться з розроблювачем», це означає, що з якоїсь причини драйвер не був установлений. Це може відбутися на системах з Windows 10/11, на яких трапилося ушкодження бази даних драйверів. Для усунення проблеми, потрібно перезавантаження Windows. В Windows 10/11 є помилка, що втручається в завантаження драйвера. Можливо буде потрібно завантажити драйвер вручну. Докладніше в наступному пункті.

2. Коли система виводить повідомлення «taking over... 3...2... 1», після чого перезавантажується, можливо, відбулося наступне. Або ви не перезавантажилися коли вам запропонували (а ми адже попереджали!!), або драйвер ушкоджений або не може бути використаний у вашій системі. У цьому випадку, дотримуючись інструкцій у наступній секції, видалите драйвер вручну а потім установите його заново. На деяких системах установлені материнські плати, що апаратно підтримують APIC таймер, але BIOS яких її не використовує. Це може перешкодити установці Mach2. Для такого випадку існує файл specialdriver.bat, що запускається з вікна DOS. Це змусить драйвер використовувати старий контролер переривань i8529. Вам прийдеться повторювати це процес щораз при установці відновлень, так як ця установка замінить спеціальний драйвер.

Тестування драйвера після аварійного завершення Mach2

Під час роботи з Mach2 можливі ситуації, коли трапляється аварійне завершення роботи. Це може бути пов'язане з часовою неполадкою встаткування або із програмною помилкою. У такому випадку потрібно запустити OCXTest якнайшвидше. Якщо ви не запустите його в плині двох мінут з моменту аварійного завершення, Windows покаже «Синій екран смерті». Запуск OCXTest скине драйвер, і він повернеться до стабільного стану, навіть якщо Mach2 раптово закрилася.

Ручна установка й видалення драйвера

Цю частину варто читати тільки якщо вам не вдалося успішно запустити програму OCXDriverTest. Драйвер (Mach2.sys) може бути вручну встановлений або вилучений через панель керування Windows. Кроки установки:

Відкрийте панель керування й два рази кликніть на іконці або рядку «Система».

Виберіть «Устаткування» і натисніть «Установка встаткування». (Як раніше згадувалося, драйвер Mach2 працює на нижчому рівні Windows) Windows зробить пошук нових пристроїв, але нічого не знайде.

Скажіть, що ви вже встановили встаткування, і переходите до наступного діалогу.

Вам покажуть список устаткування. Пролистайте долілиць список й виберіть «Додати нове обладнання». Натискайте Next.

У наступному діалозі вам не потрібно щоб Windows шукав драйвер, тому натисніть «Вибрати встаткування вручну зі списку».

Показаний список повинен містити запис Mach1/2. Виберіть її й натискайте Next.

Натисніть «Установити з диска», і вкажіть шлях до папки, у якій у вас установлений Mach2 (За замовчуванням це C:\Mach2). Windows повинен знайти файл Mach2.inf. Виберіть цей файл і натисніть «Відкрити». Windows установить драйвер.

Видалити драйвер набагато простіше.

Відкрийте панель керування й два рази кликніть на іконці або рядку «Система».

Виберіть «Устаткування», потім «Диспетчер пристроїв».

Вам покажуть список пристроїв і їхніх драйверів. Під Mach1 Pulsing Engine перебуває драйвер Mach2 Driver. При необхідності, натисніть + для розкриття деревоподібної структури. Клацнувши правою кнопкою миші на

Mach2 Driver, ви побачите опцію для його видалення. Це видалить файл Mach2.sys з каталогу Windows. Копія в каталозі Mach2 видалена не буде.

Ще одне: Windows зберігає всю інформацію про те, як ви настроїли Mach2 у файлі профілю. Ця інформація не стирається при видаленні драйвера або видаленні інших файлів Mach2, так що вона залишиться після відновлення системи. Однак, у тому випадку якщо вам буде потрібно встановити Mach2 з нуля, тоді вам доведеться видалити файл(и) профілю з розширенням .XML.

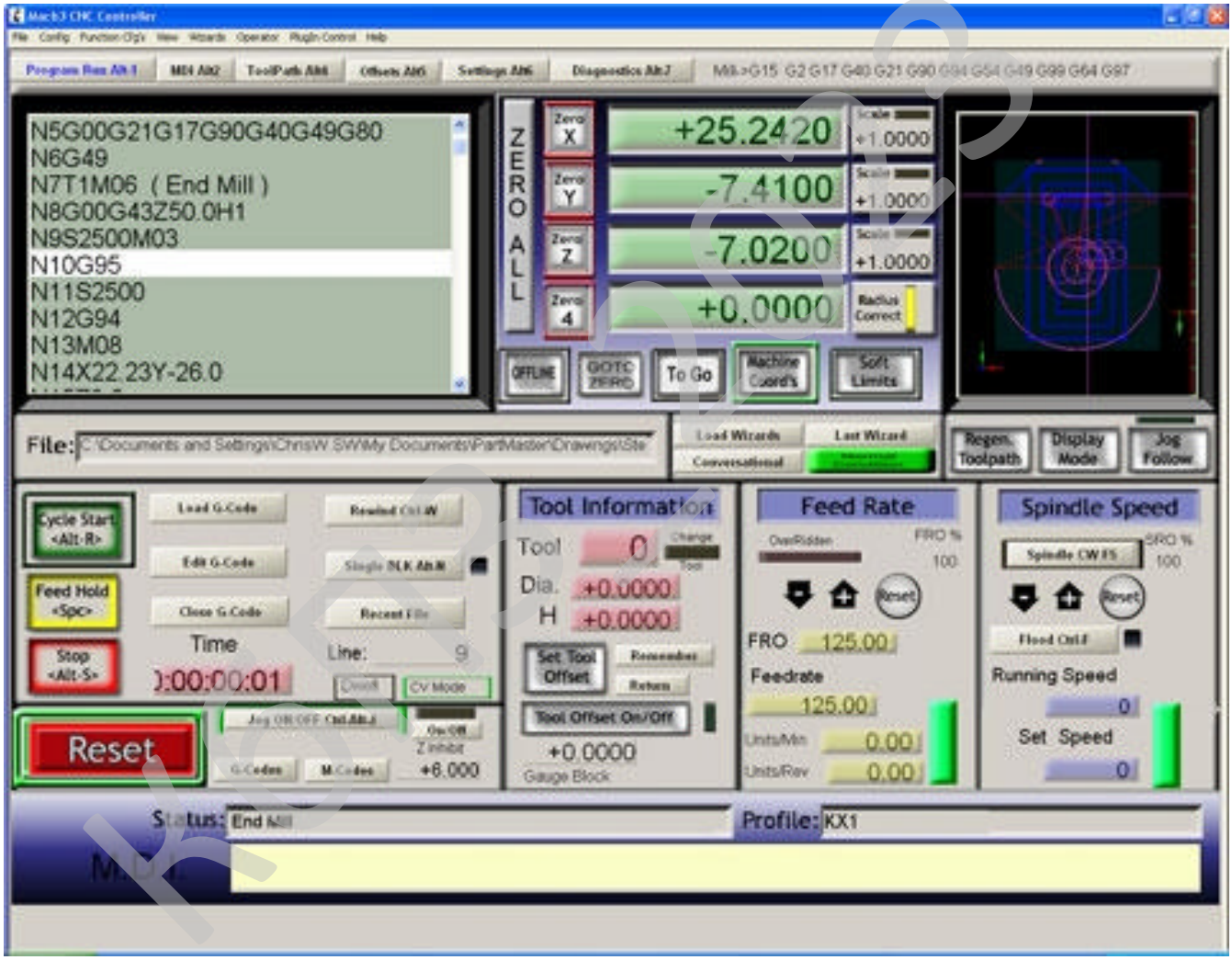

Рисунок 2.2 – Інтерфейс користувача Mach2

SprutCAM

SprutCAM єдина українська CAM-система, і одна з деяких серед закордонних, підтримуючу розробку управляючих програм (УП) для багатокоординатного, електроерозійного й токарно-фрезерного встаткування з обліком повної кінематичної 3D-моделі всіх вузлів у тому числі.

SprutCAM дозволяє створювати 3D-схеми верстатів і всіх його вузлів і робити попередню віртуальну обробку з контролем кінематики й 100 % вірогідністю, що дозволяє наочно програмувати складне багатокоординатне встаткування. Зараз для вільного використання доступні більше 45 схем різних типів верстатів.

SprutCAM використовується в метало-, дерево-, що обробляє промисловості; для електроерозійної, фрезерної, токарської, токарно-фрезерної, лазерної, плазменої і газової обробки; при виробництві оригінальних виробів, штампів, прес-форм, прототипів виробів, деталей машин, шаблонів, а також гравірування написів і зображень.

SprutCAM успішно впроваджувався з різним устаткуванням будь-якої складності (AGIE, MoriSeiki, PolyGim, Deckel, Homag, Hermle, HAAS, Okuma і ін.).

Характеристики:

– Широкий набір операцій 2, 2.5, 3 і 5 координатні обробки.

– Операції електроерозійної, токарської, фрезерної й токарно-фрезерної обробки.

– Мінімальна трудомісткість розробки УП.

– Оптимальні траєкторії інструмента.

– Сумісність і інтеграція із сучасними CAD системами (SolidWorks ™, КОМПАС ™, SolidEdge ™, Rhinoceros ™, Autodesk AutoCAD ™, Alibre Design ™, Cobalt ™).

– Убудований генератор постпроцесорів, що дозволяє зробити налаштування на будь-яку систему ЧПК.

– Розрахунок траєкторії практично з будь-якою точністю.

– Автоматичний контроль оправлення інструмента.

– Дружній інтерфейс.

– Велика бібліотека готових постпроцесорів.

– Реалістична симуляція процесу обробки.

– Легкість у використанні.

– Одержання результату відразу після установки системи.

– Невисокі вимоги до конфігурації комп'ютера.

– Навчання, сервіс, документація, підтримка, «гаряча лінія».

– Безкоштовне відновлення в межах версії.

– Швидка окупність капіталовкладень.

– Незалежно від типу вашого виробництва й галузі, SprutCAM® забезпечує цілий ряд автоматичних функцій, що дозволяють швидко й ефективно використовувати сучасну технологію розробки УП

Конфігурації:.

– SprutCAM «Різання». 2-х координатні операції газо-, плазмо-, лазерного різання.

– SprutCAM «Електроерозія». 2, 4-х координатні електроерозійні операції.

– SprutCAM «Токар». Операції токарської обробки.

– SprutCAM «Машиніст». 2-х координатні операції газо-, плазмо-, лазерного різання. 2.5 координатні фрезерні й гравірувальні операції.

– SprutCAM «Універсал». 2-х координатні операції газо-, плазмо-, лазерного різання. Операції токарської обробки. 2.5 координатні фрезерні й гравірувальні операції. 2, 4-х координатні електроерозійні операції.

– SprutCAM «Експерт». 2-х координатні операції газо-, плазмо-, лазерного різання. 2, 4-х координатні електроерозійні операції. 2.5, 3-х фрезерні й гравірувальні операції. 5 (3+2) координатні операції.

– SprutCAM «Майстер». 2-х координатні операції газо-, плазмо-, лазерного різання. Операції токарської обробки. 2.5, 3-х фрезерні й гравірувальні операції. 5

(3+2) координатні операції. 5 координатні операції. Обробка на токарно-фрезерному встаткуванні.

– SprutCAM «Профи». 2-х координатні операції газо-, плазмо-, лазерного різання. 2-х, 4-х координатні електроерозійні операції. Операції токарської обробки. 2.5, 3-х фрезерні й гравірувальні операції. 5 (3+2) координатні операції. 5 координатні операції. Обробка на токарно-фрезерному встаткуванні.

В SprutCAM доступні для вільного використання близько 600 налагоджених постпроцесорів. [1].

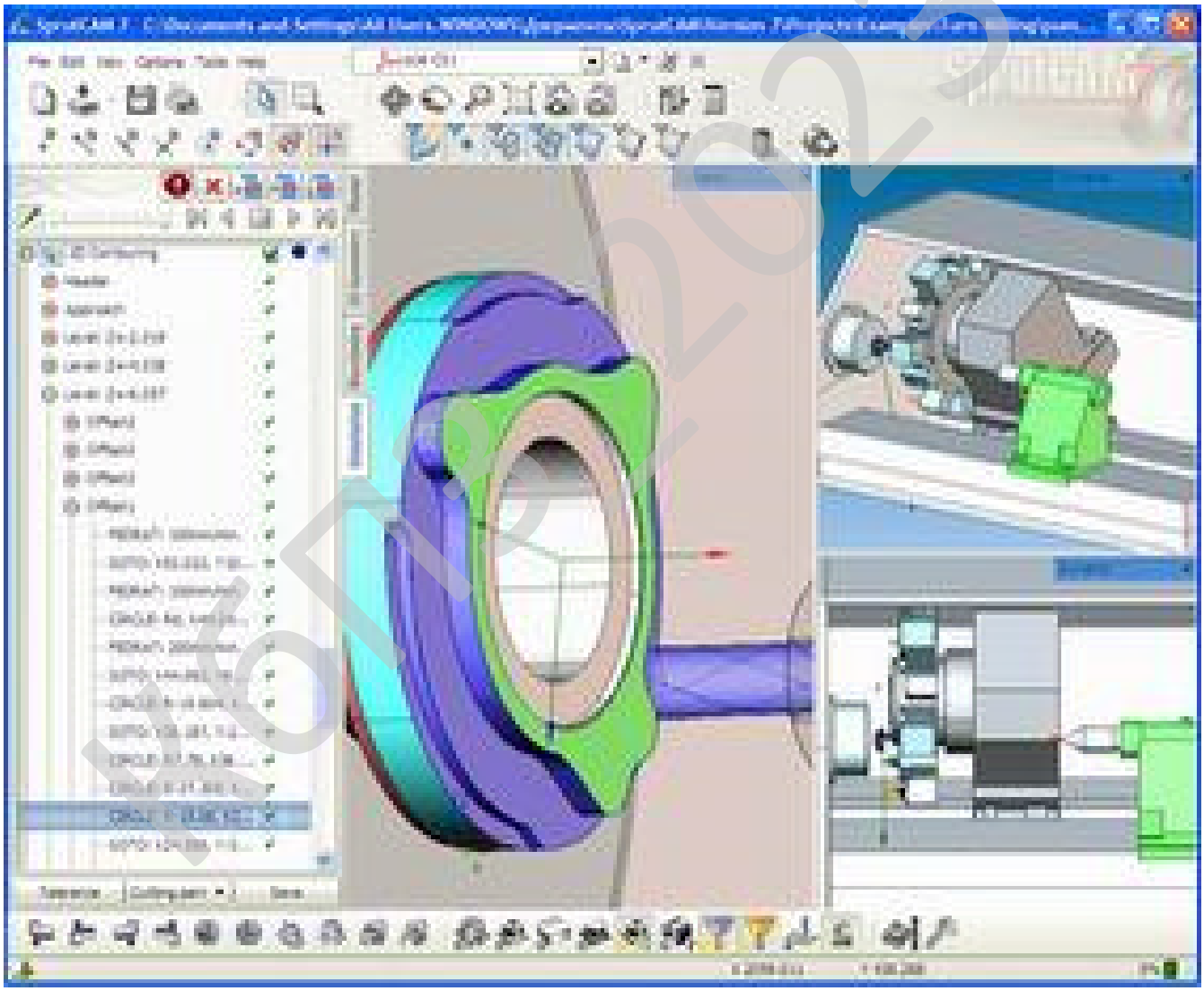

Рисунок 2.3 – Інтерфейс користувача SprutCAM

В SprutCAM реалізована інтеграція по прямій передачі даних з наступними CAD-системами:

– AutoCAD.

– Alibre Design.

– Inventor.

– Rhinoceros.

– SolidEdge.

– SolidWorks – статус сертифікованого партнера 'Solution Partner'.

– КОМПАС-3D.

SprutCAM поширюється й успішно використовується в більш ніж 45 країнах по усьому світі.[2] Серед користувачів SprutCAM Samsung, Hitachi, HP, Intel, Philips, HITTITE, Chemtura [3][4], а також велика кількість вищих навчальних закладів України, США, Великобританії, Єгипту й інших країн [5].

ADEM

ADEM (англ. Automated Design Engineering Manufacturing) – українська інтегрована CAD/CAM/CAPP система, призначена для автоматизації конструкторсько-технологічної підготовки виробництва (КТПП).

Розробка системи була почата в 90-х роках двома основними групами розроблювачів з Москви (конструкторський САПР «CherryCAD» – лауреат премії Ради Міністрів СРСР 1990 року) і Іжевська (технологічний САПР «Катран»).

ADEM був створений, як єдиний продукт, що включає в себе інструментарій для проектантів і конструкторів (CAD), технологів (CAPP) і програмістів ЧПК (CAM). Тому він містить декількох різних предметно-предметно-орієнтованих САПР під єдиною логікою керування й на єдиній інформаційній базі.

ADEM дозволяє автоматизувати наступні види робіт:

– об'ємне й плоске моделювання й проектування.

– оформлення проектно-конструкторської й технологічної документації.

– проектування технологічних процесів.

– аналіз технологічності й нормування проекту.

– програмування встаткування зі ЧПК (фрезерне, токарське, електроерозійне, лазерне й ін.).

– ведення архівів документів.

– реновацію знань (робота зі сканованими кресленнями й старими програмами ЧПК).

– підготовку актуальних даних для MES і ERP систем.

ADEM застосовується в різних галузях: авіаційної, атомної, аерокосмічної, машинобудівної, металургійної, верстатобудівної й інших.

NX CAM

NX CAM – система автоматизованої розробки керуючих програм для верстатів зі ЧПК (числовим програмним керуванням) від компанії Siemens PLM Software.[1].

Спочатку PLM система NX звалася Unigraphics. ПО Unigraphics було розроблено компанією United Computing. В 1976 компанія McDonnell Douglas (сьогодні Boeing) придбала United Computing і згодом була утворена McDonnell Douglas Automation Unigraphics Group. Компанія EDS придбала даний бізнес в 1991. Після придбання EDS компанії Structural Dynamics Research Corporation (SDRC) (англ.) в 2001, продукт Unigraphics був об'єднаний із САПР системою I-DEAS, розробленої SDRC. Поступове додавання функціональних можливостей I-DEAS в основний код системи Unigraphics стало основою існуючої лінійки продуктів NX.

NX CAM – ключовий компонент системи технологічного проектування, що також містить у собі засобу передачі інформації у виробництво й багато інших функцій.

Набір засобів для програмування верстатів зі ЧПК дозволяє застосовувати NX CAM у найрізноманітніших галузях. NX CAM впроваджений і використовується в авіаційно-космічній і оборонній промисловості,

автомобілебудуванні, машинобудуванні, виробництві споживчих товарів, медичного встаткування й багатьох інших галузях.[4].

NX CAM поставляється і як окреме робоче місце для програмування обробки, і як CAD/CAM система, а також може включати систему керування технологічними даними й бібліотеками інструментів. NX CAM підтримує спільну роботу з додатками конструкторського проектування NX, утворюючи єдине рішення. NX CAM поставляється із трансляторами, убудованими засобами візуалізації обробки, редактором постпроцесорів.

Виготовлення виробів зі складною геометрією зовнішніх обводів вимагає відповідного програмного забезпечення для розрахунку керуючої програми для верстата зі ЧПК. Залежно від складності деталі застосовується токарська обробка, фрезерна обробка на верстатах із трьома-п'ятьома керованими осями, токарно-фрезерна, електроерозійна обробка дротом. Система NX CAM має всі можливості для формування траєкторій інструмента для відповідних типів обробки.[5][6].

NX CAM має широкий набір убудованих засобів автоматизації – від майстрів і шаблонів до можливостей програмування обробки типових конструктивних елементів.

Генератор програм ЧПК містить у собі стратегії обробки, призначені для створення програм з мінімальною участю інженера.

Концепція майстер-моделі є базою, на якій будується розподіл даних між модулем проектування й інших модулів NX, у тому числі й модулями CAM. Асоціативний зв'язок між вихідною параметричною моделлю й сформованою траєкторією інструмента робить процес відновлення траєкторії швидким і легенею.

Для того щоб програму можна було запустити на певному верстаті, необхідно неї перетворити в машинні коди даного верстата. Це робиться за допомогою постпроцесору. У системі NX існує спеціальний модуль для налаштування постпроцесору для будь-яких керуючих стійок і верстатів зі ЧПК. Основні налаштування виконуються без використання програмування, однак

можливе підключення спеціальних процедур мовою Tcl, що відкриває широкі можливості по внесенню в постпроцесор будь-яких необхідних унікальних змін.

Для складного багатофункціонального встаткування компанія Siemens PLM Software розробила концепцію пакета підтримки верстата. У такий пакет входить не тільки постпроцесор, але й 3D-модель верстата, драйвер симуляції на основі G-кодів, шаблони типових технологічних операцій, приклади деталей і документація. Крім того, для Siemens Sinumerik 840D підтримується симуляція обробки із застосуванням віртуального контролера [7].

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

G-код

В цілому програма написана з використанням G-коду складається з кадрів, кожен кадр містить набір команд управління. Команди управління можуть слідувати в кадрі в будь-якому порядку, але зазвичай в цілях зручності прочитання керуючої програми системи числового програмного керування спочатку йдуть підготовчі команди, потім команди управлінням переміщенням, слідом команди вибору режимів обробки матеріалу і завершують кадр – технологічні команди.

Починається і закінчується текст керуючої програми символом «%». Далі може слідувати назва програми після символу «О». Коментарі в тексті керуючої програми розміщуються або у круглих дужках, або починаються з символу «;».

Кожна керуюча команда може мати один або декілька параметрів, які позначаються літерами латинського алфавіту.

Delphi 10.4 Sydney

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.

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 верстата ЧПК обробки електроізоляційного картону.

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

Апаратне забезпечення

Структурно, до складу ЧПК входять:

– пульт оператора (або консоль вводу/виводу), що дозволяє вводити керуючу програму, задавати режими роботи; виконати операцію вручну. Як правило, усередині шафи пульта сучасної компактної ЧПК, розміщаються її інші частини;

– дисплей (або операторська панель) – для візуального контролю режимів роботи й керуючої програми, що редагується / даних; може бути реалізований у вигляді окремого пристрою для дистанційного керування встаткуванням;

– контролер – комп'ютеризований пристрій, що вирішує завдання формування траєкторії руху різального інструменту, технологічних команд керування пристроями автоматики верстата, загальним керуванням, редагування керуючих програм, діагностики й допоміжних розрахунків (траєкторії руху різального інструменту, режимів різання);

– ПЗП – пам'ять призначена для довгострокового зберігання (роки й десятки років) системних програм і констант; інформація із ПЗП може тільки зчитуватися;

– ОЗП – пам'ять призначена для часового зберігання керуючі програми й системні програми, використовуваних у цей момент.

У ролі контролера виступає промисловий контролер як то: мікропроцесор, на якому побудована система, що вбудовується; програмувальний логічний контролер або більш складний пристрій керування – промисловий комп'ютер.

Важливою характеристикою CNC-контролера є кількість осей (каналів), які він здатний синхронізувати (управляти) – для цього потрібна висока продуктивність і відповідне ПЗ.

Як виконавчі механізми використовуються сервоприводи, крокові двигуни.

Для передачі даних між виконавчим механізмом і системою керування верстатом звичайно використовується промислова мережа (наприклад, CAN, Profibus, Industrial Ethernet).

Програмне забезпечення

Після того як створена управляюча програма, оператор за допомогою программатора вводить її в контролер. Команди керуючої програми розміщаються в ОЗП. У процесі створення або після вводу керуючої програми оператор (у даному аспекті виконуючий роль програміста) може відредагувати її, включивши в роботу системну програму редактори й виводячи на дисплей всю або потрібні частини керуючої програми й вносячи в них необхідні зміни. При роботі в режимі виготовлення деталі управляюча програма кадр за кадром надходить на виконання. Відповідно до команд керуючої програми контролер викликає із ПЗП відповідні системні підпрограми, які змушують працювати підключене до ЧПК встаткування в необхідному режимі – результати роботи контролера у вигляді електричних сигналів надходять на виконавчий пристрій – приводи подач, або на пристрої керування автоматикою верстата.

Керуюча система зчитує інструкції спеціалізованої мови програмування (наприклад, G-код) програми, що потім інтерпретатором системи ЧПК переводиться із вхідної мови в команди керування головним приводом, приводами подач, контролерами керування вузлів верстата (наприклад, включити/виключити подачу охолодної емульсії).

Розробка керуючих програм у цей час виконується з використанням спеціальних модулів для систем автоматизованого проектування (САПР) або

окремих систем автоматизованого програмування (CAM), які по електронній моделі генерують програму обробки.

Для визначення необхідної траєкторії руху робочого органа в цілому (інструмента/заготовки) відповідно до керуючої програми використовується інтерполятор, що розраховує положення проміжних точок траєкторії по заданим у програмі кінцевим.

У системі керування, крім самої програми, присутні дані інших форматів і призначення. Як мінімум, це машинні дані й дані користувача, специфічно прив'язані до конкретної системи керування або до певної серії (лінійки) однотипних моделей систем керування.

Програма для верстата (устаткування) зі ЧПК може бути завантажена із зовнішніх носіїв наприклад, магнітної стрічки, перфорованої паперової стрічки (перфострічки), дискети або флеш-накопичувачів у власну пам'ять або часово, до вимикання живлення – в оперативну пам'ять, або постійно – у ПЗП, карту пам'яті або інший накопичувач: жорсткий диск або твердотільний накопичувач. Крім цього, сучасне встаткування підключається до централізованих систем керування за допомогою заводських (цехових) мереж зв'язку.

Найпоширеніша мова програмування ЧПК для металорізального встаткування описана документом ISO 6983 Міжнародного комітету зі стандартам і називається «G-код». В окремих випадках – наприклад, системи керування гравірувальними верстатами – мова керування принципово відрізняється від стандарту. Для простих завдань, наприклад, розкрою плоских заготівель, система ЧПК як вхідну інформація може використовувати текстовий файл у форматі обміну даними – наприклад DXF або HPGL.

G-code

G-код – умовне іменування мови програмування пристроїв із числовим програмним керуванням (ЧПК). Був створений компанією Electronic Industries Alliance на початку 1960-х. Фінальна доробка була схвалена в лютому 1980 року як стандарт RS274D. Комітет ISO затвердив G-код, як стандарт ISO 6983-1:1982,

Держкомітет по стандартах СРСР – як ДСТ 20999-83. У радянській технічній літературі G-код позначається, як код ІСО 7-біт (ISO 7-bit), так як G-код для подання на перфострічці її кодували на 8-мі дорожню перфострічку в коді ISO 7-bit (розроблений для подання інформації у ЧПК у вигляді машинного коду також як і коди AEG і PC8C), восьма доріжка використовувалася для контролю парності.

Виробники систем ЧПК (CNC) як правило використовують софт керування верстатом, для якого написана (оператором) програма обробки як осмислені команди керування використовується G-код як базова підмножина мови програмування, розширюючи його за своїм розсудом.

Структура програми

Програма, написана з використанням G-коду, має тверду структуру. Всі команди керування поєднуються в кадри – групи, що складаються з однієї або більше команд. Кадр завершується символом перекладу рядка (CR/LF) і має номер, за винятком першого кадру програми й коментарів. Перший (а в деяких випадках ще й останній) кадр містить тільки один символ «%». Завершується програма командою M02 або M30. Коментарі до програми розміщаються в круглих дужках, як після програмних кодів, так і в окремому кадрі.

Порядок команд у кадрі строго не обмовляється, але традиційно передбачається, що першими вказуються підготовчі команди, (наприклад, вибір робочої площини), потім команди переміщення, потім вибору режимів обробки й технологічних команд.

Підпрограми можуть бути описані після команди M02, але до M30. Починається підпрограма з кадру виду Lxx, де xx – номер підпрограми, закінчується командою M17.

Зведена таблиця кодів

Основні (називані в стандарті підготовчими) команди мови починаються з букви G:

– Переміщення робочих органів устаткування із заданою швидкістю (лінійне й кругове).

– Виконання типових послідовностей (таких, як обробка отворів і різьблення).

– Керування параметрами інструмента, системами координат, і робітників площин.

Таблиця 3.1 – Підготовчі (основні) команди

Коди	Опис
G 00-G03	Позиціонування інструмента
G 17-G19	Перемикання робочих площин (XY, ZX, YZ)
G 20-G21	Не стандартизовано
G 40-G44	Компенсація розміру різних частин інструмента (довжина, діаметр)
G 53-G59	Перемикання систем координат
G 80-G85	Цикли свердління, розточування, нарізування різьблення
G 90-G91	Перемикання систем координат (абсолютна, відносна)

Таблиця 3.2 – Основні команди

Команда	Опис	Приклад
G00	Прискорене переміщення інструмента (холостий хід)	G0 X0 Y0 Z100
G01	Лінійна інтерполяція	G01 X0 Y0 Z100 F200
G02	Кругова інтерполяція за годинниковою стрілкою	G02 X15 Y15 R5 F200
G03	Кругова інтерполяція проти годинникової стрілки	G03 X15 Y15 R5 F200

Продовження таблиці 3.2

Команда	Опис	Приклад
G04	Затримка виконання програми, спосіб завдання величини затримки залежить від реалізації системи керування	G04
G15	Скасування полярної системи координат	G15 X15 Y22.5; G15;
G16	Полярна система координат (X радіус Y кут)	G16 X15 Y22.5
G17	Вибір робочої площини X-Y	
G18	Вибір робочої площини Z-X	
G19	Вибір робочої площини Y-Z	
G40	Скасування компенсації радіуса інструмента	G1 G40 X0 Y0 F200
G41	Компенсувати радіус інструмента ліворуч від траєкторії	G41 X15 Y15 D1 F100
G42	Компенсувати радіус інструмента праворуч від траєкторії	G42 X15 Y15 D1 F100
G43	Компенсувати довжину інструмента позитивно	G43 X15 Y15 Z100 H1 S1000 M3
G44	Компенсувати довжину інструмента негативно	G44 X15 Y15 Z4 H1 S1000 M3
G49	Скасування компенсації довжини інструмента	G49 Z100
G53	Відключити зсув початку системи координат верстата	G53 G0 X0 Y0 Z0
G 54-G59	Перемкнутися на задану оператором систему координат	G54 G0 X0 Y0 Z100
G70	Програмувати в дюймах	G70

Продовження таблиці 3.2

Команда	Опис	Приклад
G71	Програмувати в мм	G71
G80	Скасування циклів свердління, розточування, нарізування різьблення мітчиком і т.д.	G80
G81	Цикл свердління	G81 X0 Y0 Z-10 R3 F100
G82	Цикл свердління із затримкою	G82 X0 Y0 Z-10 R3 P100 F100
G83	Цикл переривчастого свердління (з повним виводом свердла)	G83 X0 Y0 Z-10 R3 Q8 F100
G84	Цикл нарізування різьблення	G95 G84 M29 X0 Y0 Z-10 R3 F1.411
G90	Завдання абсолютних координат опорних точок траєкторії	G90 G1 X0.5 Y0.5 F10
G91	Завдання координат інкрементально останньої уведеної опорної точки	G91 G1 X4 Y5 F100
G94	F (подача) – у форматі мм/хв.	G94 G80 Z100
G95	F (подача) – у форматі мм/хв.	G95 G84 X0 Y0 Z-10 R3 F1.411

максимум 4 команди в кадрі

Таблиця технологічних кодів

Технологічні команди мови починаються з букви M. Включають такі дії:

– Перемінити інструмент.

– Включити/виключити шпиндель.

– Включити/виключити охолодження.

– Робота з підпрограмами.

Таблиця 3.3 – Допоміжні (технологічні) команди

Код	Опис	Приклад
M00	Призупинити роботу верстата до натискання кнопки «старт» на пульті керування, так звана «безумовна технологічна зупинка»	G0 X0 Y0 Z100 M0
M01	Призупинити роботу верстата до натискання кнопки «старт», якщо включено режим підтвердження зупинки	G0 X0 Y0 Z100 M1
M02	Кінець програми, без скидання модальних функцій	M02
M03	Почати обертання шпинделя за годинниковою стрілкою	M3 S2000
M04	Почати обертання шпинделя проти годинникової стрілки	M4 S2000
M05	Зупинити обертання шпинделя	M5
M06	Перемінити інструмент	T15 M6
M07	Включити додаткове охолодження	M3 S2000 M7
M08	Включити основне охолодження. Іноді використання більше одного М-коду в одному рядку (як у прикладі) неприпустимо, для цього використовуються M13 і M14	M3 S2000 M8
M09	Виключити охолодження	G0 X0 Y0 Z100 M5 M9
M13	Включити охолодження й обертання шпинделя за годинниковою стрілкою	S2000 M13
M14	Включити охолодження й обертання шпинделя проти годинникової стрілки	S2000 M14
M17	Кінець підпрограми	M17
M25	Заміна інструмента вручну	M25

не більше одного коду в кадрі

Таблиця 3.3 – Допоміжні (технологічні) команди

Код	Опис	Приклад
M97	Запуск підпрограми, що перебуває в тій же програмі (де P – номер кадру, у випадку приклада перехід здійсниться до рядка N25),діє не скрізь, приблизно – тільки на верстатах HAAS	M97 P25
M98	Запуск підпрограми, що перебуває окремо від основної програми (де P – номер підпрограми, у випадку приклада перехід здійсниться до програми O1015)	M98 P1015
M99	Кінець підпрограми	M99
M30	Кінець програми, зі скиданням модальних функцій	M30

Параметри команд

Таблиця 3.4 – Параметри команд задаються буквами латинського алфавіту

Код	Опис	Приклад
X	Координата точки траєкторії по осі X	G0 X100 Y0 Z0
Y	Координата точки траєкторії по осі Y	G0 X0 Y100 Z0
Z	Координата точки траєкторії по осі Z	G0 X0 Y0 Z100
P	Параметр команди	G04 P101
F	Швидкість робочої подачі	G1 G91 X10 F100
S	Швидкість обертання шпинделя	S3000 M3
R	Параметр стандартного циклу або радіус дуги (розширення стандарту)	G81 R1 0 R2 − 10 F50 або G1 G91 X12.5 R12.5
D	Параметр корекції обраного інструмента	G1 G41 D1 X10. F150.
P	Число викликів підпрограми	L82 P10
I,J,K	Параметри дуги при круговій інтерполяції	G03 X10 Y10 I0 J0 F10
L	Виклик підпрограми з даною міткою	L12

DXF

DXF (англ. Drawing eXchange Format) – відкритий формат файлів для обміну графічною інформацією між додатками САПР. Був створений фірмою Autodesk для системи AutoCAD. Підтримується практично всіма CAD-системами на платформі PC.

DXF був уперше представлений у грудні 1982 року як частина AutoCAD 1.0, як обмінний формат даних, що надає ту ж інформацію, що й закритий внутрішній формат AutoCAD – DWG, специфікація на який ніколи не надавалася. У цей час на сайті Autodesk можна знайти специфікації всіх версій DXF, починаючи з AutoCAD Release 13 (листопад 1994 р.). Починаючи з AutoCAD Release 10 (жовтень 1988 р.) крім текстового варіанта DXF, підтримується також і двійкова версія – DXB.

У міру того, як AutoCAD ставав усе складнішим й підтримував усе більше складні типи об'єктів, DXF ставав усе менш корисний. Нові об'єкти в специфікації формату описувалися не повністю або не описувалися зовсім. Більшість розроблювачів комерційних додатків, включаючи конкурентів Autodesk, як основний формат обміну з AutoCAD використовують формат DWG, через бібліотеки, надавані некомерційною організацією Open Design Alliance, що була виконана зворотна розробка формату DWG.

Однак, для більшості практичних потреб об'єкти, що вводяться знову, такі як 3D-розширення, не є необхідними. Наприклад, відповідно до вимоги ЄСКД, креслення будь-якого виробу є двовимірним контурним зображенням (ДСТ 2.301-68, Формати). Тому DXF не тільки не зник, але став де-факто одним із двох стандартів для векторних зображень у відкритих операційних системах і додатках (другий стандарт – SVG). Наприклад (на 2009 рік), векторний графічний редактор Inkscape і САПР QCad підтримують DXF як основний формат.

HPGL

HPGL (іноді HP-GL) є основною мовою керування принтерами, використовуваним плотерами Hewlett-Packard. Його назва являє собою

абревіатуру Hewlett-Packard Graphics Language. У цей момент він є стандартним майже для всіх плотерів. Принтери Hewlett-Packard, як правило, також підтримують HPGL нарівні з PCL.

Таблиця 3.5 – Приклад HPGL файлу

Команда	Значення
IN;	ініціалізація процесу креслення
IP;	визначає початкову точку, у цьому випадку за замовчуванням 0,0
SC0,100,0,100;	установлює розміри сторінки від 0 до 100 у напрямках X і Y
SP1;	вибирає перо 1
PU0,0;	переміщає перо в початкову позицію
PD100,0,100,100,0,100,0,0;	опускає й рухає перо по заданих позиціях (креслить прямокутник навколо сторінки)
PU50,50;	піднімає й переміщає перо в позицію 50,50
CI25;	креслить окружність із радіусом 25
SS;	вибирає стандартний шрифт
DT*,1;	установлює як текстовий роздільник символ * і забороняє його печатка на папері (1 – «true»)
PU20,80;	піднімає й переміщає перо в позицію 20,80
LBHello World*;	креслить напис

Мова являє собою сполучення коду із двох букв і наступних за ним додаткових параметрів. Наприклад дуга (arc) може виводитися на печатку наступною командою:

AA100,100,50;

де AA – скорочення від Arc Absolute; 100,100 – координати центральної точки дуги; 50 – початковий кут, вимірюваний проти годинникової стрілки.

Четвертий параметр, невикористовуваний у цьому випадку, визначає кут малювання дуги й за замовчуванням дорівнює 5 градусам. Звичайно HPGL файли починаються з декількох команд, які встановлюють параметри, і тривають довгим списком графічних команд.

Координатна система була заснована на найменших одиницях, підтримуваних їх плотерами – 25 μm (тобто 40 одиниць на міліметр, 1016 на дюйм).

Координати задавалися числами із плаваючої коми в межах $\pm 2^{30}$.

HP-GL/2

Первісна мова HP-GL не підтримувала лінії різної ширини. Цей параметр визначався пір'ям, установлюваними в плоттер. З появою перших струминних плоттеров ширина ліній «пір'я», зазначених в HPGL-файлах, повинна була встановлюватися на принтері для кожного пера, що являло собою досить трудомісткий процес, що супроводжується частими помилками. В HP-GL/2 дана можливість була передбачена безпосередньо на рівні мови, що дозволило автоматизувати даний етап. Серед інших поліпшень був доданий бінарний формат. Це нововведення зменшило розмір файлів і час на їхню передачу. Також був поліпшений мінімальне розрішення.

3.2 Розробка структурної схеми

Структурна схема системи зображена на рисунку 3.1. Далі в тексті застосовуються наступні скорочення:

– ПО – пульт оператора.

– ПК – пульт корекції.

– ПВП – пристрій введення програми.

– ППМК – пристрій пам'яті мікрокоманд.

– АЛП – арифметико-логічний пристрій.

– НПОЗП – напівпровідниковий оперативний запам'ятовувальний пристрій.

– КСП – керування приводом, який стежить.

– ПЗВ – пристрій зв'язку з верстатом.

– ТП – тиристорні перетворювачі за координатами X, Z, I, K.

– БЗЗ – блок зворотного зв'язку.

– БПК – блок посилення команд S, T, M.

– М – двигуни приводів за координатами X, Z, I, K.

– ДЗЗ – датчик зворотного зв'язку.

– ВОВ – виконавчі органи верстату.

Інформація через пристрій уведення програми (ПВП) надходить у буферну частину напівпровідникового оперативного запам'ятовувального пристрою (НПОЗП). Одночасно із цим іде відпрацьовування верстатом попереднього кадру по алгоритмах лінійної або кругової інтерполяції за допомогою арифметико-логічного пристрою (АЛП).

Задана інформація про переміщення виконавчих органів верстата по осях X, Z, I, K надходить у пристрій керування приводами, що стежать, КСП, а технологічні команди S, T, M – у пристрій зв'язку з верстатом (ПЗВ) і на виконавчі органи верстата (ВОВ). Після закінчення відпрацьовування уведеного раніше кадру й за умови, що закінчено уведення наступного кадру, відбувається передача даних з буферної в робочу пам'ять НПОЗП, і цикл повторюється, тобто починається відпрацьовування наступного кадру й уведення нового. Всі алгоритми роботи пристрою ЧПК зберігаються в пристрої пам'яті мікрокоманд (ППМК), звідки вони в необхідній послідовності вибираються по запитах пристрою керування.

З пульта оператора (ПО) здійснюють ручне уведення додаткової інформації: команд, корекції виконання програми, індикації номера кадру, номери або положення інструмента й ін.

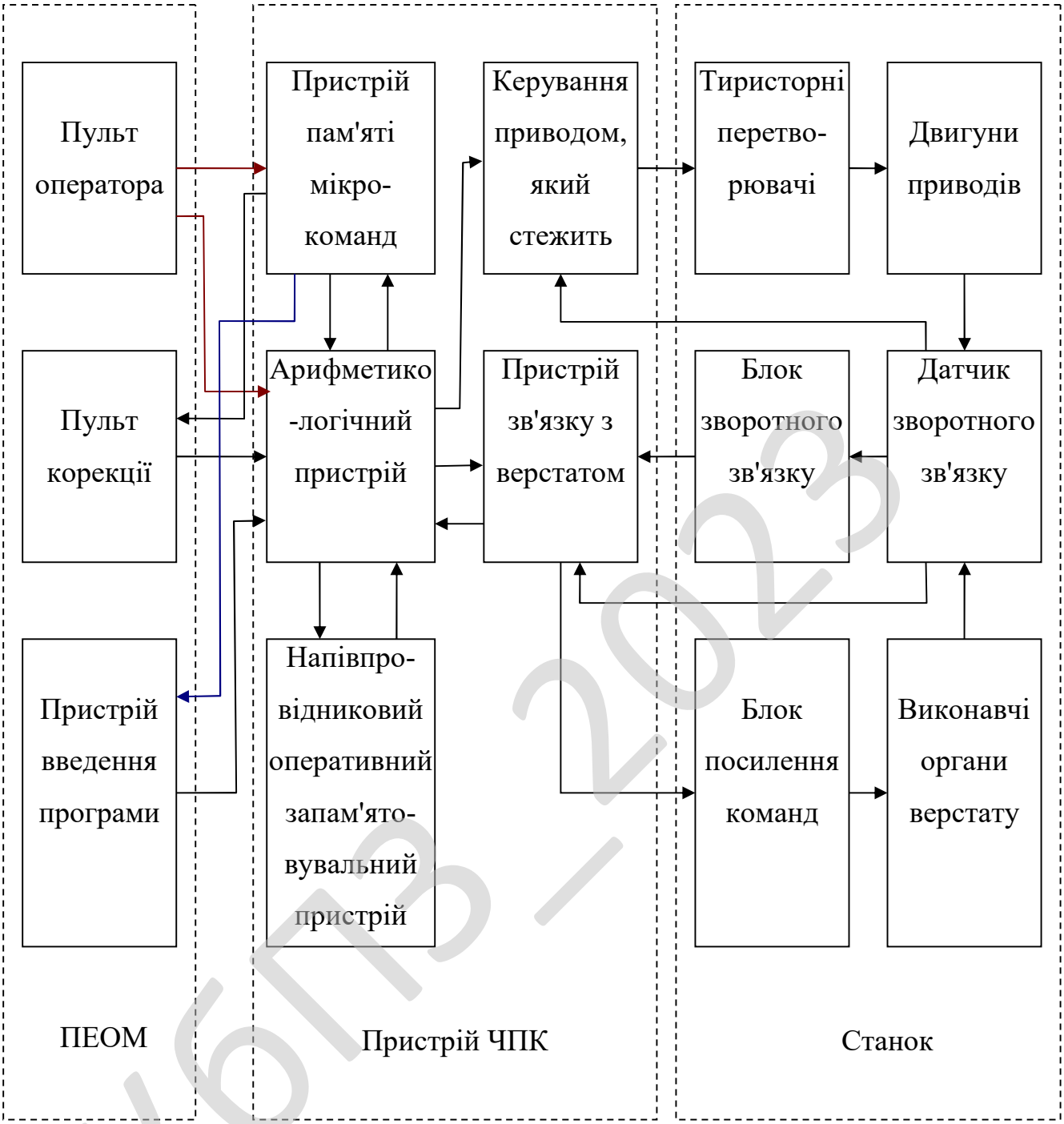

Рисунок 3.1 – Структурна схема системи

Інформацію про відпрацьовування команд, положенні й стані робочих органів верстата видають датчики зворотного зв'язка (ДЗЗ) по функціях команд у свої керуючі блоки КСП або ПЗВ, чим підтримується робота системи пристрій ЧПК-верстат по заданій керуючій програмі. У зв'язку з тим, що всі елементи електронно-обчислювального пристрою ЧПК працюють використовуючи

мікроструми й малі напруги, а виконавчі органи верстата працюють на промисловій напрузі 380 В, між системою ЧПК й приводами виконавчих органів верстата розміщають блоки керування для перетворення команд пристрою ЧПК в робочі напруги. Ці блоки об'єднані в електрошафі верстата.

Для контролю величин лінійних переміщень інструментів у верстатів ЧПК обробки електроізоляційного картону, застосовують датчики зворотного зв'язка двох типів: багатовідрахунковий пристрій на сельсинах із приводом шестірня-гайка типу Б2Р або індуктивний пристрій ДЛП.

Рейковий датчик лінійних переміщень типу Б2Р складається з набору точно виготовлених мілкомодульних рейок (m = 0,796), закріплених на одній із частин контрольованої системи, наприклад на поперечці, на стійці або на повзуну супорта верстата ЧПК обробки електроізоляційного картону. Довжина рейок у наборі відповідає величині контрольованого переміщення. Точність взаємного розташування рейок по довжині перевіряють спеціальними вимірювальними пристроями. Непаралельність рейок ходу рухливого вузла повинна лежати в межах 0,015 мм. При переміщенні вузла верстата по рейках котиться притискається до них з постійним зусиллям шестірня із числом зубів z=16. Вона встановлена на вході багатоступінчастого пристрою автоматичного контролю переміщень, що складається з набору трифазних сельсинів типу БС-155А. Сельсини з'єднані між собою послідовно розташованими безсоромними зубчастими передачами, кожна з яких має передатне відношення 1:10. Кожний сельсин датчика відраховує за один оберт 2, 20, 200, 2000 і 20 000 мм лінійного переміщення вузла. Сигнали, що знімаються із сельсинів датчика, що відповідають 1/200 оберту кожного сельсина, подаються в електронно-обчислювальний пристрій і вказуються по декадах на табло цифрової індикації як величини пройденого вузлом, що переміщається, шляхи в сотих або десятих частках міліметра або в одиницях, десятках, сотнях і тисячах міліметрів.

Погрішність відліку переміщень для рейкових датчиків верстатів ЧПК обробки електроізоляційного картону лежить у межах 0, 016-0,018 мм на довжині

шляху 500 мм. Для підвищення точності відрахунків датчик повинен бути прогрітий перед початком роботи протягом 30 хв. Якщо цього не зробити, то за рахунок теплових деформацій погрішність відліку зросте на 0, 004-0,005 мм.

Пристрій другого типу – датчик лінійних переміщень ДЛП [4], що складається з набору вимірювальних шкал і індуктивний датчики, що зчитує.

Вимірювальна шкала, що служить базою для відрахунків, укріплена аналогічно набору рейок рейкового датчика на одній із частин 3 верстати, щодо якої контролюється переміщення. Довжина шкал у наборі відповідає величині контрольованого переміщення. Датчик, що зчитує, укріплений на іншій частині верстата, рух якої контролюється, у спеціальному кронштейні, що забезпечує строго постійне положення датчика щодо шкали на всьому шляху переміщення. Величина зазору між ними постійна й лежить у межах 0,2-0,3 мм. У процесі експлуатації верстата й при ремонтах величину цього зазору необхідно перевіряти.

Кожна вхідна в набір шкала-лінійка являє собою сталеву пластинку довжиною 250 мм із наклеєною накладкою з текстоліту. На накладці методом фотодруку нанесена обмотка у вигляді зигзагоподібної мідної смужки шириною близько 1 мм і кроком 2 мм. Обмотки шкал включені в один послідовний ланцюг. Індуктивний датчик складається із двох таких же, але більше коротких обмоток, зрушених відносно один одного на 1/4 кроку. Вхідні сигнали (напруги) на його обмотки подаються із пристрою ЧПК у формі синусоїд. Вихідний сигнал знімається з послідовно включених обмоток лінійок.

Погрішність шкал лінійок на довжині 250 мм звичайно лежить у межах ±10 мкм. В окремих випадках застосовують більше точні датчики з погрішністю шкал ±2,5 мкм на тій же довжині. Погрішність відліку переміщень на верстатах ЧПК обробки електроізоляційного картону при застосуванні ДЛП становить не більше 15 мкм на довжині ходу 500 мм. Зміна цієї величини може бути наслідком забруднення шкал осілим металевим пилом або порушення взаємного їхнього

розташування; останнє усувають регулюванням. Датчики періодично варто очищати.

Всі органи керування верстатом ЧПК обробки електроізоляційного картону зосереджені в межах робочої зони верстатника-оператора. Залежно від конструктивного виконання того або іншого верстата набір керуючих елементів, їхні типи й розташування на пультах можуть бути змінені, але загальні принципи компонування зберігаються. Органи керування механізмами верстата при роботі в налагоджувальному або ручному режимі розміщені на підвісній кнопковій станції (ПКС) верстата. Тут же розташована й кнопка включення пристрою ЧПК. Органи керування пристроєм ЧПК розміщені на його пультах. Існують розходження компонування й призначення керуючих елементів позиційної й безперервної систем ЧПК.

1. Автоматичний режим – основний режим роботи, коли керування всіма функціями здійснюється системою без участі верстатника-оператора по записаній у програмному забезпеченні керуючій програмі.

2. Режим переднабору використовують при налагодженні верстата або пробній обробці деталі без зчитування команд із комп'ютера. Всі команди по черзі верстатник-оператор набирає безпосередньо кнопками й перемикачами пульта керування пристрою ЧПК.

3. Режим універсального ручного керування можливо здійснити тільки на верстатах виконання Ф2. Всі операції, пов'язані з керуванням верстатом, робітник здійснює з підвісної кнопкової станції.

Верстат зі ЧПК може виконувати практично необмежену кількість різних погоджених між собою переміщень робочих органів з точністю, обумовленою конструкцією верстата й системою ЧПК, а також здійснювати необхідні по технологічному циклі обробки включення допоміжних органів. При цьому точність обробки й витрачене на неї час не залежать від рівня кваліфікації й психологічного стану обслуговуючий цей верстат верстатника-оператора.

Послідовність обробки може бути точно повторене необмежене число раз і через будь-який проміжок часу.

Застосування верстатів зі ЧПК має наступні переваги.

1. Автоматизація процесу обробки й зниження фізичної й психологічної стомлюваності верстатника-оператора, а також зниження вимог до його кваліфікації навіть при обробці складних і точних деталей.

2. Підвищення продуктивності обробки в результаті:

– концентрації операцій і їхнього сполучення, а в ряді випадків за рахунок можливості, що з'являється, виключення окремих операцій і перевстановлення оброблюваної заготовки;

– оптимізації режимів різання й можливості багатоінструментальної обробки за допомогою двох супортів, збільшення питомої ваги машинного часу в загальному балансі штучно-калькуляційного часу за рахунок скорочення втрат на пробні проточки й виміри, що особливо відчутно при обробці великогабаритних деталей складної форми з більшим числом оброблюваних поверхонь із точними розмірами;

– впровадження технічно обґрунтованих норм, розрахованих технологом-програмістом.

3. Підвищення точності й стабільності розмірів оброблюваних поверхонь, що, у свою чергу, приводить до спрощення роботи апарата ОТК.

У той же час при перекладі обробки деталей з універсального встаткування на верстати зі ЧПК необхідно завжди брати до уваги порівняльну вартість цих верстатів, а також вартість їх технологічного й технічного обслуговування. Розробка програми й прив'язка інструмента до загальної системи координат верстата вимагають більших витрат часу кваліфікованих фахівців. З урахуванням цих факторів, як правило, виявляється, що прості за формою деталі, для обробки яких треба мало машинного часу, але велика кількість переходів, особливо при одиничному характері виробництва, більш економично обробляти

на універсальних верстатах ЧПК обробки електроізоляційного картону, оснащених пристроями цифрової індикації (виконання Ф1).

При підборі номенклатури деталей або окремих токарських операцій для обробки на верстатах ЧПК обробки електроізоляційного картону варто керуватися положеннями про те, що:

– найбільший ефект дає переклад обробки деталей складної форми з важкодоступними оброблюваними поверхнями;

– деталі складної, особливо криволінійної форми, обробка яких на універсальному встаткуванні вважається не технологічною, на верстатах зі ЧПК обробити легше.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно.

Числовим програмним керуванням верстатом відповідно до ДСТ 20523-80 (у ред. 1987 р.) мають на увазі керування обробкою заготовки на верстаті по керуючій програмі, у якій дані відображаються в цифровій формі.

Під системою числового програмного керування розуміють комплекс функціонально взаємозалежних і взаємодіючих програмних і технічних засобів, які у свою чергу постачають ЧПК верстатом і різанням зі ЧПК.

Основою системи ЧПК служить пристрій числового програмного керування (ПЧПК), що, у свою чергу, видає керуючі впливи на виконавчі органи верстата згідно з керуючою програмою й інформацією про стан керованого об'єкта, одержуваної за допомогою вимірювальних систем. Управляючою програмою (УП) називається комплекс команд мовою програмування, що відповідає вихідному алгоритму функціонування верстата по обробці конкретної заготовки або різання електрокартону.

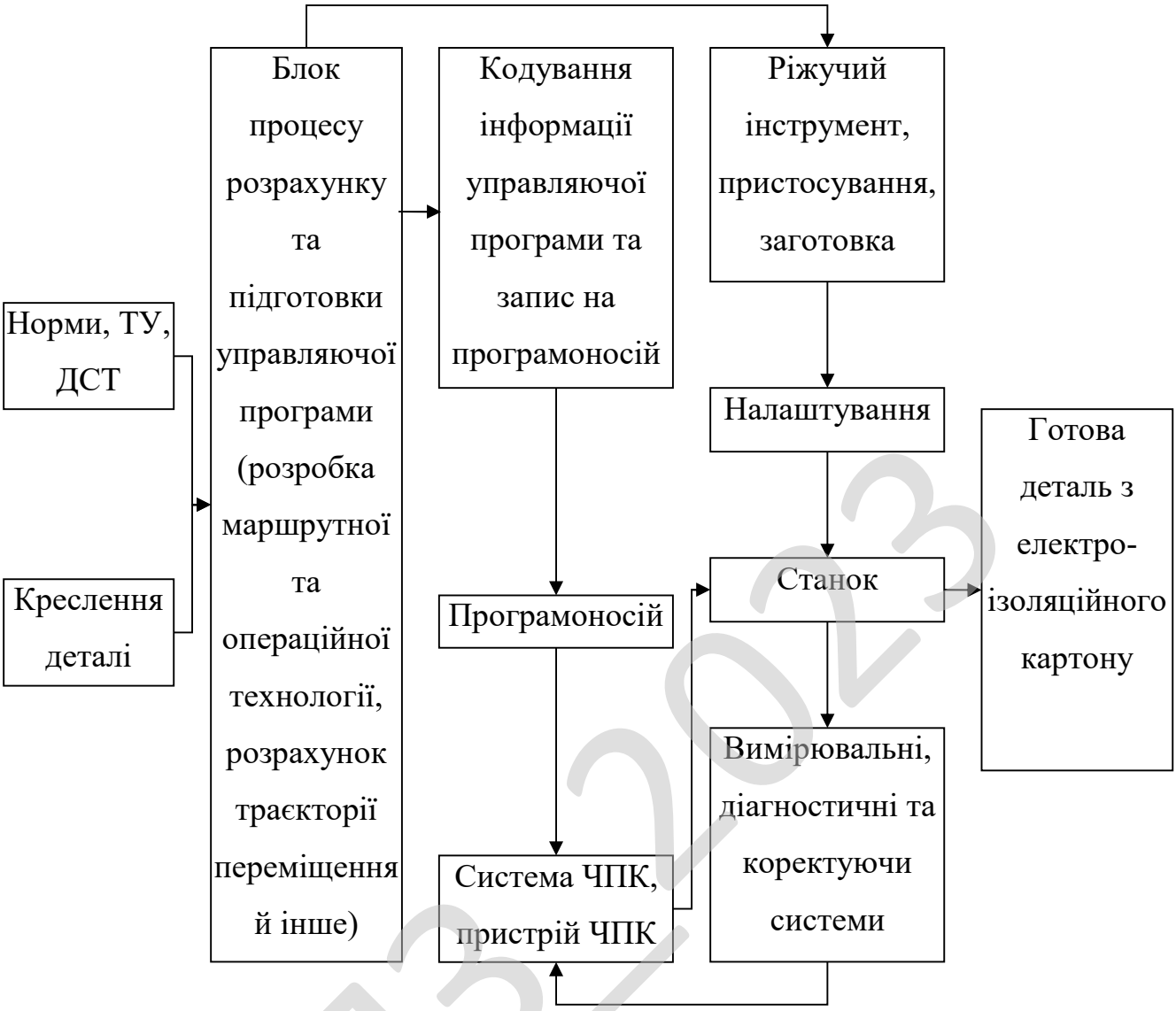

Рисунок 3.2 – Функціональна схема системи

Управляюча програма записується й уводиться в систему різання зі ЧПК за допомогою програмоносія, у якості якого можуть використовуватися перфострічка, магнітна стрічка, магнітний диск і запам'ятовувальні пристрої різного типу.

У загальному виді процес підготовки й роботи верстата зі ЧПК можна представити, розглядаючи його як процес передачі й перетворення інформації в системі "креслення деталі – готова деталь". У першу чергу виходячи із креслення деталі, а також інформації з нормалей, ТУ, РТМ, ДСТ, характеристик верстатів зі ЧПК виконується підготовка даних для проектування технологічного процесу

обробки деталі (різання зі ЧПК) зі створенням маршрутної й операційної технологій, розрахунком траєкторій переміщень робочих органів верстата з різальним інструментом і заготовкою, кодування інформації, що одержали, а також її запис на програмоносій.

По ходу розробки технологічного процесу обробки виробляється вибір і наступне налагодження на верстаті різальних інструмент і пристосування.

Після цього проводиться налагодження й контроль УП і розробленого технологічного процесу з наступною обробкою на верстаті контрольних деталей.

Далі на базі розробленої УП здійснюється керування верстатом при обробці всієї партії заданих деталей. Система різання зі ЧПК, коренем якої представляється пристрій різання зі ЧПК, виконує керування приводом головного руху, приводами подач і цикловою автоматикою (допоміжними пристроями верстата). У ході керування відбувається вимір величин пересувань робочих органів верстата (за допомогою зворотного зв'язка й може виконуватися технічне діагностування системи керування, різального інструменту, вузлів верстата, вимір оброблюваних деталей прямо на верстаті, вимір дійсного положення різального інструменту, вимір погрішностей верстата для їхньої наступної корекції й т.д.).

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

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання магістерського проектування, наведена на рисунку 3.3. Після початку роботи ПЗ ми потрапляємо до головного вікна програми звідки проводиться взаємодія з бібліотекою роботи з мовою G-код та основною, додатковою таблицею технологічних кодів і команд. Через обробник помилок проводиться введення G-коду та далі його обробка з використанням лексічного аналізатора. Після чого

при наявності COM порту, проводиться формування запиту і отримання чи відправлення даних.

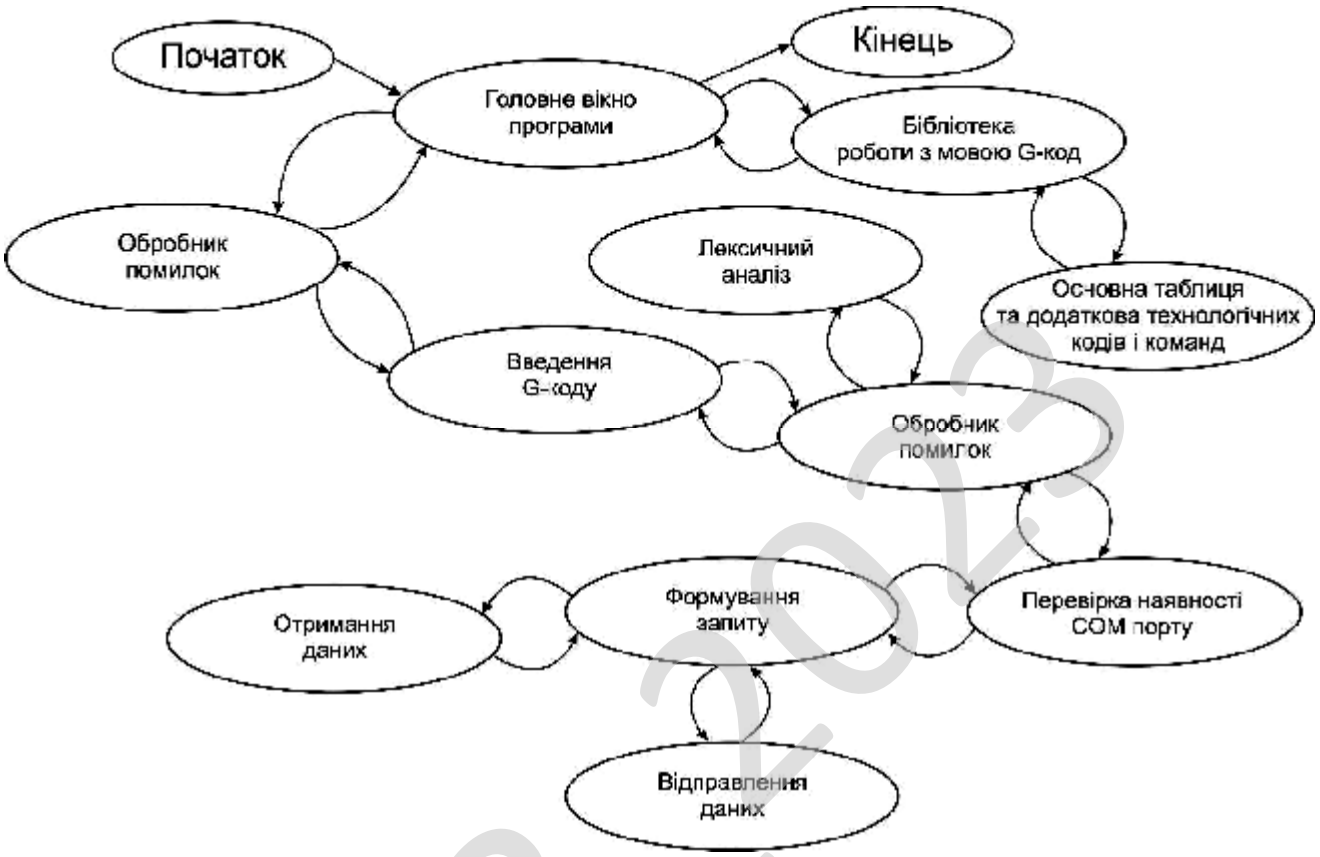

Рисунок 3.3 – Діаграма взаємодії процесів

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рисунку 4.1 наведено блок-схему основної програми. Її робота починається з ініціалізації ресурсів програми та передача управління ПЗ, створення шалонів та структур ПЗ, завантаження налаштувань ПЗ, виділення динамічних ресурсів користувача та сканування файлів ПЗ. Якщо всі файли знайдено проводиться пошук додаткових модулів з послідуючим підключенням. Якщо всі ці дії проведено успішно – ПЗ ініціалізовано.

Далі проводиться виведення даних додаткових модулів, завантаження модуля бібліотеки G-код, підключення таблиць технологічних кодів і команд та очікування дії користувача.

Якщо є запит перейти на додаткові форми проводиться виведення на екран додаткової форми.

Якщо є запит програмування верстату, викликається підпрограма формування G-коду яка зображена на рисунку 4.2. В підпрограмі відбувається виведення форми G-коду та введення самого G-коду. Коли G-код сформовано проводиться аналізу G-коду з обробкою введених даних. Якщо немає помилок у синтаксисі проводиться перевірка команд. Спочатку перевіряються основні команди (G00-G91).

Якщо основні команди (G00-G03) проводиться перевірка коду позиціонування інструмента.

Якщо основні команди (G17-G19) проводиться перевірка коду перемикання робочих площин (XY, ZX, YZ).

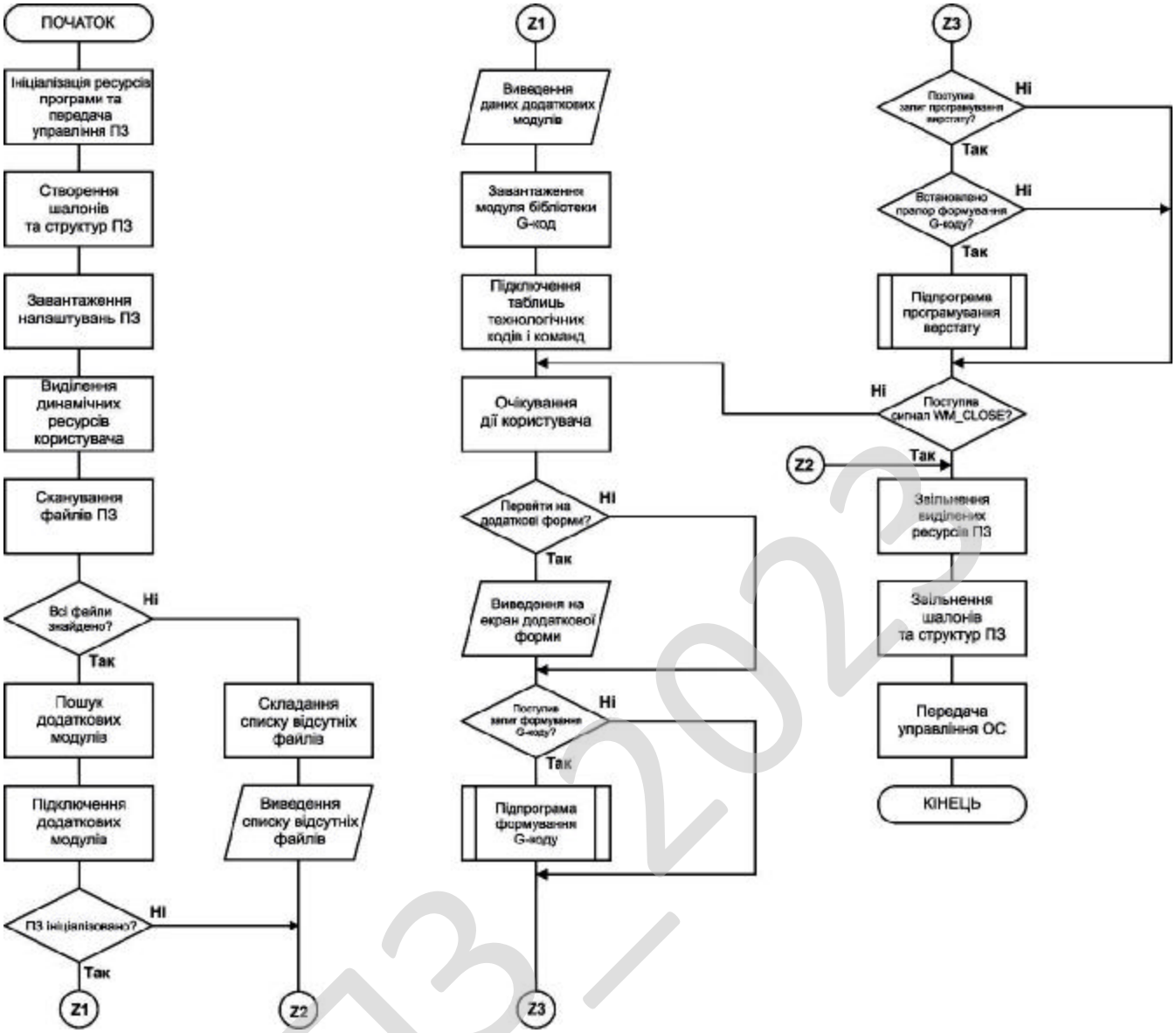

Рисунок 4.1 – Блок-схема основної програми

Якщо основні команди (G40-G44) проводиться перевірка коду компенсації розміру різних частин інструмента (довжина, діаметр).

Якщо основні команди (G53-G59) проводиться перевірка коду перемикання систем координат.

Якщо основні команди (G80-G85) проводиться перевірка коду циклів нарізування.

Якщо основні команди (G90-G91) проводиться перевірка коду перемикання систем координат (абсолютна, відносна).

Далі перевіряються допоміжні (технологічні) команди M00-M99. В кінці аналізу якщо помилок не знайдено встановлюється прапор формування G-коду.

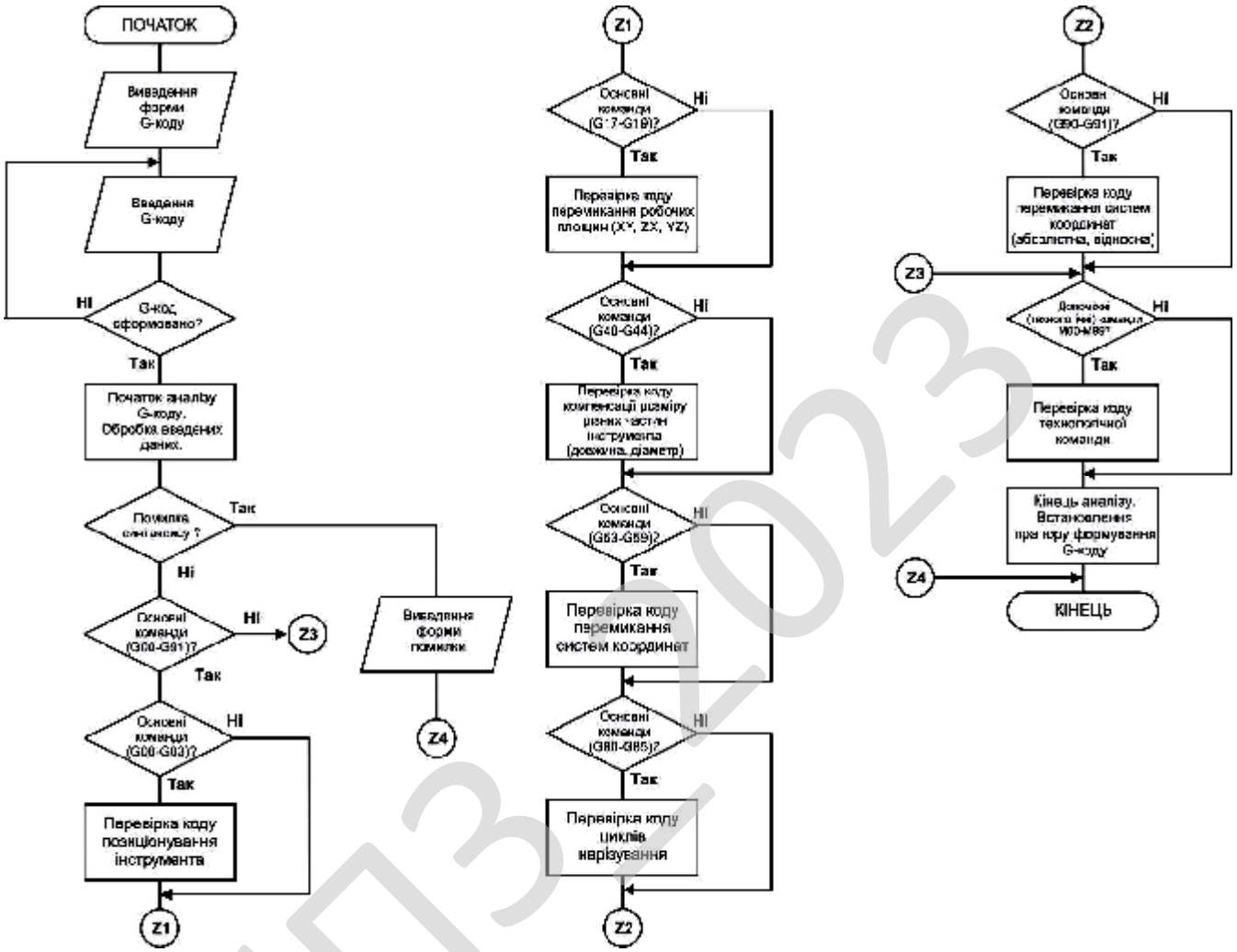

Рисунок 4.2 – Блок-схема підпрограми

Після цього проводиться повернення до основної блок-схеми яка зображена на рисунку 4.1. Та проходе запит якщо поступив запит формування G-коду та встановлено прапор формування G-коду викликається підпрограма програмування верстату. Далі якщо є сигнал WM_CLOSE проводиться звільнення виділених ресурсів ПЗ, звільнення шалонів та структур ПЗ та передача управління ОС.

Програмований порт послідовної передачі даних. Данні передаються / приймаються у вигляді послідовних імпульсів (0,1) з

використанням однієї лінії.

Для розпізнавання групи символів служить спеціальний стартовий біт, потім передаються біти даних у вигляді набору високих та низьких рівнях. Останній біт даних може супроводжуватися бітом паритету (парності), який використовується для виявлення помилок. Для виявлення кінця символу в послідовності включити один або більше стоп-бітів.

Інформаційні біти даних, біт парності, стартові і стопові біти визначають протокол для обміну інформації . Передатчик і приймач повинні використовувати один і той же протокол . Інша важлива характеристика – швидкість передачі даних . Вона повинна бути однаковою для передавача і приймача. Вимірюється в бодах або bps. Бод – це кількість бітів у секунду, враховуючи стартові стопові біти, а також біти парності. Bps- швидкість передачі даних без службових бітів.

Основні функції, які виконує УАПП(UART)

1. Забезпечує перетворення паралельного коду в послідовний при передачі даних і зворотне перетворення при їх прийомі.

2. Формування стартового, контрольного і стопового бітів при передачі даних. Здійснюється програмно.

3. Контроль вірності прийому стартового,контрольного і стопового бітів при прийомі даних.

4.Прийом / передача знаку на фіксованих швидкостях.

5. Формування і контроль стану сигналів і інтерфейсі RS 232C.

6. Організація діагностичної перевірки без використання додаткового обладнання.

Апаратна реалізація.

Комп’ютери IBM PC мають 2 порти послідовної передачі даних, так як ОС підтримує тільки 2 порти. Зовнішні пристрої підключені до вводу / виводу.

Програмування UART 8250.

Адаптер має 10 програмованих 1-байтних регістра. За допомогою яких керується і контролюється обмін інформацією.

Доступ до цих 10 регістрів здійснюється за допомогою 7 адресів. Це порти 3F8H-3F9H(COM1), 2F8H-2FEH(COM2).

Асинхронний адаптер вироблює переривання COM1 IRQ 4 (INT 0CH) та COM2 IRQ 3 (INT 0BH).

Порти асинхронного адаптера.

На етапі ініціалізації системи модуль POST BIOS тестує асинхронний адаптер, їх базові адреси розміщені за адресою 0000:0400 h.

Перший адаптер COM1 займає діапазон 3F8H-3FEH.

Формат порту 3F8H.

Цей порт відповідає регістру передаваємих даних. Для передачі в порт 3F8 необхідно записати передаваємий байт даних. Після прийому даних від зовнішнього пристрою вони можуть бути прочитані із цього порту.

Таким чином порт 3F8H використовується для прийому і передачі даних. Прийом і передача здійснюється, якщо у регістрі керування 3FBH біт Д7=0; якщо Д7=1, то порт 3F8H використовується для вводу значення молодшого байту дільника частоти тактового генератора.

Порт 3FBH. Керуючий регістр має доступ для запису і читання.

Д0, Д1 – довжина слова:

– 00 – 5 біт;

– 01 – 6 біт;

– 10 – 7біт;

– 11 – 8 біт.

Д2- кількість стопових бітів:

– 0 – один стоповий біт;

– 1 – два стопових біти.

Д3, Д4- контрольна парність:

– 00 – не використовується ;

– 01 – контроль на непарність;

– 11 – контроль на парність.

Д5 – фіксація парності.

Приймає значення.

– 0 – якщо контроль на парність.

– 1 – якщо на непарність.

Д6 – встановлення переривання.

Д7.

– 1 – порти 3F8-3F9 використовуються для завантаження константи поділювана частоти.

– 0 – порти 3F8-3F9 використовуються звичайно (3F8 прийом\передача в буфер, 3F8 встановлення обробки переривання).

Порт 3F9H.

Регістр дозволу перерви використовується для керування перериваннями від асинхронного адаптера або для введення значення старшого байту дільника частоти тактового генератора.

Якщо програміст не використовує переривання то все одно потрібно провести запис в регістр. Потрібно помістити в регістр 0. В режимі регістр керування переривань порт має формат.

Д7 – Д4 – Не використовуються.

Д3 – 1 – дозвіл вхідних CTS,DSR,RI.

Д2 – 1 – дозвіл переривань при виявленні стану помилки.

Д1 – 1 – дозвіл переривань після передачі байта (коли вихідний буфер передачі пустий).

Д0 – 1 – дозвіл переривань при готовності приймальних даних.

Порт 3FAH.

Регістр ідентифікації переривання. По його змісту програміст зможе визначити причину переривань.

Д0 – 1 – немає переривань, що очікують обслуговування.

Д1-2:

– 00 – переривання по лінії стану приймача. Помилка виникає при переповненні буфера приймача, помилка парності або формату даних, скидається після читання даних із порта 3F9H.

– 01 – дані прийняті і доступні для читання. Скидується після читання даних із порту 3FDH.

– 10 – буфер передавача пустий. Скидується після запису даних у порт 3F8H;

– 11 – стан модему. Встановлюється при зміні стану вхідних ліній (CTS,DSR,RI).

Д3-7 – Не використовуються.

Порт 3FDh. Регістр стану лінії.

Д0 – буфер регістра прийому: 0-пустий; 1-завантажений.

Д1 – переповнення при прийомі: 0-немає; 1-є.

Д2 – помилка паритету: 1-немає; 0-є.

Д3 – стопові біти: 0-немає; 1-є.

Д4 – переривання обриву лінії: 0-немає; 1-є.

Д5 – буфер передавача: 0-завантажений; 1-пустий.

Д6 – регістр зсуву: 0-завартажений; 1-пустий.

Д7 – не використовується.

Розглянемо використані алгоритми функціонування системи.

Практично будь-якому комп'ютеру доводиться зв'язуватися з зовнішніми пристроями. Практично будь-якому програмістові доводилося (доводиться, доведеться) створювати програми під ці пристрої.

Величезна кількість зовнішніх пристроїв спілкуються з комп'ютером за допомогою RS-232.

Не виняток і використовуваний верстат з ЧПК. Розглянемо застосовувані алгоритми для взаємодії з верстатом з ЧПК.

Відкриття та налаштування COM порту

```
var
DCB: TDCB;
hPort: THandle;
begin
// 1. Відкриваємо файл
hPort:= CreateFile (PChar ('COM' + IntToStr (<номер порту>)),
                        GENERIC_READ + GENERIC_WRITE, 0, nil,
                        OPEN_EXISTING, FI LE_ATTRIBUTE_NORMAL, 0);
// 2. Контроль помилок
if hPort = INVALID_HANDLE_VALUE then begin
   // Виявлена помилка, порт відкрити не вдалося
exit;
end;
// 3. Читання поточних налаштувань порту
if GetCommState (hPort, DCB) then;
  // 4. Налаштування:
  // Швидкість обміну
DCB.BaudRate:= [швидкість обміну];
// Число біт на символ
DCB.ByteSize:= [розмір "байта" при обміні по COM порту – зазвичай 8 біт];
// Стоп-біти
DCB.StopBits:= [константа, що визначає к-ть стопбітов];
// Парність
DCB.Parity:= [константа, що визначає режим контролю парності];
DCB.Flags:= 20625;
// 5. Передача налаштувань
if not SetCommState (hPort, DCB) then {помилка налаштування порту};
// 6. Налаштування буферів порту (черг введення і виведення)
if not SetupComm (hPort, 16, 16) then {помилка налаштування буферів};
// 7. Скидання буффер і черг
if PurgeComm (hPort, PURGE_TXABORT or PURGE_RXABORT or PURGE_TXCLEAR or
PURGE_RXCLEAR) then;
// 8. Закриття порту
CloseHandle (hPort);
end;
```

Послідовність дій.

Крок 1 – відкриття порту. Порт відкривається стандартною командою CreateFile, тільки в якості імені файлу передається ім'я порту (наприклад, "COM1").

Крок 2 – перевірка успішності кроку 1. Якщо порт відкрився, то функція CreateFile повертає його Handle, інакше hPort = INVALID_HANDLE_VALUE, що свідчить про помилку.

Крок 3 – завантаження поточних налаштувань порту за допомогою функції GetCommState. Цей крок не є обов'язковим, тому можна налаштувати порт, як кажуть, "з нуля". Після виконання GetCommState структура DCB завантажується поточними налаштуваннями.

Крок 4 – занесення в структуру DCB необхідних налаштувань.

Крок 5 – Налаштування порту за допомогою функції SetCommState. Після налаштування з портом вже можна працювати. (кроки 6 і 7).

Крок 6 – Налаштування розмірів буфера прийому і передачі за допомогою SetupComm.

Крок 7 – Очищення буферів і черг порту – другий параметр функції PurgeComm є бітової маскою і задає, що саме необхідно очистити – в нашому випадку все по максимуму.

Крок 8 – закриття порту після завершення роботи з ним.

Запис даних проводиться таким чином.

```
Result:= WriteFile (hPort, Buffer, Size, NumberOfBytesWritten, nil);
```

Параметр NumberOfBytesWritten після виконання містить кількість реально переданих байт.

Читання даних.

```
Result:= ReadFile (hPort, Buffer, Size, NumberOfBytesReaded, nil);
```

Параметр Size задає читаний обсяг інформації (його значення повинно бути менше обсягу буфера), NumberOfBytesReaded отримує кількість реально прочитаних байт.

Налаштування таймаутів прийому і передачі.

```
function SetComPortTimeouts: boolean;
var
CommTimeouts: TCommTimeouts;
ComErrors: DWORD;
begin
Result:= false;
```

```
// Якщо з'єднання є, то перенастроювати його
if hPort <> INVALID_HANDLE_VALUE then begin
// Читання поточних таймаутів
GetCommTimeouts (hPort, CommTimeouts);
// настройка параметрів структури CommTimeouts
// Установка таймаутів
  SetCommTimeouts (hPort, CommT imeouts);
Result:= true;
end;
end;
```

Запит помилки проводиться за допомогою функції ClearCommError.

```
var
ComErrors: DWORD;
begin
ClearCommError (hPort, ComErrors, nil);
end;
```

Ця функція скидає стан помилки, передаючи код помилки в змінну ComErrors.

Слід мати на увазі що при розробці програм що мають справу роботи з портами слід ует враховувати наступні фактори:

1. Прагнути використовувати функції високого рівня для доступу до портів (зокрема WinAPI) і не вдаватися до низькорівневих операцій читання / запису портів. Якщо ви все-таки вирішили писати низькорівневе читання те ці процедури потрібно виносити в окрему DLL або VXD, з наступних причин – відомо, що операційна система Windows95/98 а особливо NT є по своїй суті багатозадачними системами. Тобто якщо ваша програма звертається конкретно до порту не через динамічний виклик функції DLL або VXD (використання механізму DLL) а напряму то це може позначитися на коректній роботі системи або навіть завалити її. І навіть якщо в Windows95/98 такий підхід цілком може працювати те в Windows NT внаслідок його архітектури не вирішить безпосереднє читання / запис безпосередньо, а використання механізму DLL або VXD дозволяє обійти цю проблему.

2. Якщо робота проводиться з якимось нестандартним портом вводу-виводу (наприклад портом що зберігає стан кнопок пульта ДУ TVTunera то напевно в

комплекті постачання рідного софтвера знайдеться DLL або VXD для управління цим пристроєм і відпаде потреба писати код, так я при роботі з пультом ДУ TVTunera використовую стандартну DLL поставляється в комплекті, це відразу вирішило питання пов'язані з управлінням портами даного тюнера).

Формат передачі даних в верстат з ЧПК наступний.

```
$GPRMC,101072.00,A,5029.0728,N,03028.7404,E,000.0,000.0,230111,02.2,E,A*0D
```

Розглянемо частину коду, який реалізує формування даних для верстата з ЧПК.

```
        unit Main;
        interface
        uses
Windows, Messa ges, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, Registry, ReadThread;
        type
        TMainForm = class (TForm)
        OpenPort: TButton;
        ClosePort: TButton;
        SendData: TButton;
        ReadData: TButton;
        PortStateLabel: TLabel;
        Label1: TLabel;
        Label2: TLabel;
        nToReadLabel: TLabel;
        nReadLabel: TLabel;
        Label3: TLabel;
        RcDataLabel: TLabel;
        Label4: TLabel;
        Label5: TLabel;
        bRefreshComPrt: TButton;
        cbCOMPrts: TComboBox;
        cbSpeed: TComboBox;
        procedure OpenPortClick (Sender: TObject);
        procedure ClosePortClick (Sender: TObject);
        procedure SendDataClick (Sender: TObject);
        procedure ReadDataClick (Sender: TObject);
        procedure FormCreate (Sender: TObject);
        procedure FormClose (Sender: TObject; var Action: TCloseAction);
        procedure bRefreshComPrtClick (Sender: TObject);
        procedure RefreshComPrt;
```

```
        procedure FormShow (Sender: TObject);
        private
        {Private declarations}
        public
        {Public declarations}
        Port: THandle;
        end;
        var
        MainForm: TMainForm;
        ReadThr: TReadThread;
        implementation
        {$ R *. Dfm}
        procedure TMainForm.OpenPortClick (Sender: TObject);
        Var
        DCB: TDCB; // структура, що містить настройки порту
        CommTimeouts: TCommTimeouts;
        begin
Port:= CreateFile (
// Відкриваємо перший порт
        pWideChar (cbCOMPrts.Text),
// Відкриваємо порт для читання і запису
        GENERIC_READ or GENERIC_WRITE,
// Загальний доступ до ресурсу заборонений
        0,
// Атрибути захисту, не використовуються і тому nil
        nil,
// Атрибути відкриття, для портів OPEN _ EXISTING
        OPEN_EXISTING,
// Синхронна робота
        FILE _ ATTRIBUTE _ NORMAL, 0
        );
        if (port = INVALID_HANDLE_VALUE) // якщо порт НЕ відкрився
        then showmessage ('Помилочка вийшла!')
// виводимо повідомлення про помилку
        else POrtStateLabel.Caption:= 'Порт відкритий';
// Якщо порт відкрився, то пишемо що відкрився
// Що б не заповнювати всю структуру самим, спочатку зчитуємо її,
// потім поміняємо потрібні поля
        GetCommState (port, DCB)
        DCB.BaudRate:= StrToInt (cbSpeed.Text); // швидкість обміну
        DCB.Parity:= NoParity; // немає контролю парності
        DCB.ByteSize:= 8; // байт з восьми біт
```

```
        DCB.StopBits:= ONESTOPBIT; // один стоповий біт
        // Записуємо змінену структуру, для відкритого порту
        SetCommState (port, DCB);
// Отримуємо м структуру CommTimeouts що б
// не заповнювати все вручну
        GetCo mmTimeouts (Port, CommTimeouts);
// Функція ReadFile повертає
// Негайно всі наявні
// Байти в приймальному буфері
        CommTimeouts.ReadIntervalTimeout:= MAXDWORD;
          CommTimeouts.ReadTotalTimeoutMultiplier:= 0;
        CommTimeouts.ReadTotalTimeoutConstant:= 0;
        CommTimeouts.WriteTotalTimeoutMultiplier:= 0;
// Загальний тайм-аут для
// Операції запису не використовується.
        CommTimeouts.WriteTotalTimeoutConstant:= 0;
        // Операції запису не використовується.
        SetCommTimeouts (Port, CommTimeouts);
        end;

        procedure TMainForm.RefreshComPrt;
        var
        reg: TRegistry;
        ts: TStrings;
        i: integer;
        begin
        cbCOMPrts.Items.Clear;
        reg:= TRegistry.Create;
        reg.RootKey:= HKEY_LOCAL_MACHINE;
        reg.OpenKey ('hardware \devicemap\serialcomm',false);
        ts:= TStringList.Create;
        reg.GetValueNames (ts);
        for i:= 0 to ts.Count -1 do begin
        cbCOMPrts.Items.Add (reg.ReadString (ts.Strings [i]));
        end;
        if cbCOMPrts.Items.Count> 0 then
        cbCOMPrts.ItemIndex:= 0;
        ts.Free;
        reg.CloseKey;
        reg.free;
        end;
        procedure TMainForm.bRefreshComPrtClick (Sender: TObject);
```

```
        begin
        RefreshComPrt;
        end;
        procedure TMainForm.ClosePortClick (Sender: TObject);
        begin
        if not CloseHandle (Port)
// Якщо порт   НЕ закрився, то пишемо повідомлення
          then showmessage ('Не закрилося')
        else PortStateLabel.Caption:= 'Порт не відкритий'
// Якщо закрився, то пишемо повідомлення
        end;
        procedure TMainForm.SendDataClick (Sender: TObject);
        var
        TRBuf: PChar; // буфер даних для передачі
        nToWrite: DWord; // число байт для запису
        nWrite: DWord; // число записаних байт
        begin
        TRBuf:= '1 '; // заповнюємо буфер даними
        nToWrite:= length (TRBuf) +1; // число переданих байт
// Відправляємо дані
        WriteFile (port, TRBuf ^, nToWrite, nWrite, nil);
        // WriteFile (port, Edit1.Text [1], nToWrite, nWrite, nil);
        // WriteFile (port, TRBuf [0], nToWrite, nWrite, nil);
        end;
        procedure TMainForm.ReadDataClick (Sender: TObject);
        Var
        RCBuf: PChar; // Буфер даних для прийому
        nToRead: Cardinal; // Число байт для читання
        ReadedBytes: integer; // число всього прочитаних байт
        nRead: Cardinal; // Число прочитаних байт
        ComStat: TComStat; // стан порту
          Errs: Dword;
        Data: TStringList; // прочитані дані
        begin
        ReadedBytes:= 0;
          Data:= TStringList.Create;
        ClearCommError (POrt, Errs, @ ComStat);
// Зчитуємо стан порту
        nToRead:= ComStat.cbInQue;
// Зчитуємо число байт для читання зі структури
        nToReadLabel.Caption:= IntToStr (nToRead);
// Виводимо на форму число байт для читання
```

```
          while ReadedBytes <nToRead do
        begin
        ReadFile (Port, RCBuf ^, nToRead, nRead, nil);
// Зчитуємо дані
        nReadLabel.Caption:= IntToStr (nRead);
// Виводимо на форму число прочитаних байт
        ReadedBytes:= ReadedBytes + nRead;
        RcDataLabel.Caption:= RCBuf;
        Data.Add (RCBuf);
        end;
Data.SaveToFile (ExtractFilePath (ParamStr (0)) + 'Output.txt');
        Data.Free;
        end;
        procedure TMainForm.FormCreate (Sender: TObject);
        begin
        nToreadLabel.Caption:='';
// Очищаємо мітки
        nReadLabel.Caption:='';
        RcDataLabel.Caption:='';
        ReadThr:= TReadThread.Create (True);
        ReadThr.Priority:= tpNormal;
        ReadThr.FreeOnTerminate:= True;
        ReadThr.Start;
        end;
        procedure TMainForm.FormShow (Sender: TObject);
        begin
        RefreshComPrt;
        end;
procedure TMainForm.FormClo se (Sender: TObject; var Action: TCloseAction);
        begin
        CloseHandle (Port);
// закриваємо порт
        end;
        end.
```


4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою Serpent – симетричний блочний алгоритм шифрування, розроблений Россом Андерсоном, Елі Біхамом та Ларсом Кнудсеном. Алгоритм був одним з

фіналістів 2-го етапу конкурсу AES. Як і інші алгоритми, які брали участь у конкурсі AES, Serpent має розмір блоку 128 біт і можливі довжини ключа 128, 192 або 256 біт. Алгоритм являє собою 32-раундовий шифр на основі SP-мережі, і працює з блоком з чотирьох 32-бітових слів. Serpent був розроблений так, що всі операції можуть бути виконані паралельно, використовуючи 32-а 1-бітних «потоків».

При розробці Serpent використовувався консервативніший підхід до безпеки, ніж у інших фіналістів AES, проектувальники шифру вважали, що 16 раундів достатньо, щоб протистояти відомим видам криптоаналізу, але збільшили число раундів до 32, щоб алгоритм міг краще протистояти ще не відомим методам криптоаналізу.

Шифр Serpent не запатентований і є громадським надбанням.

Алгоритм створювався під гаслом «криптографічний алгоритм 21 століття» для участі в конкурсі AES. При створенні нового алгоритму Serpent його автори дотримувалися консервативних поглядів на проектування, що підтверджується первісним рішенням про використання таблиць підстановки з відомого багато років раніше алгоритму шифрування DES, який протягом довгого часу вивчався провідними фахівцями в області криптографії та захисту інформації і чиї властивості і особливості були добре відомі науковому світу. Одночасно з цим до нового алгоритму міг бути застосований вичерпний аналіз, вже розроблений для DES. Не використовувалися нові, неперевірені і невипробувані технології при створенні шифру, який у разі прийняття був би використаний для захисту величезних масивів фінансових транзакцій та урядової інформації. Основною вимогою до учасників конкурсу AES було те, що алгоритм-претендент повинен бути швидшим, ніж 3DES, і надавати як мінімум такий же рівень безпеки: він повинен мати блок даних довжиною 128 біт і ключ завдовжки 256 біт. 16-раундовий Serpent був би таким же надійним, як 3DES, але в два рази швидшим. Однак, автори вирішили, що для більшої надійності варто

збільшити кількість раундів до 32. Це зробило шифр таким же швидким, як DES, і набагато надійнішим, ніж 3DES.

Структура алгоритму

Алгоритм Serpent є SP-мережею, у котрій весь блок даних довжиною 128 біт на кожному раунді розбивається на 4 слова довжиною 32 біти. Всі значення, що використовуються при шифруванні є бітовими потоками. Бітові індекси змінюють значення від 0 до 31 для 32-бітових слів, від 0 до 127 – для 128-бітових блоків та від 0 до 255 для 256-бітових ключів тощо. Для внутрішніх обчислень всі біти величин представлені в прямому порядку (little-endian).

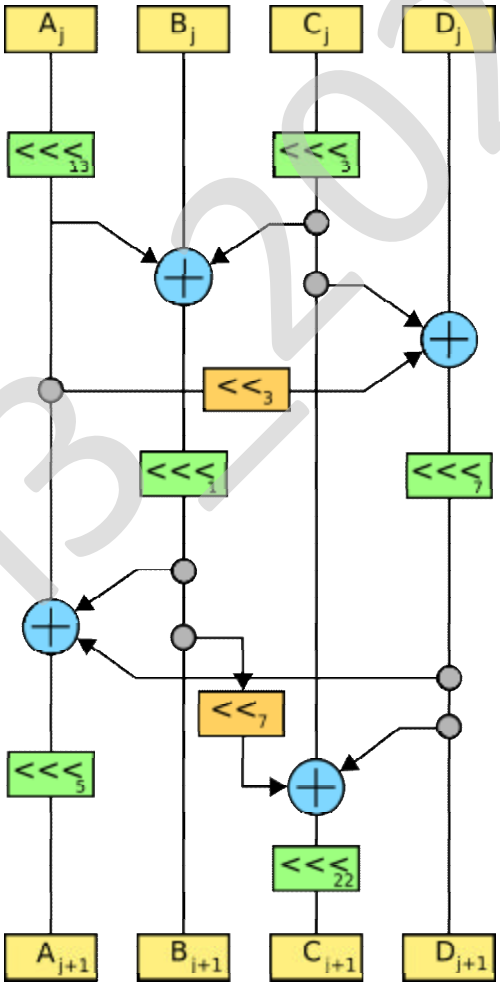

Рисунок 4.3 – Структура алгоритму Serpent

Serpent шифрує відкритий текст P довжиною 128 біт в шифротекст C довжиною таких же 128 біт за 32 раунд за допомогою 33 подключів {\displaystyle K_{0},...,K_{32}\,}{\displaystyle K_{0},...,K_{32}\,} довжиною 128 біт. Довжина використовуваного ключа може приймати різні значення, але для конкретики зафіксуємо їх довжину в 128, 192 або 256 біт. Короткі ключі довжиною менше 256 біт доповнюються до повної довжини в 256 біт.

Шифрування складається з наступних основних кроків:

– Початкова перестановка.

– 32 раунд, кожен з яких складається з операції змішування з 128-бітовим ключем (побітове логічне виключаюче «або»), таблична заміна (S-box) і лінійне перетворення. В останньому раунді лінійне перетворення замінюється додатковим накладанням ключа.

– Кінцева перестановка.

Початкова і кінцева перестановки не мають будь-якої криптографічного значущості. Вони використовуються для спрощення оптимізованої реалізації алгоритму і підвищення обчислювальної ефективності.

Розширення ключа

Як і інші алгоритми, що брали участь в конкурсі AES, Serpent має можливі довжини ключа 128, 192 або 256 біт. «Неповний» ключ довжиною менше 256 біт доповнюється за наступним правилом: додається одиничний біт справа, за ним слід стільки нульових бітів, щоб довжина ключа стала дорівнює 256 бітам.

Початкова перестановка IP

Дана перестановка IP задається таблицею, де вказується позиція, на яку перейде відповідний біт (наприклад, біт 1 перейде на 32 позицію):

S-бокси (таблиці замін)

В алгоритмі Serpent таблиці замін є 4-бітовими перестановками з властивостями стійкості до диференціального криптоаналізу, до лінійного криптоаналізу і такою властивістю, що порядок вихідних біт, як функції вхідних повинен бути максимальний, тобто бути рівним 3.

Таблиця підстановки генерується з відомих і добре вивчених таблиць для алгоритму DES в ітераційному процесі, поки не будуть отримані бажані диференціальні й лінійні властивості. Таким чином, створюється 8 таблиць підстановки.

Лінійне перетворення LT

Лінійне перетворення LTзадається таблицею, де біти перераховані від 0 до 127 (наприклад, вихідний 2 біт утворений 2, 9, 15, 30, 76, 84, 126 битами, складеними за модулем 2) . В кожному рядку описується 4 вихідних біти, які разом складають вхідні дані на одну таблицю замін в наступному раунді. Варто зазначити, що даний набір являє собою таблицю IP(LT(FP(x))), де LT і є те лінійне перетворення.

Таблиця зворотного лінійного перетворення, яке використовується при розшифровці ILT.

Кінцева перестановка FP

Дана перестановка є зворотною до початкової, тобто $FP=IP^{-1}$ і задається наступною таблицею.

Ефективна реалізація алгоритму

Бажання авторів зробити алгоритм саме таким, яким він є стає зрозумілим при розгляді його ефективної низькорівневої реалізації.

Serpent був створений таким чином, щоб всі операції в процесі шифрування і розшифрування одного блоку могли бути виконані паралельно в 32 потоках. До того ж низькорівневий опис алгоритму набагато простішій, ніж стандартний опис. Ніяких початкових і кінцевих перестановок не потрібно.

Шифрування складається з 32 раундів. Відкритий текст є першими проміжними даними $B_0 = P$. Потім виконується 32 раунди, кожен і-й раунд складається з:

– Змішування з ключем. Проводиться побітове виключаюче «або» проміжних даних B_i з ключем довжиною 128 біт.

– Застосування таблиць підстановки. Вхідні дані довжиною 128 біт поділяються на 4 слова по 32 біта. Таблиця підстановки, реалізована послідовністю логічних операцій (як якщо це було б реалізовано апаратно), застосовується до цих 4 словам. В результаті виходить 4 вихідних слова. Таким чином, центральний процесор виконує підстановку по 32 копій таблиці одночасно.

– Лінійне перетворення. 32-бітові слова перетворюються заданим порядком.

Першою причиною вибору такого лінійного перетворення є максимізація лавинного ефекту. Такі таблиці підстановки мають властивість, що зміна кожного вхідного біта призведе до зміни 2 вихідних бітів. Таким чином, кожен вхідний біт відкритого тексту вже через 3 раунди впливає на всі вихідні біти. Аналогічно кожен біт ключа впливає на результат шифрування.

Друга причина полягає в простоті перетворення. Воно може бути реалізоване на будь-якому сучасному процесорі з мінімальними витратами.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено головне вікно розробленого ПЗ. Вікно розподіляється на наступні частини:

– Позиція. Тут зображено поточні координати лазерного різця. За допомогою G–коду можна керувати діями різця на поверхні картону.

– Вікно команд. Забезпечує відображення послідовності введеного чи імпортованого G–коду.

– Позиція. Відображує поточну позицію лазерного різця, з можливістю встановлення на паркову – положення при вимкненому стані.

– Ручне введення даних. Забезпечує введення команд G–коду з перевіркою лексичного аналізатора.

– Верхнє меню. Дозволяє налаштовувати роботу ПЗ та переглядати довідкову інформацію та форму авторського права.

Форма авторського права відображена на рисунку 5.3. Розглянемо обрану ліцензію. Початок тексту ліцензії.

Надається дозвіл безкоштовно будь-якій особі отримати копію цього програмного забезпечення та пов'язаних з ними файлів документації (далі "Програмне забезпечення"), для користування без будь-яких обмежень, включаючи, без обмеження прав на використання, копіювання, зміну, об'єднання з іншим забезпеченням, публікування, поширювання, ліцензування та/або продаж копій програмного забезпечення. Все це також дозволяється особам, яким це програмне забезпечення буде надано. Все це дозволяється при дотриманні наступних умов:

Наведені вище повідомлення про авторські права, і це повідомлення повинно бути включене у всі копії або в суттєву частину програмного забезпечення.

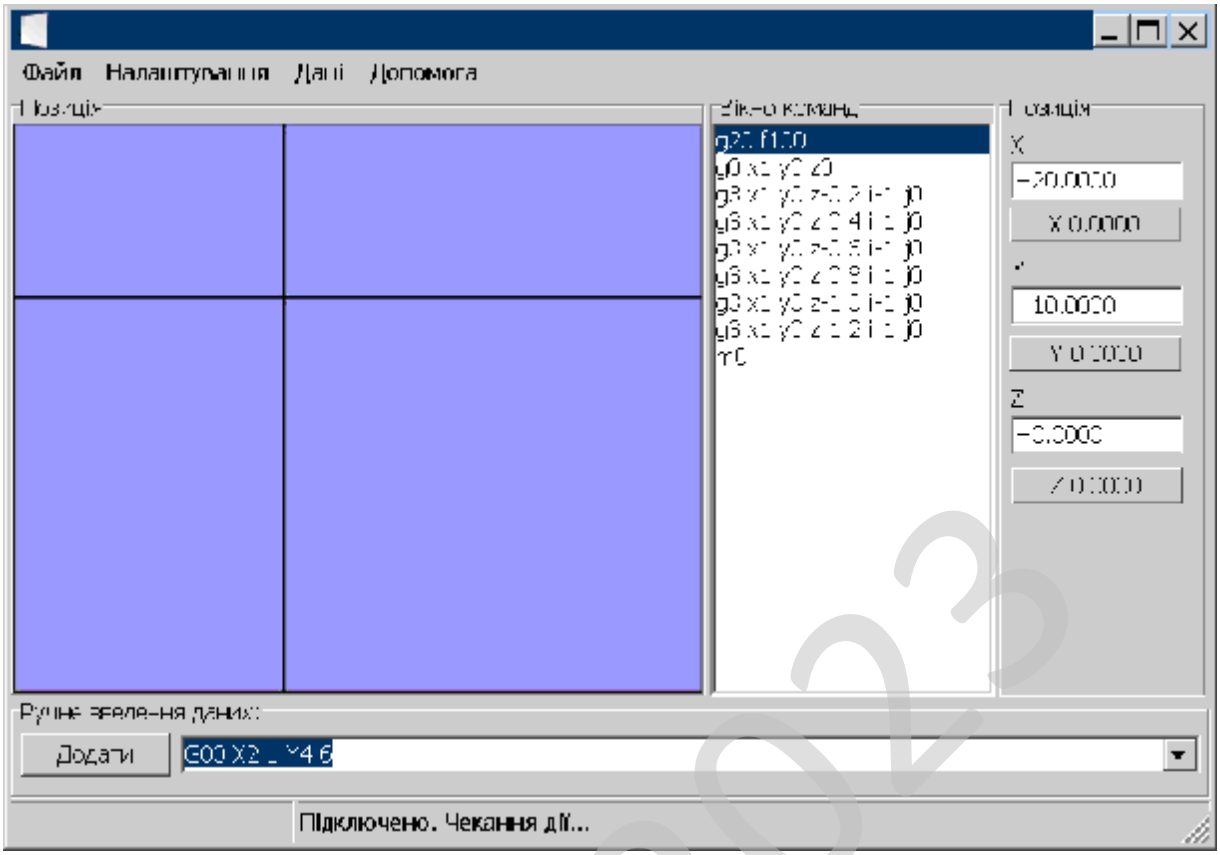

Рисунок 5.1 – Головне вікно ПЗ

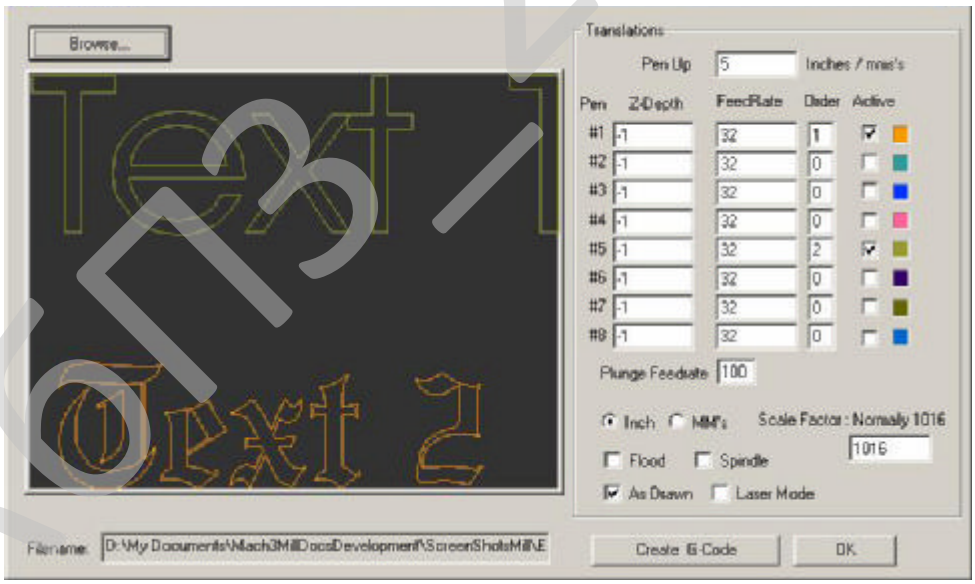

Рисунок 5.2 – Вікно завантаження шаблону

На рисунку 5.2 зображено вікно завантаження шаблону за допомогою якого можна створювати складні візерунки на картоні.

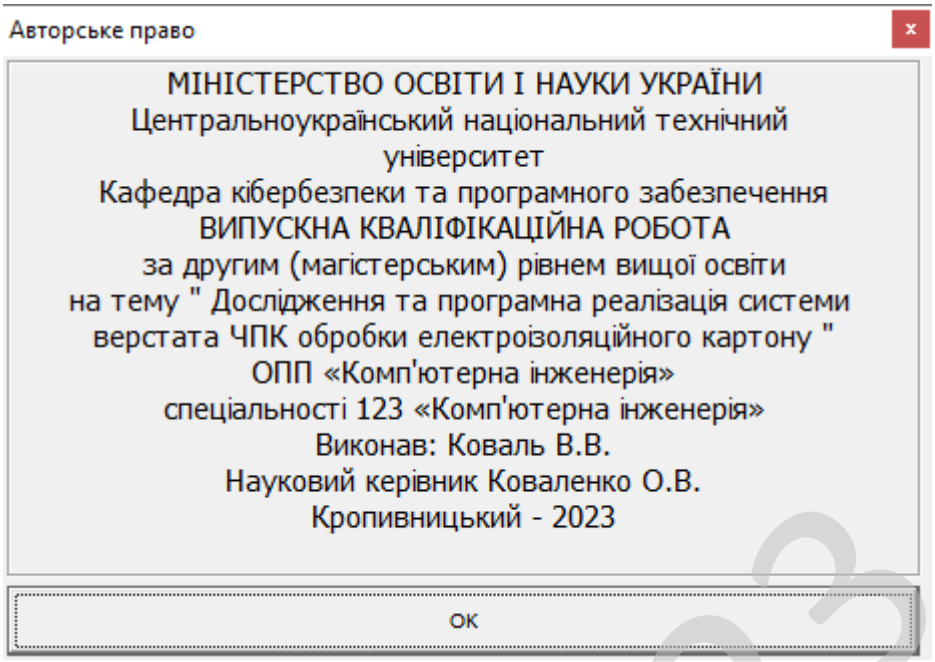

Рисунок 5.3 – Вікно авторського права ПЗ

Програмне забезпечення надається "як є", без гарантій будь-якого виду, виражених або домислених, у тому числі, але не обмежуючись гарантіями комерційної вигоди, придатності для конкретної мети і можливих помилок. Ні в якому разі автори або власники авторських прав не повинні нести відповідальність за будь-які претензії, збитки або іншу відповідальність за будь-які дії які виникають у зв'язку цім з програмним забезпеченням або застосуванням його або інших операцій з ним.

Кінець тексту ліцензії.

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи верстата ЧПК обробки електроізоляційного картону.

Метою розробки є дослідження та програмна реалізація системи верстата ЧПК обробки електроізоляційного картону.

Об'єктом дослідження є процес верстата ЧПК обробки електроізоляційного картону.

Предметом дослідження є методи верстата ЧПК обробки електроізоляційного картону.

Методи дослідження базуються на методах чисельно-програмного керування, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод верстата ЧПК обробки електроізоляційного картону.

– Розроблено вітчизняний продукт верстата ЧПК обробки електроізоляційного картону, 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 60 днів (три місяці).

В магістерській роботі було проведене дослідження та виконана програмна реалізація системи верстата ЧПК обробки електроізоляційного картону.

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	50
3. Запланований термін розробки, днів	Fpq	60 (3 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2
7. Кількість макетів вхідної інформації	–	3

Продовження табл. 7.1

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

Продовження табл. 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПО для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн	–	50000
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	55
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

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

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

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

де 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.

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

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

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

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

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

$$T_{уточн}= 6{,}78\cdot(0{,}88\cdot0{,}93\cdot0{,}88\cdot0{,}91\cdot0{,}95\cdot1\cdot1\cdot0{,}87\cdot1{,}22\cdot1{,}16\cdot1{,}1\cdot1{,}1\cdot$$
$$\cdot1{,}12\cdot1{,}1\cdot1\cdot1{,}1\cdot1{,}1)=9{,}37 \text{ люд-міс.}$$

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

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

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

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

$$T_{РП}=0{,}3\cdot2{,}66\cdot9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)}\cdot57=96{,}5 \text{ люд/день}$$

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

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	9	Д7
Робочий проект	96,5	Ф 7.1-7.4
Впровадження	13	Д13
Всього	137,5	–

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 Fpq – плановий фонд робочого часу одного спеціаліста, днів, Тпз – трудомісткість розробки програмного забезпечення люд-дні,

$$Ч = \frac{137,5 \cdot 1}{60\text{-}5} = 2,5 \text{ ставки}$$

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року.

Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3

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

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	6	540	9
Монітор	60	6	360	6
Клавіатура	30	6	180	3
Маніпулятор «мишка»	30	6	180	3
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	1	120	2
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор-маршрутизатор	30	2	60	1
Кабельні господарства ЛОМ на 1 м. п.	2,5	200	500	8,33
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	35,99

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

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

$$Ф_{др}^{с} = \frac{36 \cdot 3}{1,2} = 129,6 \text{ год}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

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

$Ч_{ел}$=129,6/(60·8)=0,25 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	Кількість штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, серверу доступу АДСЛ (ОС Linux), Wi-Fi налаштування ADSL, VPN, PPPoE, Frame Relay	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 д мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

Посада	Вид роботи	Час	Кількість 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	Середньо-місячний оклад, грн.	Всього за період роз-робки, грн.
Керівник (ІТ-менеджер)	0,25	15360	11520
Продакт-менеджер	0,25	12000	9000
Інженер-програміст	2,5	12000	90000
Інженер-електронщик	0,25	12000	9000
Інженер-системотехнік	0,25	12000	9000
Адміністратор мережі	0,5	12000	18000
Системний програміст	0,25	12000	9000
Дизайнер WEB	0,25	12000	9000
Інженер-верстальник	0,25	12000	9000
Бухгалтер-економіст	0,5	12000	18000
Всього за період розробки	$R_{сп}$=5,25	-	$Ф_{роб}$=191520

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.

$$З_{сд} = \frac{191520}{5,25 \cdot 60} = 608 \; грн.$$

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. Sy – питома площа на одне робоче місце, $м^2$; $Ц_{пл}$ – вартість одного квадратного метра площі, грн.

Згідно даних інтернет ресурсу DOM.RIA (https://dom.ria.com) ціна одного квадратного метра площі, вік якої не перевищує 30 років, по місту складає 400…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 38 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно $8 м^2$. З урахуванням цього:

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

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

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

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

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

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

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

Специфікація на обчислювальну техніку наведена в таблиці 7.7. Дані по оптовій ціні на обладнання та комплектуючі вибирались за пропозицією інтернет ресурсу hotline за 18.10.23 – джерело https://hotline.ua

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10947
Системний блок Dell OptiPlex 3060 USFF		7347
Процесор	Intel Core i3-8100T (4 ядра по 3.1 GHz), 6 MB Smart Cache	-
Системна плата	Dell OptiPlex 3060 4x USB 3.0, 2x USB 2.0, 1x Ethernet, 1x DisplayPort, 1x HDMI, 2x Audio	-
Відеокарта	Інтегрована в процесор Intel HD Graphic 630 (до 1792 MB з ОЗП)	-
Жорсткий диск	SSD 120Gb + HDD Seagate Barracuda 750 Gb	-
Оперативна пам’ять	8Gb DDR4 non-Reg., no-ECC, (модулі 2x4Gb)	-
DVD-привод	DVD -RW/+RW , LG SATA SuperMulti Bulk 22x, SecurDisc, black	-
Корпус	Dell OptiPlex USFF, 65W (80mm fan), full-ATX, БЖ 2xSATA, 1xFDD, Air Duct, 2xUSB 3.0, 2xUSB 2.0, Audio, silver/black	-
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-B, int. 3.5", 1*USB2.0+AUDIO+1394, multi: All Type Cards, black	-
інше	Клавіатура, мишка	-

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Монітор	LG W2363V-WF Wide LCD 2ms, 70 000:1, 300кд/м2, 170/160, D-Sub / Glossy White	3600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струменевий	Epson Stylus Photo P50 (C11CA45341) + USI cable	5500
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965

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

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп’ютери	8	10947	8757,6	96333,6
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	114885,1

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400
Група 4			
3. Обчислювальна техніка	114885	-	-
Всього по групі	114885	50	57442,5
Група 5			
4. Вимірювальні пристрої	5190	-	-
5. Господарський інвентар	28000	-	-
Всього по групі	33190	25	8297,5
Нематеріальні активи			
6. Нематеріальні активи	50000	10	5000
Разом	$К_р$= 1606075		$А_р$= 141140

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 Ne – Кількість екземплярів програм, шт.

$$З_о = 608 \cdot 137,5/50 = 1672 \text{ грн}$$

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, %

$$З_д = 1672 \cdot 10 \cdot 0,01 = 167,2 \text{ грн}$$

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

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

де $Н_с$ – відрахування на соціальні потреби, %

$$С_{оц} = 0,01 \cdot 22(1672+167,2) = 681 \text{ грн}$$

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

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

де $Н_г$ – загальногосподарські витрати, %

$$Г_{осп} = 1672 \cdot 15 \cdot 0,01 = 251 \text{ грн}$$

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, картриджів, тонеру, грн., $N_е$ – кількість екземплярів програм, шт.

Згідно прийнятих норм на підприємстві $n_{вит}$ приймаємо 0,5 пачки паперу на період розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=210 грн., визначаємо вартість паперу за період розробки:

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

$$З_{М1} = 210 \cdot 1 \cdot 0,5 = 105 \text{ грн.}$$

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

$$З_{М2} = \sum Ц_д \,, \tag{7.17}$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR box – 23,6 грн./шт., DVD-R box – 41,2 грн./шт.

$$З_{М2} = 41{,}2 \cdot 10 = 412 \text{ грн.}$$

Згідно норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_з., \tag{7.18}$$

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

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

$$З_М = (105 + 412 + 1702)/50=44 \text{ грн.}$$

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

$$О_п = З_о \cdot Н_п \cdot 0{,}01, \tag{7.19}$$

де Нп – норматив витрат на освоєння нових мов програмування, %

$$О_п = 1672 \cdot 15 \cdot 0{,}01=251 \text{ грн}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 50 прим.)

$$А_м = \frac{А_р \cdot N_{міс}}{N_е \cdot 12}, \tag{7.20}$$

де Ар – загальна річна сума амортизаційних відрахувань, грн.

$$А_м = 141140 \cdot 3/(50 \cdot 12)=706 \text{ грн}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції

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

$$C_п = 1672+167+681+251+44+251+706 = 3772 \text{ грн.}$$

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

Для даного програмного забезпечення рівень рентабельності складає 55%

$$П_р = 0{,}01 \cdot Р_п \cdot С_п, \quad (7.22)$$

де Рс – рівень рентабельності, %

$$П_р = 0{,}01 \cdot 55 \cdot 3772 = 2075 \text{ грн.}$$

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. Основна зарплата виконавців	$З_о$	1672
2. Додаткова зарплата виконавців	$З_д$	167
3. Відрахування на соціальні потреби	$С_{оц}$	681
4. Загальногосподарські витрати	$Г_{осп}$	251
5. Витрати на матеріали	$З_М$	44
6. Освоєння нових операційних систем, мов програмування	$О_п$	251
7. Амортизація основних фондів	$А_м$	706
8. Повна собівартість програмного забезпечення	$С_п$	3772
9. Плановий прибуток	$П_р$	2075
10.Ціна підприємства $Ц_п=С_п+П_р$	$Ц_п$	5847
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	1169,4
12.Відпускна ціна програмної продукції Ц = Цп + ПДВ	$Ц$	7922

7.6 Визначення об’єму капітальних вкладень у споживача програмної продукції

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

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

Найменування капітальних вкладень	Сума за варіантами, грн	
	Базовий	Новий
Вартість програмної продукції	–	7922
Всього капітальних витрат	–	7922

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	$З_р$	42944	29524
2. Витрати на електроенергію	$З_{ел}$	722	497
3. Витрати на амортизацію	$З_{ам}$	0	1981
Всього витрат за рік	$І$	43666	32002

Витрати на оплату праці:

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

де Тр – кількість годин обслуговування системи за рік, *год.*; Зг – заробітна плата обслуговуючого персоналу, *грн / год*.

Після купівлі нового програмного забезпечення кількість годин обслуговування системи зменшилась з 400 годин до 275 годин на рік, тому витрати на обслуговування склали:

$$З_{р\ баз} = 400 \cdot 80 \cdot 1{,}1 \cdot 1{,}22 = 42944\ грн.$$

до

$$З_{р\ нов} = 275 \cdot 80 \cdot 1{,}1 \cdot 1{,}22 = 29524\ грн.$$

Витрати на електроенергію визначаються з урахуванням спожитої потужності (Пел) в кіловатах, часу експлуатації технічних засобів (Тр) в годинах та ціни однієї кіловат-години (Цел).

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

Зел баз=·0,475·400·3,8 = 722 грн

Зел нов=·0,475·275·3,8 = 497 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	–	7922	–	1980,5
Всього відрахувань	-	–	7922	–	1980,5

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

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

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

$$Ев = (5847 - 3772) \cdot 50 - (0,05 \cdot 1408000 + 0,5 \cdot 114885 +$$
$$+0,25 \cdot 33190 + 0,1 \cdot 50000) \cdot 3/12 = 68465 \ грн.$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

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

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

$$T_в = \frac{198075}{(5847 - 3772) \cdot 50 \cdot 12 / 3} = 0,5 \ років.$$

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

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

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

$$E_{сп} = (43666 - 32002) - 0,25 \cdot 7922 = 9684 \text{ грн.}$$

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

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

$$T_{сп} = \frac{7922}{43666 - 32002} = 0,7 \text{ року}$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	50
2. Повна собівартість розробленої програми	Грн.	3772
3. Ціна розробленої програми	Грн.	5847
4. Плановий прибуток від реалізації розробленої програми	Грн.	2075
5. Рентабельність програмної продукції	%	55
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1606075
7. Загальний прибуток від реалізації програмної продукції	Грн.	103750
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	68465
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	0,5
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	7922
11.Величина економічного ефекту у користувача програмної продукції	Грн.	9684
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,7

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

Законом України “Про охорону праці” [3] 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 польців та м'язи рук) при роботі з клавіатурою мають теж істотне навантаженням [1]. До шкідливих факторів, які впливають на робитників 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 вулючають 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 визначемо фактори, які можуть викликати професійне захворювання і впливають на працездатність програміста,

8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

Електронно-обчислювальні машини (ЕОМ) та інше обладнання є джерелами небезпеки ураження електричним струмом.

На робочому місці програміста виникають небезпечні та шкідливі фактори: підвищений рівень шуму, несприятливі мікрокліматичні умови, недостатній рівень освітленості, шкідливі речовини, підвищений рівень електромагнітних випромінювань радіочастот, висока напруга електричної мережі, статична електрика та інші.

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;

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

– несприятливі мікрокліматичні умови;

– нервово-емоційна напруженість праці;

– інтелектуальні навантаження;

– монотонність праці;

– невідповідність ергономічних показників робочого місця діючим вимогам;

– шум;

– статичні навантаження на кістково-м’язовий апарат.

Вивчення умов праці на робочому місці програміста є необхідною умовою запобігання впливу небезпечних та шкідливих факторів на організм персоналу.

Дослідження санітарно-гігієнічних умов у приміщенні звичайно проводять з використанням пристроїв вимірювання параметрів мікроклімату – вологи повітря (психрометр), освітленості (люксметр) та шуму (шумомір), а також відомих розрахункових методик.

8.3 Пожежна безпека

Пожежна безпека на підприємстві it-індустрії являє собою комплекс організаційних заходів щодо забезпечення нормального протипожежного стану об’єкта нерухомості – житлового, виробничого, складського, офісного або торгово-розважальної споруди, приміщення або комплексу приміщень.

Нормативною основою цього питання на підприємствах усіх форм власності є Кодекс цивільного захисту України та його 13-й розділ, а також Правила пожежної безпеки України. Згідно цих нормативно-правових актів керівник і власник підприємства, компанії або організації несе повну адміністративну та кримінальну відповідальність за своєчасне і правильне введення і підтримання протипожежного режиму. Згідно Державного стандарту України 2272:2006 «Пожежна безпека. Терміни та визначення основних понять» [8] під протипожежним режимом “розуміється звід правил поведінки людей, виконання робіт і експлуатації об’єкта нерухомості, призначений для

забезпечення пожежної безпеки". Керівник в свою чергу може делегувати частину своїх обов'язків і повноважень спеціально призначеному наказом по підприємству співробітнику. У більшості випадків призначається одна відповідальна посадова особа по підприємству або компанії в цілому і окремо – в кожному з окремих будівель, філій, відділів, дивізійних і структурних підрозділів.

Призначений співробітник приступає до виконання своїх обов'язків на підставі наказу по підприємству. Відповідальну особу в обов'язковому порядку знайомлять з цим документом, який він запевняє своїм підписом. У кожному з окремих приміщень, що охороняються повинна бути розташована інформаційна бирка з прізвищем, ім'ям, по батькові та номером контактного телефону особи, відповідальної за протипожежну безпеку. Конкретний перелік обов'язків щодо забезпечення режиму на об'єкті, комплектації, збереження та експлуатації засобів та інструментів протипожежного захисту вказується в посадових інструкціях, положеннях і наказах.

Чинними нормами і законодавчими актами України не передбачений точний список вимог, що висуваються до відповідальної особи – вік, гендерна приналежність та інші фактори значення не мають. Найчастіше компанії та організації висувають на цю посаду висококваліфікованого фахівця з вищою або середньою технічною професійною освітою, що має безперервний стаж роботи в галузі діяльності підприємства не менше ніж три роки. Крім того, бажано, що б здобувач мав мінімальний досвід діяльності в сфері пожежної безпеки, деякі технічні вміння та навички, Досвід роботи з комп'ютером. Відповідальна особа повинна пройти навчання з питань пожежної безпеки з відривом від виробництва, а в подальшому проходити курси в навчальному центрі не рідше, ніж один раз на три роки.

До кола обов'язків особи, відповідальної за протипожежну безпеку в компанії, організації або на підприємстві входять:

– розробка і ведення документації – планів з евакуації персоналу в разі пожежі, ведення журналів обліку, наказів, і спеціальних інструкцій щодо захисту об’єктів і використання первинних засобів гасіння, інструкцій з техніки безпеки, технічного обслуговування вогнегасників різних марок і типів, систем пожежної сигналізації та інше;

– організація навчання працівників, проведення інструктажів;

– матеріально-технічне забезпечення підприємства протипожежним інвентарем – первинними засобами пожежогасіння, включаючи переносні і пересувні вогнегасники, інформаційними стендами з актуальними матеріалами, пожежними щитами і знаками пожежної безпеки та іншими;

– забезпечення технічної справності протипожежного обладнання, інструментів та спеціального інвентарю;

– періодичні перевірки стану ввіреного об’єкта;

– здача звітності до контролюючих органів, допомога у здійсненні планових та позапланових перевірок;

– організація безпечної евакуації співробітників і персоналу, матеріальних цінностей, організація гасіння вогнищ загоряння або задимлення в разі виникнення пожежі.

– Крім евакуаційного плану в будівлях і приміщеннях з масовим скупченням людей повинні бути інструкції з евакуації. Плани та інструкції повинні бути розвішені в приміщеннях на добре видимих місцях. Не рідше, ніж один раз на календарний рік на підприємствах повинні проводитися тренування на випадок виникнення надзвичайних ситуацій.

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 значеннь;

Таким чином можна припустити, що основною причиною можнивого 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 опору ланцюга).

Так як при ураженні електричним струмом у людини може статися фібриляція шлуночків серця, в організації бажано мати дефібрилятор і підготовлений персонал для роботи з ним.

8.5 Розрахункова частина

Для захисного штучного заземлення застосовуються вертикальні електроди: металевий куток *80·50·6 мм*., (згідно з ДСТУ 8769:2018 «Кутики сталеві гарячекатані нерівнополичні. Сортамент») довжиною *L=1,7 м.,* та горизонтальний електрод – металева полоса з перетином *60·5 мм*. Напруга – *220/380 В*. Розрахункова схема розташування заземлюючих електродів – у ряд.

Розрахунок проведемо за допустимим опором розтіканню струму заземлювача.

Початкові дані для розрахунку захисного заземлення: тип верхнього шару грунта – чорнозем, нижнього шару грунта – глина (питомий опір $\rho_2 = 40$ *Ом·м)*. Умовна товщина верхнього шару грунта: $H=0,5$ *м*. Відстань між вертикальними заземлювачами (електродами) $A=1,7$ *м.* Глибина закладення горизонтального контура заземлення $t=0,6$ *м*. Опір заземлювача, який нормується: $R_{ЗН} = 4$ *Ом*. Необхідно визначити необхідну кількість вертикальних заземлювачів та длину полоси (горизонтального заземлювача).

Розрахунок

Відстань від центра вертикального заземлювача до поверхні землі:

$$T=t+L/2=0,6+1,7/2=1,45 \text{ м.}$$

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

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

де $\psi = 1,36$ – табличне значення коефіцієнта сезонності для відповідної климатичной зони у багатошаровому грунті [10];

$\rho_2 = 40$ *Ом·м.* – табличне значення питомого опору нижнього шару грунта (глина) [10].

Еквівалентний діаметр вертикального електрода (кутка) [10]:

$$Dв=0,95 \cdot K = 0,95 \cdot 65 = 59,85 \text{ мм.} = 0,62 \text{ м.}$$

де $K = (80+50)/2 = 65$ *мм.* – середній розмір полиці металевого кутка.

Відношення $A/L=1,7/1,7=1$

Опір розтіканню електричного струму одного електрода вертикального заземлювача з урахуванням заглиблення заземлювача [10]:

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

$$=0,366(54,5/1,7)[(\lg(2 \cdot 1,7/0,62)+(1/2)\lg((4 \cdot 1,45+1,7)/(4 \cdot 1,45-1,7))] =$$

$$=21,9 \text{ Ом.}$$

Визначаємо коефіцієнт екранування вертикальних електродів $Кев=0,8$ при орієнтовній кількості вертикальних електродів, яке дорівнює 5 [10].

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

$$N=R_O /(Кев\ R_{ЗН})= 21,9 / (0,8\cdot4)= 6,85 \approx 7\ шт.$$

Визначаємо длину з'єднуючої полоси:

$$L_П =1,05\cdot A\cdot N= 1,05\cdot1,7\cdot7=12,22 \approx 12\ м.$$

Опір розтіканню електричного струму з'єднуючої полоси з урахуванням кліматичного коефіцієнта питомого опору грунта $K_П$ [10]:

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

$$=0,366(40\cdot5/12)\cdot lg((2\cdot12^{\,2})/(0,06\cdot0,6))=23,5\ Ом.$$

де $K_П=5$ – табличне значення кліматичного коефіцієнта питомого опору грунта для відповідної климатичной зони для з'єднуючої полоси [10]:

$B = 60$ *мм.* $= 0,06$ *м.*- ширина з'єднуючої полоси (задана).

Загальньний опір розтіканню електричного струму заземлювача [10]:

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

$$=(21,9\cdot 23,5)/(21,9\cdot 0,75+ 9\cdot 23,5\cdot 0,8)=3,48\ Ом.$$

де $\eta_П = 0,75$ – табличне значення коефіцієнта екранування з'єднуючої полоси [10].

Умова $R \le R_{ЗН}$ виконується ($3,48 \le 4$).

Так як при *7* вертикальних електродах R суттево менше $R_{ЗН}$, зменшимо кількість вертикальних електродів N до *6* і виконаємо перерахунок. У результаті остаточно отримали: $R = 3,98$ *Ом.* при кількості вертикальних електродів $N=6$.

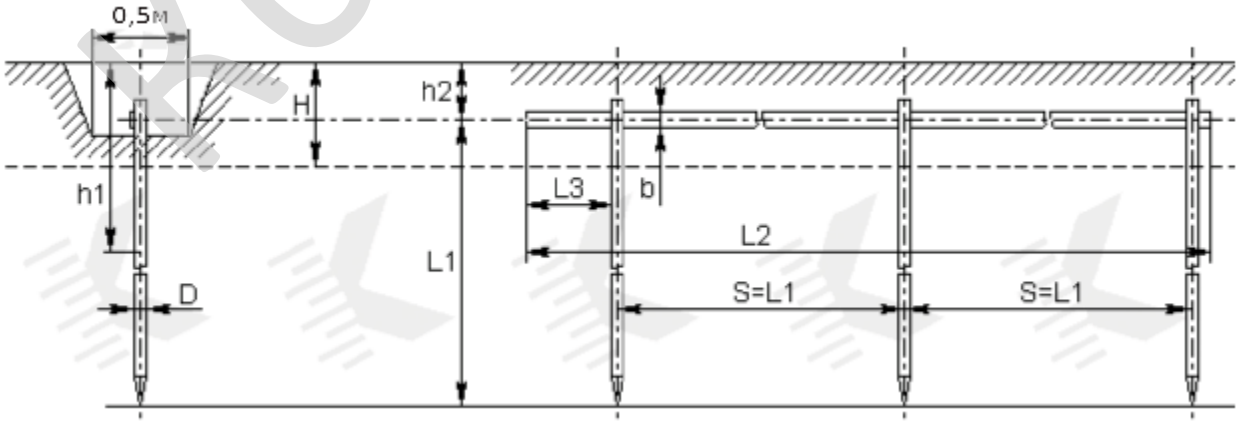

Рисунок 8.1 – Схема штучного заземлення

Висновки до розділу

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

З цих міркувань було здійснено аналіз умов праці на робочому місці програміста, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи, розроблені заходи з умов поліпшення охорони прац.

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

Список використаних джерел інформації

1. Державні будівельні норми України: ДБН В.2.5-28:2018. – Режим доступу до ресурсу: https://goo.su/9AkQ (дата звернення 19.10.23).

2. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин: ДСанПІН 3.3.2-007-98. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/v0007282-98 (дата звернення 19.10.23).

3. Закон України «Про охорону праці» від 14.10.1992 р. № 2694-XII. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2694-12 (дата звернення 19.10.23).

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

5. Наказ Міністерства соціальної політики України 14.02.2018 № 207 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z0508 (дата звернення 19.10.23).

6. Постанова № 42 від 01.12.1999 Головного державного санітарного лікаря України «Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/va042282-99 (дата звернення 19.09.23).

7. Оришака, О. В. Основи охорони праці: навч. посіб. / О. В. Оришака, Г. П. Горбачова, К. М. Марченко; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. – Кропивницький : ЦНТУ, 2022. – 175 с. – Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/12161 (дата звернення 19.09.23).

8. Оришака О.В. Охорона праці в галузі та цивільний захист / О.В Оришака, Г.П. Горбачова, О.М. Мезенцева , К.М. Марченко, К.О. Буравченко; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. – Кропивницький: Видавець Лисенко В.Ф., 2019. − 226 с. – Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/9258 (дата звернення 19.09.23).

9. Методичні рекомендації до виконання розділу "Заходи з охорони праці та техніки безпеки" випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти для здобувачів вищої освіти спеціальностей 123 "Комп'ютерна інженерія" та 122 "Комп'ютерні науки" / М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т, каф. кібербезпеки та програм. забезпечення; [укл. О.В. Оришака, К.М. Марченко]. – Кропивницький: ЦНТУ, 2022. – 19 с. [Електронний ресурс]. – Режим доступу: http://dspace.kntu.kr.ua/jspui/handle/123456789/12240 (дата звернення 19.09.23).

10. Охорона праці. Ч. 1. Захисне заземлення : метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ сумісного типу / Кіровоград. ін-т с.-г. машинобуд. ; [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака]. – Кіровоград : КІСМ, 1997. – 20 с. – Режим доступу: http://dspace.kntu.kr.ua/jspui/handle/123456789/4358 (дата звернення 19.09.23).

11. Наказ Міністерства регіонального розвитку, будівництва та житлово-комунального господарства України 31.10.2016 «Про затвердження ДБН В.1.1-702016» – Режим доступу до ресурсу: ttps://ips.ligazakon.net/document/fn025551 (дата звернення 19.09.23).

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи верстата ЧПК обробки електроізоляційного картону.

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 верстата ЧПК обробки електроізоляційного картону.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем верстата ЧПК обробки електроізоляційного картону.

– Досліджена система верстата ЧПК обробки електроізоляційного картону.

– На основі отриманих результатів досліджень створена програмна реалізація системи верстата ЧПК обробки електроізоляційного картону.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання верстата ЧПК обробки електроізоляційного картону.

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, G-CODE. 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 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм Serpent.

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 9684 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,7 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Коваль В.В. Дослідження та програмна реалізація системи верстата ЧПК обробки електроізоляційного картону // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2023.

2. І.В.Чихіра, А.Г. Микитишин Конспект лекцій з дисципліни «Програмування систем реального часу» / Укладачі : Чихіра І.В., Микитишин А.Г., – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя , 2016. – 76 с.

3. Jack Ganssle and Michael Barr. 2003. Embedded Systems Dictionary. CMP Books.

4. Технології інтернету речей. Навчальний посібник [Електронний ресурс]: / Б. Ю. Жураковський, І.О. Зенів; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 12,5 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2021. – 271 с.

5. Greg Dunko, Joydeep Misra, Josh Robertson, Tom Snyder “A reference guide to the Internet of Things” / 2017 Bridgera LLC, RIoT.

6. Donald Norris “Programming with STM32. Getting started with the Nucleo Board and C/C++” 416 p. 2018.

7. Neil Kolban “Kolban’s book on ESP32”. Texas, USA. 951 p.

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

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

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

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

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

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

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

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

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

17. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated

with the Markov chains», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 633-645.

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

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

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

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

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

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

24. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in

Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 353-358.

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

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

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

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

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

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

31. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик

запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»,* № 2 (307). С. 46-52. 2022.

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

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

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

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

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

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

38. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга

Маркова». *Центральноукраїнський науковий вісник. Технічні науки.* № 2(33). с. 161-172, 2019.

39. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В. Поліщук Л.І. Проектування комп'ютерних систем та мереж. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2019. – 264 с.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи верстата ЧПК обробки електроізоляційного картону.

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. № 34-13 від 04.08.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи верстата ЧПК обробки електроізоляційного картону.

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 верстата ЧПК обробки електроізоляційного картону;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Delphi 10, G-CODE.

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 2023 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинна бути розглянута пожежна безпека.

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 – 121 аркуш.

10 Етапи розробки

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

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

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

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

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

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 12.12.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

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

_____ Коваленко О.В.

Дослідження та програмна реалізація системи верстата ЧПК обробки електроізоляційного картону

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 28

Літера: РП

Кропивницький – 2023 року

VERSTAT.DPR – ФАЙЛ ОСНОВНОЇ ПРОГРАМИ

```
program VERSTAT; //Назва програми

// РОЗРОБИВ КОВАЛЬ ВЛАДИСЛАВ ВІКТОРОВИЧ
// КРОПИВНИЦЬКИЙ – 2023 рік

uses // бібліотеки
  Forms,
  main in 'main.PAS' {Main},
  about in 'about.pas' {About},
  UserRegs in 'UserRegs.pas' {UserRegisters},
  Com_D in 'Com_D.pas';

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TMainForm, MainForm);
  Application.CreateForm(TAboutBox, AboutBox);
  Application.Title:='VERSTAT';
  Application.Run; //Запукс ПЗ
end.
```

MAIN.PAS – ФАЙЛ, ГОЛОВНЕ ВІКНО ПЗ

```
unit MAIN; // Назва модулю

interface // розділ оголошення

uses // які бібліотекі використовувались
  Windows, SysUtils, Classes, Graphics, Forms, Controls, Menus,
  StdCtrls, Dialogs, Buttons, Messages, ExtCtrls, ComCtrls,
  StdActns, ActnList, ToolWin, ImgList, UserRegs,  SysRegs,
  MemEdit, DebugView, LogToLine, LineToPhysic, CPU, jpeg;

Type // об'ява власних типів даних
  TMainForm = class(TForm)
    MainMenu: TMainMenu;
    File1: TMenuItem;
    FileNewItem: TMenuItem;
    FileOpenItem: TMenuItem;
    Window1: TMenuItem;
    Help1: TMenuItem;
    N1: TMenuItem;
    FileExitItem: TMenuItem;
    HelpAboutItem: TMenuItem;
    OpenDialog: TOpenDialog;
    FileSaveItem: TMenuItem;
    FileSaveAsItem: TMenuItem;
    Edit1: TMenuItem;
    CutItem: TMenuItem;
    CopyItem: TMenuItem;
    PasteItem: TMenuItem;
    StatusBar: TStatusBar;
    ActionList1: TActionList;
    EditCut1: TEditCut;
    EditCopy1: TEditCopy;
    EditPaste1: TEditPaste;
    FileNew1: TAction;
    FileSave1: TAction;
    FileExit1: TAction;
    FileOpen1: TAction;
    FileSaveAs1: TAction;
    WindowCascade1: TWindowCascade;
    WindowTileHorizontal1: TWindowTileHorizontal;
    WindowArrangeAll1: TWindowArrange;
    WindowMinimizeAll1: TWindowMinimizeAll;
    HelpAbout1: TAction;
    FileClose1: TWindowClose;
    WindowTileVertical1: TWindowTileVertical;
    ImageList1: TImageList;
    SaveDialog: TSaveDialog;
    Run: TAction;
    Step: TAction;
    StepInto: TAction;
    N2: TMenuItem;
    Reset: TAction;
    Image1: TImage;
    N3: TMenuItem;
    N11: TMenuItem;
    N21: TMenuItem;
    N31: TMenuItem;
    N41: TMenuItem;
    N51: TMenuItem;
    procedure FileNew1Execute(Sender: TObject);
    procedure FileOpen1Execute(Sender: TObject);
    procedure HelpAbout1Execute(Sender: TObject);
    procedure FileExit1Execute(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure MenuTreeKeyPress(Sender: TObject; var Key: Char);
    procedure tool_btn_saveClick(Sender: TObject);
    procedure FileSaveAsItemClick(Sender: TObject);
```

```
    procedure RunExecute(Sender: TObject);
    procedure StepExecute(Sender: TObject);
    procedure StepIntoExecute(Sender: TObject);
    procedure ResetExecute(Sender: TObject);
    procedure N3Click(Sender: TObject);
    procedure N11Click(Sender: TObject);
    procedure N21Click(Sender: TObject);
    procedure N31Click(Sender: TObject);
    procedure N41Click(Sender: TObject);
    procedure N51Click(Sender: TObject);
  private
    { Private declarations }
    stateDefault  :boolean;
    stateFileName :string;
    function  FormInstanceExists(FormClassName: string) :boolean;
  public
    { Public declarations }
    C :TDATA;
    { Формы просмотров }
    UserRegisters:   TUserRegisters;
    SystemRegisters: TSystemRegisters;
    DebugerView:     TDebugerView;
    procedure UpdateViews(SenderFormName: string = '');
  end;

var
  MainForm: TMainForm;

Implementation // розділ реалізації

{$R *.dfm} // ресурси форми

uses // які бібліотекі використовувались
  strman,
  about,
  Common,
  ErrorLog,
  Msg;

var
  sm :TStringManager;

//-----------------------------------------------------------------// MDI Form
Actions, тип форми
//-----------------------------------------------------------------//Main form
create - створення головної форми
procedure TMainForm.FormCreate(Sender: TObject);
begin
  MenuTree.Items.GetFirstNode.Selected := True;
  MenuTree.Items.GetFirstNode.Expand(True);

  //створення DATA
  C := TDATA.Create;

  //Load state
  stateDefault  := true;
  stateFileName := ExtractFilePath(Application.ExeName)+'1.dat';
  if not C.LoadState(ExtractFilePath(Application.ExeName)+'Files\1.dat') then
  begin
    Application.MessageBox('Невірний формат',
                           'Error');
    Close;
  end;
MainForm.Width:=453;
MainForm.Height:=579;
end;

//-----------------------------------------------------------------
//Open/Save
```

```
procedure TMainForm.FileNew1Execute(Sender: TObject);
begin
 if Application.MessageBox(PAnsiChar('Збереження даних'+stateFileName+'?'), '',
MB_YESNO OR MB_ICONWARNING) = 6 then
  C.SaveState(stateFileName);
  stateDefault  := true;
  stateFileName := ExtractFilePath(Application.ExeName)+'1.dat';
  C.LoadState(ExtractFilePath(Application.ExeName)+'Files\1.dat');
  UpdateViews;
end;

procedure TMainForm.FileOpen1Execute(Sender: TObject);
var
  openflag :boolean;
begin
  openflag := false;
  if OpenDialog.Execute then
  begin
    if fileexists(OpenDialog.FileName) then
      openflag := true
    else
      Application.MessageBox('Файл незнайдено', 'Error');
  end;

  if openflag then
  begin
    //Save
    if Application.MessageBox(PAnsiChar('Збереження'+stateFileName+'?'), '',
MB_YESNO OR MB_ICONWARNING) = 6 then
   C.SaveState(stateFileName);
    if C.LoadState(OpenDialog.FileName) then
    begin
      stateDefault  := false;
      stateFileName := OpenDialog.FileName;
    end
    else
      Application.MessageBox('Невірний формат файлу', 'Error')
  end;
  UpdateViews;
end;

procedure TMainForm.tool_btn_saveClick(Sender: TObject);
begin
  if stateDefault then
  begin
    SaveDialog.FileName := stateFileName;
    if SaveDialog.Execute then
    begin
      SaveDialog.FileName := sm.Strip(SaveDialog.FileName);
      if not sm.IsLast('.dat', SaveDialog.FileName) then
        SaveDialog.FileName := SaveDialog.FileName + '.dat';
      stateDefault  := false;
      stateFileName := SaveDialog.FileName;
    end;
  end;

  C.SaveState(stateFileName);
end;

procedure TMainForm.FileSaveAsItemClick(Sender: TObject);
begin
  SaveDialog.FileName := stateFileName;
  if SaveDialog.Execute then
  begin
    SaveDialog.FileName := sm.Strip(SaveDialog.FileName);
    if not sm.IsLast('.dat', SaveDialog.FileName) then
      SaveDialog.FileName := SaveDialog.FileName + '.dat';
    stateFileName := SaveDialog.FileName;
  end;
```

```
  C.SaveState(stateFileName);
end;
//---------------------------------------------------------------------
//Процедура запуску
procedure TMainForm.RunExecute(Sender: TObject);
begin
   DATA.Run;
end;

procedure TMainForm.StepExecute(Sender: TObject);
begin
  C.Step;
  UpdateViews;
end;

procedure TMainForm.StepIntoExecute(Sender: TObject);
begin
  C.StepInto;
  UpdateViews;
end;

procedure TMainForm.ResetExecute(Sender: TObject);
begin
  C.Reset;
  C.setRegister('EIP', 0);
  UpdateViews;
end;

// авторське право
procedure TMainForm.HelpAbout1Execute(Sender: TObject);
begin
  AboutBox.show;
end;

// завершення роботи
procedure TMainForm.FileExit1Execute(Sender: TObject);
begin
  Close;
end;

//---------------------------------------------------------------------
//Menu
procedure TMainForm.MenuTreeKeyPress(Sender: TObject; var Key: Char);
begin
  if Key = #13 then MenuTreeClick(Sender);
end;

// Перевіряє, чи існує відкритий екземпляр форми
function TMainForm.FormInstanceExists(FormClassName: string) :boolean;
var
  i:integer;
begin
  FormInstanceExists := false;

  for i := 1 to MDIChildCount do
    if(MDIChildren[i-1].Name = FormClassName) then
      FormInstanceExists := true;
end;

procedure TMainForm.UpdateViews(SenderFormName: string = '');
var
  i :integer;
begin
  for i := 1 to MDIChildCount do
    if(MDIChildren[i-1].Name <> SenderFormName) then
    begin
      if MDIChildren[i-1].Name = '1' then
        (MDIChildren[i-1] as TUserRegisters).UpdateView;
```

```
      if MDIChildren[i-1].Name = '2' then
        (MDIChildren[i-1] as TSystemRegisters).UpdateView;

      if MDIChildren[i-1].Name = '3' then
        (MDIChildren[i-1] as TDebugerView).UpdateView;

      if MDIChildren[i-1].Name = '4' then
        (MDIChildren[i-1] as TLogicalToLine).UpdateView;

      if MDIChildren[i-1].Name = '5' then
        (MDIChildren[i-1] as TLineToPhysical).UpdateView;
    end;
end;

procedure TMainForm.N3Click(Sender: TObject);
begin
LogicalToLine.Show; // показати вікно LogicalToLine та
MainForm.Hide; // приховати головне вікно - MainForm
end;

procedure TMainForm.N11Click(Sender: TObject);
begin
Image1.Picture.LoadFromFile('1.jpg'); // завантаження зображення
end;

procedure TMainForm.N21Click(Sender: TObject);
begin
Image1.Picture.LoadFromFile('2.jpg'); // завантаження зображення
end;

procedure TMainForm.N31Click(Sender: TObject);
begin
Image1.Picture.LoadFromFile('3.jpg'); // завантаження зображення
end;

procedure TMainForm.N41Click(Sender: TObject);
begin
Image1.Picture.LoadFromFile('4.jpg'); // завантаження зображення
end;

procedure TMainForm.N51Click(Sender: TObject);
begin
Image1.Picture.LoadFromFile('5.jpg'); // завантаження зображення
end;

end.
```

БІБЛІОТЕКА COM_ DATA.PAS – РОБОТА З COM ПОРТОМ

```
unit COM_ DATA;

interface

uses
  // Модулі Delphi
  Windows, Messages, SysUtils, Classes, Forms;

//---------------------------------------------------------------
// Property types
//---------------------------------------------------------------

type
  // Швидкість передачі даних (custom or 110...256k bauds)
  TBaudRate = ( brCustom,
                br110, br300, br600, br1200, br2400, br4800,
                br9600, br14400, br19200, br38400, br56000,
                br57600, br115200, br128000, br256000 );
  // Port Numbers ( custom or COM1..COM16 )
  TPortNumber = ( pnCustom,
         pnCOM1, pnCOM2, pnCOM3, pnCOM4, pnCOM5, pnCOM6, pnCOM7,
         pnCOM8, pnCOM9, pnCOM10, pnCOM11, pnCOM12, pnCOM13,
         pnCOM14, pnCOM15, pnCOM16 );
  // Дані bits ( 5, 6, 7, 8 )
  TDataBits = ( db5BITS, db6BITS, db7BITS, db8BITS );
  // Stop bits ( 1, 1.5, 2 )
  TStopBits = ( sb1BITS, sb1HALFBITS, sb2BITS );
  TParity = ( ptNONE, ptODD, ptEVEN, ptMARK, ptSPACE );
  THwFlowControl = ( hfNONE, hfNONERTSON, hfRTSCTS );
    TSwFlowControl = ( sfNONE, sfXONXOFF );
   TPacketMode = ( pmDiscard, pmPass );

//---------------------------------------------------------------
// Типи обробників
//---------------------------------------------------------------

type
 TReceiveDataEvent = procedure( Sender: TObject; DataPtr: pointer;
                                DataSize:DWORD ) of object;
  TReceivePacketEvent = procedure( Sender: TObject; Packet:
                           pointer; DataSize:DWORD ) of object;

//---------------------------------------------------------------
// Інші типи
//---------------------------------------------------------------

type
  //Статус Clear To Send, Data Set Ready, Ring, Carrier Detect
  TLineStatus = ( lsCTS, lsDSR, lsRING, lsCD );
  TLineStatusSet = set of TLineStatus;

//---------------------------------------------------------------// Константи
//---------------------------------------------------------------

const
  RELEASE_NOCLOSE_PORT = HFILE(INVALID_HANDLE_VALUE-1);

type
  TCommPortDriver = class( TComponent )
  protected
    FHandle: HFILE;
    FPort                     : TPortNumber;
    FPortName                 : string;
    FBaudRate                 : TBaudRate;
    // Baud rate ( actual numeric value )
    FBaudRateValue            : DWORD;
```

```
  // Data bits size (dbXXX)
  FDataBits                  : TDataBits;
  FStopBits                  : TStopBits;
  FParity                    : TParity;
  FHwFlow                    : THwFlowControl;
  FSwFlow                    : TSwFlowControl;
  FInBufSize                 : DWORD;
  FOutBufSize                : DWORD;
  FPacketSize                : smallint;
  FPacketTimeout             : integer;
  FPacketMode                : TPacketMode;
  FOnReceiveData             : TReceiveDataEvent;
  FOnReceivePacket           : TReceivePacketEvent;
  FPollingDelay              : word;
  FEnableDTROnOpen           : boolean;
  FOutputTimeout             : word;
  FInputTimeout              : DWORD;
  FCkLineStatus              : boolean;
  FNotifyWnd                 : HWND;
  FTempInBuffer              : pointer;
  FFirstByteOfPacketTime     : DWORD;
  FRXPollingPauses           : integer;

  procedure SetHandle( Value: HFILE );
  procedure SetPort( Value: TPortNumber );
  procedure SetPortName( Value: string );
  procedure SetBaudRate( Value: TBaudRate );
  procedure SetBaudRateValue( Value: DWORD );
  procedure SetDataBits( Value: TDataBits );
  procedure SetStopBits( Value: TStopBits );
  // Selects the kind of parity
  procedure SetParity( Value: TParity );
  // Selects the kind of hardware flow control
  procedure SetHwFlowControl( Value: THwFlowControl );
  procedure SetSwFlowControl( Value: TSwFlowControl );
  procedure SetInBufSize( Value: DWORD );
  procedure SetOutBufSize( Value: DWORD );
  procedure SetPacketSize( Value: smallint );
  // Sets the timeout for incoming packets
  procedure SetPacketTimeout( Value: integer );
  procedure SetPollingDelay( Value: word );
  function ApplyCOMSettings: boolean;
  procedure TimerWndProc( var msg: TMessage );
public
  // конструктор
  constructor Create( AOwner: TComponent ); override;
  // Деструктор
  destructor Destroy; override;
  function Connect: boolean;
  // Закриття порту
  procedure Disconnect;
  function Connected: boolean;
  // Повернення статусу
  function GetLineStatus: TLineStatusSet;
  function IsPolling: boolean;
  procedure PausePolling;
  procedure ContinuePolling;
  function FlushBuffers( inBuf, outBuf: boolean ): boolean;
  function CountRX: integer;
  function OutFreeSpace: word;
  function SendData( DataPtr: pointer; DataSize: DWORD ): DWORD;
  function SendDataEx( DataPtr: pchar; DataSize, Timeout: DWORD ): DWORD;
  // Sends a byte. Returns true if the byte has been sent
  function SendByte( Value: byte ): boolean;
  // Sends a char. Returns true if the char has been sent
  function SendChar( Value: char ): boolean;
  function SendString( s: string ): boolean;
  function SendZString( s: pchar ): boolean;
  function ReadData( DataPtr: pchar; MaxDataSize: DWORD ): DWORD;
```

```
    // Reads a byte. Returns true if the byte has been read
    function ReadByte( var Value: byte ): boolean;
    function ReadChar( var Value: char ): boolean;
    // Set DTR line high (onOff=TRUE) or low (onOff=FALSE).
    procedure ToggleDTR( onOff: boolean );
    procedure ToggleRTS( onOff: boolean );
    property Handle: HFILE read FHandle write SetHandle;
  published
    property Port: TPortNumber read FPort write SetPort default pnCOM2;
    // Повернення використовує мого поточного COM порту
    property PortName: string read FPortName write SetPortName;
    // Швидкість ( Baud Rate )
    property BaudRate: TBaudRate read FBaudRate write SetBaudRate default
br9600;
    property BaudRateValue: DWORD read FBaudRateValue write SetBaudRateValue
default 9600;
    property DataBits: TDataBits read FDataBits write SetDataBits default
db8BITS;
    property StopBits: TStopBits read FStopBits write SetStopBits default
sb1BITS;
    property Parity: TParity read FParity write SetParity default ptNONE;
    property HwFlow: THwFlowControl read FHwFlow write SetHwFlowControl default
hfNONERTSON;
    property SwFlow: TSwFlowControl read FSwFlow write SetSwFlowControl default
sfNONE;
    property InBufSize: DWORD read FInBufSize write SetInBufSize default 2048;
    property OutBufSize: DWORD read FOutBufSize write SetOutBufSize default
2048;
    property PacketSize: smallint read FPacketSize write SetPacketSize default -
1;
    property PacketTimeout: integer read FPacketTimeout write SetPacketTimeout
default -1;
    property PacketMode: TPacketMode read FPacketMode write FPacketMode default
pmDiscard;
    property PollingDelay: word read FPollingDelay write SetPollingDelay default
50;
    property EnableDTROnOpen: boolean read FEnableDTROnOpen write
FEnableDTROnOpen default true;
    property OutputTimeout: word read FOutputTimeOut write FOutputTimeout
default 500;
    property InputTimeout: DWORD read FInputTimeOut write FInputTimeout default
200;
    property CheckLineStatus: boolean read FCkLineStatus write FCkLineStatus
default false;
    property OnReceiveData: TReceiveDataEvent read FOnReceiveData write
FOnReceiveData;
    property OnReceivePacket: TReceivePacketEvent read FOnReceivePacket write
FOnReceivePacket;
  end;

function BaudRateOf( bRate: TBaudRate ): DWORD;
function DelayForRX( bRate: TBaudRate; DataSize: DWORD ): DWORD;

implementation // реалізація

const
  Win32BaudRates: array[br110..br256000] of DWORD =
    ( CBR_110, CBR_300, CBR_600, CBR_1200, CBR_2400, CBR_4800, CBR_9600,
      CBR_14400, CBR_19200, CBR_38400, CBR_56000, CBR_57600, CBR_115200,
      CBR_128000, CBR_256000 );
//константи
const
  dcb_Binary              = $00000001;
  dcb_ParityCheck         = $00000002;
  dcb_OutxCtsFlow         = $00000004;
  dcb_OutxDsrFlow         = $00000008;
  dcb_DtrControlMask      = $00000030;
    dcb_DtrControlDisable   = $00000000;
    dcb_DtrControlEnable    = $00000010;
```

```
    dcb_DtrControlHandshake = $00000020;
  dcb_DsrSensivity        = $00000040;
  dcb_TXContinueOnXoff    = $00000080;
  dcb_OutX                = $00000100;
  dcb_InX                 = $00000200;
  dcb_ErrorChar           = $00000400;
  dcb_NullStrip           = $00000800;
  dcb_RtsControlMask      = $00003000;
    dcb_RtsControlDisable   = $00000000;
    dcb_RtsControlEnable    = $00001000;
    dcb_RtsControlHandshake = $00002000;
    dcb_RtsControlToggle    = $00003000;
  dcb_AbortOnError        = $00004000;
  dcb_Reserveds           = $FFFF8000;

// отримання весії ОС
function GetWinPlatform: string;
var ov: TOSVERSIONINFO;
begin
  ov.dwOSVersionInfoSize := sizeof(ov);
  if GetVersionEx( ov ) then
  begin
    case ov.dwPlatformId of
      VER_PLATFORM_WIN32s: // Win32s чи Windows 3.1
        Result := 'W32S';
      VER_PLATFORM_WIN32_WINDOWS: // Win32 чи Windows 95/98
        Result := 'W95';
      VER_PLATFORM_WIN32_NT: //    Windows NT
        Result := 'WNT';
    end;
  end
  else
    Result := '??';
end;

function GetWinVersion: DWORD;
var ov: TOSVERSIONINFO;
begin
  ov.dwOSVersionInfoSize := sizeof(ov);
  if GetVersionEx( ov ) then
    Result := MAKELONG( ov.dwMinorVersion, ov.dwMajorVersion )
  else
    Result := $00000000;
end;

function BaudRateOf( bRate: TBaudRate ): DWORD;
begin
  if bRate = brCustom then
    Result := 0
  else
    Result := Win32BaudRates[ bRate ];
end;

function DelayForRX( bRate: TBaudRate; DataSize: DWORD ): DWORD;
begin
  Result := round( DataSize / (BaudRateOf(bRate) / 10) * 1000 );
end;

constructor TCommPortDriver.Create( AOwner: TComponent );
begin
  inherited Create( AOwner );
  // Initialize to default values -----------------------
  // Not connected
  FHandle                    := INVALID_HANDLE_VALUE;
  FPort                      := pnCOM2;
  FPortName                  := '\\.\COM2';
  FBaudRate                  := br9600;
  FBaudRateValue             := BaudRateOf( br9600 );
  // 8 data bits
```

```
  FDataBits                  := db8BITS;
  // 1 stop bit
  FStopBits                  := sb1BITS;
  // no parity
  FParity                    := ptNONE;
  FHwFlow                    := hfNONERTSON;
  FSwFlow                    := sfNONE;
  // Розмір буферу 2048 bytes
  FInBufSize                 := 2048;
  FOutBufSize                := 2048;
  FPacketSize                := -1;
  FPacketTimeout             := -1;
  FPacketMode                := pmDiscard;
  FPollingDelay              := 50;
  FOutputTimeout             := 500;
  FInputTimeout              := 200;
  FEnableDTROnOpen           := true;
  FFirstByteOfPacketTime     := DWORD(-1);
  FCkLineStatus              := false;
  FRXPollingPauses := 0;
  FTempInBuffer := AllocMem( FInBufSize );
messages
  if not (csDesigning in ComponentState) then
    FNotifyWnd := AllocateHWnd( TimerWndProc );
end;

destructor TCommPortDriver.Destroy;
begin
  // Звільнення порту
  Disconnect;
  // Free the temporary buffer
  FreeMem( FTempInBuffer, FInBufSize );
  // Звільнення таймеру
  if not (csDesigning in ComponentState) then
    DeallocateHWnd( FNotifyWnd );
  inherited Destroy;
end;

procedure TCommPortDriver.SetHandle( Value: HFILE );
begin
  if FHandle = Value then
    exit;
  if Value = RELEASE_NOCLOSE_PORT then
  begin
    // Стоп таймеру
    if Connected then
      KillTimer( FNotifyWnd, 1 );
    FHandle := INVALID_HANDLE_VALUE;
  end
  else
  begin
    // Відключення
    Disconnect;
    if Value = INVALID_HANDLE_VALUE then
      exit;
    // Отримання показчика
    FHandle := Value;
    // Старт таймера
    SetTimer( FNotifyWnd, 1, FPollingDelay, nil );
  end;
end;

// Обрання COM порту
procedure TCommPortDriver.SetPort( Value: TPortNumber );
begin
  if Connected then
    exit;
  FPort := Value;
  if FPort <> pnCustom then
```

```
    FPortName := Format( '\\.\COM%d', [ord(FPort)] );
end;

procedure TCommPortDriver.SetPortName( Value: string );
begin
  if Connected then
    exit;
  FPort := pnCustom;
  FPortName := Value;
end;

procedure TCommPortDriver.SetBaudRate( Value: TBaudRate );
begin
  // Set new COM speed
  FBaudRate := Value;
  if FBaudRate <> brCustom then
    FBaudRateValue := BaudRateOf( FBaudRate );
  // Apply changes
  if Connected then
    ApplyCOMSettings;
end;

procedure TCommPortDriver.SetBaudRateValue( Value: DWORD );
begin
  // Set new COM speed
  FBaudRate := brCustom;
  FBaudRateValue := Value;
  // Apply changes
  if Connected then
    ApplyCOMSettings;
end;

procedure TCommPortDriver.SetDataBits( Value: TDataBits );
begin
  FDataBits := Value;
  if Connected then
    ApplyCOMSettings;
end;

// Обрання стопових бітів - Selects the number of stop bits
procedure TCommPortDriver.SetStopBits( Value: TStopBits );
begin
  // Set new stop bits
  FStopBits := Value;
  // Apply changes
  if Connected then
    ApplyCOMSettings;
end;

procedure TCommPortDriver.SetParity( Value: TParity );
begin
  // Set new parity
  FParity := Value;
  // Apply changes
  if Connected then
    ApplyCOMSettings;
end;

procedure TCommPortDriver.SetHwFlowControl( Value: THwFlowControl );
begin
  // Set new hardware flow control
  FHwFlow := Value;
  // Apply changes
  if Connected then
    ApplyCOMSettings;
end;

procedure TCommPortDriver.SetSwFlowControl( Value: TSwFlowControl );
begin
```

```
  // Set new software flow control
  FSwFlow := Value;
  // Apply changes
  if Connected then
    ApplyCOMSettings;
end;

procedure TCommPortDriver.SetInBufSize( Value: DWORD );
begin
  // Do nothing if connected
  if Connected then
    exit;
  // Free the temporary input buffer
  FreeMem( FTempInBuffer, FInBufSize );
  // Set new input buffer size
  if Value > 8192 then
    Value := 8192
  else if Value < 128 then
    Value := 128;
  FInBufSize := Value;
  // Allocate the temporary input buffer
  FTempInBuffer := AllocMem( FInBufSize );
  // Adjust the RX packet size
  SetPacketSize( FPacketSize );
end;

// Sets the TX buffer size
procedure TCommPortDriver.SetOutBufSize( Value: DWORD );
begin
  // Do nothing if connected
  if Connected then
    exit;
  // Set new output buffer size
  if Value > 8192 then
    Value := 8192
  else if Value < 128 then
    Value := 128;
  FOutBufSize := Value;
end;

procedure TCommPortDriver.SetPacketSize( Value: smallint );
begin
  // PackeSize <= 0 if data isn't to be 'packetized'
  if Value <= 0 then
    FPacketSize := -1
  // If the PacketSize if greater than then RX buffer size then
  // increase the RX buffer size
  else if DWORD(Value) > FInBufSize then
  begin
    FPacketSize := Value;
    SetInBufSize( FPacketSize );
  end;
end;

procedure TCommPortDriver.SetPacketTimeout( Value: integer );
begin
  // PacketTimeout <= 0 if packet timeout is to be disabled
  if Value < 1 then
    FPacketTimeout := -1
  // PacketTimeout cannot be less than polling delay + some extra ms
  else if Value < FPollingDelay then
    FPacketTimeout := FPollingDelay + (FPollingDelay*40) div 100;
end;

procedure TCommPortDriver.SetPollingDelay( Value: word );
begin
  // Make it greater than 4 ms
  if Value < 5 then
    Value := 5;
```

```
  // If new delay is not equal to previous value...
  if Value <> FPollingDelay then
  begin
    // Stop the timer
    if Connected then
      KillTimer( FNotifyWnd, 1 );
    // Store new delay value
    FPollingDelay := Value;
    // Restart the timer
    if Connected then
      SetTimer( FNotifyWnd, 1, FPollingDelay, nil );
    SetPacketTimeout( FPacketTimeout );
  end;
end;

// Збереження налаштувань
function TCommPortDriver.ApplyCOMSettings: boolean;
var dcb: TDCB;
begin
  // Do nothing if not connected
  Result := false;
  if not Connected then
    exit;

  fillchar( dcb, sizeof(dcb), 0 );
  // структура DCB, розмір
  dcb.DCBLength := sizeof(dcb);
  // Baud rate
  dcb.BaudRate := FBaudRateValue;
  dcb.Flags := dcb_Binary;
  if EnableDTROnOpen then
    dcb.Flags := dcb.Flags or dcb_DtrControlEnable;
  case FHwFlow of
    hfNONE:;
    hfNONERTSON:
      dcb.Flags := dcb.Flags or dcb_RtsControlEnable;
    hfRTSCTS:
      dcb.Flags := dcb.Flags or dcb_OutxCtsFlow or dcb_RtsControlHandshake;
  end;
  case FSwFlow of
    sfNONE:;
    sfXONXOFF:
      dcb.Flags := dcb.Flags or dcb_OutX or dcb_InX;
  end;
  if (GetWinPlatform = 'WNT') and (GetWinVersion >= $00040000) then
  begin
    if FInBufSize div 4 > 4096 then
      dcb.XONLim := 4096
    else
      dcb.XONLim := FInBufSize div 4;
  end
  else
    dcb.XONLim := FInBufSize div 4;
  dcb.XOFFLim := dcb.XONLim;
  dcb.ByteSize := 5 + ord(FDataBits);
  dcb.Parity := ord(FParity);
  dcb.StopBits := ord(FStopbits);
  dcb.XONChar := #17;
  dcb.XOFFChar := #19;

  // Примінити нові налаштування
  Result := SetCommState( FHandle, dcb );
  if not Result then
    exit;
  Result := FlushBuffers( true, true );
  if not Result then
    exit;
  Result := SetupComm( FHandle, FInBufSize, FOutBufSize );
end;
```

```
function TCommPortDriver.Connect: boolean;
var tms: TCOMMTIMEOUTS;
begin
  Result := Connected;
  if Result then
    exit;
  // Відкритяя COM порту
  FHandle := CreateFile( pchar(FPortName),
                         GENERIC_READ or GENERIC_WRITE,
                         0, // невикористовується
                         nil,
                         OPEN_EXISTING,
                         FILE_ATTRIBUTE_NORMAL,
                         0
                        ) ;
  Result := Connected;
  if not Result then
    exit;
  // Примінити налаштування
  Result := ApplyCOMSettings;
  if not Result then
  begin
    Disconnect;
    exit;
  end;

  tms.ReadIntervalTimeout := 1;
  tms.ReadTotalTimeoutMultiplier := 0;
  tms.ReadTotalTimeoutConstant := 1;
  tms.WriteTotalTimeoutMultiplier := 0;
  tms.WriteTotalTimeoutConstant := 10;
  // Apply timeouts
  SetCommTimeOuts( FHandle, tms );
  SetTimer( FNotifyWnd, 1, FPollingDelay, nil );
end;

procedure TCommPortDriver.Disconnect;
begin
  if Connected then
  begin
    KillTimer( FNotifyWnd, 1 );
    // Release the COM port
    CloseHandle( FHandle );
    // No more connected
    FHandle := INVALID_HANDLE_VALUE;
  end;
end;

function TCommPortDriver.Connected: boolean;
begin
  Result := FHandle <> INVALID_HANDLE_VALUE;
end;


// Повернення CTS, DSR, RING and RLSD (CD)
function TCommPortDriver.GetLineStatus: TLineStatusSet;
var dwS: DWORD;
begin
  Result := [];
  if not Connected then
    exit;
control-register
  if not GetCommModemStatus( FHandle, dwS ) then
    exit;
  if dwS and MS_CTS_ON <> 0 then Result := Result + [lsCTS];
  if dwS and MS_DSR_ON <> 0 then Result := Result + [lsDSR];
  if dwS and MS_RING_ON <> 0 then Result := Result + [lsRING];
  if dwS and MS_RLSD_ON <> 0 then Result := Result + [lsCD];
```

```
end;

function TCommPortDriver.IsPolling: boolean;
begin
  Result := FRXPollingPauses <= 0;
end;

procedure TCommPortDriver.PausePolling;
begin
  inc( FRXPollingPauses );
end;

procedure TCommPortDriver.ContinuePolling;
begin
  dec( FRXPollingPauses );
end;

function TCommPortDriver.FlushBuffers( inBuf, outBuf: boolean ): boolean;
var dwAction: DWORD;
begin
  Result := false;
  if not Connected then
    exit;
  dwAction := 0;
  if outBuf then
    dwAction := dwAction or PURGE_TXABORT or PURGE_TXCLEAR;
  if inBuf then
    dwAction := dwAction or PURGE_RXABORT or PURGE_RXCLEAR;
  Result := PurgeComm( FHandle, dwAction );
  if Result then
    FFirstByteOfPacketTime := DWORD(-1);
end;

function TCommPortDriver.CountRX: integer;
var stat: TCOMSTAT;
    errs: DWORD;
begin
  Result := 65535;
  if not Connected then
    exit;
  ClearCommError( FHandle, errs, @stat );
  Result := stat.cbInQue;
end;

function TCommPortDriver.OutFreeSpace: word;
var stat: TCOMSTAT;
    errs: DWORD;
begin
  if not Connected then
    Result := 65535
  else
  begin
    ClearCommError( FHandle, errs, @stat );
    Result := FOutBufSize - stat.cbOutQue;
  end;
end;
//Відправленнч даних з додатковими налаштуваннями
function TCommPortDriver.SendDataEx( DataPtr: pchar; DataSize, Timeout: DWORD ):
DWORD;
var nToSend, nSent, t1: DWORD;
begin
  Result := 0;
  if not Connected then
    exit;
  t1 := GetTickCount;
  while DataSize > 0 do
  begin
    nToSend := OutFreeSpace;
    if nToSend > 0 then
```

```
    begin
      // Перевірка сигналу
      if FCkLineStatus and (GetLineStatus = []) then
        exit;
      if nToSend > DataSize then
        nToSend := DataSize;
      // Відправлення
      WriteFile( FHandle, DataPtr^, nToSend, nSent, nil );
      nSent := abs( nSent );
      if nSent > 0 then
      begin
        Result := Result + nSent;
        DataSize := DataSize - nSent;
        DataPtr := DataPtr + nSent;
        // Отримання поточного часу
        t1 := GetTickCount;
        continue;
      end;
    end;
    if DWORD(GetTickCount-t1) > Timeout then
      exit;
  end;
end;

// Відправлення даних
function TCommPortDriver.SendData( DataPtr: pointer; DataSize: DWORD ): DWORD;
begin
  Result := SendDataEx( DataPtr, DataSize, FOutputTimeout );
end;

function TCommPortDriver.SendByte( Value: byte ): boolean;
begin
  Result := SendData( @Value, 1 ) = 1;
end;

function TCommPortDriver.SendChar( Value: char ): boolean;
begin
  Result := SendData( @Value, 1 ) = 1;
end;

function TCommPortDriver.SendString( s: string ): boolean;
var len: DWORD;
begin
  len := length( s );
  {$IFOPT H+}
  Result := SendData( pchar(s), len ) = len;
  {$ELSE}
  Result := SendData( pchar(@s[1]), len ) = len;
  {$ENDIF}
end;

function TCommPortDriver.SendZString( s: pchar ): boolean;
var len: DWORD;
begin
  len := strlen( s );
  Result := SendData( s, len ) = len;
end;

function TCommPortDriver.ReadData( DataPtr: pchar; MaxDataSize: DWORD ): DWORD;
var nToRead, nRead, t1: DWORD;
begin
  Result := 0;
  if not Connected then
    exit;
  PausePolling;
  t1 := GetTickCount;
  while MaxDataSize > 0 do
  begin
    nToRead := CountRX;
```

```
    if nToRead > 0 then
    begin
      if nToRead > MaxDataSize then
        nToRead := MaxDataSize;
      ReadFile( FHandle, DataPtr^, nToRead, nRead, nil );
      Result := Result + nRead;
      MaxDataSize := MaxDataSize - nRead;
      DataPtr := DataPtr + nRead;
      t1 := GetTickCount;
      continue;
    end;
    if (GetTickCount-t1) > FInputTimeout then
      break;
  end;
  ContinuePolling;
end;

function TCommPortDriver.ReadByte( var Value: byte ): boolean;
begin
  Result := ReadData( @Value, 1 ) = 1;
end;

function TCommPortDriver.ReadChar( var Value: char ): boolean;
begin
  Result := ReadData( @Value, 1 ) = 1;
end;

procedure TCommPortDriver.ToggleDTR( onOff: boolean );
const funcs: array[boolean] of integer = (CLRDTR,SETDTR);
begin
  if Connected then
    EscapeCommFunction( FHandle, funcs[onOff] );
end;

procedure TCommPortDriver.ToggleRTS( onOff: boolean );
const funcs: array[boolean] of integer = (CLRRTS,SETRTS);
begin
  if Connected then
    EscapeCommFunction( FHandle, funcs[onOff] );
end;

procedure TCommPortDriver.TimerWndProc( var msg: TMessage );
var nRead, nToRead, dummy: DWORD;
    comStat: TCOMSTAT;
begin
  if (msg.Msg = WM_TIMER) and Connected then
  begin
    if FRXPollingPauses > 0 then
      exit;
    ClearCommError( FHandle, dummy, @comStat );
    if FPacketSize > 0 then
    begin
      if DWORD(comStat.cbInQue) >= DWORD(FPacketSize) then
      begin
        repeat
          nRead := 0;
          if ReadFile( FHandle, FTempInBuffer^, FPacketSize, nRead, nil ) then
            if (nRead <> 0) and Assigned(FOnReceivePacket) then
              FOnReceivePacket( Self, FTempInBuffer, nRead );
          if comStat.cbInQue >= FPacketSize then
          FFirstByteOfPacketTime := FFirstByteOfPacketTime +
          DelayForRX( FBaudRate, FPacketSize );
          comStat.cbInQue := comStat.cbInQue - WORD(FPacketSize);
          if comStat.cbInQue = 0 then
            FFirstByteOfPacketTime := DWORD(-1);
        until DWORD(comStat.cbInQue) < DWORD(FPacketSize);
        // Done
        exit;
      end;
```

```
      if (FPacketTimeout > 0) and (FFirstByteOfPacketTime <> DWORD(-1)) and
         (GetTickCount - FFirstByteOfPacketTime > DWORD(FPacketTimeout)) then
      begin
        nRead := 0;
        if ReadFile( FHandle, FTempInBuffer^, comStat.cbInQue, nRead, nil ) then
          if (FPacketMode <> pmDiscard) and (nRead <> 0) and
Assigned(FOnReceivePacket) then
            FOnReceivePacket( Self, FTempInBuffer, nRead );
        FFirstByteOfPacketTime := DWORD(-1);
        // Done
        exit;
      end;
      // Start time
      if (comStat.cbInQue > 0) and (FFirstByteOfPacketTime = DWORD(-1)) then
        FFirstByteOfPacketTime := GetTickCount;
      // Done
      exit;
    end;
    nRead := 0;
    nToRead := comStat.cbInQue;
    if (nToRead > 0) and ReadFile( FHandle, FTempInBuffer^, nToRead, nRead, nil
) then
      if (nRead <> 0) and Assigned(FOnReceiveData) then
        FOnReceiveData( Self, FTempInBuffer, nRead );
  end
  else
    Msg.Result := DefWindowProc( FNotifyWnd, Msg.Msg, Msg.wParam, Msg.lParam ) ;
end;

end.
```

ФАЙЛ ABOUT.PAS – ФОРМА АВТОРСЬКОГО ПРАВА ПЗ

```
unit ABOUT; // назва модулю

interface // розділ оголошення

uses Windows, Classes, Graphics, Forms, Controls, StdCtrls,
  Buttons, ExtCtrls, jpeg;

Type // об'ява власних типів даних
  TAboutBox = class(TForm)
    Panel1: TPanel;
    OKButton: TButton;
    ProgramIcon: TImage;
    ProductName: TLabel;
    Version: TLabel;
    Copyright: TLabel;
    Comments: TLabel;
    procedure OKButtonClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var // об'ява перемінних
  AboutBox: TAboutBox;

Implementation // розділ реалізації

{$R *.dfm} // ресурси форми

procedure TAboutBox.OKButtonClick(Sender: TObject);
begin
  AboutBox.hide;// приховання форми
end;

end.
```

КОД НА МОВІ G. – ПАРАМЕТРИЧНА ФУНКЦІЯ. G300(3-D).

```
rem 2/p60000 p600010 G201T100D11H235.62
 a=getparameter(65)
 b=getparameter(66)
 c=getparameter(67)
 d=getparameter(68)
 f=getparameter(70)
 i=getparameter(73)
 j=getparameter(74)
 k=getparameter(75)
 l=getparameter(76)
 m=getparameter(77)
 o=getparameter(79)
 p=getparameter(80)
 q=getparameter(81)
 r=getparameter(82)
 s=getparameter(83)
 w=getparameter(87)
 x=getparameter(88)
 y=getparameter(89)
 z=getparameter(90)
 g=getdatacadr(66)
 z18=z
 if getsystemdata(1200)<>235.62 then goto 9999
 if g=54 then g56789=400
 if g=55 then g56789=500
 if g=56 then g56789=600
 if g=57 then g56789=700
 if g=58 then g56789=800
 if g=59 then g56789=900
 if paramactive(65)=1 then pamx=getsystemdata(g56789+1)
 if paramactive(66)=1 then pamy=getsystemdata(g56789+2)
 if paramactive(84)=0 then t=_t
 if paramactive(84)=1 then t=getparameter(84)
 hh=t+100
 f2=2*f
 num=1
 kord=getsystemdata(t+1000)
 pamd=kord
 open "c:\lib\program\9" for output as #1
 gosub 60000
 if s=90 then gosub 60010
 if paramactive(77)=1 then goto 6022
 if s<100 then goto 6020
 formula$=getstring
 write #1 formula$
 write #1 "M99"
 close #1
 if s=100 then goto 6100
 if s=200 then goto 7100
 N6020a$=getstring
 if compare(a$,"endcont")=0 then goto 6040
 write #1 a$
 if s=90 then gosub 60100
 num=num+1
 goto 6020
 N6022m$=concat("N",u$)
 m=m+100
 gosub 60000
 e$=concat("N",u$)
 goto 6027
 N6025a$=getstring
 if length(a$)=0 then goto 6025
 if compare(a$,"endcont")=0 then goto 6040
 if compare(a$,e$)=0 then goto 6040
```

```
N6027if compare(a$,m$)<>0 then goto 6025
N6030write #1 a$
a$=getstring
if compare(a$,"endcont")=0 then goto 6040
if s=90 then gosub 60100
num=num+1
if compare(a$,e$)<>0 then goto 6030
N6040write #1 "M99"
close #1
if s=0 then goto 6108
if s=1 then goto 6100
if s=2 then goto 7100
if s=11 then goto 6110
if s=12 then goto 6120
if s=90 then goto 6200
if s=200 then goto 7100
rem S1 DEC Tdp Xia Yjb Zkc
N6100n=1
if paramactive(65)=1 then n=abs(int(i/a))
if paramactive(66)=1 then n=abs(int(j/b))
if paramactive(67)=1 then n=abs(int(k/c))
if paramactive(80)=1 then n=abs(int(d/p))
if paramactive(65)=0 then q=abs(int(i/n))
if paramactive(66)=0 then b=abs(int(j/n))
if paramactive(67)=0 then c=abs(int(k/n))
if paramactive(80)=0 then p=(d/n)
c1=c
G25Ln
G9G90G40G1F2000GgXxYy
if paramactive(80)=1 then setsystemdata((t+1000),kord)
if paramactive(80)=1 then goto 6101
if paramactive(68)=1 then setsystemdata((t+1000),kord)
N6101G9G43HhhZ(z18+w)
F1000Zz18
G91Z-cFf2
G90G1Ff
P9
G200TtGgW(w+(abs(-z))+(abs(-k)))
if paramactive(65)=0 then goto 6102
G202GgXa
N6102if paramactive(66)=0 then goto 6104
G202GgYb
N6104if paramactive(80)=1 then kord=kord+p
if paramactive(80)=1 then goto 6106
if paramactive(68)=1 then kord=kord+p
N6106c=c1+c
M25
goto 8888
rem S0 step=0
N6108G9G90G40G1F2000GgXxYy
G9G43HhhZ(z18+w)
F1000Zz18
G91ZkFf2
G90G1Ff
P9
G200TtGgW(w+(abs(-z))+(abs(-k)))
goto 8888
rem S11 R0 L1
N6110
if paramactive(76)=0 then goto 6112
n=abs(1+(int(k/c)))
kord=kord-r
kord1=kord
c1=c
G25Ln
G9G90G40G1F2000GgXxYy
setsystemdata((t+1000),kord)
G9G43HhhZ(z18+w)
F1000Zz18
```

```
G91Z-cFf2
G90G1Ff
P9
G200TtGgW(w+(abs(-z))+(abs(-k)))
kord=kord1+(sqrt((r^2)-((r-c)^2)))
c=c1+c
M25
goto 8888
rem S11 R1 L0
N6112
n=abs(1+(int(k/c)))
kord=kord-r
kord1=kord
c1=c
G25Ln
G9G90G40G1F2000GgXxYy
setsystemdata((t+1000),kord)
G9G43HhhZ(z18+w)
F1000Zz18
G91Z-cFf2
G90G1Ff
P9
G200TtGgW(w+(abs(-z))+(abs(-k)))
kord=kord1+(sqrt((r^2)-((r-c)^2)))
c=c1+c
M25
goto 8888
rem S12 R0 L1
N6120
if paramactive(76)=0 then goto 6122
n=abs(1+(int(k/c)))
kord=kord-r
kord1=kord
c1=c
G25Ln
G9G90G40G1F2000GgXxYy
setsystemdata((t+1000),kord)
G9G43HhhZ(z18+w)
F1000Zz18
G91Z-cFf2
G90G1Ff
P9
G200TtGgW(w+(abs(-z))+(abs(-k)))
kord=kord1+(sqrt((r^2)-((r-c)^2)))
c=c1+c
M25
goto 8888
rem S12 R1 L0
N6122
n=abs(1+(int(k/c)))
kord1=kord
c1=c
G25Ln
G9G90G40G1F2000GgXxYy
setsystemdata((t+1000),kord)
G9G43HhhZ(z18+w)
F1000Zz18
G91Z-cFf2
G90G1Ff
P9
G200TtGgW(w+(abs(-z))+(abs(-k)))
kord=kord1+r-(sqrt((r^2)-((c1+c)^2)))
c=c1+c
M25
goto 8888
N60210if pamg4142=41 then goto 60220
write #1 "G90G1G42D_tX",xxx[num-1],"Y",yyy[num-1]
goto 60299
N60220write #1 "G90G1G41D_tX",xxx[num-1],"Y",yyy[num-1]
```

```
goto 60299
N60230write #1 "G90G1G40X",xxx[num-1],"Y",yyy[num-1]
goto 60299
N60240write #1 "G90G1X",xxx[num-1],"Y",yyy[num-1]
goto 60299
N60252if rrr[num]=0 then goto 60270
write #1 "G90G2X",xxx[num-1],"Y",yyy[num-1],"R",rrr[num]
goto 60299
N60263if rrr[num]=0 then goto 60270
write #1 "G90G3X",xxx[num-1],"Y",yyy[num-1],"R",rrr[num]
goto 60299
N60270 msg "ejik v tumane"
M98
N60299num=num-1
M25 write #1 "M99"
close #1
return
```

КОД НА МОВІ G. – ОБРОБКА ТИПОВИХ ЕЛЕМЕНТІВ. G205.

```
a=getparameter(65)
b=getparameter(66)
c=getparameter(67)
d=getparameter(68)
e=getparameter(69)
f=getparameter(70)
g=getparameter(71)
h=getparameter(72)
i=getparameter(73)
j=getparameter(74)
k=getparameter(75)
l=getparameter(76)
m=getparameter(77)
o=getparameter(79)
p=getparameter(80)
q=getparameter(81)
r=getparameter(82)
s=getparameter(83)
u=getparameter(85)
v=getparameter(86)
w=getparameter(87)
x=getparameter(88)
y=getparameter(89)
z=getparameter(90)
xn=getdatacadr(1)
yn=getdatacadr(2)
gn=getdatacadr(66)
xg=getdatacadr(115)
yg=getdatacadr(116)
zg=getdatacadr(117)
z18=z
f15=f*1.5
f20=f*2
if paramactive(84)=0 then t=_t
if paramactive(84)=1 then t=getparameter(84)
if getsystemdata(1200)<>235.62 then goto 9999
hh=t+100
G1G40G90G80
if s=3 then goto 7300
if s=4 then goto 7400
if s=6 then goto 7600
if s=7 then goto 7700
if paramactive(88)=0 then goto 10
if paramactive(89)=0 then goto 10
F2000G9XxYy
N10if s=1 then goto 7100
if s=2 then goto 7200
if s=5 thengoto 7500
if s=8 thengoto 7800
```

КОД НА МОВІ G. – ПАРАМЕТРИЧНА ФУНКЦІЯ. G300(3-D).

```
rem S1 RA N7100z=0
 u=u/2
 a30=u-0.1
 a29=u+a30
 a33=a29/2
 G1F2000G43HhhZ(z18+w)
 F1000Zz18
 if m=1 then goto 7130
 if m=21 then goto 7130
 if m=2 then goto 7120
 if m=22 then goto 7120
 G1G41DtG91X0Y-uFf
 N7102G17G10G3I0JuK-cZk
 NM1
 if m=3 then goto 7104
 G3X0Yu*2I0Ju
 Ff15G3X0Y-u*2I0J-u
 Ff20G3X0Ya29I0Ja33
 Ff20G1G40Y-a30
 goto 9999
 N7104Ff20G1G40G91Yu
 goto 9999
 N7120G1G42DtG91X0Y-uFf
 N7122G17G10G2I0JuK-cZk
 NM1
 if m=2 then goto 7124
 G2X0Yu*2I0Ju
 Ff15G2X0Y-u*2I0J-u
 Ff20G2X0Ya29I0Ja33
 Ff20G1G40Y-a30
 goto 9999
 N7124Ff20G1G40G91Yu
 goto 9999
 rem m1/m21
 N7130n=c
 if a=0 then goto 7131
 G202XxYy
 G37A(a+_a)
 N7131FfG1G40G91X(i/2)Z-(c/2)
 N7132G40X-iZ-c
 n=n+c
 a50=-1
 if n>=(abs(-k)) then goto 7134
 G40XiZ-c
 n=n+c
 a50=1
 if n<(abs(-k)) then goto 7132
 N7134if a50=-1 then goto 7136
 G40X-(i/2)Z-(c/2)
 NM1
 if m=21 then goto 7140
 goto 9999
 N7136G40X(i/2)Z-(c/2)
 NM1
 if m=21 then goto 7142
 goto 9999
 N7140X-(i/2)
 Xi
 goto 9999
 N7142X(i/2)
 X-I
 goto 9999
```

G-КОДИ (G-CODE), ОПИС:

G00 - швидкий хід.
G01 - лінійна інтерполяція.
G02 - кругова інтерполяція за годинниковою стрілкою.
G03 - кругова інтерполяція проти годинникової стрілки.
G04 - пауза.
G06 - параболічна інтерполяція.
G08 - розгін.
G09 - гальмування.
G10-G16 - не визначені.
Площина інтерполяції визначається G-функціями:
G17 - площина XY.
G18 - площина XZ.
G19 - площину YZ.
G40/G41/G42/G43/G44 - корекція на радіус інструменту.
G40 - скасування компенсації на радіус інструменту.
G41 - компенсація зліва.
G42 - компенсація праворуч.
G43 - компенсація позитивна.
G44 - компенсація негативна.
G52 - локальне зсув робочої системи координат.
G53 - скасування заданого зсуву.
G54-G59 - заданий зсув.
G61 - режим точного зупину.
G64 - режим різання (cutting mode).
G80 - відміна постійного циклу.
G81 - цикл багатопрохідного свердління отвору на задану глибину.
G90 - абсолютний розмір.
G91 - розмір в прирости.
G92 - установка абсолютних накопичувачів положення.
G93 - швидкість подачі в функції, зворотної часу.
G94/G95 - режим хвилинної та оборотної подач.
G96 - постійна швидкість різання.
G97 - обороти в хвилину.
G98/G99 - вибір точки повернення з жорсткого (постійного) циклу.
G70 - обробка отворів, з центрами розташованими на одній окружності.
G70.1 - скасування обробки центрального отвору.
G71 - обробка отворів на дузі.
G72 - обробка ряду отворів лежать на похилій лінії (прямий).