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 диспетчерського керування та збору даних технологічного процесу”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-22М-2
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Петченко Д.П.
«____» _____ 2023 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Смірнов С.А.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _12 _____*"Інформаційні технології"*_____

Спеціальність 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

Петченку Даниілу Павловичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи диспетчерського керування та збору даних технологічного процесу*

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

затверджені наказом вищого навчального закладу № 35-13 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 диспетчерського керування та збору даних технологічного процесу*

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

Петченко Д.П. Дослідження та програмна реалізація системи диспетчерського керування та збору даних технологічного процесу. 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 диспетчерського керування та збору даних технологічного процесу.

Метою розробки є дослідження та програмна реалізація системи диспетчерського керування та збору даних технологічного процесу.

Об'єктом дослідження є процес диспетчерського керування та збору даних технологічного процесу.

Предметом дослідження є методи диспетчерського керування та збору даних технологічного процесу.

Методи дослідження базуються на методах Інтернету речей, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи диспетчерського керування та збору даних технологічного процесу.

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 Builder C++.

Ключові слова: комп'ютерна інженерія, диспетчерське керування та збір даних, технологічний процес

ABSTRACT

Petchenko D.P. Research and software implementation of the system of dispatch control and data collection of the technological process. 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 dispatch control and data collection of the technological process.

The purpose of the development is the research and software implementation of the system of dispatch control and data collection of the technological process.

The object of the study is the process of dispatch control and data collection of the technological process.

The subject of research is methods of dispatch control and data collection of the technological process.

Research methods are based on Internet of Things methods, mathematical statistics methods, and software development methods.

The result of the work is the software implementation of the system of dispatch control and data collection of the technological process.

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 Builder C++ environment.

Keywords: computer engineering, dispatch control and data collection, technological process

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

АСУ	–	автоматизована система управління
АЦП	–	аналогово-цифровий перетворювач
КПК	–	кишеньковий персональний комп’ютер
ПЗ	–	програмне забезпечення
ПЗО	–	пристрій зв'язку з об'єктом
ПК	–	персональний комп’ютер
СУБД	–	системи управління базами даних
ТП	–	технологічний процес
HMI	–	людино-машиний інтерфейс
SCADA	–	диспетчерське управління й збір даних
WAP	–	Wireless Application Protocol

ВСТУП

Актуальність теми. Диспетчерське керування й збір даних (SCADA Supervisory Control And Data Acquisition) є основним і в цей час залишається найбільш перспективним методом автоматизованого керування складними динамічними системами (процесами) у життєво важливі й критичні з погляду безпеки й надійності областях. Саме на принципах диспетчерського керування будуються великі автоматизовані системи в промисловості й енергетиці, на транспорті, у космічній і військовій областях, у різних державних структурах.

За останні 10-15 років за рубежем різко зріс інтерес до проблем побудови високоефективних і високонадійних систем диспетчерського керування й збору даних. З одного боку, це зв'язано зі значним прогресом в області обчислювальної техніки, програмного забезпечення й телекомунікацій, що збільшує можливості й розширює сферу застосування автоматизованих систем. З іншого боку, розвиток інформаційних технологій, підвищення ступеня автоматизації й перерозподіл функцій між людиною й апаратурою загострило проблему взаємодії людини-оператора із системою керування.

Розслідування й аналіз більшості аварій і подій в авіації, наземному й водному транспорті, промисловості й енергетиці, частина з яких привела до катастрофічних наслідків, показали, що, якщо в 60-х роках помилка людини була первісною причиною лише 20% інцидентів (80%, відповідно, за технологічними несправностями й відмовами), те в 90-х роках частка людського фактора зросла до 80%, причому, у зв'язку з постійним удосконалюванням технологій і підвищенням надійності електронного встаткування й машин, частка ця може ще зрости [1-3].

Основною причиною таких тенденцій є старий традиційний підхід до побудови складних автоматизованих систем керування, що застосовується часто й у цей час: орієнтація в першу чергу на застосування новітніх технічних (технологічних) досягнень, прагнення підвищити ступінь автоматизації й

функціональні можливості системи й, у той же час, недооцінка необхідності побудови ефективного людино-машинного інтерфейсу (HMI Human-Machine Interface), тобто інтерфейсу, орієнтованого на користувача (оператора). Не випадково саме на останні 15 років, тобто період появи потужних, компактних і недорогих обчислювальних засобів, довівся пік досліджень у США із проблем людського фактора в системах керування, у тому числі по оптимізації архітектури й HMI-інтерфейсу систем диспетчерського керування й збору даних.

Вивчення матеріалів із проблем побудови ефективних і надійних систем диспетчерського керування показало необхідність застосування нового підходу при розробці таких систем: human-centered design (або top-down, долілиць), тобто орієнтація в першу чергу на людину-оператора (диспетчера) і його завдання, замість традиційного й повсюдно застосовуваного hardware-centered (або bottom-up, нагору), у якому при побудові системи основна увага приділялася вибору й розробці технічних засобів (устаткування й програмного забезпечення). Застосування нового підходу в реальних космічних і авіаційних розробках і порівняльні випробування систем у Національному керуванні по аеронавтиці й дослідженню космічного простору (NASA), США, підтвердили його ефективність, дозволивши збільшити продуктивність операторів, на порядок зменшити процедурні помилки й звести до нуля критичні (некоректуємі) помилки операторів [4, 5].

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи диспетчерського керування та збору даних технологічного процесу.

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

– Огляд існуючих систем диспетчерського керування та збору даних технологічного процесу.

– Дослідження системи диспетчерського керування та збору даних технологічного процесу.

– Програмна реалізація системи диспетчерського керування та збору даних технологічного процесу.

Об'єктом дослідження є процес диспетчерського керування та збору даних технологічного процесу.

Предметом дослідження є методи диспетчерського керування та збору даних технологічного процесу.

Методи дослідження базуються на методах Інтернету речей, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод диспетчерського керування та збору даних технологічного процесу.

– Розроблено вітчизняний продукт диспетчерського керування та збору даних технологічного процесу, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі диспетчерського керування та збору даних технологічного процесу.

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 диспетчерського керування та збору даних технологічного процесу, 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

SCADA процес збору інформації реального часу з віддалених точок (об'єктів) для обробки, аналізу й можливого керування віддаленими об'єктами. Вимога обробки реального часу обумовлено необхідністю доставки (видачі) всіх необхідних подій (повідомлень) і даних на центральний інтерфейс оператора (диспетчера). У той же час поняття реального часу відрізняється для різних SCADA-систем.

Прообразом сучасних систем SCADA на ранніх стадіях розвитку автоматизованих систем керування були системи телеметрії й сигналізації.

Всі сучасні SCADA-системи включають три основних структурних компоненти:

– Remote Terminal Unit (RTU) – віддалений термінал, що здійснює обробку завдання (керування) у режимі реального часу. Спектр його втілень широкий від примітивних датчиків, що здійснюють знімання інформації з об'єкта, до спеціалізованих багатопроцесорних відказостійких обчислювальних комплексів, що здійснюють обробку інформації й керування в режимі твердого реального часу. Конкретна його реалізація визначається конкретним застосуванням. Використання пристроїв низькорівневої обробки інформації дозволяє знизити вимоги до пропускної здатності каналів зв'язку із центральним диспетчерським пунктом.

– Master Terminal Unit (MTU), Master Station (MS) – диспетчерський пункт керування (головний термінал); здійснює обробку даних і керування високого рівня, як правило, у режимі м'якого (квазі-) реального часу; одна з основних функцій забезпечення інтерфейсу між людиною-оператором і системою (HMI, MMI). Залежно від конкретної системи MTU може бути реалізований у

найрізноманітнішому виді від одиночного комп'ютера з додатковими пристроями підключення до каналів зв'язку до більших обчислювальних систем (мейнфреймів) і/або об'єднаних у локальну мережу робочих станцій і серверів. Як правило, і при побудові MTU використовуються різні методи підвищення надійності й безпеки роботи системи.

– Communication System (CS) – комунікаційна система (канали зв'язку), необхідна для передачі даних з віддалених точок (об'єктів, терміналів) на центральний інтерфейс оператора-диспетчера й передачі сигналів керування на RTU (або віддалений об'єкт залежно від конкретного виконання системи).

1.2 Область застосування

Основними областями застосування систем диспетчерського керування є:

– керування передачею й розподілом електроенергії;

– промислове виробництво;

– виробництво електроенергії;

– водозабір, водоочищення й водорозподіл;

– видобуток, транспортування й розподіл нафти й газу;

– керування космічними об'єктами;

– керування на транспорті (всі види транспорту: авіа, метро, залізничний, автомобільний, водний);

– телекомунікації;

– військова область.

У цей час у розвинених закордонних країнах спостерігається справжній підйом по впровадженню нових і модернізації існуючих автоматизованих систем керування в різних галузях економіки; у переважній більшості випадків ці системи будуються за принципом диспетчерського керування й збору даних. Характерно, що в індустріальній сфері (в обробній і добувній промисловості, енергетику й ін.) найбільше часто згадуються саме модернізація існуючих

виробництв SCADA-системами нового покоління. Ефект від впровадження нової системи керування обчислюється, залежно від типу підприємства, від сотень тисяч до мільйонів доларів у рік; наприклад, для однієї середньої теплової станції він становить, по підрахунках фахівців, від 200000 до 400000 доларів. Велика увага приділяється модернізації виробництв, що представляють собою екологічну небезпеку для навколишнього середовища (хімічні і ядерні підприємства), а також граючу ключову роль у життєзабезпеченні населених пунктів (водопровід, каналізація та ін.). З початку 90-х років у США почалися інтенсивні дослідження й розробки в області створення автоматизованих систем керування наземним (автомобільним) транспортом ATMS (Advanced Traffic Management System).

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи диспетчерського керування та збору даних технологічного процесу, 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

CitectSCADA

CitectSCADA – програмний продукт, що представляє собою систему моніторингу, керування й збору даних (SCADA – Supervisory, Control And Data Acquisition). Дана система призначена для керування технологічними процесами. RTSoft є офіційним дистриб'ютором програмних продуктів Citect.

CitectSCADA – це вибір багатьох найвідоміших у світі компаній яким потрібно масштабоване, гнучке й надійне рішення.

SCADA-система CitectSCADA проектувалася й розроблялася як засіб реалізації всіх вимог підприємства у вигляді єдиної інтегрованої системи. CitectSCADA містить всі необхідні компоненти, що усувають як необхідність використання додаткового програмного забезпечення, так і фрагментацію даних. Перший же пакет Citect для Windows, установлений в 1992 році, підняв планку для SCADA-систем на базі ПК на новий рівень продуктивності – 33000 точок дискретного введення, 16000 точок аналогового введення, 4000 трендів, 50 операторських станцій, резервування в стилі розподілених DCS-систем, загальна база даних, конфігурування з будь-якого комп'ютера. З такими характеристиками клієнти Citect завжди можуть бути впевнені, що Citect упорається з будь-яким завданням незалежно від розмірів системи.

Подальші розробки Citect будувалися на цьому потужному фундаменті. Розроблювачі пакета прагнуть дати користувачеві реальні можливості, щоб на основі останніх досягнень ще більше розширити функціональні характеристики продукту, спростити його застосування, підвищити продуктивність і скоротити

строки розробки прикладних систем. Високий ступінь автоматизації операцій, функціональна повнота, контроль якості й тестування гарантують максимально можливу надійність Citect, що завдяки цьому може використовуватися в різних відповідальних додатках атомної енергетики, авіації й інших областей.

CitectSCADA – програмний продукт, що представляє собою повнофункціональну систему моніторингу, керування й збору даних, що дозволяє забезпечити:

– Візуалізацію процесу в графічному режимі.

– Керування алармами.

– Відстеження трендів у реальному часі й доступ до архівних трендів.

– Підготовку деталізованих звітів.

– Статичний контроль процесу.

– Багатопотокове виконання підпрограм розроблених на CitectVBA і CiCode.

SCADA-система "КАСКАД"

SCADA-система "КАСКАД" призначена для побудови автоматизованих систем керування технологічними процесами – АСУТП, автоматизованих систем контролю й обліку енергетики – АСКОЕ, автоматизованих систем оперативно-диспетчерського керування – АСОДК, і інших систем промислової автоматизації.

SCADA-система "КАСКАД" може з успіхом застосовуватися в хімічній, нафтохімічній, газовій, металургійній, електроенергетичній промисловостях і ЖКГ.

Функції:

– Збір і реєстрація первинної інформації про хід технологічного процесу.

– Обробка інформації з алгоритмів користувача.

– Надання інформації у вигляді мнемосхем технологічного процесу.

– Оперативне, диспетчерське керування.

– Ведення історії технологічного процесу.

– Перегляд і аналіз ходу технологічного процесу.

– Формування звітної документації.

– Експорт оперативної й історичної інформації в WEB.

– Сигналізація й реєстрація подій і порушень у ході технологічного процесу.

– Реєстрація всіх дій операторів.

– Механізм настроювання прав користувачів, рівні доступу.

Основні характеристики:

– Функціонування на платформі Windows 10/11.

– Підтримка стандартів OPC DA 2.0/3.0.

– Підтримка стандарту OPC HDA.

– Резервування робочих станцій, серверів, баз даних.

– Розподілена архітектура клієнт-сервер.

– Використання SQL-сервера Firebird 2.0 для керування базами даних.

– Ведення вторинних баз даних з можливістю використання будь-якої СУБД – MS SQL Server, Oracle, MySQL і т.д.

– Модуль візуалізації з великим вибором різноманітних об'єктів для відображення оперативної й історичної інформації й керування.

– Розширювана архітектура модуля візуалізації за принципом "плагинів".

– Потужний і швидкодіючий механізм зберігання історії.

– Самодостатній модуль формування звітної документації з можливістю експорту у формати xls, html, txt і т.д, автоматичного формування звітів за розкладом, ініціативі користувача або події з можливістю відправлення звітів по електронній пошті.

– Система обробки аварійних ситуацій, що контролює технологічний процес по заданих алгоритмах і яка інформує користувача за допомогою текстової, звукової сигналізації, а також за допомогою sms і електронної пошти.

– Убудована підтримка розповсюджених типів вітчизняної й закордонної контролерної техніки.

– Можливість підключення специфічних пристроїв.

– Відкриті інтерфейси для розширення функціональності.

– Інтеграція з SoftLogic-системою «KLogic», і, як наслідок, перехід до наскрізної технології програмування алгоритмів контролерів і робочих станцій:.

– Підтримка протоколів передачі даних МЕК 870-5-101/104.

– Підсистема синхронізації часу всіх серверів і робочих станцій.

SCADA-система Zenon

Zenon – легкий у використанні й у теж час потужний програмний пакет для створення систем автоматизації, що випускається світовим піонером HMI/ SCADA-Рішень, компанією COPA-DATA. Він використовується безліччю компаній по усьому світі в сфері візуалізації процесів, машинних операцій і керування виробництвом. Zenon пропонує просте об’єктно-орієнтовне проектування, повну сумісність і об'єднання в єдину систему пристроїв починаючи від одиничних терміналів і закінчуючи диспетчерськими пунктами керування, рівень безпеки відповідним міжнародним стандартам.

Рисунок 2.1 – Інтерфейс користувача Zenon

Його відкритість дозволяє реалізувати швидке й ефективне з'єднання з будь-якими апаратними й програмними засобами (напр. ERP програми). Ідеально працює на промислових ПК і пристроях з Windows 10/11. До послуг розроблювача найсучасніші програмні інтерфейси, такі як VSTA і VBA. Компанії в багатьох різних галузях – таких як машинобудування, автомобілебудування, фармацевтика, напої й продукти харчування, хімічне виробництво, поставка енергії, автоматизація будинків – одержують вигоду від використання HMI/SCADA Zenon.

Технологія програмування близька до інтуїтивного сприйняття автоматизуємого процесу. Плюс потужне об’єктно-орієнтоване програмування, використовуване в більшості цих пакетів, робить ці продукти легкими в освоєнні й доступним для широкого кола користувачів.

Всі системи можна вважати відкритими, що забезпечують можливість доповнення функціями власної розробки, що мають відкритий протокол для розробки власних драйверів, розвинену мережну підтримку, можливість включення ActiveX-об'єктів і доступність до стандартних баз даних.

Важливою особливістю всіх SCADA-систем є кількість підтримуваних різноманітних ПЛК. Системи InTouch, Factory Link, GENESIS, RealFlex підтримують десятки й сотні драйверів, що робить їхніми безумовними лідерами по цьому показнику.

Побудова прикладної системи на основі кожної з розглянутих SCADA-систем різко скорочує набір необхідних знань в області класичного програмування, дозволяючи концентрувати зусилля по освоєнню знань у самій прикладній області.

У розроблювачів SCADA-систем на платформі Windows Server 2022 / Windows 10/11 з'явилася можливість використовувати розширення реального часу (RTX), щоб перебороти недоліки Windows Server 2022 / Windows 10/11 у завданнях реального часу.

Слід зазначити тенденції включення SCADA-систем у системи комплексної автоматизації підприємства. Це забезпечує точну, своєчасну інформацію на кожному рівні виробництва.

Застосування в SCADA-системах нових технологій, розробка інструментальних засобів комплексної автоматизації підприємства свідчать про прагнення й можливість фірм-розроблювачів постійно вдосконалювати свої продукти, що є немаловажним чинником при виборі інструментального засобу, навіть якщо не всі його технологічні рішення найближчим часом будуть використані Вами.

Як тільки загальне поле діяльності провідних компаній-виробників описуваних інструментальних систем сьогодні концентрується в області MS Windows Server 2022 / Windows 10/11, як тільки загальні технічні можливості систем досить близькі, те головний упор робиться на якість технічної підтримки, на якість навчання користувачів, на концентрацію і якість додаткових комплексних послуг з освоєння й впровадження кінцевої системи керування. Іншими словами на скорочення витрат системних інтеграторів і кінцевих користувачів на інжиніринг і менеджмент своїх проектів, на зменшення вартості супроводу кінцевої системи. Саме ці показники сьогодні, в основному, впливають на рейтинг і ринковий успіх тієї або іншої SCADA-системи. Мабуть, ці показники навіть більше важливі, ніж абсолютні вартісні характеристики SCADA.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Оскільки потрібно розробити просту та легку у користуванні програму, яка б виконувалась під операційною системою Windows, то для її реалізації я обрав Builder C++. Існує велике число бібліотек написаних під Builder C++ , тому це одна з важливих причин вибору мови програмування. Середовище Builder C++ досить просте в користуванні, його вихідний код значно менше по об’єму в

порівнянні з Delphi чи деякими іншими програмами такого типу. Досить легко організувати взаємодію між модулями програм, об’єктно-орієнтований підхід дає можливість значно скоротити код програми, а отже і час його виконання.

На заміну старого розробленого набору елементів управління у Builder C++ інтегрована бібліотека візуальних компонентів VCL, представлених на палітрі компонентів. Після переносу на форму методом перетягування (drag-and-drop) компоненти відразу становляться діючими об’єктами вашої програми. Окрім типізованих інтерфейсних елементів Windows (кнопки, смуги прокручування, редагуємі текстові області, прості та комбіновані списки, та інше) у бібліотеку включені елементи підтримки діалогових вікон, обслуговування баз даних та багато іншого. Можливо не тільки модифікувати поведінку існуючих компонентів, але і будувати нові.

Builder C++ підтримує останні розширення стандарту мови C++ та забезпечує швидку компіляцію та складання 32-розрядних програм для Windows. Результуючі програми оптимізовані з точки зору швидкості виконання програм та затрат пам’яті. Зручний відладгоджувальник (з асемблерним вікном, можливістю крокового виконання, завдання точок зупинки, трасування та інше) повністю інтегрований у систему проектування. Дизайнер форм, редактор коду, інспектор об’єктів та інші інструменти зостаються доступними під час виконання програми, саме через це вносити зміни до коду можна прямо у процесі відлагодження.

Дизайнер форм, Інспектор об'єктів і інші засоби залишаються доступними під час роботи програми, тому вносити зміни можна в процесі відлагодження.

Builder C++ поставляється в трьох варіантах: Standard (стандартний), Professional (для професіоналів розробників, орієнтованих на мережеву архітектуру) і Client/Server Suite (для розробки систем в архітектурі клієнт/сервер). Останні два варіанти доповнюють стандартний початковими текстами візуальних компонентів, різномасштабним словником даних, новими функціями мови запитів SQL для бази даних, пакетом підтримки систем Internet, службою моніторингу програм, а також рядом інших засобів.

Builder C++ підтримує зв'язок з різними базами даних 3-х видів: dBASE і Paradox: Sybase, Oracle, InterBase і Informix; Excel, Access, FoxPro і Btrieve. Механізм BDE (Borland Database Engine) додає обслуговуванню зв'язків з базами даних дивовижну простоту і прозорість. Провідник Database Explorer дозволяє зображати зв'язки і об'єкти баз даних графічно. Використовуючи компоненти баз даних, я побудував електронний записник згідно таблиці dBASE за півгодини роботи на комп'ютері. Спадкоємство готових форм і їх "підгонка" під специфічні вимоги помітно скорочують тимчасові витрати на вирішення подібних завдань.

Довідкова служба Builder C++ надавала мені допомогу в цій і багатьох інших подібних ситуаціях. Є повний опис кожного управляємого компонента, включаючи списки властивостей і методів, а також численні приклади. Виклад матеріалу в книзі був значно покращуваний і систематизований завдяки відомостям, почерпнутим мною з довідкової служби.

Завдяки засобам управління проектами, двосторонній інтеграції додатку і синхронізації між засобами візуального і текстового редагування, а також вбудованому відладнику (з асемблерним вікном прокрутки, покрокового виконання, точок останову, трасуванням і тому подібне) – Builder C++ корпорації Borland надає собою вражаюче середовище розробки, яка, мабуть, витримає конкурентну боротьбу з такими модними продуктами як Developer Studio фірми Microsoft.

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 диспетчерського керування та збору даних технологічного процесу.

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

Визначення й загальна структура SCADA

Існує два типи керування віддаленими об'єктами в SCADA: автоматичне й ініціюєме оператором системи.

Шеридан (Sheridan) виділив чотири основних функціональних компоненти систем диспетчерського керування й збору даних людина-оператор, комп'ютер взаємодії з людиною, комп'ютер взаємодії із завданням (об'єктом), завдання (об'єкт керування), а також визначив п'ять функцій людини-оператора в системі диспетчерського керування й охарактеризував їх як набір вкладених циклів, у яких оператор:

– планує, які наступні дії необхідно виконати;

– навчає (програмує) комп'ютерну систему на наступні дії;

– відслідковує результати автоматичної роботи системи;

– втручається в процес у випадку критичних подій, коли автоматика не може впоратися, або при необхідності підстроювання (регулювання) параметрів процесу;

– навчається в процесі роботи (одержує досвід).

Дане подання SCADA з'явилося основою для розробки сучасних методологій побудови ефективних диспетчерських систем.

Особливості SCADA як процесу керування

Особливості процесу керування в сучасних диспетчерських системах:

– процес SCADA застосовується системах, у яких обов'язкова наявність людини (оператора, диспетчера);

– процес SCADA був розроблений для систем, у яких будь-який неправильний вплив може привести до відмови (втраті) об'єкта керування або навіть катастрофічних наслідків;

– оператор несе, як правило, загальну відповідальність за керування системою, що, при нормальних умовах, тільки зрідка вимагає підстроювання параметрів для досягнення оптимальної продуктивності;

– активна участь оператора в процесі керування відбувається нечасто й у непередбачені моменти часу, звичайно у випадку настання критичних подій (відмови, позаштатні ситуації та ін.);

– дії оператора в критичних ситуаціях можуть бути жорстко обмежені за часом (декількома мінутами або навіть секундами).

Основні вимоги до диспетчерських систем керування

До SCADA-систем пред'являються наступні основні вимоги:

– надійність системи (технологічна й функціональна);

– безпека керування;

– точність обробки й подання даних;

– простота розширення системи.

Вимоги безпеки й надійності керування в SCADA включають наступні:

– ніяка одинична відмова встаткування не повинен викликати видачу помилкового вихідного впливу (команди) на об'єкт керування;

– ніяка одинична помилка оператора не повинна викликати видачу помилкового вихідного впливу (команди) на об'єкт керування;

– всі операції по керуванню повинні бути інтуїтивно-зрозумілими й зручними для оператора (диспетчера).

Тенденції розвитку технічних засобів систем диспетчерського керування

Прогрес в області інформаційних технологій обумовив розвиток всіх 3-х основних структурних компонентів систем диспетчерського керування й збору даних RTU, MTU, CS, що дозволило значно збільшити їхні можливості; так, число контрольованих віддалених точок у сучасної SCADA-системі може досягати 100000.

Основна тенденція розвитку технічних засобів (апаратного й програмного забезпечення) SCADA міграція убік повністю відкритих систем. Відкрита архітектура дозволяє незалежно вибирати різні компоненти системи від різних виробників; у результаті розширення функціональних можливостей, полегшення обслуговування й зниження вартості SCADA-систем.

Віддалені термінали (RTU)

Головна тенденція розвитку віддалених терміналів збільшення швидкості обробки й підвищення їхніх інтелектуальних можливостей. Сучасні термінали будуються на основі мікропроцесорної техніки, працюють під керуванням операційних систем реального часу, при необхідності поєднуються в мережу, безпосередньо або через мережу взаємодіють із інтелектуальними електронними датчиками об'єкта керування й комп'ютерами верхнього рівня.

Конкретна реалізація RTU залежить від області застосування. Це можуть бути спеціалізовані (бортові) комп'ютери, у тому числі мультипроцесорні системи, звичайні мікрокомп'ютери або персональні ЕОМ (PC); для індустріальних і транспортних систем існує два конкуруючі напрямки в техніці: RTU індустріальні (промислові) PC і програмувальні логічні контролери (у українському перекладі часто зустрічається термін промислові контролери) PLC.

Індустріальні комп'ютери являють собою, як правило, програмно сумісні зі звичайними комерційними PC машини, але адаптовані для жорстких умов експлуатації буквально для установки на виробництві, у цехах, газокомпресорних станціях і т.д. Адаптація ставиться не тільки до конструктивного виконання, але й до архітектури й схемотехниці, тому що зміни температури навколишнього середовища приводять до дрейфу електричних параметрів. Як пристрої сполучення з об'єктом керування дані системи комплектуються додатковими платами (адаптерами) розширення, яких на ринку існує велика розмаїтість від різних виготовлювачів (як, втім, і самих постачальників промислових PC). Як операційна система в промислових PC, що працюють у ролі віддалених терміналів, все частіше починає застосовуватися лінійки Windows NT, у тому

числі різні розширення реального часу, спеціально розроблені для цієї операційної системи.

Промислові контролери (PLC) являють собою спеціалізовані обчислювальні пристрої, призначені для керування процесами (об'єктами) у реальному часі. Промислові контролери мають обчислювальне ядро й модулі введення-виводу, що приймають інформацію (сигнали) з датчиків, перемикачів, перетворювачів, інших пристроїв і контролерів, і здійснюючe керування процесом або об'єктом видачею керуючих сигналів на приводи, клапани, перемикачі й інші виконавчі пристрої. Сучасні PLC часто поєднуються в мережу (RS-485, Ethernet, різні типи індустріальних шин), а програмні засоби, розроблювальні для них, дозволяють у зручній для оператора формі програмувати й управляти ними через комп'ютер, що перебуває на верхньому рівні SCADA-системи диспетчерському пункті керування (MTU). Дослідження ринку PLC показало, що найбільш розвиненою архітектурою, програмним забезпеченням і функціональними можливостями володіють контролери фірм Siemens, Fanuc Automation (General Electric), Allen-Bradley (Rockwell), Mitsubishi. Становить інтерес також продукція фірми CONTROL MICROSYSTEMS промислові контролери для систем моніторингу й керування нафто- і газопромиселами, трубопроводами, електричними підстанціями, міським водопостачанням, очищенням стічних вод, контролю забруднення навколишнього середовища.

Багато матеріалів і досліджень по промисловій автоматизації присвячено конкуренції двох напрямків PC і PLC; кожний з авторів приводить велику кількість доводів за й проти по кожному напрямку. Проте, можна виділити основну тенденцію: там, де потрібна підвищена надійність і керування у твердому реальному часі, застосовуються PLC. У першу чергу це стосується застосувань у системах життєзабезпечення (наприклад, водопостачання, електропостачання), транспортних системах, енергетичних і промислових підприємствах, що представляють підвищену екологічну небезпеку. Прикладами

можуть служити застосування PLC сімейства Simatic (Siemens) у керуванні електроживленням монорейкової дороги в Німеччині або застосування контролерів компанії Allen-Bradley (Rockwell) для модернізації застарілої диспетчерської системи аварійної вентиляції й кондиціювання на плутонієвому заводі 4 у Лос-Аламосе. Апаратні засоби PLC дозволяють ефективно будувати відказостійкые системи для критичних додатків на основі багаторазового резервування. Індустріальні PC застосовуються переважно в менш критичних областях (наприклад, в автомобільній промисловості, модернізація виробництва фірмою General Motors), хоча зустрічаються приклади й більше відповідальних застосувань (метро у Варшаві керування рухом поїздів). По оцінках експертів, побудова систем на основі PLC, як правило, є менш дорогим варіантом у порівнянні з індустріальними комп'ютерами.

Канали зв'язку (CS)

Канали зв'язку для сучасних диспетчерських систем відрізняються більшою розмаїтістю; вибір конкретного рішення залежить від архітектури системи, відстані між диспетчерським пунктом (MTU) і RTU, числа контрольованих точок, вимог по пропускній здатності й надійності каналу, наявності доступних комерційних ліній зв'язку.

Тенденцією розвитку CS як структурного компонента SCADA-систем можна вважати використання не тільки великої розмаїтості виділених каналів зв'язку (ISDN, ATM та ін.), але також і корпоративних комп'ютерних мереж і спеціалізованих індустріальних шин.

У сучасних промислових, енергетичних і транспортних системах більшу популярність завоювали індустріальні шини спеціалізовані швидкодіючі канали зв'язку, що дозволяють ефективно вирішувати завдання надійності й завадостійкості з'єднань на різних ієрархічних рівнях автоматизації. Існує три основних категорії індустріальних шин, що характеризують їхнє призначення (місце в системі) і складність переданої інформації: Sensor, Device, Field. Багато індустріальних шин охоплюють дві або навіть всі три категорії.

Із усього різноманіття індустріальних шин, що застосовуються по усьому світі (тільки по Німеччині їх установлено в різних системах близько 70 типів) варто виділити промисловий варіант Ethernet і PROFIBUS, найбільш популярні в цей час і, очевидно, найбільш перспективні. Застосування спеціалізованих протоколів у промисловому Ethernet дозволяє уникнути властивого цій шині недетермінізму (через метод доступу абонентів CSMA/CD), і в той же час використовувати його переваги як відкритого інтерфейсу. Шина PROFIBUS у цей час є однієї з найбільш перспективних для застосування в промислових і транспортних системах керування; вона забезпечує високошвидкісну (до 12 Мбод) завадостійку передачу даних (кодова відстань = 4) на відстань до 90 км. На основі цієї шини побудована, наприклад, система автоматизованого керування рухом поїздів у варшавському метро.

Диспетчерські пункти керування (MTU)

Головною тенденцією розвитку MTU (диспетчерських пунктів керування) є перехід більшості розроблювачів SCADA-систем на архітектуру клієнт-сервер, що складається з 4-х функціональних компонентів.

1. User (Operator) Interface (інтерфейс користувача/оператора) винятково важлива складова систем SCADA. Для неї характерні а) стандартизація інтерфейсу користувача навколо декількох платформ; б) усе більше зростаючий вплив лінійки Windows NT; в) використання стандартного графічного інтерфейсу користувача (GUI); г) технології об’єктно-орієнтованого програмування: DDE, OLE, ActiveX X, OPC (OLE for Process Control), DCOM; д) стандартні засоби розробки додатків, найбільш популярні серед яких, Visual Basic for Applications (VBA), Visual C++; е) поява комерційних варіантів програмного забезпечення класу SCADA/MMI для широкого спектра завдань. Об'єктна незалежність дозволяє інтерфейсу користувача представляти віртуальні об'єкти, створені іншими системами. Результат розширення можливостей по оптимізації HMI-інтерфейсу.

2. Data Management (керування даними) відхід від вузькоспеціалізованих баз даних убік підтримки більшості корпоративних реляційних баз даних (Microsoft SQL, Oracle). Функції керування даними й генерації звітів здійснюються стандартними засобами SQL, 4GL; ця незалежність даних ізолює функції доступу й керування даними від цільових завдань SCADA, що дозволяє легко розробляти додаткові додатки по аналізі й керуванню даними.

3. Networking & Services (мережі й служби) перехід до використання стандартних мережних технологій і протоколів. Служби мережного керування, захисти й керування доступом, моніторингу транзакцій, передачі поштових повідомлень, сканування доступних ресурсів (процесів) можуть виконуватися незалежно від коду цільової програми SCADA, розробленої іншим вендором.

4. Real-Time Services (служби реального часу) звільнення MTU від навантаження перерахованих вище компонентів дає можливість сконцентруватися на вимогах продуктивності для завдань реального й квазі-реального часу. Дані служби являють собою швидкодіючі процесори, які управляють обміном інформацією з RTU і SCADA-процесами, здійснюють керування резидентною частиною бази даних, оповіщення про події, виконують дії по керуванню системою, передачу інформації про події на інтерфейс користувача (оператора).

Операційні системи

Незважаючи на триваючі суперечки серед фахівців із систем керування на тему що краще UNIX або лінійки Windows NT, ринок однозначно зробив вибір на користь останньої. Вирішальними для швидкого росту популярності лінійки Windows NT стала її відкрита архітектура й ефективні засоби розробки додатків, що дозволило численним фірмам-розроблювачам створювати програмні продукти для рішення широкого спектра завдань.

Ріст застосування лінійки Windows NT в автоматизованих системах керування обумовлений у значній мірі появою ряду програмних продуктів, які дозволяють використовувати неї як платформа для створення відповідальних

додатків у системах реального часу, а також у вбудовуються конфигурациях, що. Найбільш відомими розширеннями реального часу для лінійки Windows NT є продукти компаній VenturCom, Nematron, RadiB.

Рішення фірми VenturCom стали стандартом де-факто для створення відповідальних додатків твердого реального часу на платформі лінійки Windows NT. При розробці інтерфейсу для додатків реального часу розроблювачі фірми пішли по шляху модифікації модуля лінійки Windows NT шаруючи апаратних абстракцій (HAL Hardware Abstraction Layer), відповідального за виробіток високопріоритетних системних переривань, що заважають завданню здійснювати керування у твердому реальному часі. Програмний продукт Component Integrator компанії VenturCom є засобом прискореної розробки й впровадження додатків реального часу для лінійки Windows NT; він поставляється у вигляді інтегрованого пакета, що складає з інструментів для створення додатків, що вбудовуються, (ECK Embedded Component Kit) і властиво розширень реального часу (RTX 4.1), що дозволяють додаткам, створюваним для роботи під лінійки Windows NT, працювати а режимі реального часу.

Компанія RadiSys застосувала інший підхід до розробки розширень реального часу: лінійки Windows NT завантажується як низькопріоритетне завдання під добре перевіреною й відомою от уже років 20 операційною системою реального часу iRMX. Всі функції обробки й керування реального часу виконуються як високопріоритетні завдання під iRMX, ізольовані в пам'яті від додатків і драйверів лінійки Windows NT механізмом захисту процесора. Даний підхід має та перевага в порівнянні з рішенням VenturCom, що завдання реального часу не залежить від роботи лінійки Windows NT: у випадку збоїти або катастрофічної системної помилки в роботі лінійки Windows NT керуюче завдання реального часу буде продовжувати працювати. Це рішення дозволяє інформувати основне завдання про проблеми, що виникли в роботі NT, і залишати тільки за нею право продовження роботи або зупинка всієї системи.

Слід зазначити, що в SCADA-системах вимога твердого реального часу (тобто здатність відгуку/обробки подій у чітко певні, гарантовані інтервали часу) ставиться, як правило, тільки до віддалених терміналів; у диспетчерських пунктах керування (MTU) відбувається обробка/керування подіями (процесами, об'єктами) у режимі м'якого (квази-) реального часу.

Прикладне програмне забезпечення

Орієнтація на відкриті архітектури при побудові систем диспетчерського керування й збору даних дозволяє розроблювачам цих систем сконцентруватися безпосередньо на цільовому завданні SCADA збір і обробка даних, моніторинг, аналіз подій, керування, реалізація HMI-інтерфейсу.

Як правило, цільове програмне забезпечення для автоматизованих систем керування розробляється під конкретне застосування самими постачальниками цих систем. Однак останнім часом на ринку з'явилася велика кількість програмних продуктів класу SCADA/MMI для індустріальних систем, що дозволяють вирішувати завдання автоматизації для дискретного виробництва, індустрії процесів, виробництва електроенергії. Найбільших успіхів у цьому напрямку домоглися компанії Intellution і Wonderware.

3.2 Розробка структурної схеми

Звичайно системний інтегратор або кінцевий користувач, приступаючи до розробки прикладного програмного забезпечення (ППЗ) для створення системи керування, вибирає один з наступних шляхів:

– програмування з використанням "традиційних" засобів (традиційні мови програмування, стандартні засоби налагодження та ін.);

– використання існуючих, готових (COTS Commercial Off The Shelf) інструментальних проблемно-орієнтованих засобів.

Безумовно, немає нічого краще якісного, добре налагодженого ППЗ, написаного висококваліфікованим програмістом спеціально для деякого проекту.

Але наступне завдання цей програміст змушений вирішувати знову практично з нуля. Процес створення ППЗ для складних розподілених систем стає неприпустимо тривалим, а витрати на його розробку дуже високими. Сьогодні, в умовах усе більше зростаючої частки ППЗ у витратах на створення кінцевої системи й, відповідно, все більшої інтенсифікації праці програмістів, варіант із безпосереднім програмуванням відносно привабливий лише для простих систем або невеликих фрагментів великої системи, для яких немає стандартних рішень (не написаний, наприклад, що підходить драйвер) або вони не влаштовують по тим або інших причинах у принципі. У кожному разі процес розробки власного ППЗ важливо спростити, скоротити тимчасові й прямі фінансові витрати на розробку ППЗ, мінімізувати витрати праці висококласних програмістів, по можливості залучаючи до розробки фахівців в області автоматизуємих процесів.

Сучасний бізнес в області розробки ПЗ усе більше й більше сегментується й спеціалізується. Причина проста: ПЗ стає усе більше складним і дорогим. Розроблювачі операційних систем, розроблювачі інструментальних засобів, розроблювачі прикладного ПЗ й т.п., по суті, розмовляють різними мовами. Таким чином, сама логіка розвитку сучасного бізнесу в частині розробки ППЗ для кінцевих систем керування вимагає використання усе більше розвинених інструментальних засобів типу SCADA-систем (від Supervisory Control And Data Acquisition). Розробка сучасної SCADA-системи вимагає більших вкладень і виконується в тривалий термін. І саме тому в більшості випадків розроблювачам керуючого ППЗ, зокрема ППЗ для АСУ ТП, представляється доцільним іти по другому шляху, здобуваючи, освоюючи й адаптуючи який-небудь готовий, уже випробуваний універсальний інструментарій.

Схематично зупинимося на традиційному наборі властивостей і характеристик SCADA-систем і загострімо увагу на нових, що з'явилися недавно, зв'язках SCADA-систем з навколишнім світом (OPC-сервери, розширення реального часу для лінійки Windows NT) і на моментах, які нечасто знаходять висвітлення в публікаціях про нішу SCADA-систем у комплексі програмних

компонентів наскрізної автоматизації виробництва. SCADA-системи закривають цеховий рівень автоматизації, зв'язаний, насамперед, з одержанням і візуалізацією інформації від програмувальних контролерів, розподілених систем керування. Інформація, що поставляється на даний рівень, недоступна на рівні керування виробництвом. Тому важливо відзначити, що деякі фірми розробляють системи керування виробництвом і забезпечують обмін між цими рівнями.

Перелічимо характеристики, важливі для оцінки функціональності SCADA-системи, що реалізується, з коротким їхнім аналізом.

Програмно-апаратні платформи, на яких реалізована SCADA-система

Аналіз переліку таких платформ необхідний, оскільки від нього залежить відповідь на питання поширення SCADA-системи на наявні обчислювальні засоби, а також оцінка вартості експлуатації системи (прикладна програма, розроблена в одному операційному середовищі, може виконуватися в будь-який інший, котру підтримує обраний SCADA-пакет). У різних SCADA-системах це питання вирішене по різному. Так, FactoryLink має досить широкий список підтримуваних програмно-апаратних платформ:

У той же час у таких SCADA-системах, як RealFlex і Sitex основу програмної платформи принципово становить єдина, хоча й задовольняюча багатьом вимогам, операційна система реального часу QNX.

Переважну більшість SCADA-систем реалізовано на платформах MS Windows. Саме такі системи пропонують найбільш повні й легко нарощувані людино-машинні інтерфейсні (Man Machine Interface MMI) засоби. З огляду на триваюче посилення позицій Microsoft на ринку операційних систем (ОС) слід зазначити, що навіть розроблювачі багатоплатформених SCADA-систем, такі як United States DATA Co, пріоритетним уважають подальший розвиток своїх SCADA-систем на платформі Windows Server 2022 / Windows 10/11. Деякі фірми, що дотепер підтримували SCADA-системи на базі ОС реального часу (РВ), почали міняти орієнтацію, вибираючи системи на платформі Windows Server 2022 / Windows 10/11. Усе більше очевидним стає застосування ОС реального часу, в

основному, у вбудовуються системах, що. Таким чином, основним полем, де сьогодні розвертаються головні події глобального ринку SCADA-систем, стала ОС MS Windows Server 2022 / Windows 10/11.

Наявні засоби мережної підтримки

Одна з основних особливостей сучасного миру систем автоматизації високий ступінь інтеграції цих систем. У кожній з них можуть бути задіяні об'єкти керування, виконавчі механізми, апаратура, що реєструє й обробляє інформацію, робочі місця операторів, сервери баз даних і т.д. Очевидно, що для ефективного функціонування в цьому різнорідному середовищі SCADA-система повинна забезпечувати високий рівень мережного сервісу. Бажано, щоб вона підтримувала роботу в стандартних мережних середовищах (ARCNET, ETHERNET і т.д.) з використанням стандартних протоколів (NETBIOS, TCP/IP і ін.), а також забезпечувала підтримку найбільш популярних мережних стандартів із класу промислових інтерфейсів (PROFIBUS, CANBUS, LON, MODBUS і т.д.).

Структурна схема системи, що розробляється, наведена на рисунку. 3.1.

Цим вимогам у тому або іншому ступені задовольняють практично всі SCADA-системи, з тим тільки розходженням, що набір підтримуваних мережних інтерфейсів, звичайно ж, різний.

Убудовані командні мови

SCADA-система, що реалізується, має убудовані мови високого рівня, VBasic-подібні мови, що дозволяють згенерувати адекватну реакцію на події, пов'язані зі зміною значення змінної, з виконанням деякої логічної умови, з натисканням комбінації клавіш, а також з виконанням деякого фрагмента із заданою частотою щодо всього додатка або окремого вікна.

Підтримувані бази даних

SCADA-система, що реалізується, використовує синтаксис ANSI SQL, що не залежить від типу бази даних. Таким чином, додатки віртуально ізольовані, що дозволяє міняти базу даних без серйозної зміни самого прикладного завдання, створювати незалежні програми для аналізу інформації, використовувати вже напрацьоване програмне забезпечення, орієнтоване на обробку даних.

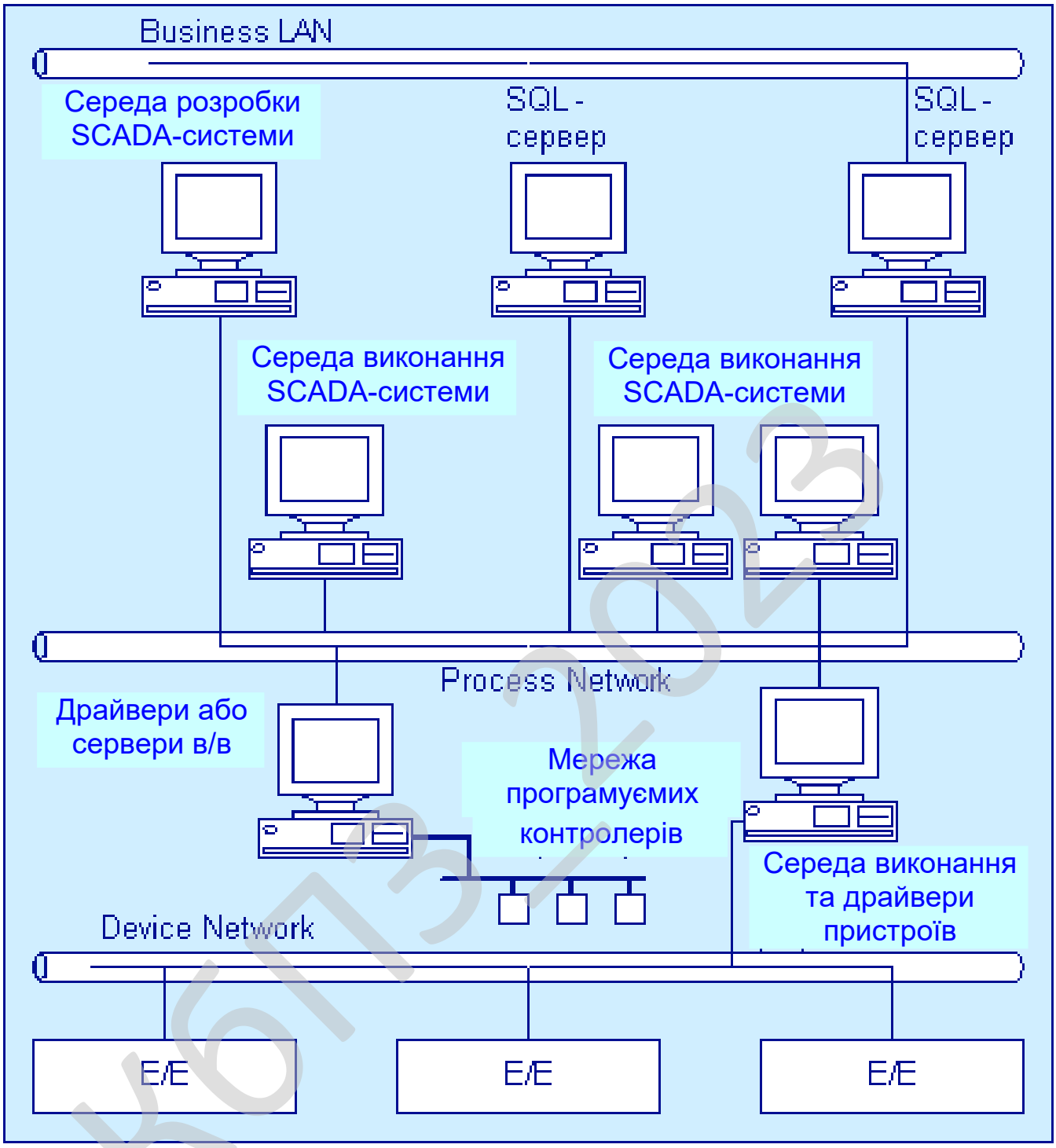

Рисунок 3.1 – Структурна схема системи

Графічні можливості

Для фахівця-розроблювача системи автоматизації, також як і для фахівця-технолога, чиє робоче місце створюється, дуже важливий графічний користувальницький інтерфейс (Graphic Users Interface MMI). Функціонально

графічні інтерфейси SCADA-систем дуже схожі. У кожній з них існує графічний об’єктно-орієнтований редактор з певним набором анімаційних функцій. Використовувана векторна графіка дає можливість здійснювати широкий набір операцій над обраним об'єктом, а також швидко обновляти зображення на екрані, використовуючи засобу анімації.

Украй важливе також питання про підтримку в розглянутих системах стандартних функцій GUI. Оскільки SCADA-система, що реалізується, працює під керуванням Windows, це й визначає тип використовуваного GUI.

Відкритість системи

Програмна система є відкритою, якщо для неї визначені й описані використовувані формати даних і процедурний інтерфейс, що дозволяє підключити до неї зовнішні, незалежно розроблені компоненти.

Розробка власних програмних модулів

Перед фірмами-розроблювачами систем автоматизації часто встає питання про створення власних (не передбачених у рамках систем SCADA) програмних модулів і включення їх у створювану систему автоматизації. Тому питання про відкритість системи є важливою характеристикою SCADA-систем. Фактично відкритість системи означає доступність специфікацій системних (у змісті SCADA) викликів, що реалізують той або інший системний сервіс. Це може бути й доступ до графічних функцій, функціям роботи з базами даних і т.д.

Драйвери введення-виводу

SCADA-система, що реалізується, не обмежує вибору апаратури нижнього рівня, тому що надають великий набір драйверів або серверів введення-виводу й мають добре розвинені засоби створення власних програмних модулів або драйверів нових пристроїв нижнього рівня. Самі драйвери розробляються з використанням стандартних мов програмування. Питання, однак, у тому, чи досить тільки специфікацій доступу до ядра системи, що поставляються фірмою-розроблювачем у штатному комплекті (система Trace Mode), або для створення драйверів необхідні спеціальні пакети (системи FactoryLink, InTouch), або ж, взагалі, розробку драйвера потрібно замовляти у фірми-розроблювача.

Для приєднання драйверів введення-виводу до SCADA використовуються два механізми:

– стандартний динамічний обмін даними (Dynamic Data Exchange DDE);

– обмін по внутрішньому протоколу.

В SCADA-системах основним механізмом, використовуваним для зв'язку із зовнішнім світом, дотепер залишається механізм DDE. Але через свої обмеження по продуктивності й надійності він не зовсім придатний для обміну інформацією в реальному масштабі часу. Замість DDE компанія Microsoft запропонувала більше ефективний і надійний засіб передачі даних між процесами OLE (Object Linking and Embedding включення й вбудовування об'єктів). Механізм OLE підтримується в RSView, Fix, InTouch, Factory Link і ін. На базі OLE з'являється новий стандарт OPC (OLE for Process Control OLE для АСУТП), орієнтований на ринок промислової автоматизації. Новий стандарт, по-перше, дозволяє поєднувати на рівні об'єктів різні системи керування й контролю, що функціонують у розподіленому гетерогенному середовищі; по-друге, усуває необхідність використання різного нестандартного встаткування й відповідних комунікаційних програмних драйверів. З погляду SCADA-систем, поява OPC-серверів означає розробку програмних стандартів обміну з технологічними пристроями. Оскільки виробники повністю розбираються у своїх пристроях, те ці специфікації є для них керівництвом до розробки відповідних драйверів. Тому що ці програмні драйвери вже з'являються на ринку, розроблювачі SCADA-систем пропонують свої механізми зв'язку з OPC-драйверами. OPC-інтерфейс допускає різні варіанти обміну: одержання сирих даних з фізичних пристроїв, з розподіленої системи керування або з будь-якого додатка. На ринку з'явилися інструментальні пакети для написання OPC-Компонентів, наприклад, OPC-Toolkits фірми FactorySoft Inc., що включає OPC Server Toolkit, OPC Client Toolkit, приклади OPC-програм.

Об'єкти ActiveX, що вбудовуються

Об'єкти ActiveX це об'єкти, в основі яких лежить модель складених об'єктів Microsoft COM (Component Object Model). Технологія COM визначає загальну схему взаємодії компонентів програмного забезпечення в середовищі Windows і надає стандартну інфраструктуру, що дозволяє об'єктам обмінюватися даними й функціями між прикладними програмами. Більшість SCADA-систем є контейнерами, які повідомляються ActiveX про події, що відбулися. Будь-які ActiveX-об'єкти можуть завантажуватися в систему розробки більшості SCADA-систем і використовуватися при створенні прикладних програм. Керування ActiveX-об'єктами здійснюється за допомогою даних, методів і подійних функцій, властивих обраному об'єкту.

Розробки третіх фірм

Багато компаній займаються розробкою драйверів, ActiveX-об'єктів і іншого програмного забезпечення для SCADA-систем. Цей факт дуже важливо оцінювати при виборі SCADA-пакета, оскільки це розширює область застосування системи непрофесійними програмістами.

Для реалізації вищевказаних технологій розроблені спеціальні бібліотеки й інструментальні системи для ОС Windows. Використання ж тільки специфікацій стандартів для цього не тільки досить трудомістко, але й вимагає високого професіоналізму програмістів і, отже, важко для не-Windows платформ.

Реальний час для Windows Server 2022 / Windows 10/11

Один з істотних недоліків SCADA-систем на платформах Windows 3.xx/95 у порівнянні з SCADA-системами на платформах ОСРВ відсутність підтримки твердого реального часу. Ситуація стала змінюватися з появою Windows Server 2022 / Windows 10/11. Вихід у світ цієї ОС стимулював розробку нових підходів у підтримці твердого реального часу. Насамперед, сама по собі Windows Server 2022 / Windows 10/11 робить досить успішні спроби потіснити ОСРВ. Проте, Windows Server 2022 / Windows 10/11 має ряд обмежень. Такі її особливості, як перевага апаратного переривання програмному (навіть якщо цей простий рух миші), виконання в підпрограмі обробки апаратних переривань лише необхідних

дій з виконанням наступної обробки через чергу відкладених процедур, відсутність пріоритетної обробки процесів у черзі відкладених процедур, не дозволяють віднести Windows Server 2022 / Windows 10/11 до категорії класичних ОС реального часу.

Ряд фірм (LP Elektronik, Imagination Systems, RadSys, Spectron Microsystems, VenturCom) почали більше радикальні спроби перетворити Windows Server 2022 / Windows 10/11 в ОС твердого реального часу. Розглянемо деякі ключові особливості реалізації такої ідеї на підсистемі реального часу RTX (Real Time Extension), запропонованою фірмою Ventur Com. Саме ця реалізація одержує в цей час найбільш широке поширення. Фірми-розроблювачі SCADA-систем негайно почали пропонувати застосування нових рішень. Так, набір прикладних інтерфейсів програмування RTX 4.1 (Ventur Com) в FIX дозволяє:

– здійснювати повний контроль над завданнями реального часу;

– використовувати фіксовану систему з 128 пріоритетів для контролю RTX-Завдань;

– застосовувати стандартні засоби обміну даними між завданнями;

– звертатися до стандартних функцій з Win32 API.

Поява подібних рішень, по-перше, завдає чергового удару по SCADA-системах на базі ОСРВ, оскільки віднімає (хоча й досить штучно) у них дуже важлива перевага твердого реального часу, закладені в ОСРВ, і, по-друге, тіснить застосування ОСРВ у системах, що вбудовуються.

Експлуатаційні характеристики

Експлуатаційні характеристики SCADA-системи мають велике значення, оскільки від них залежить швидкість освоєння продукту й розробки прикладних систем. Вони в остаточному підсумку відбиваються на вартості реалізації проектів.

3.3 Розробка функціональної схеми

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

У силу тих вимог, які пред'являються до систем SCADA, спектр їхніх функціональних можливостей визначений і реалізований практично у всіх пакетах.

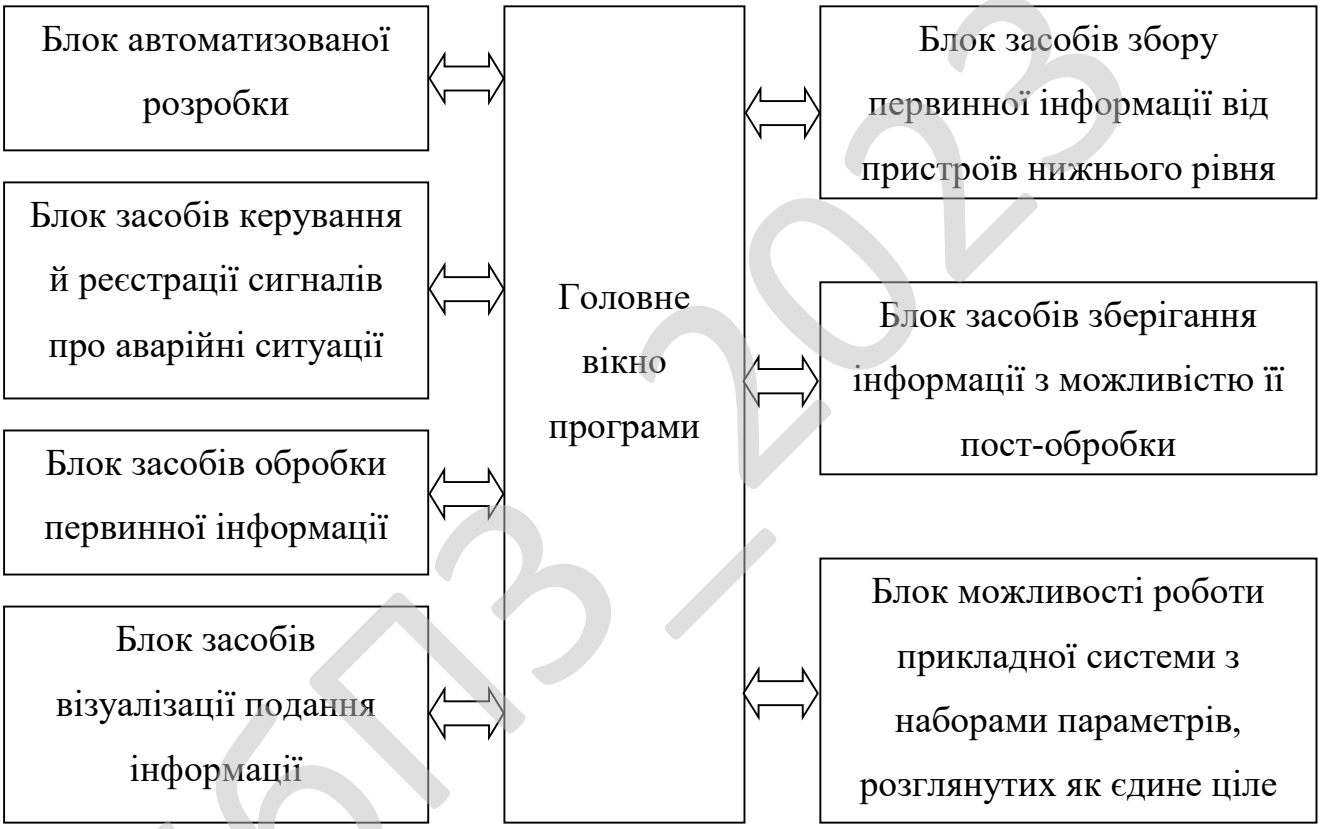

Рисунок 3.2 – Функціональна схема системи

Перелічимо основні можливості й засоби, властивим всім системам і реалізації, що тільки розрізняються технічними особливостями:

– автоматизована розробка, що дає можливість створення програмного забезпечення (ПЗ) системи автоматизації без реального програмування;

– засоби збору первинної інформації від пристроїв нижнього рівня;

– засоби керування й реєстрації сигналів про аварійні ситуації;

– засоби зберігання інформації з можливістю її пост-обробки (як правило, реалізується через інтерфейси до найбільш популярних баз даних);

– засоби обробки первинної інформації;

– засоби візуалізації подання інформації у вигляді графіків, гістограм і т.п.;

– можливість роботи прикладної системи з наборами параметрів, розглянутих як єдине ціле (recipe , або установки).

Основу більшості SCADA-пакетів становлять кілька програмних компонентів (база даних реального часу, введення-виводу, передісторії, аварійних ситуацій) і адміністраторів (доступу, керування, повідомлень).

Слід зазначити, що технологія проектування систем автоматизації на основі різних SCADA-систем багато в чому схожа й включає наступні етапи. Розробка архітектури системи автоматизації в цілому. На цьому етапі визначається функціональне призначення кожного вузла системи автоматизації. Рішення питань, пов'язаних з можливою підтримкою розподіленої архітектури, необхідністю введення вузлів з гарячим резервуванням і т.п. Створення прикладної системи керування для кожного вузла. На цьому етапі фахівець в області автоматизуємих процесів наповнює вузли архітектури алгоритмами, сукупність яких дозволяє вирішувати завдання автоматизації. Приведення параметрів прикладної системи у відповідність із інформацією, який обмінюються пристрої нижнього рівня (наприклад, програмувальні логічні контролери ПЛ=+=) із зовнішнім миром (датчики температури, тиску й ін.). Налагодження створеної прикладної програми в режимі емуляції (у деяких системах, наприклад IGSS, режим налагодження практично відсутній) і в реальному режимі.

Перераховані вище можливості систем SCADA значною мірою визначають вартість і строки створення ПЗ, а також строки її окупності.

Зручність використання

Слід зазначити, що сервіс, надаваний SCADA-системами на етапі розробки прикладного ПЗ, звичайно дуже високий це випливає з основних вимог до таких систем. Майже всі вони мають Windows-подібний користувальницький інтерфейс, що багато в чому підвищує зручність їхнього використання, як у процесі розробки, так і в період експлуатації прикладного завдання.

Наявність і якість підтримки

Необхідно звертати увагу не тільки на наявність технічної підтримки SCADA-систем, як такий, але й на її якість. Для закордонних систем у України можливі наступні рівні підтримки: послуги фірми-розроблювача; обслуговування регіональними представниками фірми-розроблювача; взаємодія із системними інтеграторами. Судячи з великої кількості установок закордонних систем, що обчислюються в тисячах (InTouch 80000, Genesis 30000), можна припустити, що підтримка цих систем досить ефективна.

Інтеграція багаторівневих систем автоматизації

SCADA-системи відповідальні за одержання інформації з рівня Керування, знизу, тобто від різних датчиків через пристрої сполучення, від програмувальних контролерів, що поставляють інформацію для безпосереднього керування виробничим процесом. Далі інформація з рівня Керування надходить на вхід SCADA-систем. На SCADA-рівні можливо оперативне керування процесом, прийняття тактичних рішень на основі інформації, отриманої на рівні Керування. Сам процес надходження інформації на виробництві відбувається й зверху, і знизу. Зверху формується інформація, відповідальна за роботу підприємства в цілому, здійснюється планування виробництва. Точна, своєчасна, достовірна інформація на кожному рівні виробництва дозволяє оцінити рівень витрат, якість і конкурентоспроможність продукції. Для організації зв'язку між інформацією зверху й знизу необхідний клас інструментальних засобів керування виробництвом, відповідальний за доставку даних у реальному часі з рівня Керування наверх і у зворотному напрямку, з можливою обробкою цих даних.

Тому досить важливим критерієм порівняння інструментальних засобів, що підтримують розробку АСУ ТП, є наявність засобів доставки інформації з SCADA-рівня наверх, на рівень планування виробництва. Ряд фірм (Intellution, Wonderware) пропонує продукти (Fix BOS, InTrack, InBatch), що представляють собою системи керування виробництвом. Основне їхнє призначення полягає в створенні прикладних програм, що моделюють і простежують кожну стадію виробничих процесів від завантаження сировини до випуску готової продукції.

Величезне стратегічне значення мають те, наскільки інструментальні системи АСУ ТП пов'язані з Microsoft BackOffice Suite, оскільки останні сталі розповсюдженими офісними програмними продуктами. Тому, наприклад, всі продукти FactorySuite компанії Wonderware легко інтегруються з такими продуктами, як Microsoft SQL Server, Windows Server 2022 / Windows 10/11 Server, System Management Server, SNA Server і Mail-Server. Фірма Wonderware пропонує IndustrialSQL Server, що дозволяє реєструвати дані в реальному часі. Джерелом даних можуть бути InTouch-сервери висновку-виводу-вводу-виводу. Побудований же IndustrialSQL Server на базі Microsoft SQL Server. Це істотно розширює можливості всього виробничого персоналу в змісті можливості доступу до повної інформації про будь-який етап виробництва.

Усе більше актуальним стає вимога передачі як статичної (у певні моменти часу), так і динамічної (постійно) інформації на вузол-вузол-web-вузли. В деяких web-браузерах об'єкти з’явилися ActiveX дозволяють передавати дані з SCADA-системи на web-сторінки. Але існують і більше багатофункціональні компоненти типу Scout фірми Wonderware, що забезпечують можливість доступу до систем автоматизації на базі InTouch через Internet/Intranet і які дозволяють віддаленому користувачеві взаємодіяти із прикладним завданням автоматизації, як із простою WEB-сторінкою.

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

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. Після початку роботи розробленого ПЗ ми потрапляємо до головного блоку системи звідки через ланку дій відбувається наступне:

– Інтерфейс ПЗ.

– БД ПЗ.

– Налаштування системи.

– Встановлення змінних параметрів роботи системи.

– Обрання типу сигналізації виключних ситуацій.

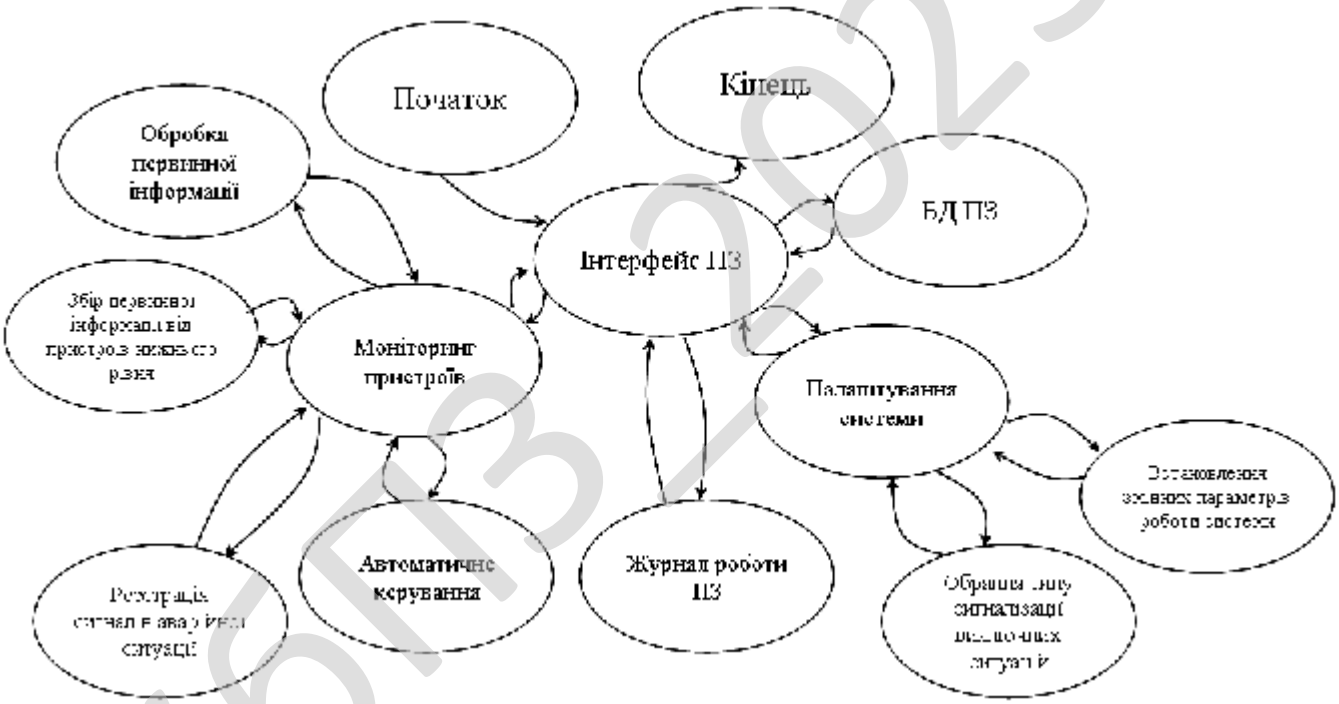

Рисунок 3.3 – Діаграма взаємодії процесів

– Журнал роботи ПЗ.

– Моніторинг пристроїв.

– Автоматичне керування.

– Реєстрація сигналів аварійної ситуації.

– Збір первинної інформації від пристроїв нижнього рівня.

– Обробка первинної інформації.

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

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування). Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи.

Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

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

Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно розробка блок-схем.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми. З якої видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ.

При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

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

При виборі початкової точки відліку при побудові схем я враховував, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю системи диспетчерського керування та збору даних технологічного процесу.

При складанні блок-схем програмного забезпечення і напрацювання алгоритмів я зіткнувся з масою проблем, які вимагали напрацювання процедур і функцій над основною проблематикою. Для чого були створені додаткові класи, типи даних і константи, що забезпечило вирішення проблем.

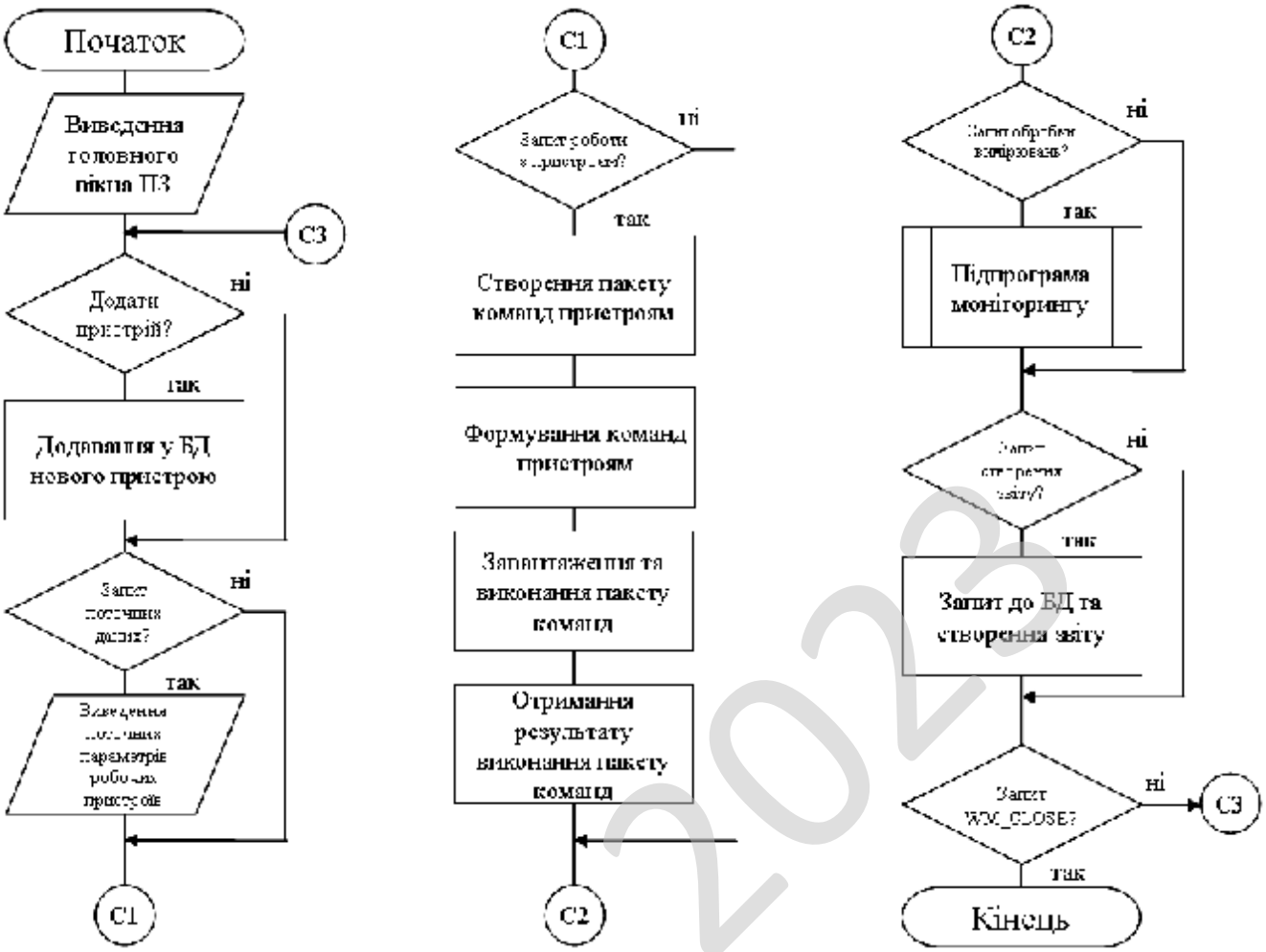

Рисунок 4.1 – Блок схема основної програми

Архітектурно система диспетчерського керування та збору даних технологічного процесу ділиться на підсистеми.

Підсистеми можуть бути двох типів: звичайні й модульні. Модульні підсистеми мають властивість розширення за допомогою модулів. Кожна модульна підсистема може містити безліч модульних об'єктів. Наприклад, модульна підсистема «Бази даних» містить модульні об'єкти типів баз даних. Модульний об'єкт є коренем усередині модуля.

Усього система диспетчерського керування та збору даних технологічного процесу містить 9 підсистем з них 7 підсистем є модульними. 9 підсистем системи є базовими й присутні в будь-якій конфігурації. До списку 9 підсистем можуть додаватися нові підсистеми за допомогою модулів.

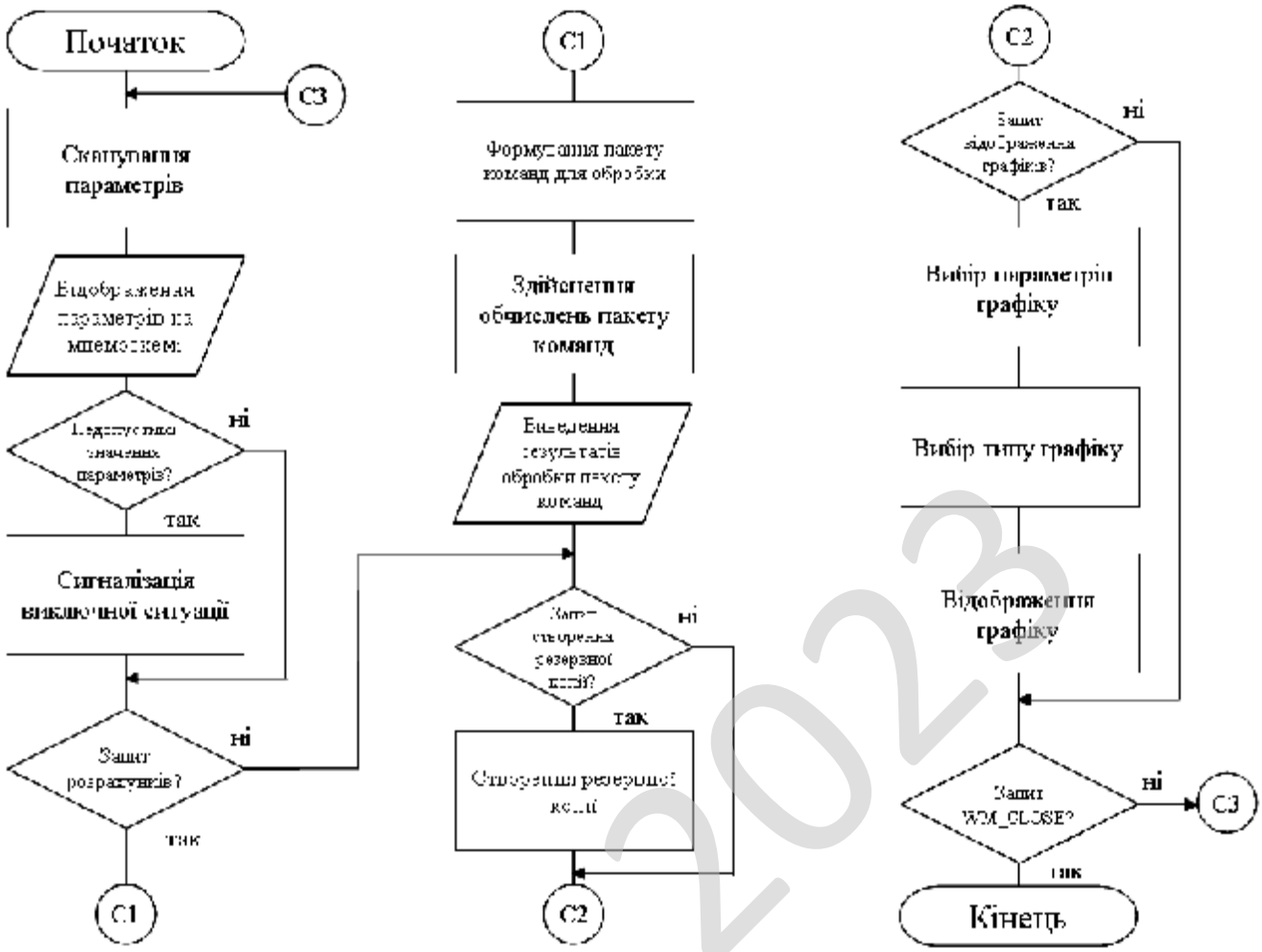

Рисунок 4.2 – Блок схема підпрограми

Підсистеми системи диспетчерського керування та збору даних технологічного процесу:

– Архіви (модульна).

– Бази даних (модульна).

– Безпека.

– Інтерфейси користувача (модульна).

– Керування модулями.

– Збір даних (модульна).

– Транспортні протоколи (модульна).

– Спеціальні (модульна).

– Транспорти (модульна).

Розглянемо розроблені програмні об'єкти.

Абстрактний об'єкт являє собою асоціативний контейнер властивостей і функцій. Властивості можуть містити як дані чотирьох базових типів, так і інші об'єкти.

Доступ до властивостей об'єкта звичайно здійснюється за допомогою запису імен властивостей через крапку до об'єкта <obj.prop>, а також за допомогою висновку ім'я властивості у квадратні дужки <obj["prop"]>.

Очевидно, що перший механізм статичний, а другий дозволяє вказувати ім'я властивості через змінну. Базове визначення об'єкта не містить функцій. Операції копіювання об'єкта насправді роблять посилання на вихідний об'єкт. При видаленні об'єкта здійснюється зменшення лічильника посилань, а при досягненні лічильника посилань нуля об'єкт віддаляється фізично.

Різні компоненти можуть довизначати базовий об'єкт особливими властивостями й функціями. Стандартним розширенням об'єкта є масив "Array".

Об'єкт Array. Особливістю масиву є те, що він працює із властивостями, як з індексами, і повне їхнє іменування безглуздо, а значить доступний механізм обігу тільки висновком індексу у квадратні дужки <arr[1]>.

Масив зберігає властивості у власному контейнері одномірного масиву. Цифрові властивості масиву використовуються для доступу безпосередньо до масиву, а символьні працюють як властивості об'єкта.

Масив надає спеціальна властивість "length" для одержання розміру масиву <var = arr.length;>. Також масив надає наступні функції:

– `string join( string sep = ","), string toString( string sep = ","), string valueOf( string sep = ",")` – повертає рядок з елементами масиву розділеними <sep> або символом

– `Array concat( Array arr);` – Додає до вихідного масиву елементи масиву <arr>. Повертає вихідний масив зі змінами.

– `int push( ElTp var, ...);` – Поміщає елемент(ы) <var> у кінець масиву, як у стек. Повертає новий розмір масиву.

– `ElTp pop();` – Видалення останнього елемента масиву й повернення його значення, як зі стека.

– `Array reverse();` – Зміна порядку розташування елементів масиву. Вертається вихідний масив зі змінами.

– `ElTp shift();` – Зрушення масиву у верх. При цьому перший елемент масиву віддаляється, а його значення вертається.

– `int unshift( ElTp var, ...);` – Засовує елемент(ы) <var> у масив. Перший елемент в 0, другий в 1 і т.д.

– `Array slice( int beg, int end);` – Повертає фрагмент масиву від <beg> до <end>. Якщо значення початку або кінця негативно, то відлік ведеться з кінця масиву. Якщо кінець не зазначений, то кінцем є кінець масиву.

– `Array splice( int beg, int rem, ElTp val1, ElTp val2, ...);` - Вставляє, видаляє або заміняє елементи масиву. Повертає вихідний масив зі змінами. У першу чергу здійснюється видалення елементів з позиції <beg> і кількістю <rem>, а потім вставляються значення <val> і т.д., починаючи з позиції <beg>.

– `Array sort();` – Сортування елементів масиву в лексикографічному порядку.

Об'єкт RegExp. Об'єкт роботи з регулярними вираженнями, заснований на бібліотеці PCRE. При глобальному пошуку встановлюється атрибут об'єкта "lastIndex", що дозволяє продовжити пошук при наступному виклику функції. У випадку невдалого пошуку атрибут "lastIndex" скидається в нуль.

Як аргументи створення об'єкта передається рядок з текстом регулярного вираження й прапори у вигляді рядка символів:

– 'g' – режим глобального пошуку;

– 'i' – режим регістронезалежного пошуку;

– 'm' – режим багаторядкового пошуку;

– 'p' – тестування вираження по звичайному шаблоні із ключовими символами: '?', '*' і '\'.

Властивості об'єкта:

– `source` – Вихідний шаблон регулярного вираження, тільки читання.

– `global` – Ознака глобального пошуку, тільки читання.

– `ignoreCase` – Ознака ігнорувати регістр символів при пошуку, тільки читання.

– `multiline` – Ознака багаторядкового пошуку, тільки читання.

– `lastIndex` – Індекс символу за підрядком останнього пошуку. Використовується в глобальному режимі для продовження сканування, при наступному виклику.

Функції об'єкта:

– `Array exec(string val);` – Виклик пошуку по рядку <val>. Повертає знайдену підрядок (0) і підвиразу (>0) у масиві. Установлює атрибут масиву "index" у позицію знайденого підрядка. Установлює атрибут масиву "input" у значення вихідного рядка.

– `var re = new RegExp("(\\d\\d)[-/](\\d\\d)[-/](\\d\\d(?:\\d\\d)?)","");`

```
var rez = re.exec("12/30/1969"); var month = rez[1]; var day = rez[2]; var year = rez[3];
```

– `bool test(string val); - Повертає "true" якщо подрядок знайдений в <val>.`

```
var re = new RegExp("(\\d\\d)[-/](\\d\\d)[-/](\\d\\d(?:\\d\\d)?)","");
var OK = re.test("12/30/1969");
```

Об'єкт XMLNodeObj. Функції:

– `string name()` – Ім'я вузла, XML-тегу.

– `string text()` – Текст вузла, уміст XML-тегу.

– `string attr( string id)` – Значення атрибута вузла <id>.

– `XMLNodeObj setName( string vl)` – Установка ім'я вузла в <vl>. Повертає поточний вузол.

– `XMLNodeObj setText( string vl)` – Установка тексту вузла в <vl>. Повертає поточний вузол.

– `XMLNodeObj setAttr( string id, string vl)` – Установка атрибута <id> у значення <vl>. Повертає поточний вузол.

– `int childSize()` – Кількість вкладених вузлів.

– `XMLNodeObj childAdd( ElTp no = XMLNodeObj); XMLNodeObj childAdd(string no)` – Додавання об'єкта <no> як вкладеного. <no> може бути як безпосередньо об'єктом-результатом функції B.XMLNode(), так і рядком з ім'ям нового тегу. Вертається вкладений вузол.

– `XMLNodeObj childIns( int id, ElTp no = XMLNodeObj); XMLNodeObj childIns(int id, string no)` – Вставка об'єкта <no> як вкладеного в позицію <id>. <no> може бути як безпосередньо об'єктом-результатом функції B.XMLNode(), так і рядком з ім'ям нового тегу. Вертається вкладений вузол.

– `XMLNodeObj childDel( int id)` - Видалення вкладеного вузла в позиції <id>. Повертає поточний вузол.

– `XMLNodeObj childGet( int id)` - Одержання вкладеного вузла в позиції <id>.

– `XMLNodeObj parent()` - Одержати батьківський вузол.

– `String load( string str, bool file = false, bool full = false, string cp = " UTF-8")` – Завантаження XML з рядка <str> або з файлу зі шляхом в <str> якщо <file> "true", з кодуванням <cp>. <full> – для повного завантаження, із блоками тексту й коментарями в спеціальних вузлах.

– `string save( int opt = 0, string path = "", string cp = " UTF-8")` – Збереження дерева XML у рядок або у файл <path> з параметрами форматування <opt> і кодуванням <cp>. Повертає текст XML або код помилки. Передбачено наступні опції форматування <opt>:

– `0x01` – переривати рядок перед відкриваючим тегом;

– `0x02` – переривати рядок після відкриваючого тегу;

– `0x04` – переривати рядок після закриваючого тегу;

– `0x08` – переривати рядок після тексту;

– `0x10` – переривати рядок після інструкції;

– `0x1E` – переривати рядок після всіх;

– `0x20` – вставляти стандартний XML-заголовок;

– `0x40` – вставляти стандартний XHTML-заголовок.

— `XMLNodeObj getElementBy( string val, string attr = "id")` — одержати елемент із дерева по атрибуті <attr> зі значенням <val>.

Система (SYS). Функції об'єкта:

— `string system( string cmd, bool noPipe = false);` — здійснює виклик консольних команд <cmd> ОС із поверненням результату по каналі. Якщо <noPipe> установлений то вертається код повернення виклику й можливий запуск програм у тлі ("sleep 5 &"). Функція відкриває широкі можливості користувачеві диспетчерського керування та збору даних технологічного процесу шляхом виклику будь-яких системних програм, утиліт і скриптів, а також одержання за допомогою їх доступу до величезного обсягу системних даних. Наприклад команда "ls -l" поверне деталізований уміст робочої директорії.

— `string fileRead( string file`); – Повертає вміст файлу <fle> у рядку.

— `int fleWrite( string file, string str, bool append = false);` – Записує рядок <str> у файл <file>, видаляючи присутній файл або додаючи в нього <append>. Повертає кількість записаних байт.

— `int message( string cat, int level, string mess);` – формування системного повідомлення <mess> з категорією <cat>, рівнем <level>. Негативне значення рівня формує порушення (Alarm).

— `int messDebug(string cat, string mess); int messInfo(string cat, string mess); int messNote(string cat, string mess); int messWarning(string cat, string mess); int messErr(string cat, string mess); int messCrit(string cat, string mess); int messAlert(string cat, string mess); int messEmerg(string cat, string mess);` – формування системного повідомлення <mess> з категорією <cat> і відповідним рівнем.

— `XMLNodeObj XMLNode( string name = "");` – створення об'єкта вузла XML с ім'ям <name>.

— `string cntrReq( XMLNodeObj req, string stat = "");` - запит інтерфейсу керування до системи за допомогою XML. Звичайний запит записується у вигляді <get path="/OPath/ %2felem"/>. При вказівці станції здійснюється запит до зовнішньої станції.

```
req = B.XMLNode("get").setAttr("path","/%2fgen%2fid");
```

```
B.cntrReq(req);
idSt = req.text();
```

— `string sleep(int tm, int ntm = 0);` — приспати потік виконання на <tm> секунд і <ntm> наносекунд. Функція додана тільки для повноти й украй не рекомендована до використання, особливо в процедурах користувальницького інтерфейсу оскільки це приведе до блокування інтерфейсу.

— `int time( int usec);` — повертає абсолютний час у секундах від епохи 1.1.1970 і мікросекундах, якщо <usec> зазначений.

— `int localtime( int fullsec, int sec, int min, int hour, int mday, int month, int year, int wday, int yday, int isdst);` — повертає повну дату в секундах (sec), хвилинах (min), годинниках (hour), днях місяця (mday), місяці (month), році (year), днях тижня (wday), днях у році (yday) і ознака літнього часу (isdst), виходячи з абсолютного часу в секундах <fullsec> від епохи 1.1.1970.

— `string strftime( int sec, string form = "% Y-Y-%m-%d %H:%M:%S");` – Перетворить абсолютний час <sec> у рядок потрібного формату <form>. Запис формату відповідає POSIX-функції strftime.

— `int strptime( string sttr, string form = "% Y-Y-%m-%d %H:%M:%S");` – Повертає час у секундах від епохи 1.1.1970, виходячи зі строкового запису часу <str>, відповідно до зазначеного шаблона <form>. Наприклад, шаблону "% Y-Y-%m-%d %H:%M:%S" відповідає час " 2006-08-08 11:21:55". Опис формату шаблона можна одержати з документації на POSIX-функцію "strptime".

— `int cron( string cronreq, int base = 0);` — Повертає час, спланований у форматі стандарту Cron <cronreq>, починаючи від базового часу <base> або від поточного, якщо базове не зазначено.

— `string strFromCharCode( int char1, int char2, int char3, ...);` — Створення рядка з кодів символів char1, char2 ... char.

— `string strCodeConv( string src, stringfromCP, string toCP);` — Кодування тексту <src> з кодування <fromCP> в <toCP>. Якщо кодування опущене, то використовується внутрішня.

Будь-який об'єкт (TCntrNode) дерева диспетчерського керування та збору даних технологічного процесу (B.*). Функції об'єкта:

– `TArrayObj nodeList(string grp = "", string path = "");` - Одержання списку дочірніх вузлів для групи <grp> і вузла по шляху <path>. Якщо <grp> порожня то вертаються вузли всіх груп.

– `TCntrNodeObj nodeAt(string path, string sep="");` - Підключення до вузла <path> у дереві об'єктів диспетчерського керування та збору даних технологічного процесу. Якщо вказується роздільник в <sep> то шлях обробляється як рядок з роздільником.

– `TCntrNodeObj nodePrev();` – Одержати попередній, батьківський, вузол.

– `string nodePath(string sep = "", bool fromroot = true);` - Одержання шляху до поточного вузла, у дереві об'єктів диспетчерського керування та збору даних технологічного процесу. Один символ роздільника вказується в <sep> для одержання шляху через роздільник, наприклад, "A.ModBus.PLC1.P1.var", інакше "/DAQ/ModBus/PLC1/P1/var". <from_root> указує на необхідність формувати шлях від кореня й без вказівки ідентифікатора станції.

Підсистема "Безпека" (B.Security). Функції об'єкта підсистеми (B.Security):

– `int access(string user, int mode, string owner, string group, int access)` – Перевірка для користувача <user> доступу до ресурсу який належить <owner> і <group> з доступом <access> для режим <mode>:

– `user` – користувач для перевірки доступу; mode – режим доступу (4-R, 2-W, 1-X); owner – власник ресурсу; group – група ресурсу;

– `access` – режим доступу до ресурсу (RWXRWXRWX – 0777).

Функції об'єкта користувача `(B.Security["usr_User"])` і групи `(B.Security["grp_Group"])`:

– `ElTp cfg(string nm)` – одержання значення конфігураційного поля <nm> об'єкта.

– `ElTp cfgSet(string nm, ElTp val)` – установка конфігураційного поля <nm> об'єкта в значення <val>.

Підсистема "БД" (B.BD). Функції об'єкта БД (B.BD["TypeDB"] ["DB"]):

– ElTp cfg(string nm) – одержання значення конфігураційного поля <nm> об'єкта.

– ElTp cfgSet(string nm, ElTp val) – установка конфігураційного поля <nm> об'єкта в значення <val>.

– Array SQLReq(string req); – Формування SQL-запиту до БД.

```
DBTbl=B.BD.MySQL.GenDB.SQLReq("SELECT *  from DB;");
for(var i rw = 0;  i rw < DBTbl.length;  i rw++)
{
var rec = "";
for( var i_fld = 0;  i_fld < DBTbl[i_rw].length;  i_fld++)
rec += DBTbl[i_rw][i_fld]+"\t"; B.messDebug("TEST DB","Row "+i_rw+":   "+rec);
if(i_rw)      B.messDebug("TEST  DB:      ","Row  "+i_rw+":      'NAME'"+DBTbl[i_rw]
["NAME"]);
}
```

Функції об'єкта Таблиці (B.BD["TypeDB"]["DB"]["Table"]):

– XMLNodeObj fieldStruct(); – Одержання структури таблиці у вигляді XML вузла "field" з дочірніми вузлами-колонками <RowId type="real" len="10.2" key="1" def="Default value">{Value}</RowId>, де:

– {RowId} – ідентифікатор колонки;

– {Value} – значення колонки;

– type – тип значення колонки: str – рядок, int – ціле, real – речовинне й bool – логічне;

– len – розмір значення колонки, у знаках;

– key – ознака того, що колонка є ключем, і пошук здійснюється по його значенню;

– def – значення колонки за замовчуванням.

– string fieldSeek(int row, XMLNodeObj fld); – Запит поля <row> таблиці. Якщо поле отримане то вертається "1" інакше "0". У випадку помилки вертається "0:Error".

– string fieldGet(XMLNodeObj fld); – Запит значень поля. У випадку помилки вертається "0:Error".

```
req = B.XMLNode("field");
req.childAdd("user").setAttr("type","str").setAttr("key","1").setText("root");
req.childAdd("id").setAttr("type","str").setAttr("key","1").setText("/Lang2CodeBas
e");
req.childAdd("val").setAttr("type","str");
B.BD.MySQL.GenDB.B.fieldGet(req);
B.messDebug("TEST DB","Value:   "+req.childGet(2).text());
```

— `string fieldSet(XMLNodeObj fld);` – Установка поля. У випадку помилки вертається "0:Error".

— `string fieldDel(XMLNodeObj fld);` – Видалення поля. У випадку помилки вертається "0:Error".

Підсистема "Збір даних" (B.DAQ). Функції об'єкта підсистеми (B.DAQ):

— `bool funcCall(string progLang, TVarObj args, string prog);` – виклик тексту функції <prog> з аргументами в об'єкті <args> для мови програмування <progLang>. Повертає "true" при коректному виклику.

```
var args = new Object(); args.y = 0; args.x = 123;
B.A.funcCall("BuilderCLikeCalc.BuilderCScript",args,"y=2*x;");
B.messDebug("TEST Calc","TEST Calc rezult:   "+args.y);
```

Функції об'єкта контролера (B.DAQ["Modul"]["Controller"]):

— `ElTp cfg(string nm)` – одержання значення конфігураційного поля <nm> об'єкта.

— `ElTp cfgSet(string nm, ElTp val)` – установка конфігураційного поля <nm> об'єкта в значення <val>.

— `string name()` – ім'я контролера.

— `string descr()` – опис контролера.

— `string status()` - статус контролера.

— `bool alarmSet(string mess, int lev = -5, string prm = "")` – установка/зняття порушення <mess> з рівнем <lev> (негативний для установки інакше зняття), для параметра <prm>.

— `bool enable(bool newSt = EVAL)` – одержання стану "Включена" або зміна його призначенням атрибута <newSt>.

– `bool start(bool newSt = EVAL)` – одержання стану "Запущена" або зміна його призначенням атрибута <newSt>.

Функції об'єкта параметра контролера (B.DAQ["Modul"]["Controller"]["Parameter"]):

– `ElTp cfg(string nm)` – одержання значення конфігураційного поля <nm> об'єкта.

– `ElTp cfgSet(string nm, ElTp val)` – установка конфігураційного поля <nm> об'єкта в значення <val>.

Функції об'єкта атрибута параметра контролера (B.DAQ["Modul"]["Controller"] ["Parameter"]["Attribute"]):

– `ElTp get( int tm = 0, int utm = 0, bool sys = false)` – запит значення атрибута на час <tm:utm> і ознакою системного доступу <sys>.

– `bool set( ElTp val, int tm = 0, int utm = 0, bool sys = false)` - запис значення <val> в атрибут з міткою часу <tm:utm> і ознакою системного доступу <sys>.

– `TCntrNodeObj arch()` – одержання об'єкта архіву, пов'язаного із цим атрибутом. У випадку відсутності зв'язаного архіву вертається "false".

– `string descr()` – опис атрибута.

– `int time(int utm)` – час останнього значення в секундах і мікросекундах в <utm>.

– `int len()` – довжина поля.

– `int dec()` – дозвіл для дійсного числа.

– `int flg()` - прапори поля.

– `string def()` – значення за замовчуванням.

– `string values()` – список припустимих значень або діапазон.

– `string selNames()` – список імен припустимих значень.

– `string reserve()` - резервна властивість значення.

Функції об'єкта бібліотеки шаблона (B.DAQ[tmplb_Lib"]) і шаблона (B.DAQ[tmplb_Lib"]["Tmpl"]) параметра контролера:

– `ElTp cfg(string nm)` - одержання значення конфігураційного поля <nm> об'єкта.

– `ElTp cfgSet(string nm, ElTp val)` - установка конфігураційного поля <nm> об'єкта в значення <val>.

Підсистема "Архіви" (B.Archive). Функції об'єкта підсистеми:

– `Array messGet( int btm, int etm, string cat = "", int lev = 0, string arch= "");` - запит системних повідомлень за час від <btm> до <etm> для категорії <cat>, рівня <lev> і архіватора <arch>. Вертається масив об'єктів повідомлень із властивостями:

– `tm` – час повідомлення, секунди;

– `utm` – час повідомлення, мікросекунди;

– `categ` – категорія повідомлення;

– `level` – рівень повідомлення;

– `mess` – текст повідомлення.

Функції об'єкта архіватора повідомлень (B.Archive["mod_Modul"]["mess_Archivator"]):

– `ElTp cfg(string nm)` – одержання значення конфігураційного поля <nm> об'єкта.

– `ElTp cfgSet(string nm, ElTp val)` - установка конфігураційного поля <nm> об'єкта в значення <val>.

– `bool status()` – одержання статусу архіватора.

– `int end()` - повчання часу закінчення даних архіватора.

– `int begin()` – повчання часу початку даних архіватора.

Функції об'єкта архіву (B.Archive["va_Archive"]) і архіватора значень (B.Archive["val_Modul"]["val_Archivator"]):

– `ElTp cfg(string nm)` – одержання значення конфігураційного поля <nm> об'єкта.

– `ElTp cfgSet(string nm, ElTp val)` – установка конфігураційного поля <nm> об'єкта в значення <val>.

Підсистема "Транспорти" (B.Transport). Функції об'єкта вхідного транспорту (B.Transport["Modul"]["in_Transp"]):

– `ElTp cfg(string nm)` – одержання значення конфігураційного поля <nm> об'єкта.

— `ElTp cfgSet(string nm, ElTp val)` — установка конфігураційного поля <nm> об'єкта в значення <val>.

Функції об'єкта вихідного транспорту (B.Transport["Modul"] ["out_Transp"]):

— `ElTp cfg(string nm)` — одержання значення конфігураційного поля <nm> об'єкта.

— `ElTp cfgSet(string nm, ElTp val)` - установка конфігураційного поля <nm> об'єкта в значення <val>.

— `string messIO( string mess, real timeOut = 0);` — відправлення повідомлення <mess> через транспорт із таймаутом очікування <timeOut> (у секундах). У випадку нульового таймаута цей час береться з настроювань вихідного транспорту.

```
rez=B.Transport.Serial.out_ttyUSB0.messIO(B.strFromCharCode(0x4B,0x0 0,
0x37,0x40),0.2);
while(true)
{
trez = B.Transport.Serial.out_ttyUSB0.messIO("");
if( !trez.length) break;
rez+=trez;
}
```

— `int messIO( XMLNodeObj req, string prt);` — відправлення запиту <req> до протоколу <prt> для здійснення сеансу зв'язку через транспорт за допомогою протоколу.

```
req = B.XMLNode("TCP");
req.setAttr("id","test").setAttr("reqTm",1).setAttr("node",1).setAttr("req
Try",2).setText(B.strFromCharCode(0x03,0x0 0,0x0 0,0x0 0,0x05));
 B.Transport.Sockets.out_testModBus.messIO(req,"ModBus");
test = Special.FLibB.strDec4Bin(req.text());
```

Об'єкт "Віджет" (this).

— `TCntrNodeObj ownerSess()` - одержати об'єкт сеансу даного віджета.

— `TCntrNodeObj ownerPage()` — одержати об'єкт батьківської сторінки даного віджета.

— `TCntrNodeObj ownerWdg(bool base = false)` - одержати батьківський віджет даного віджета. При вказівці <base> буде повернуті й об'єкти сторінок.

— `TCntrNodeObj wdgAdd(string wid, string wname, string parent)` — додавання віджета <wid> з ім'ям <wname> на основі бібліотечного віджета <parent>.

```
//Додати новий віджет на основі віджета текстового примітива
nw = this.wdgAdd("nw",   "Новий віджет",   "/wlb originals/wdg Text");
nw.attrSet("geom",   50).attrSet("geom",   50)
```

— `bool wdgDel(string wid)` — видалення віджета <wid>.

— `bool attrPresent(string attr)` — перевірка атрибута віджета <attr> на факт присутності.

— `ElTp attr(string attr)` — одержання значення атрибута віджета <attr>. Для відсутніх атрибутів вертається порожній рядок.

— `TCntrNodeObj attrSet(string attr, ElTp vl)` — установка атрибута віджета <attr> у значення <vl> . Вертається поточний об'єкт для конкатенації функцій установки.

— `string link(string attr, bool prm = false)` — одержання посилання для атрибута віджета <attr>. При установці <prm> запитується посилання для групи атрибутів (параметр), представлених зазначеним атрибутом.

— `string linkSet(string attr, string vl, bool prm)` — установка посилання для атрибута віджета <attr> . При установці <prm> здійснюється установка посилання для групи атрибутів (параметр), представлених зазначеним атрибутом.

```
//Установити посилання восьмого тренда параметром
this.linkSet("el8.name",   "prm:/LogicLev/experiment/Pi",   true);
```

4.2 Захист розробленого програмного забезпечення

Розроблене програмне забезпечення захистимо за допомогою національного стандарту захисту інформації на основі алгоритму шифрування/дешифрування ДСТУ 4145-2002 з використанням еліптичних кривих над двійковим розширеним полем Галуа. У системі шифрування/дешифрування як параметри розглядається еліптична крива E_p (a,b) і точка G на ній. Учасник В вибирає закритий ключ n і обчислює відкритий ключ $P_B = n \times G$. Щоб зашифрувати повідомлення P_m використовується відкритий ключ одержувача В P_B. Учасник А вибирає випадкове ціле позитивне число k і обчислює зашифроване повідомлення C_m, що є точкою на еліптичній кривій.

$$C_m = \{k \times G,\ P_m + k \times P_B\}. \quad (4.1)$$

Щоб дешифрувати повідомлення, учасник В множить першу координату точки на свій закритий ключ і віднімає результат від другої координати:

$$P_m + k \times P_B - n_B \times (k \times G) = P_m + k \times (n_B \times G) - n_B \times (k \times G) = P_m. \quad (4.2)$$

Учасник А зашифрував повідомлення P_m додаванням до нього $k \times P_B$. Ніхто не знає значення k, тому, хоча P_B і є відкритим ключем, ніхто не знає $k \times P_B$. Супротивнику для відновлення повідомлення доведеться обчислити k. Зробити це буде нелегко. Одержувач також не знає k, але йому як підказку посилається $k \times G$. Помноживши $k \times G$ на свій закритий ключ, одержувач одержить значення, що було додано відправником до незашифрованого повідомлення. Тим самим одержувач, не знаючи k, але маючи свій закритий ключ, може відновити незашифроване повідомлення.

Нехай задано просте число $p > 4$. Тоді еліптичною кривою E, визначеною над розширеним двійковим полем полем F_{2^m}, називається безліч пар чисел (x, y), $x, y \in F$, що задовольняють тотожності:

$$y^2 \equiv x^3 + a \cdot x + b (\mathrm{mod}\, 2^m), \quad (4.3)$$

де $4 \cdot a^3 + 27 \cdot b^2$ не рівно з нулю по модулю 2^m.

Інваріантом еліптичної кривої називається величина $J(E)$, що задовольняє тотожності:

$$J(E) \equiv 1728 \frac{4a^3}{4a^3 + 27b^2} (\mathrm{mod}\, 2^m). \quad (4.4)$$

Коефіцієнти a, b еліптичної кривої E, по відомому інваріанту $J(E)$, визначаються таким чином:

$$\begin{cases} a \equiv 3k (\mathrm{mod}\, 2^m) \\ b \equiv 2k (\mathrm{mod}\, 2^m) \end{cases} \text{де } k \equiv \frac{J(E)}{1728 - J(E)} (\mathrm{mod}\, 2^m), J(E) \neq 0 \text{ або } 1728. \quad (4.5)$$

Пари (x,y), що задовольняють тотожності (4.1), називаються точками еліптичної кривої E, x та y – відповідно x– та y-координатами точки.

Точки еліптичної кривої позначатимемо $Q(x,y)$ або просто Q. Дві точки еліптичної кривої рівні, якщо рівні їх відповідні x– і y-координати.

На безлічі всіх точок еліптичною кривою Е введемо операцію додавання, яку позначатимемо знаком "+". Для двох довільних точок $Q_1(x_1, y_1)$ та $Q_2(x_2, y_2)$

еліптичної кривої Е, розглянемо декілька варіантів.

Нехай координати точок Q_1 та Q_2 задовольняють умові $x_1 \neq x_2$. В цьому випадку їх сумою називатимемо точку $Q_3(x_3, y_3)$ координати якої визначаються порівняннями:

$$\begin{cases} x_3 \equiv \lambda^2 - x_1 - x_2 (\mathrm{mod}\, 2^m), \\ y_3 \equiv \lambda(x_1 - x_3) - y_1 (\mathrm{mod}\, 2^m), \end{cases} \text{де } \lambda \equiv \frac{y_2 - y_1}{x_2 - x_1} (\mathrm{mod}\, 2^m). \quad (4.6)$$

Якщо виконана рівність $x_1 = x_2$ та $y_1 = y_2 \neq 0$, то визначимо координати точки Q_3 таким чином:

$$\begin{cases} x_3 \equiv \lambda^2 - 2x_1 (\mathrm{mod}\, 2^m), \\ y_3 \equiv \lambda(x_1 - x_3) - y_1 (\mathrm{mod}\, 2^m), \end{cases} \text{де } \lambda \equiv \frac{3x_1^2 + a}{2y_1} (\mathrm{mod}\, 2^m). \quad (4.7)$$

У разі, коли виконана умова $x_1 = x_2$ та $y_1 = -y_2 (\mathrm{mod}\, p)$, суму точок Q_1 та Q_2 називатимемо нульовою точкою О, не визначаючи її х– і у-координати. В цьому випадку, точка Q_2 називається запереченням точки Q_1. Для нульової точки О виконана рівність:

$$Q + 0 = 0 + Q = Q, \quad (4.8)$$

де Q – довільна точка еліптичної кривої Е.

Щодо введеної операції складання безліч всіх точок еліптичною кривою Е, разом з нульовою точкою, утворюють кінцеву абельову (комутативну) групу порядку т, для якого виконана нерівність:

$$p + 1 - 2\sqrt{p} \leq m \leq p + 1 + 2\sqrt{p} \quad (4.9)$$

Точка Q називається точкою кратності k, або просто – кратною точкою еліптичної кривої E, якщо для деякої точки P виконана рівність:

$$Q = P + ... + P = kP \quad (4.10)$$

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ яке зображено на рисунку 5.1. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні розділи:

– Блок меню.

– Блок клавіш швидкого доступу.

– Блок відображення елементів системи диспетчерського керування та збору даних технологічного процесу.

– Блок контролю та параметрів.

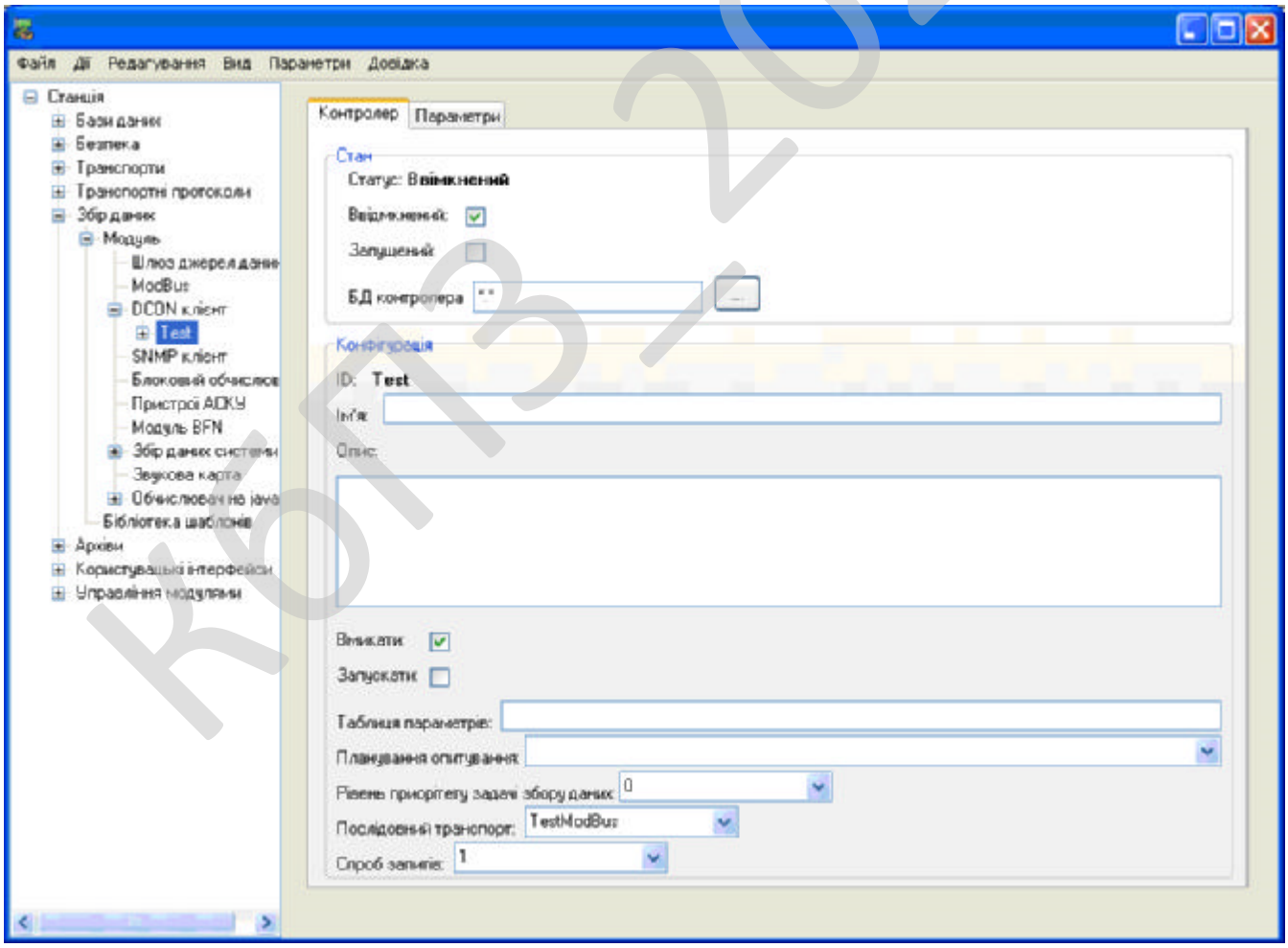

Рисунок 5.1 – Головне вікно програми

Блок меню складається з наступних елементів: Файл; Дії; Редагування; Вид; Параметри; Довідка. Блок клавіш швидкого доступу складається з наступних елементів: Зчитати дані; Записати дані; Рівень вверх; Рівень нижче; Відновити крок; На крок назад; Додати рівень; Видалити рівень; Записати в буфер; Вирізати; Копіювати; Перезапустити; Зупинити; Контролер; Параметри; Довідка.

Розроблена програма має дуже простий і інтуїтивно зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий.

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

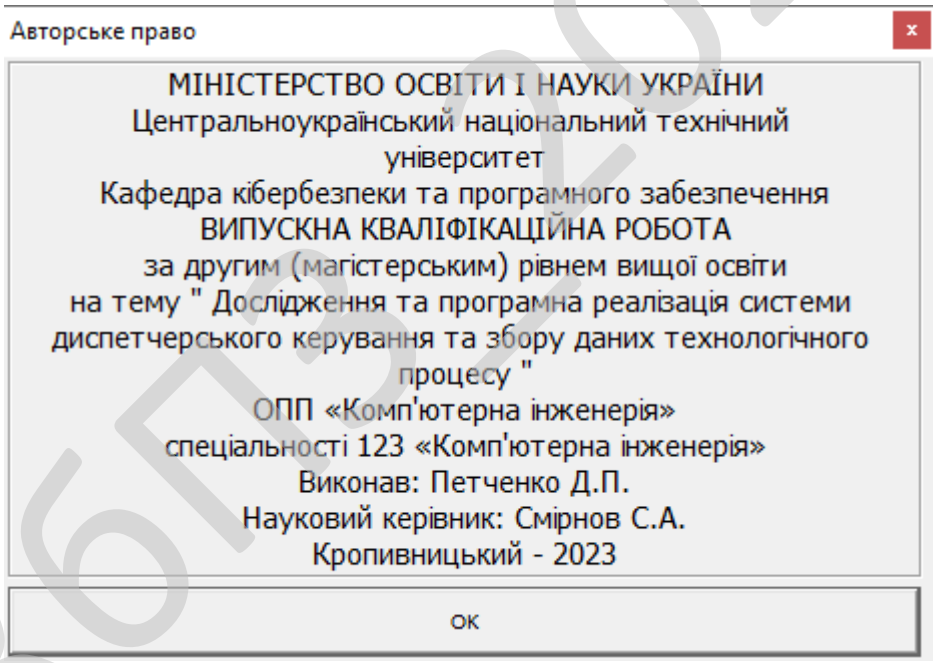

Рисунок 5.2 – Авторське право

Обрано умови розповсюдження – commercial software. Програмне забезпечення, створене комерційною організацією з метою отримання прибутку від його використання іншими, наприклад, шляхом продажу копій.

Найважливішою особливістю комерційний програмних продуктів є підтримка великих компаній, прямо зацікавлених у поширенні програм. Багато організацій надають виключно платну підтримку своїх продуктів, такий підхід, як правило, використовують організації надають відкриті вихідні коди. Для продуктів, що розповсюджуються на комерційній основі діють зазвичай безкоштовні служби підтримки, покликані збільшити рівень довіри у клієнтів і потенційних покупців.

Далеко не завжди, але як правило терміни критично важливих змін в комерційних продуктах значно менше, ніж у некомерційних проектів. Це пов'язано з тим, що над комерційним продуктом працюють цілі групи розробників і ця робота є їх основним заняттям. Розробникам-початківцям як правило доводиться шукати додаткові способи заробітку, і це збільшує час, що витрачається на доповнення і зміни програм. Так як основним рушійним фактором створення комерційного ПЗ є одержання прибутку, то комерційні програмні продукти першими заповнюють вільні ніші та пропонують варіанти вирішення завдань відразу по мірі виявлення вакууму в будь-якому секторі ринку.

Окремий вид комерційних програм, коли їх розробка оплачується безпосередньо замовником. Такі програми найчастіше позбавлені всіх переваг комерційних продуктів, оскільки мають обмежений бюджет, але більш адаптовані до вимог замовника, ніж аналоги.

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи диспетчерського керування та збору даних технологічного процесу.

Метою розробки є дослідження та програмна реалізація системи диспетчерського керування та збору даних технологічного процесу.

Об'єктом дослідження є процес диспетчерського керування та збору даних технологічного процесу.

Предметом дослідження є методи диспетчерського керування та збору даних технологічного процесу.

Методи дослідження базуються на методах Інтернету речей, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод диспетчерського керування та збору даних технологічного процесу.

– Розроблено вітчизняний продукт диспетчерського керування та збору даних технологічного процесу, 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 диспетчерського керування та збору даних технологічного процесу. 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	40
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	–	40000
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.

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

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)%

$$T_{РП}=0{,}3\cdot2{,}66\cdot9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)}\cdot45=76 \text{ люд/день}$$

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	76	Ф 7.1-7.4
Впровадження	13	Д13
Всього	117	–

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,

$$Ч = \frac{117 \cdot 1}{60\text{-}5} = 2{,}1 \text{ ставки}$$

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

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

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

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	5	450	7,5
Монітор	60	5	300	5
Клавіатура	30	5	150	2,5
Маніпулятор «мишка»	30	5	150	2,5
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	1	30	0,5
Кабельні господарства ЛОМ на 1 м.п.	2,5	180	450	7,5
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	31,16

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

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

$$Ф_{др}^{с} = \frac{31,16 \cdot 3}{1,2} = 78 \text{ год}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

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

$Ч_{ел}$=78/(60·8)=0,15 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)	0,5	18200	27300
Продакт-менеджер	0,25	12000	9000
Інженер-програміст	2,1	16000	100800
Інженер-електронщик	0,15	12000	5400
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	-	$Ф_{роб}$=214500

Розрахуємо середньоденну зарплату одного виконавця:

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

де Фроб – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{214500}{5 \cdot 60} = 715 \; грн.$$

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. Sу – питома площа на одне робоче місце, $м^2$; Цпл – вартість одного квадратного метра площі, грн.

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

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

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

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

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

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

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

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

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались за прайсом фірми Компбест за 03.11.23 – джерело https://compbest.com.ua/.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10947
Системний блок		7347
Процесор	Intel Core™ i3 10105F (BX8070110105F) 1200, 4 ядра, 8 потоків, 3.7 GHz, 4.4 GHz, TDP – 65 Вт, 14nm	-
Системна плата	ASUS PRO H410T/CSM сокет – 1200, SO-DIMM DDR4, LAN – 1 Гбит/с, DisplayPort, HDMI, 1 x M.2 Socket 3, 1 x M.2 Socket 1, 2 x Sata 6.0 Gb/s, Mini ITX	-
Відеокарта	Radeon R7 350 2Gb Afox (AFR7350-2048D5H4-V3) PCI-Express 3.0, 2 ГБ, GDDR5, 128 Bit	-
Жорсткий диск	SSD 2.5" 256GB Mibrand (MI2.5SSD/CA256GBST) 256 GB, 3D TLC NAND, 2.5", SATA III (6Gb/s)	-
Оперативна пам’ять	DDR4 8GB 2666 MHz eXceleram (E408266A)	-
Корпус	GRESSO GE-7525, 500W (120mm big fan), 2xIDE, full-ATX,БЖ 2xSATA, 1xFDD, Air Duct, 2xUSB 3.0, Mic+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	220
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) + USB 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	40000	10	4000
Разом	$К_р$= 1596075		$А_р$= 140140

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.

$$З_о = 715 \cdot 117/40 = 2090 \text{ грн}$$

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, %

$$З_д = 2090 \cdot 10 \cdot 0{,}01 = 209 \text{ грн}$$

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, %

$$С_{оц} = 0{,}01 \cdot 22(2090+209) = 506 \text{ грн}$$

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, %

$$Г_{осп} = 2090 \cdot 15 \cdot 0{,}01 = 314 \text{ грн}$$

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:

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

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

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 – 33,6 грн./шт., DVD-R box – 49,2 грн./шт.

$$З_{М2} = 49{,}2 \cdot 10 = 492 \text{ грн.}$$

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.

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

$$З_М = (105 + 492 + 1702)/40=57 \text{ грн.}$$

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, %

$$О_п = 2090 \cdot 15 \cdot 0{,}01=314 \text{ грн}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 40 прим.)

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

де Ар – загальна річна сума амортизаційних відрахувань, грн.

$$А_м = 140140 \cdot 3/(40 \cdot 12)=876 \text{ грн.}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції

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

$$C_п = 2090+209+506+314+57+314+876 = 4366 \text{ грн.}$$

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, %

$$П_р = 0{,}01 \cdot 55 \cdot 4366 = 2401 \text{ грн.}$$

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	$З_о$	2090
2. Додаткова зарплата виконавців	$З_д$	209
3. Відрахування на соціальні потреби	$С_{оц}$	506
4. Загальногосподарські витрати	$Г_{осп}$	314
5. Витрати на матеріали	$З_М$	57
6. Освоєння нових операційних систем, мов програмування	$О_п$	314
7. Амортизація основних фондів	$А_м$	876
8. Повна собівартість програмного забезпечення	$С_п$	4366
9. Плановий прибуток	$П_р$	2401
10.Ціна підприємства $Ц_п=С_п+П_р$	$Ц_п$	6767
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	1353,4
12.Відпускна ціна програмної продукції Ц = Цп + ПДВ	$Ц$	9168

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	–	9168
Всього капітальних витрат	–	9168

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 оплату праці	$З_р$	72146	35576
2. Витрати на електроенергію	$З_{ел}$	0	0
3. Витрати на амортизацію	$З_{ам}$	0	2292
Всього витрат за рік	$І$	72146	37868

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 профілактичних годин робіт зменшилася з 480 годин на рік до 250 годин на рік, тому витрати на технічне обслуговування зменшилися з

$$З_{р\,баз} = 480 \cdot 112 \cdot 1{,}1 \cdot 1{,}22 = 72146 \; грн.$$

до

$$З_{р\,нов} = 250 \cdot 112 \cdot 1{,}1 \cdot 1{,}22 = 37576 \; грн.$$

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

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

Витрати на електроенергію залишились без змін $З_{ел\,баз} = З_{ел\,нов}$

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	–	9168	–	2292
Всього відрахувань	-	–	9168	–	2292

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.

$$Ев = (6767 - 4366) \cdot 40 - (0,05 \cdot 1408000 + 0,5 \cdot 114885 +$$
$$+0,25 \cdot 33190 + 0,1 \cdot 40000) \cdot 3/12 = 61005 \; грн.$$

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.

$$T_в = \frac{1596075}{(6767 - 4366) \cdot 40 \cdot 12 / 3} = 4,15 \; років.$$

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

$$E_{сп} = (72146 - 37868) - 0,25 \cdot 9168 = 31986 \text{ грн.}$$

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

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

$$T_{сп} = \frac{9168}{72146 \ - \ 37868} = 0,3 \ року$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	40
2. Повна собівартість розробленої програми	Грн	4366
3. Ціна розробленої програми	Грн.	6767
4. Плановий прибуток від реалізації розробленої програми	Грн.	2401
5. Рентабельність програмної продукції	%	55
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1596075
7. Загальний прибуток від реалізації програмної продукції	Грн.	96040
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	61005
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Рік	4,15
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	9168
11.Величина економічного ефекту у користувача програмної продукції	Грн.	31986
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Рік	0,3

7.9 Висновки

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

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Впровадження комп'ютерних технологій принципово змінило характер праці різних категорій фахівців. Працівники, використовують комп'ютерну техніку, на своєму досвіді оцінили її величезні можливості. Одночасно виникла певна безтурботність при її експлуатації.

Характерною ознакою сучасного науково-технічного прогресу практично у всіх сферах діяльності людини є широке застосування комп'ютерних технологій, заснованих на використанні електронно-обчислювальних машин (ЕОМ). Сьогодення, а тим більше, майбутнє, вже важко уявити без комп'ютерів та іншої електронної техніки. Адже саме завдяки їм стала можливою швидка переробка величезних обсягів інформації, проведення необхідних розрахунків, виконання різних видів робіт, пов'язаних обробкою текстових та ілюстраційних зображень, організація оперативного отримання та передачі інформації, збереження її значних обсягів електронним способом.

Недотримання вимог безпеки призводить до того, що й через кілька днів роботи за комп'ютером співробітник починає відчувати певний дискомфорт: в нього виникає головний біль і різь у власних очах, з'являються почуття виснаження й дратівливість. В окремих людей порушується сон, погіршується зір, занедужують руки, шия, поперек тощо.

До недоліків умов праці користувачів комп’ютерної техніки можна віднести:

– недостатню площу і обсяг виробничого приміщення;

– недотримання вимог, мікроклімату на робочих місцях;

– низький рівень освітленості у приміщеннях і на робочих поверхнях апаратури;

– підвищений рівень низькочастотних магнітних полів від моніторів;

– порушення вимог організації робочих місць;

– недотримання вимог до режимам праці та відпочинку;

– надмірне виробничу навантаження працівників;

– відсутність навичок зниження впливу психоемоційного напруги.

Відповідно до ст.14 Закону «Про охорони праці» [3] на роботодавця покладено обов'язок забезпечити: безпеку працівників при експлуатації устаткування; застосування коштів індивідуальної захисту працівників; відповідні вимоги охорони праці, умови праці в кожному робоче місце; дотримання режиму праці та відпочинку працівників; навчання безпечним методам і прийомам виконання; інструктаж з охорони праці; організацію контролю над станом умов праці в робочих місць; проведення атестації робочих місць в умовах праці.

Максимально зменшити кількість шкідливих впливів на людину при високій продуктивності праці, створити комфортні умови для роботи людей – ось одна з головних задач охорони праці.

Електронно-обчислювальни машини (ЕОМ) та інше обладнання є джерелами небезпеки ураження електричним струмом. Так як робота програміста характеризується істотним зоровим навантаженням, то вимагає належного освітлення. У приміщенні, в якому працюють люди (у т.ч. програмісти) необхідно створити належний мікроклімат, праметри якого регламентуються, Державними санітарними правилами і нормами, зокрема ДСанПіН 3.3.2.007-98 [2].

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.

8.2 Аналіз умов праці

Приміщення розташовано на третьому поверсі п'ятиповерхового будинку. У приміщенні розташовано 3 робочих місць з комп'ютерами (далі ПК). Відповідно до норм «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98 [2] площа, що відводиться для робочого місця з комп'ютером повинна бути не менше 6 м^2, об'єм не менше 20 м^3. Розміри даного приміщень складають: довжина – 6 м, ширина – 4,5 м, висота – 3,5 м, тобто загальна фактична площа складає 27 м^2. Необхідна площа на 3 робочих місця із установленими ПК складає 18 м^2, що не перевищує фактичну. Обсяг кабінету на одного працюючого складає 31,5м^3, отже відповідає нормі ДСанПіН 3.3.2-007-98 – не менше 20 м^3.

При роботі з ПК людина може піддатися впливу шкідливих та небезпечних факторів. Під шкідливими виробничими факторами розуміють фактори, тривалий вплив яких викликає розвиток професійних захворювань. Небезпечні виробничі фактори – вплив яких на працюючого викликає травму, тобто пошкодження

організму. Шкідливі і небезпечні чинники, з якими стикається бібліограф при роботі з ПК, приведені в таблиці 8.1.

Таблиця 8.1 – Перелік шкідливих та небезпечних виробничих факторів

Найменування факторів	Можливі джерела їх виникнення	Характер дії
Небезпека ураження електричним струмом	Мережа живлення	Небезпечний
Пожежонебезпечність приміщень	Наявність матеріалів, що згорають і джерел запалення (електроапаратура)	Небезпечний та шкідливий
Іонізація повітря	Статична електрика випромінювання	Шкідливий
Підвищений рівень шуму	Шум створюється перетворювачем напруги ЕОМ, її технічною периферією, а також людьми, що працюють в приміщенні	Шкідливий
Несприятлива освітленість	Недостатнє штучне і природне освітлення	Шкідливий
Незадовільні параметри мікроклімату	Незадовільний стан системи опалення і вентиляції	Шкідливий
Психофізіологічні напруження	Монотонність праці, перенапруженість зорових аналізаторів, розумова напруженість, незручність і статичність пози	Шкідливий

По категорії вибухо – і пожежонебезпеки, згідно дане приміщення відноситься до категорії В – пожежонебезпечне, тому що присутні тверді матеріали, що горять, такі як дерев'яні столи, папір і інше. Виходячи з категорії пожежонебезпеки і поверховості будинку, ступінь вогнестійкості будівлі II. Згідно з ДБН В 1.1-7-2016 «Пожежна безпека об’єктів будівництва» [3] ЕОМ повинні розташовуватись в будівлі не менше ніж II ступню вогнестійкості.

По ступені небезпеки поразки людей електричним струмом відділ, згідно, класифікується як приміщення з підвищеною небезпекою, тому що не виключена можливість одночасного дотику людини до маючих з'єднання з землею конструкціям будинку, з одного боку, і до металевих корпусів електроустаткування, що можуть виявити під напругою – з іншого.

Для забезпечення вищевказаних оптимальних метеорологічних умов у помешканні передбачена система опалення (загальне парове) в холодному періоді, та вентиляція і кондиціювання в теплий період року, згідно ДБН2.5–67–2013 «Опалення, вентиляція та кондиціювання» [4]. При виконанні замірів параметрів мікроклімату, значення їх відповідали оптимальним та допустимим параметрам відповідно до ДСанПіН3.3.2.007 – 98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно – обчислювальних машин».

Припустимий рівень іонізації повітря помешкання відповідно до СН 21.52-80 повинен складати 1500 – 3000 один./м^3.

Нормування освітлення здійснюється відповідно до ДБН В.2.5 – 28 – 2006 «Природне та штучне освітлення». [5]

Відділ забезпечений комбінованим освітленням. В темний час доби передбачається загальне і/або місцеве рівномірне штучне, а в світлий – бокове одностороннє природне освітлення два віконних прорізи.

Одним з найбільш поширеніших чинників зовнішнього середовища, який несприятливо впливає на людину, є шум. Вплив шуму на організм людини

залежить від рівня звукового тиску, частотних характеристик, тривалості дії, а також індивідуальних особливостей людини.

При тривалій дії шуму у оператора ЕОМ виявляються симптоми утомленості, нервового збудження, що сприяють погіршенню працездатності і допущенні помилок при роботі. Для уникнення шкідливої дії шуму на організм працюючого, необхідне дотримання нормованих параметрів, які не повинні перевищувати допустимих величин. При роботі на комп'ютері рівень шуму не повинен перевищувати 50 дБА. Приміщення розташоване вікнами у двір і знаходиться далеко від проїжджої частини вулиці. Основними джерелами шуму в приміщенні є устаткування і люди. Розглянута кімната не призначена для прийому відвідувачів і тому в ній не спостерігається великого скупчення людей. Тому основним джерелом шуму є комп'ютерна техніка.

Джерелами шуму при роботі ЕОМ є механічні частини принтера, що рухаються, і вентилятори(Lпк = 35 дБА, , Lprn= 48 дБА.) При роботі вентиляційної системи, що забезпечує оптимальний температурний режим електронних блоків ЕОМ і вмонтована в задню панель, створюється аеродинамічний шум. Шум, створюваний працюючим комп'ютером, може бути охарактеризований як широко смуговий постійний з аперіодичним посиленням при роботі принтера. Час роботи ПЕОМ – 6 – 8 год. за добу; принтери працюють не більш 1,5-2 год. за добу.

При наявності великої кількості джерел шуму еквівалентне значення шуму $L_{ЭКВ}$, дБА розраховують по наступній формулі:

$$L_{ЭКВ} = 10 \cdot \lg\left(\frac{1}{T}\sum_{i=1}^{n}\left(t_i \cdot 10^{0.1 L_i}\right)\right) \quad (8.1),$$

де L_i – рівень шуму і-го джерела (пристрою),

t_i – час роботи і-го джерела (пристрою),

T – загальний час роботи,

n – кількість джерел шуму даного типу;

Для даного приміщення необхідні змінні складають:

Загальний час роботи – робітник день, тобто Т=8 годин.

Для фонового шуму (вентиляторів):

L_1 = 35 дБА, T_1 = 8 годин, n_1=15 (5х3);

Для лазерного принтера Lexmark Jet:

L_2 = 48дБА, Т2 = 2 години, n_2=1, для сканера L_3 = 46дБА, Т3 = 2 години.

Підставляємо отримані величини у формулу (8.1):

$$L_{экв} = 10 \cdot \lg\left(\frac{1}{8} \cdot \left(15 \cdot 8 \cdot 10^{0.1 \cdot 55} + 1 \cdot 2 \cdot 10^{0.1 \cdot 48} + 1 \cdot 2 \cdot 10^{0.1 \cdot 46}\right)\right) = 46{,}3 \text{ дБА}$$

Таким чином, еквівалентний рівень шуму в приміщенні за робітник день Lэкв = 46,3 дБА , тобто не перевищує норму 50 дБА.

8.3 Техніка безпеки та протипожежна профілактика

Відповідно ДБН В 1.1-7-2016 «Пожежна безпека об’єктів будівництва» будинок можна віднести до II групи по ступені вогнестійкості й до категорії Д по ступені пожежонебезпеки.

Від розподільного щита по праву й ліву сторони встановлені кондиціонери, зовнішня електропроводка, поміщена в ізольований кабель. Висота проводки становить 2,2м від рівня підлоги, її кріплення здійснюється за допомогою металевих власників. Біля кожного стола організований розподільний щит, розташований на текстолітовій пластинці, закріпленої на стіні на рівні 1м від підлоги. Усього до складу входять п'ять розеток і дві клеми заземлення. Всі обчислювальні машини з'єднані із клемами заземлення. Чотири з п'яти розеток забезпечують подачу напруги 220 V, а одна, забезпечує подачу напруги в 36 V. Про це є відповідні написи на кожному розподільному щиті.

Робота обслуговуючого персоналу полягає в інсталяції необхідного програмного забезпечення й наступному його використанні в діалоговому режимі роботи з ЕОМ. Іноді може виникати необхідність написання допоміжних програм для поліпшення роботи вузла або для зниження витрат. З погляду забезпечення умов праці й вимог техніки безпеки для роботи програміста необхідно наступне: достатнє висвітлення екрана дисплея й робочого місця; повна технічна справність

устаткування, його електробезпечність; достатня пожежобезпечність приміщення; оптимальний мікроклімат, що сприяє продуктивній роботі; відповідність робочого місця вимогам ергономіки. До небезпечних і шкідливих факторів, дії яких піддається програміст, можна віднести: можливість поразки електричним струмом, при електроні справності встаткування, порушенні заземлення або техніки безпеки; робота в мікрокліматі з неприпустимими параметрами; робота при недостатній освітленості екрана дисплея й робочого місця.

Відповідно НПАОП 40.1-1.21-98 “Правил безпечної експлуатації електроустановок споживачів” [6], приміщення можна віднести до приміщень без підвищеної небезпеки, оскільки це приміщення, сухе, з нормальною температурою й ізолюючими підлогами, що не має заземлених металоконструкцій.

Персональні ЕОМ можна віднести до першого класу електротехнічних виробів по способі захисту людини від поразки електричним струмом, оскільки їхні корпуси зроблені з ізолюючої пластмаси й кожен пристрій має заземлення. Відповідно правилам пристрою електроустановок ЕОМ можна віднести до електроустановок з робочою напругою до 1000 В.

Однієї з достовірних причин пожежі в приміщенні з обчислювальною технікою може бути коротке замикання, що спричиняє спалах електропроводки. Для його попередження вся обчислювальна техніка, а також інші електричні пристрої повинні бути обладнані плавкими запобіжниками, а на вході електромережі повинен бути передбачений автомат захисту. Не слід користуватися електричними подовжувачами й трійниками, що не мають сертифікатів відповідності вимогам безпеки.

Необхідно передбачити наявність у межах досяжності первинних засобів гасіння пожежі (вогнегасників) для локалізації вогню власними засобами до приїзду команди пожежної охорони. Повинен бути розроблений план екстреної евакуації персоналу при виникненні загоряння. Кількість евакуаційних виходів

повинне бути не менш двох. Допускається використання одного евакуаційного виходу, якщо відстань найбільш віддаленого робочого місця до цього виходу не перевищує 25 м.

8.4 Розробка заходів з охорони праці

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

Для зменшення шуму в приміщенні пропоную використовувати замість матричного принтера, що створює багато шуму, більш тихий – лазерний принтер.

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

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

Як міри по зниженню шуму можна запропонувати:

– облицювання стелі і стін звукопоглинаючим матеріалом (знижують шум на 6-8 до);

– екранування робочого місця (постановкою перегородок, діафрагм);

– установка в комп'ютерних приміщеннях устаткування, що робить мінімальний шум;

– раціональне планування приміщення.

З точки зору забезпечення пожежної безпеки до цих заходів можна віднести наявність схеми евакуації з приміщення вузла, у випадку пожежі, повішену на вхідні двері.

8.5 Висновки до розділу

У даному розділі магістерської роботи були виконано аналіз умов праці користувачів ПК, які працюють у зазначеному приміщенні Проведено перевірка організації робочого місця із відповідними замірами параметрів мікроклімату, освітлення, рівня шуму та розрахунком рівня шуму.

Розроблені заходи щодо поліпшення умов праці дотримання техніки безпеки та проведення протипожежної профілактики дозволить створити умови, які будуть забезпечувати більш комфортну роботу.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи диспетчерського керування та збору даних технологічного процесу.

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 диспетчерського керування та збору даних технологічного процесу.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем диспетчерського керування та збору даних технологічного процесу.

– Досліджена система диспетчерського керування та збору даних технологічного процесу.

– На основі отриманих результатів досліджень створена програмна реалізація системи диспетчерського керування та збору даних технологічного процесу.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання диспетчерського керування та збору даних технологічного процесу.

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 Builder C++. 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 ДСТУ 4145-2002.

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 31986 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,3 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Петченко Д.П. Дослідження та програмна реалізація системи диспетчерського керування та збору даних технологічного процесу // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2023.
2. Peter Hoddie, Lizzie Prader «IoT Development for ESP32 and ESP8266 with JavaScript: A Practical Guide to XS and the Moddable SDK» ISBN-13 (pbk): 978-1-4842-5069-3 ISBN-13 (electronic): 978-1-4842-5070-9
3. STM32CubeMX for STM32 configuration and initialization C code generation. User manual. June 2022. 397 p.
4. 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.
5. Jack Ganssle and Michael Barr. 2003. Embedded Systems Dictionary. CMP Books.
6. 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.
7. Greg Dunko, Joydeep Misra, Josh Robertson, Tom Snyder “A reference guide to the Internet of Things” / 2017 Bridgera LLC, RIoT.
8. Donald Norris “Programming with STM32. Getting started with the Nucleo Board and C/C++” 416 p. 2018.
9. Neil Kolban “Kolban’s book on ESP32”. Texas, USA. 951 p.
10. 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.

11. 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.

12. 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.

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

14. 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.

15. 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.

16. 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.

17. 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.

18. 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.

19. 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.

20. 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.

21. 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.

22. 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.

23. 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.

24. 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.

25. 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.

26. 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.

27. 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.

28. 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.

29. 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.

30. 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.

31. 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.

32. 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.

33. 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.

34. 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.

35. 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

36. 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.

37. 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.

38. 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.

39. 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.

40. 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.

41. 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.

42. 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).

43. 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

44. 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.

45. 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.

46. 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.

47. 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.

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

49. 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.

50. 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.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи диспетчерського керування та збору даних технологічного процесу.

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. № 35-13 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 диспетчерського керування та збору даних технологічного процесу.

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 диспетчерського керування та збору даних технологічного процесу;

– 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 Builder C++.

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 техніка безпеки та протипожежна профілактика.

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 – 103 аркушів.

10 Етапи розробки

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

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

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

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

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

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 13.12.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

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

_____ Смірнов С.А.

Дослідження та програмна реалізація системи диспетчерського керування та збору даних технологічного процесу

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 68

Літера: РП

Кропивницький – 2023 року

resalloc.cpp – Розподіл ресурсів у системі

```
// Петченко Даниіл Павлович
//SCADA_system системний файл: resalloc.cpp


#include "errno.h"

#include "tsys.h"
#include "resalloc.h"

using namespace WSCADA;

//*****************************************
//* Об'єкт ресурсу                        *
//*****************************************
Res::Res( )
{
#if !__GLIBC_PREREQ(2,4)
    wThr = 0;
#endif
    if(pthread_rwlock_init(&rwc,NULL))
      throw TError("ResAlloc",_("Помилка відкриття семафору!"));
}

Res::~Res( )
{
    pthread_rwlock_wrlock(&rwc);
    pthread_rwlock_destroy(&rwc);
}

void Res::resRequestW( unsigned short tm )
{
    int rez = 0;
#if !__GLIBC_PREREQ(2,4)
    //EDEADLK імітація
    if(wThr && wThr == pthread_self()) rez == EDEADLK;
    else
#endif
    if(!tm) rez = pthread_rwlock_wrlock(&rwc);
    else
    {
      timespec wtm;
      clock_gettime(CLOCK_REALTIME,&wtm);
      wtm.tv_nsec += 1000000*(tm%1000);
      wtm.tv_sec += tm/1000 + wtm.tv_nsec/1000000000; wtm.tv_nsec =
wtm.tv_nsec%1000000000;
      rez = pthread_rwlock_timedwrlock(&rwc,&wtm);
    }
    if(rez == EDEADLK) throw TError(10,"ResAlloc",_("Ресурс пробує заблокувати
потік!"));
    else if(tm && rez == ETIMEDOUT) throw TError("ResAlloc",_("Ресурс не
відповідає!"));
#if !__GLIBC_PREREQ(2,4)
    wThr = pthread_self();
#endif
}

bool Res::resTryW( )
{
    int rez = pthread_rwlock_trywrlock(&rwc);
    if(rez == EBUSY) return false;
    else if(rez == EDEADLK) throw TError(10,"ResAlloc",_("Ресурс пробує
заблокувати потік!"));
    return true;
}
```

```
void Res::resRequestR( unsigned short tm )
{
    int rez = 0;
#if !__GLIBC_PREREQ(2,4)
    //EDEADLK імітація
    if(wThr && wThr == pthread_self()) rez == EDEADLK;
    else
#endif
    if(!tm) rez = pthread_rwlock_rdlock(&rwc);
    else
    {
      timespec wtm;
      clock_gettime(CLOCK_REALTIME,&wtm);
      wtm.tv_nsec += 1000000*(tm%1000);
      wtm.tv_sec += tm/1000 + wtm.tv_nsec/1000000000; wtm.tv_nsec =
wtm.tv_nsec%1000000000;
      rez = pthread_rwlock_timedrdlock(&rwc,&wtm);
    }
    if(rez == EDEADLK) throw TError(10,"ResAlloc",_("Ресурс пробує заблокувати
потік!"));
    else if(tm && rez == ETIMEDOUT) throw TError("ResAlloc",_("Ресурс не
відповідає!"));
}

bool Res::resTryR( )
{
    int rez = pthread_rwlock_tryrdlock(&rwc);
    if(rez == EBUSY) return false;
    else if(rez == EDEADLK) throw TError(10,"ResAlloc",_("Ресурс пробує
заблокувати потік!"));
    return true;
}

void Res::resRelease( )
{
    pthread_rwlock_unlock(&rwc);
#if !__GLIBC_PREREQ(2,4)
    if(wThr == pthread_self()) wThr = 0;
#endif
}

//*****************************************
//* Автоматичний розподіл ресурсів *
//*****************************************
ResAlloc::ResAlloc( Res &rid ) : mId(rid), mAlloc(false)
{

}

ResAlloc::ResAlloc( Res &rid, bool write, unsigned short tm ) : mId(rid),
mAlloc(false)
{
    request(write, tm);
}

ResAlloc::~ResAlloc( )
{
    if(mAlloc) release();
}

void ResAlloc::request( bool write, unsigned short tm )
{
    if(mAlloc) release();
    mAlloc = false;
    try
    {
      if(write) mId.resRequestW(tm);
      else mId.resRequestR(tm);
```

```
      mAlloc = true;
    }catch(TError err) { if(err.cod!=10) throw; }
}

void ResAlloc::release()
{
    if(!mAlloc) return;
    mId.resRelease( );
    mAlloc = false;
}

//*****************************************
//* Рядок + ресурс для                    *
//*****************************************
ResString::ResString( const string &vl )
{
    pthread_mutex_init(&mRes, NULL);
    setVal(vl);
}

ResString::~ResString( )
{
    pthread_mutex_lock(&mRes);
    pthread_mutex_destroy(&mRes);
}

size_t ResString::size( )     { return getVal().size(); }

bool   ResString::empty( )    { return getVal().empty(); }

void ResString::setVal( const string &vl )
{
    pthread_mutex_lock(&mRes);
    str = vl;
    pthread_mutex_unlock(&mRes);
}

string ResString::getVal( )
{
    string rez;
    pthread_mutex_lock(&mRes);
    rez = str;
    pthread_mutex_unlock(&mRes);
    return rez;
}

ResString &ResString::operator=( const string &val )
{
    setVal(val);
    return *this;
}
```

tarchives.cpp – Робота з архівом

```
// Петченко Даниіл Павлович
//SCADA_system системний файл: tarchives.cpp

#include <unistd.h>
#include <getopt.h>
#include <signal.h>
#include <sys/time.h>
#include <string.h>
#include <algorithm>

#include "tsys.h"
#include "tarchives.h"

#define BUF_SIZE_DEF 500
#define BUF_SIZE_MAX 100000

using namespace WSCADA;

//************************************************
//* Підсистема архівування                       *
//************************************************

//************************************************
//* TArchiveS                                    *
//************************************************
TArchiveS::TArchiveS( ) :
    TSubSYS(SARH_ID,"Archives",true), elMess(""), elVal(""), elAval(""),
bufErr(0), mMessPer(10), prcStMess(false),
    headBuf(0), headLstread(0), mValPer(1000), mValPrior(10), prcStVal(false),
endrunReqVal(false)
{
    mAval = grpAdd("va_");

    //> архіватор повідомлення у структурі БД
    elMess.fldAdd( new TFld("ID",_("ID"),TFld::String,TCfg::Key,"20") );
    elMess.fldAdd( new TFld("MODUL",_(" Ім’я модуля
(плагіна)"),TFld::String,TCfg::Key,"20") );
    elMess.fldAdd( new
TFld("NAME",_("Ім’я"),TFld::String,TCfg::TransltText,"50") );
    elMess.fldAdd( new
TFld("DESCR",_("Дескриптор"),TFld::String,TCfg::TransltText,"200") );
    elMess.fldAdd( new TFld("START",_("Початок архіву"),TFld::Boolean,0,"1") );
    elMess.fldAdd( new TFld("CATEG",_("Категорії
повідомлень"),TFld::String,0,"100") );
    elMess.fldAdd( new TFld("LEVEL",_("Рівні
повідомлень"),TFld::Integer,0,"1","","0;7") );
    elMess.fldAdd( new TFld("ADDR",_("Адреса"),TFld::String,0,"100") );

    //> Значення архіватора у структурі БД
    elVal.fldAdd( new TFld("ID",_("ID"),TFld::String,TCfg::Key,"20") );
    elVal.fldAdd( new TFld("MODUL",_("Ім’я модуля
(плагіна)"),TFld::String,TCfg::Key,"20") );
    elVal.fldAdd( new TFld("NAME",_(" Ім’я
"),TFld::String,TCfg::TransltText,"50") );
    elVal.fldAdd( new
TFld("DESCR",_("Дескриптор"),TFld::String,TCfg::TransltText,"200") );
    elVal.fldAdd( new TFld("START",_("Початок архіву"),TFld::Boolean,0,"1","0")
);
    elVal.fldAdd( new TFld("ADDR",_("Адреса"),TFld::String,0,"50") );
    elVal.fldAdd( new TFld("V_PER",_("Value period
(sec)"),TFld::Real,0,"12.6","1","0;1000000") );
    elVal.fldAdd( new TFld("A_PER",_("Період архівації
(sec)"),TFld::Integer,0,"4","60","0;1000") );

    //> Значення архіву у структурі БД
    elAval.fldAdd( new TFld("ID",_("ID"),TFld::String,TCfg::Key,"20") );
```

```
    elAval.fldAdd( new TFld("NAME",_(" Ім'я
"),TFld::String,TCfg::TransltText,"50") );
    elAval.fldAdd( new
TFld("DESCR",_("Дескриптор"),TFld::String,TCfg::TransltText,"200") );
    elAval.fldAdd( new TFld("START",_("Початок архіву"),TFld::Boolean,0,"1","0")
);
    elAval.fldAdd( new TFld("SrcMode",_("Режим джерела"),TFld::Integer,0,"1") );
    elAval.fldAdd( new TFld("Source",_("Джерело"),TFld::String,0,"100") );
    elAval.fldAdd( new TFld("VTYPE",_("Тип значення"),TFld::Integer,0,"1") );
    elAval.fldAdd( new TFld("BPER",_("Період буферізації
(sec)"),TFld::Real,0,"9.6","1","0;10000") );
    elAval.fldAdd( new TFld("BSIZE",_("Розмір
буферу()"),TFld::Integer,0,"6","100","0;1000000") );
    elAval.fldAdd( new TFld("BHGRD",_("Буфер у режимі грід-
системи"),TFld::Boolean,0,"1","1") );
    elAval.fldAdd( new TFld("BHRES",_("Значення буфера останім
часом"),TFld::Boolean,0,"1","0") );
    elAval.fldAdd( new TFld("ArchS",_("Процес
архівування"),TFld::String,0,"500") );

    setMessBufLen( BUF_SIZE_DEF );

    //> Створення повідомлення часу архівування
    struct sigevent sigev;
    memset(&sigev,0,sizeof(sigev));
    sigev.sigev_notify = SIGEV_THREAD;
    sigev.sigev_value.sival_ptr = this;
    sigev.sigev_notify_function = ArhMessTask;
    sigev.sigev_notify_attributes = NULL;
    timer_create(CLOCK_REALTIME,&sigev,&tmIdMess);
}

TArchiveS::~TArchiveS(  )
{
    //> Поввідомлення про закінчення архівування
    timer_delete(tmIdMess);

    //> Перерривання архівування
    if(prcStVal) SYS->taskDestroy(nodePath('.',true)+".vals", &endrunReqVal);

    //> Визволення усіх ресурсів
    nodeDelAll();
}

int TArchiveS::valPeriod( )         { return vmax(1,mValPer); }

void TArchiveS::setValPrior( int ivl )    { mValPrior = vmax(-1,vmin(99,ivl));
modif(); }

void TArchiveS::load_( )
{
    //> Завантажуємо параметри з командного рядка
    int next_opt;
    const char *short_opt="h";
    struct option long_opt[] =
    {
      {"help"    ,0,NULL,'h'},
      {NULL      ,0,NULL,0  }
    };

    optind=0,opterr=0;
    do
    {
      next_opt=getopt_long(SYS->argc,(char * const *)SYS-
>argv,short_opt,long_opt,NULL);
      switch(next_opt)
      {
          case 'h': fprintf(stdout,"%s",optDescr().c_str()); break;
          case -1 : break;
```

```
      }
    } while(next_opt != -1);

    //> Завантажуємо параметри
    setMessBufLen(
atoi(TBDS::genDBGet(nodePath()+"MessBufSize",TSYS::int2str(messBufLen())).c_str(
)) );
    setMessPeriod(
atoi(TBDS::genDBGet(nodePath()+"MessPeriod",TSYS::int2str(mMessPer)).c_str()) );
    setValPeriod(
atoi(TBDS::genDBGet(nodePath()+"ValPeriod",TSYS::int2str(mValPer)).c_str()) );
    setValPrior(
atoi(TBDS::genDBGet(nodePath()+"ValPriority",TSYS::int2str(mValPrior)).c_str())
);

    //> LidDB
    //>> Повідомлення завантаження архіватора
    string id,type;
    map<string, bool>   itReg;
    try
    {
      TConfig c_el(&elMess);
      c_el.cfgViewAll(false);
      vector<string> db_ls;

      //>> Шукаємо у БД і створюємо новий архів
      SYS->db().at().dbList(db_ls,true);
      db_ls.push_back("<cfg>");
      for(unsigned i_db = 0; i_db < db_ls.size(); i_db++)
          for(int fld_cnt=0; SYS-
>db().at().dataSeek(db_ls[i_db]+"."+subId()+"_mess_proc",nodePath()+subId()+"_me
ss_proc",fld_cnt++,c_el); )
          {
            id = c_el.cfg("ID").getS();
            type = c_el.cfg("MODUL").getS();
            if(modPresent(type) && !at(type).at().messPresent(id))
                at(type).at().messAdd(id,(db_ls[i_db]==SYS-
>workDB())?"*.*":db_ls[i_db]);
            itReg[type+"."+id] = true;
          }

      //>>> Перевіряємо для видалення з БД
        if(!SYS->selDB().empty())
        {
            vector<string> m_ls;
            modList(m_ls);
            for(unsigned i_m = 0; i_m < m_ls.size(); i_m++)
            {
                at(m_ls[i_m]).at().messList(db_ls);
                for(unsigned i_it = 0; i_it < db_ls.size(); i_it++)
                    if(itReg.find(m_ls[i_m]+"."+db_ls[i_it]) == itReg.end() &&
SYS->chkSelDB(at(m_ls[i_m]).at().messAt(db_ls[i_it]).at().DB()))
                        at(m_ls[i_m]).at().messDel(db_ls[i_it]);
            }
        }
    }catch( TError err )
    {
      mess_err(err.cat.c_str(),"%s",err.mess.c_str());
      mess_err(nodePath().c_str(),_("Повідомлення вомилки завантаження
архіватора."));
    }

    //>> Завантажуємо значення архіватора
    try
    {
      TConfig c_el(&elVal);
      c_el.cfgViewAll(false);
      vector<string> db_ls;
      itReg.clear();
```

```
      //>> Шукаємо у БД та створюємо новий архів
      SYS->db().at().dbList(db_ls,true);
      db_ls.push_back("<cfg>");
      for(unsigned i_db = 0; i_db < db_ls.size(); i_db++)
          for(int fld_cnt=0; SYS-
>db().at().dataSeek(db_ls[i_db]+"."+subId()+"_val_proc",nodePath()+subId()+"_val
_proc",fld_cnt++,c_el); )
          {
            id = c_el.cfg("ID").getS();
            type = c_el.cfg("MODUL").getS();
            if(modPresent(type) && !at(type).at().valPresent(id))
                at(type).at().valAdd(id,(db_ls[i_db]==SYS-
>workDB())?"*.*":db_ls[i_db]);
            itReg[type+"."+id] = true;
          }

      //>>> Перевіряємо для видалення з БД
        if(!SYS->selDB().empty())
        {
            vector<string> m_ls;
            modList(m_ls);
            for(unsigned i_m = 0; i_m < m_ls.size(); i_m++)
            {
                at(m_ls[i_m]).at().valList(db_ls);
                for(unsigned i_it = 0; i_it < db_ls.size(); i_it++)
                    if(itReg.find(m_ls[i_m]+"."+db_ls[i_it]) == itReg.end() &&
SYS->chkSelDB(at(m_ls[i_m]).at().valAt(db_ls[i_it]).at().DB()))
                        at(m_ls[i_m]).at().valDel(db_ls[i_it]);
            }
        }
    }catch( TError err )
    {
      mess_err(err.cat.c_str(),"%s",err.mess.c_str());
      mess_err(nodePath().c_str(),_("Помилка завантаження значення
архіватора."));
    }

    //>> Завантажуємо значення архіватора
    try
    {
      TConfig c_el(&elAval);
      c_el.cfgViewAll(false);
      vector<string> db_ls;
      itReg.clear();

      //>> Шукаємо у БД та створюємо новий архів
      SYS->db().at().dbList(db_ls,true);
      db_ls.push_back("<cfg>");
      for(unsigned i_db = 0; i_db < db_ls.size(); i_db++)
          for(int fld_cnt=0; SYS-
>db().at().dataSeek(db_ls[i_db]+"."+subId()+"_val",nodePath()+subId()+"_val",fld
_cnt++,c_el); )
          {
              id = c_el.cfg("ID").getS();
              if(!valPresent(id)) valAdd(id,(db_ls[i_db]==SYS-
>workDB())?"*.*":db_ls[i_db]);
              itReg[id] = true;
          }

      //>>> Перевіряємо для видалення з БД
        if(!SYS->selDB().empty())
        {
            valList(db_ls);
            for(unsigned i_it = 0; i_it < db_ls.size(); i_it++)
                if(itReg.find(db_ls[i_it]) == itReg.end() && SYS-
>chkSelDB(valAt(db_ls[i_it]).at().DB()))
                    valDel(db_ls[i_it]);
        }
```

```
    }catch(TError err)
    {
      mess_err(err.cat.c_str(),"%s",err.mess.c_str());
      mess_err(nodePath().c_str(),_("Помилка завантаження значення
архіватора."));
    }
}

void TArchiveS::save_( )
{
    vector<string> t_lst, o_lst;

    //> Зберігаємо параметри
    TBDS::genDBSet(nodePath()+"MessBufSize",TSYS::int2str(messBufLen()));
    TBDS::genDBSet(nodePath()+"MessPeriod",TSYS::int2str(messPeriod()));
    TBDS::genDBSet(nodePath()+"ValPeriod",TSYS::int2str(valPeriod()));
    TBDS::genDBSet(nodePath()+"ValPriority",TSYS::int2str(valPrior()));
}

void TArchiveS::valAdd( const string &iid, const string &idb )
{
    if( valPresent(iid) ) return;
    chldAdd(mAval,new TVArchive(iid,idb,&aValE()));
}

string TArchiveS::optDescr(  )
{
    char buf[STR_BUF_LEN];
    snprintf(buf,sizeof(buf),_(
      "======================== Підсистема \"Архів\" Опції
==================\n"
      "------------ Параметри частини '%s' у кофігураційний файл -----------\n"
      "MessBufSize   <items>       Повідомлення розміру буферу.\n"
      "MessPeriod    <sec>         Повідомлення періоду архівування.\n"
      "ValPeriod     <msec>        Значення періоду архівування.\n"
      "ValPriority   <level>      Значення завдання пріорітетного рівня.\n"
      "MaxReqMess    <items>       Повідомлення максимального запиту.\n"
      "MaxReqVals    <items>       Значення  максимального запиту.\n\n"
      ),nodePath().c_str());

    return buf;
}

void TArchiveS::subStart( )
{
    mess_info(nodePath().c_str(),_("Запускаємо підсистеми."));

    SubStarting = true;

    vector<string> t_lst, o_lst;

    modList(t_lst);
    for(unsigned i_t = 0; i_t < t_lst.size(); i_t++)
    {
      AutoHD<TTipArchivator> mod = modAt(t_lst[i_t]);

      //> Повідомлення про початок роботи архіватора
      mod.at().messList(o_lst);
      for(unsigned i_o = 0; i_o < o_lst.size(); i_o++)
      {
          AutoHD<TMArchivator> mess = mod.at().messAt(o_lst[i_o]);
          if( /*!mess.at().startStat() &&*/ mess.at().toStart() )
            try{ mess.at().start(); }
            catch(TError err)
            {
                mess_err(err.cat.c_str(),"%s",err.mess.c_str());
                mess_err(nodePath().c_str(),_("Повідомлення про помилку роботи
архіватора."),o_lst[i_o].c_str());
            }
```

```
      }
      //> Значення початку роботи архіватора
      mod.at().valList(o_lst);
      for(unsigned i_o = 0; i_o < o_lst.size(); i_o++)
      {
          AutoHD<TVArchivator> val = mod.at().valAt(o_lst[i_o]);
          if( /*!val.at().startStat() &&*/ val.at().toStart() )
            try{ val.at().start(); }
            catch(TError err)
            {
                mess_err(err.cat.c_str(),"%s",err.mess.c_str());
                mess_err(nodePath().c_str(),_("Помилка початку роботи
архіватора."),val.at().workId().c_str());
            }
      }
    }

    //> Значення початку роботи архіватора
    valList(o_lst);
    for(unsigned i_o = 0; i_o < o_lst.size(); i_o++)
    {
      AutoHD<TVArchive> aval = valAt(o_lst[i_o]);
      if( /*!aval.at().startStat() &&*/ aval.at().toStart() )
          try{ aval.at().start(); }
          catch(TError err)
          {
            mess_err(err.cat.c_str(),"%s",err.mess.c_str());
            mess_err(nodePath().c_str(),_("Помилка початку роботи
архіватора"),o_lst[i_o].c_str());
          }
    }

    //> Повідомлення про початок роботи інтервального таймера
    struct itimerspec itval;
    itval.it_interval.tv_sec = itval.it_value.tv_sec = messPeriod();
    itval.it_interval.tv_nsec = itval.it_value.tv_nsec = 0;
    timer_settime(tmIdMess, 0, &itval, NULL);

    //> Значення початку роботи завдань
    if(!prcStVal) SYS->taskCreate(nodePath('.',true)+".vals", valPrior(),
TArchiveS::ArhValTask, this);

    TSubSYS::subStart( );

    SubStarting = false;
}

void TArchiveS::subStop( )
{
    mess_info(nodePath().c_str(),_("Зупинка підсистеми."));

    TSubSYS::subStop( );

    vector<string> t_lst, o_lst;

    //> Зупинка інтервального таймера для періодичних потоків, для створення
повідомлення архівації структур;
    itval.it_interval.tv_sec = itval.it_interval.tv_nsec =
      itval.it_value.tv_sec = itval.it_value.tv_nsec = 0;
    timer_settime(tmIdMess, 0, &itval, NULL);
    if(TSYS::eventWait( prcStMess, false, nodePath()+"mess_stop",10))
      throw TError(nodePath().c_str(),_("Архівація повідомлень потоків не може
бути зупинена!"));

    //> Значення зупинки роботи завдань
    if(prcStVal) SYS->taskDestroy(nodePath('.',true)+".vals", &endrunReqVal);

    //> Виклик останього повідомлення архіватора
    sigval obj; obj.sival_ptr = this;
```

```
    ArhMessTask(obj);

    //> Зупинка архіватора
    modList(t_lst);
    for(unsigned i_t = 0; i_t < t_lst.size(); i_t++)
    {
      AutoHD<TTipArchivator> mod = modAt(t_lst[i_t]);
      //Значення зупинки архіватора
      mod.at().valList(o_lst);
      for(unsigned i_o = 0; i_o < o_lst.size(); i_o++)
      {
          AutoHD<TVArchivator> val = mod.at().valAt(o_lst[i_o]);
          if( val.at().startStat() )
            try{ val.at().stop(); }
            catch(TError err)
            {
                mess_err(err.cat.c_str(),"%s",err.mess.c_str());
                mess_err(nodePath().c_str(),_("Значення архіватора '%s' stop
error."),o_lst[i_o].c_str());
            }
      }
      // Повідомлення про зупинку архіватора
      mod.at().messList(o_lst);
      for(unsigned i_o = 0; i_o < o_lst.size(); i_o++)
      {
          AutoHD<TMArchivator> mess = mod.at().messAt(o_lst[i_o]);
          if( mess.at().startStat() )
            try{ mess.at().stop(); }
            catch(TError err)
            {
                mess_err(err.cat.c_str(),"%s",err.mess.c_str());
                mess_err(nodePath().c_str(),_("Повідомлення про помилку зупинки
архіватора"),o_lst[i_o].c_str());
            }
      }
    }

    //> Значення зупинки архіватора
    valList(o_lst);
    for(unsigned i_o = 0; i_o < o_lst.size(); i_o++)
    {
      AutoHD<TVArchive> aval = valAt(o_lst[i_o]);
      if( aval.at().startStat() )
          try{ aval.at().stop(); }
          catch(TError err)
          {
            mess_err(err.cat.c_str(),"%s",err.mess.c_str());
            mess_err(nodePath().c_str(),_("Значення помилки зупинки
архіватора"),o_lst[i_o].c_str());
          }
    }
}

void TArchiveS::messPut( time_t tm, int utm, const string &categ, int8_t level,
const string &mess )
{
    //> Відправляємо повідомлення у буфер
    ResAlloc res(mRes,true);
    mBuf[headBuf].time  = tm;
    mBuf[headBuf].utime = utm;
    mBuf[headBuf].categ = categ;
    mBuf[headBuf].level = (TMess::Type)abs(level);
    mBuf[headBuf].mess  = mess;
    if( ++headBuf >= mBuf.size() ) headBuf = 0;
    //> Перервіряємо, чи це не повідомлення архіватора
    if( headBuf == headLstread )
    {
      if( !(bufErr&0x01) )
      {
```

```
            bufErr |= 0x01;
            res.release();
            mess_err(nodePath().c_str(),_("Буфер заповнений. Останнє
повідомлення!"));
            res.request(true);
        }
        if( ++headLstread >= mBuf.size() ) headLstread = 0;
    }
    //> Перевіряємо швидкість заповнення буфера.
    else if( headBuf-headLstread > messBufLen( )/2 )
    {
        if( !(bufErr&0x02) )
        {
            bufErr |= 0x02;
            res.release();
            mess_warning(nodePath().c_str(),_("Повідомлення про т, що швидкість
заповнення буфера дуже висока!"));
            res.request(true);
        }
    }
    else bufErr = 0;

    //> Обробка тривог. Для рівня менше 0 тривоги встановлено
    map<string,TMess::SRec>::iterator p;
    if( level < 0 ) mAlarms[categ] =
TMess::SRec(tm,utm,categ,(TMess::Type)abs(level),mess);
    else if( (p=mAlarms.find(categ)) != mAlarms.end() ) mAlarms.erase(p);
}

void TArchiveS::messPut( const vector<TMess::SRec> &recs )
{
    for(unsigned i_r = 0; i_r < recs.size(); i_r++)
      messPut(recs[i_r].time,recs[i_r].utime,recs[i_r].categ,recs[i_r].level,rec
s[i_r].mess);
}

void TArchiveS::messGet( time_t b_tm, time_t e_tm, vector<TMess::SRec> & recs,
const string &category, int8_t level, const string &arch, time_t upTo )
{
    recs.clear();

    ResAlloc res(mRes,false);
    if(!upTo) upTo = time(NULL)+STD_INTERF_TM;
    TRegExp re(category, "p");

    //> Отримуємо записи з буфера
    unsigned i_buf = headBuf;
    while(level >= 0 && (!arch.size() || arch==BUF_ARCH_NM) && time(NULL) <
upTo)
    {
      if(mBuf[i_buf].time >= b_tm && mBuf[i_buf].time != 0 && mBuf[i_buf].time
<= e_tm &&
          mBuf[i_buf].level >= level && re.test(mBuf[i_buf].categ))
            recs.push_back(mBuf[i_buf]);
      if(++i_buf >= mBuf.size()) i_buf = 0;
      if(i_buf == headBuf) break;
    }

    //> Отримуємо записи з архівів
    vector<string> t_lst, o_lst;
    modList(t_lst);
    for(unsigned i_t = 0; level >= 0 && i_t < t_lst.size(); i_t++)
    {
      at(t_lst[i_t]).at().messList(o_lst);
      for(unsigned i_o = 0; i_o < o_lst.size() && time(NULL) < upTo; i_o++)
      {
          AutoHD<TMArchivator> archtor = at(t_lst[i_t]).at().messAt(o_lst[i_o]);
          if(archtor.at().startStat() && (!arch.size() ||
arch==archtor.at().workId()))
```

```
            archtor.at().get(b_tm,e_tm,recs,category,level);
      }
    }

    //> Запит процесу тривоги
    if(level < 0)
    {
      vector< pair<int64_t,TMess::SRec* > > mb;
      for(map<string,TMess::SRec>::iterator p = mAlarms.begin(); p !=
mAlarms.end() && time(NULL) < upTo; p++)
        if((p->second.time >= b_tm || b_tm == e_tm) && p->second.time <= e_tm
&&
            p->second.level >= abs(level) && re.test(p->second.categ))
          mb.push_back(pair<int64_t,TMess::SRec* >(FTM(p->second),&p-
>second));
      sort(mb.begin(),mb.end());
      for(unsigned i_b = 0; i_b < mb.size(); i_b++)
recs.push_back(*mb[i_b].second);
    }
}

time_t TArchiveS::messBeg( const string &arch )
{
    time_t rez = 0;
    ResAlloc res(mRes,false);
    if(arch.empty() || arch == BUF_ARCH_NM)
    {
      unsigned i_buf = headBuf;
      while(!arch.size() || arch == BUF_ARCH_NM)
      {
          rez = rez ? vmin(rez,mBuf[i_buf].time) : mBuf[i_buf].time;
          if(++i_buf >= mBuf.size()) i_buf = 0;
          if(i_buf == headBuf) break;
      }
      if( !arch.empty() ) return rez;
    }

    //- Отримуємо записи з архівів -
    vector<string> t_lst, o_lst;
    modList(t_lst);
    AutoHD<TMArchivator> archtor;
    for(unsigned i_t = 0; i_t < t_lst.size(); i_t++)
    {
      at(t_lst[i_t]).at().messList(o_lst);
      for(unsigned i_o = 0; i_o < o_lst.size(); i_o++)
      {
          archtor = at(t_lst[i_t]).at().messAt(o_lst[i_o]);
          if(archtor.at().startStat() && (!arch.size() ||
arch==archtor.at().workId()))
            rez = rez ? vmin(rez,archtor.at().begin()) : archtor.at().begin();
      }
    }

    return rez;
}

time_t TArchiveS::messEnd( const string &arch )
{
    time_t rez = 0;
    ResAlloc res(mRes,false);
    if(arch.empty() || arch == BUF_ARCH_NM)
    {
      unsigned i_buf = headBuf;
      while(!arch.size() || arch == BUF_ARCH_NM)
      {
          rez = rez ? vmax(rez,mBuf[i_buf].time) : mBuf[i_buf].time;
          if(++i_buf >= mBuf.size()) i_buf = 0;
          if(i_buf == headBuf) break;
      }
```

```
      if(!arch.empty()) return rez;
    }

    //> Отримуємо записи з архівів
    vector<string> t_lst, o_lst;
    modList(t_lst);
    AutoHD<TMArchivator> archtor;
    for(unsigned i_t = 0; i_t < t_lst.size(); i_t++)
    {
      at(t_lst[i_t]).at().messList(o_lst);
      for(unsigned i_o = 0; i_o < o_lst.size(); i_o++)
      {
          archtor = at(t_lst[i_t]).at().messAt(o_lst[i_o]);
          if(archtor.at().startStat() && (!arch.size() ||
arch==archtor.at().workId()))
            rez = rez ? vmax(rez,archtor.at().end()) : archtor.at().end();
      }
    }

    return rez;
}

void TArchiveS::setMessBufLen(unsigned len)
{
    ResAlloc res(mRes,true);
    len = vmin(BUF_SIZE_MAX,vmax(BUF_SIZE_DEF,len));
    while(mBuf.size() > len)
    {
      mBuf.erase(mBuf.begin() + headBuf);
      if(headBuf >= mBuf.size()) headBuf = 0;
      if(headLstread >= mBuf.size())     headLstread = mBuf.size()-1;
    }
    while(mBuf.size() < len) mBuf.insert(mBuf.begin() + headBuf, TMess::SRec());
    modif();
}

void TArchiveS::setActValArch( const string &id, bool val )
{
    unsigned i_arch;

    ResAlloc res(vRes,true);
    for( i_arch = 0; i_arch < actUpSrc.size(); i_arch++ )
      if( actUpSrc[i_arch].at().id() == id ) break;

    if( val && i_arch >= actUpSrc.size() )
      actUpSrc.push_back(valAt(id));
    if( !val && i_arch < actUpSrc.size() )
      actUpSrc.erase(actUpSrc.begin()+i_arch);
}

void TArchiveS::setMessPeriod( int ivl )
{
    mMessPer = ivl;
    modif();

    if( subStartStat( ) )
    {
      struct itimerspec itval;
      itval.it_interval.tv_sec = itval.it_value.tv_sec = mMessPer;
      itval.it_interval.tv_nsec = itval.it_value.tv_nsec = 0;
      timer_settime(tmIdMess, 0, &itval, NULL);
    }
}

void TArchiveS::ArhMessTask( union sigval obj )
{
    TArchiveS &arh = *(TArchiveS *)obj.sival_ptr;
    if( arh.prcStMess )  return;
    arh.prcStMess = true;
```

```
    //> Читаємо повідомлення з буфера
    if( arh.headLstread != arh.headBuf )
      try
      {
          ResAlloc res(arh.mRes,false);

          //>> Беремо нове повідомлення
          unsigned new_headLstread = arh.headBuf;
          unsigned i_m = arh.headLstread;
          vector<TMess::SRec> o_mess;
          while( i_m != new_headLstread )
          {
            o_mess.push_back(arh.mBuf[i_m]);
            if( ++i_m >= arh.mBuf.size() ) i_m = 0;
          }
          arh.headLstread = new_headLstread;

          res.release();

          //>> Архівуємо
          vector<string> t_lst, o_lst;
          arh.modList(t_lst);
          for(unsigned i_t = 0; i_t < t_lst.size(); i_t++)
          {
            arh.at(t_lst[i_t]).at().messList(o_lst);
            for(unsigned i_o = 0; i_o < o_lst.size(); i_o++)
                if(arh.at(t_lst[i_t]).at().messAt(o_lst[i_o]).at().startStat())
                  arh.at(t_lst[i_t]).at().messAt(o_lst[i_o]).at().put(o_mess);
          }
      }
      catch(TError err)
      {
          mess_err(err.cat.c_str(),"%s",err.mess.c_str());
          mess_err(arh.nodePath().c_str(),_("Помилка читання повідомлення з
буфера."));
      }

    arh.prcStMess = false;
}

void *TArchiveS::ArhValTask( void *param )
{
    TArchiveS &arh = *(TArchiveS *)param;
    arh.endrunReqVal = false;
    arh.prcStVal = true;

    while( !arh.endrunReqVal )
    {
      int64_t work_tm = SYS->curTime();

      arh.vRes.resRequestR( );
      for(unsigned i_arh = 0; i_arh < arh.actUpSrc.size(); i_arh++)
          try
          {
            if( work_tm/arh.actUpSrc[i_arh].at().period() >
arh.actUpSrc[i_arh].at().end()/arh.actUpSrc[i_arh].at().period() )
                arh.actUpSrc[i_arh].at().getActiveData();
          }
          catch(TError err)
          { mess_err(err.cat.c_str(),"%s",err.mess.c_str()); }
      arh.vRes.resRelease( );

      TSYS::taskSleep((int64_t)arh.valPeriod()*1000000);
    }

    arh.prcStVal = false;

    return NULL;
```

```
}

TVariant TArchiveS::objFuncCall( const string &iid, vector<TVariant> &prms,
const string &user )
{
    // Array messGet(int btm, int etm, string cat = "", int lev = 0, string arch
= ""); - запит системних повідомлень, для часу з <btm>
    //       до <etm> для категорії <cat>, рівня <lev> та архіву <arch>
    //  btm - час початку
    //  etm - час закінчення
    //  cat - категорія повідомлення
    //  lev - рівні повідомлень
    //  arch - архівація повідомлення
    if( iid == "messGet" && prms.size() >= 2 )
    {
      vector<TMess::SRec> recs;
      messGet( prms[0].getI(), prms[1].getI(), recs, ((prms.size()>=3) ?
prms[2].getS() : string("")),
          ((prms.size()>=4) ? prms[3].getI() : 0), ((prms.size()>=5) ?
prms[4].getS() : string("")) );
      TArrayObj *rez = new TArrayObj();
      for(unsigned i_m = 0; i_m < recs.size(); i_m++)
      {
          TVarObj *am = new TVarObj();
          am->propSet("tm",(int)recs[i_m].time);
          am->propSet("utm",recs[i_m].utime);
          am->propSet("categ",recs[i_m].categ);
          am->propSet("level",recs[i_m].level);
          am->propSet("mess",recs[i_m].mess);
          rez->propSet(TSYS::int2str(i_m),am);
      }
      return rez;
    }

    return TCntrNode::objFuncCall(iid,prms,user);
}

void TArchiveS::cntrCmdProc( XMLNode *opt )
{
    string a_path = opt->attr("path");
    //> Сервіс управління процесами
    if(a_path == "/serv/mess")           //Повідомлення доступу
    {
      if(ctrChkNode(opt,"info",RWRWRW,"root",SARH_ID,SEC_RD))
      //Інформаційні повідомлення
      {
          string arch = opt->attr("arch");
          opt->setAttr("end",TSYS::uint2str(messEnd(arch)));
          opt->setAttr("beg",TSYS::uint2str(messBeg(arch)));
      }
      else if(ctrChkNode(opt,"get",RWRWRW,"root",SARH_ID,SEC_RD))//Запит
значення дати
      {
          time_t tm      = strtoul(opt->attr("tm").c_str(),0,10);
          time_t tm_grnd = strtoul(opt->attr("tm_grnd").c_str(),0,10);
          string arch    = opt->attr("arch");
          string cat     = opt->attr("cat");
          int    lev     = atoi(opt->attr("lev").c_str());
          vector<TMess::SRec> rez;
          messGet( tm_grnd, tm, rez, cat, (TMess::Type)lev, arch );
          for(unsigned i_r = 0; i_r < rez.size(); i_r++)
            opt->childAdd("el")->
                setAttr("time",TSYS::uint2str(rez[i_r].time))->
                setAttr("utime",TSYS::uint2str(rez[i_r].utime))->
                setAttr("cat",rez[i_r].categ)->
                setAttr("lev",TSYS::int2str(rez[i_r].level))->
                setText(rez[i_r].mess);
      }
      return;
```

```
    }

    //> Беремо сторінку інформації
    if(opt->name() == "info")
    {
      TSubSYS::cntrCmdProc(opt);
      ctrMkNode("grp",opt,-1,"/br/va_",_("Архів
значень"),RWRWR_,"root",SARH_ID,2,"idm","1","idSz","20");
      if(ctrMkNode("area",opt,1,"/m_arch",_("Архів
повідомлень"),R_R_R_,"root",SARH_ID))
      {
          ctrMkNode("fld",opt,-1,"/m_arch/size",_("Повідомлення розміру
буферу"),RWRWR_,"root",SARH_ID,2,"tp","dec","min",TSYS::int2str(BUF_SIZE_DEF).c_
str());
          ctrMkNode("fld",opt,-1,"/m_arch/per",_("Період архівування
(s)"),RWRWR_,"root",SARH_ID,1,"tp","dec");
          if(ctrMkNode("area",opt,-1,"/m_arch/view",_("Перегляд
повідомлень"),R_R___,"root",SARH_ID))
          {
            ctrMkNode("fld",opt,-
1,"/m_arch/view/tm",_("Time"),RWRW__,"root",SARH_ID,1,"tp","time");
            ctrMkNode("fld",opt,-1,"/m_arch/view/size",_("Розмір
(s)"),RWRW__,"root",SARH_ID,1,"tp","dec");
            ctrMkNode("fld",opt,-1,"/m_arch/view/cat",_("Зразок
категорії"),RWRW__,"root",SARH_ID,2,"tp","str","help",
                _("Шаблон категорії повідомлень або регулярне вираження.\n"
                        "Використовуйте тимчасові символи для групового
виділення:\n  '*' - будь-які підрядки;\n  '?' - будь-які символи.\n"
                      "Регулярне вираження складається з символів'/'
(/mod_(System|LogicLev)/)."));
            ctrMkNode("fld",opt,-
1,"/m_arch/view/lvl",_("Рівень"),RWRW__,"root",SARH_ID,4,"tp","dec","min","-
7","max","7",
                "help",_("Отримуємо повідомлення для рівня більшого і
дорівнюючого цьому."));
            ctrMkNode("fld",opt,-
1,"/m_arch/view/archtor",_("Архіватор"),RWRW__,"root",SARH_ID,4,"tp","str","dest
","select","select","/m_arch/lstAMess",
                "help",_("Архівація повідомлення.\n Не встановлюємо архіватор
для процесів у буфері та інших архіваторів.\Встановлюємо '<buffer>' для процесів
у буфері ."));
            if(ctrMkNode("table",opt,-
1,"/m_arch/view/mess",_("Messages"),R_R___,"root",SARH_ID))
            {
                ctrMkNode("list",opt,-
1,"/m_arch/view/mess/0",_("Time"),R_R___,"root",SARH_ID,1,"tp","time");
                ctrMkNode("list",opt,-
1,"/m_arch/view/mess/0a",_("mcsec"),R_R___,"root",SARH_ID,1,"tp","dec");
                ctrMkNode("list",opt,-
1,"/m_arch/view/mess/1",_("Category"),R_R___,"root",SARH_ID,1,"tp","str");
                ctrMkNode("list",opt,-
1,"/m_arch/view/mess/2",_("Lev."),R_R___,"root",SARH_ID,1,"tp","dec");
                ctrMkNode("list",opt,-
1,"/m_arch/view/mess/3",_("Message"),R_R___,"root",SARH_ID,1,"tp","str");
            }
          }
      }
      if(ctrMkNode("area",opt,2,"/v_arch",_("Архів
значеньь"),R_R_R_,"root",SARH_ID))
      {
          ctrMkNode("fld",opt,-1,"/v_arch/per",_("Беремо дані періоду
(ms)"),RWRWR_,"root",SARH_ID,1,"tp","dec");
          ctrMkNode("fld",opt,-1,"/v_arch/prior",_("Беремо дані завдання
пріорітетного рівня "),RWRWR_,"root",SARH_ID,1,"tp","dec");
          ctrMkNode("fld",opt,-
1,"/v_arch/nmb",_("Number"),R_R_R_,"root",SARH_ID,1,"tp","str");
          ctrMkNode("list",opt,-1,"/v_arch/archs",_("Архів
значень"),RWRWR_,"root",SARH_ID,5,"tp","br","idm","1","s_com","add,del","br_pref
","va_","idSz","20");
```

```
      }
      ctrMkNode("fld",opt,-1,"/help/g_help",_("Опції
допомоги"),R_R___,"root",SARH_ID,3,"tp","str","cols","90","rows","10");
      return;
    }

    //> Процес управління на сторінці
    if(a_path == "/m_arch/per")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt-
>setText(TSYS::int2str(messPeriod()));
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))
      setMessPeriod(atoi(opt->text().c_str()));
    }
    else if(a_path == "/m_arch/size")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt-
>setText(TSYS::int2str(messBufLen()));
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))
      setMessBufLen(atoi(opt->text().c_str()));
    }
    else if(a_path == "/m_arch/view/tm")
    {
      if(ctrChkNode(opt,"get",RWRW__,"root",SARH_ID,SEC_RD))
      {
          opt->setText(TBDS::genDBGet(nodePath()+"messTm","0",opt-
>attr("user")));
          if( !atoi(opt->text().c_str()) )    opt-
>setText(TSYS::int2str(time(NULL)));
      }
      if(ctrChkNode(opt,"set",RWRW__,"root",SARH_ID,SEC_WR))
          TBDS::genDBSet(nodePath()+"messTm",(atoi(opt-
>text().c_str())>=time(NULL))?"0":opt->text(),opt->attr("user"));
    }
    else if(a_path == "/m_arch/view/size")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt-
>setText(TBDS::genDBGet(nodePath()+"messSize","60",opt->attr("user")));
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))
      TBDS::genDBSet(nodePath()+"messSize",opt->text(),opt->attr("user"));
    }
    else if(a_path == "/m_arch/view/cat")
    {
      if(ctrChkNode(opt,"get",RWRW__,"root",SARH_ID,SEC_RD))     opt-
>setText(TBDS::genDBGet(nodePath()+"messCat","",opt->attr("user")));
      if(ctrChkNode(opt,"set",RWRW__,"root",SARH_ID,SEC_WR))
      TBDS::genDBSet(nodePath()+"messCat",opt->text(),opt->attr("user"));
    }
    else if(a_path == "/m_arch/view/lvl")
    {
      if(ctrChkNode(opt,"get",RWRW__,"root",SARH_ID,SEC_RD))     opt-
>setText(TBDS::genDBGet(nodePath()+"messLev","0",opt->attr("user")));
      if(ctrChkNode(opt,"set",RWRW__,"root",SARH_ID,SEC_WR))
      TBDS::genDBSet(nodePath()+"messLev",opt->text(),opt->attr("user"));
    }
    else if(a_path == "/m_arch/view/archtor")
    {
      if(ctrChkNode(opt,"get",RWRW__,"root",SARH_ID,SEC_RD))     opt-
>setText(TBDS::genDBGet(nodePath()+"messArch","",opt->attr("user")));
      if(ctrChkNode(opt,"set",RWRW__,"root",SARH_ID,SEC_WR))
      TBDS::genDBSet(nodePath()+"messArch",opt->text(),opt->attr("user"));
    }
    else if(a_path == "/m_arch/lstAMess" && ctrChkNode(opt,"get",R_R___))
    {
      opt->childAdd("el")->setText("");
      opt->childAdd("el")->setText(BUF_ARCH_NM);
      vector<string> lsm, lsa;
      modList(lsm);
      for(unsigned i_m = 0; i_m < lsm.size(); i_m++)
```

```
      {
          at(lsm[i_m]).at().messList(lsa);
          for(unsigned i_a = 0; i_a < lsa.size(); i_a++)
            opt->childAdd("el")->setText(lsm[i_m]+"."+lsa[i_a]);
      }
    }
    else if(a_path == "/m_arch/view/mess" &&
ctrChkNode(opt,"get",R_R___,"root",SARH_ID))
    {
      vector<TMess::SRec> rec;
      time_t gtm = atoi(TBDS::genDBGet(nodePath()+"messTm","0",opt-
>attr("user")).c_str());
      if( !gtm ) gtm = time(NULL);
      int gsz = atoi(TBDS::genDBGet(nodePath()+"messSize","60",opt-
>attr("user")).c_str());
      messGet( gtm-gsz, gtm, rec,
             TBDS::genDBGet(nodePath()+"messCat","",opt->attr("user")),
             atoi(TBDS::genDBGet(nodePath()+"messLev","0",opt-
>attr("user")).c_str()),
             TBDS::genDBGet(nodePath()+"messArch","",opt->attr("user")) );

      XMLNode *n_tm     = ctrMkNode("list",opt,-
1,"/m_arch/view/mess/0","",R_R___,"root",SARH_ID);
      XMLNode *n_tmu    = ctrMkNode("list",opt,-
1,"/m_arch/view/mess/0a","",R_R___,"root",SARH_ID);
      XMLNode *n_cat    = ctrMkNode("list",opt,-
1,"/m_arch/view/mess/1","",R_R___,"root",SARH_ID);
      XMLNode *n_lvl    = ctrMkNode("list",opt,-
1,"/m_arch/view/mess/2","",R_R___,"root",SARH_ID);
      XMLNode *n_mess   = ctrMkNode("list",opt,-
1,"/m_arch/view/mess/3","",R_R___,"root",SARH_ID);
      for(int i_rec = rec.size()-1; i_rec >= 0; i_rec--)
      {
          if(n_tm)      n_tm->childAdd("el")-
>setText(TSYS::int2str(rec[i_rec].time));
          if(n_tmu)     n_tmu->childAdd("el")-
>setText(TSYS::int2str(rec[i_rec].utime));
          if(n_cat)     n_cat->childAdd("el")->setText(rec[i_rec].categ);
          if(n_lvl)     n_lvl->childAdd("el")-
>setText(TSYS::int2str(rec[i_rec].level));
          if(n_mess)    n_mess->childAdd("el")->setText(rec[i_rec].mess);
      }
    }
    else if(a_path == "/v_arch/per")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt-
>setText(TSYS::int2str(valPeriod()));
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))
      setValPeriod(atoi(opt->text().c_str()));
    }
    else if(a_path == "/v_arch/prior")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt-
>setText(TSYS::int2str(valPrior()));
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))
      setValPrior(atoi(opt->text().c_str()));
    }
    else if(a_path == "/v_arch/nmb" && ctrChkNode(opt))
    {
      vector<string> list;
      valList(list);
      unsigned e_c = 0;
      for(unsigned i_a = 0; i_a < list.size(); i_a++)
          if(valAt(list[i_a]).at().startStat()) e_c++;
      opt->setText(TSYS::strMess(_("All: %d; Enabled: %d"),list.size(),e_c));
    }
    else if(a_path == "/br/va_" || a_path == "/v_arch/archs")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))
```

```
      {
          vector<string> list;
          valList(list);
          for(unsigned i_a=0; i_a < list.size(); i_a++)
            opt->childAdd("el")->setAttr("id",list[i_a])-
>setText(valAt(list[i_a]).at().name());
      }
      if(ctrChkNode(opt,"add",RWRWR_,"root",SARH_ID,SEC_WR))
      {
          string vid = TSYS::strEncode(opt->attr("id"),TSYS::oscdID);
          valAdd(vid); valAt(vid).at().setName(opt->text());
      }
      if(ctrChkNode(opt,"del",RWRWR_,"root",SARH_ID,SEC_WR))
      chldDel(mAval,opt->attr("id"),-1,1);
    }
    else if(a_path == "/help/g_help" &&
ctrChkNode(opt,"get",R_R___,"root",SARH_ID))    opt->setText(optDescr());
    else TSubSYS::cntrCmdProc(opt);
}

//************************************************
//* TTipArchivator                                *
//************************************************
TTipArchivator::TTipArchivator( const string &id ) : TModule(id)
{
    mVal = grpAdd("val_");
    mMess = grpAdd("mess_");
}

TTipArchivator::~TTipArchivator()
{
    nodeDelAll();
}

TArchiveS &TTipArchivator::owner()
{
    return (TArchiveS &)TModule::owner();
}

void TTipArchivator::messAdd(const string &name, const string &idb )
{
    chldAdd(mMess, AMess(name,idb));
}

void TTipArchivator::valAdd( const string &iid, const string &idb )
{
    chldAdd(mVal, AVal(iid,idb));
}

void TTipArchivator::cntrCmdProc( XMLNode *opt )
{
    //> Беремо сторінку інформації
    if(opt->name() == "info")
    {
      TModule::cntrCmdProc(opt);
      ctrMkNode("area",opt,0,"/arch",_("Архіватори"));
      ctrMkNode("grp",opt,-1,"/br/mess_",_("Повідомлення
архіватора"),RWRWR_,"root",SARH_ID,2,"idm","1","idSz","20");
      ctrMkNode("grp",opt,-1,"/br/val_",_("Значення
архіватора"),RWRWR_,"root",SARH_ID,2,"idm","1","idSz","20");
      ctrMkNode("list",opt,-1,"/arch/mess",_("Повідомлення
архіваторів"),RWRWR_,"root",SARH_ID,5,"tp","br","idm","1","s_com","add,del","br_
pref","mess_","idSz","20");
      ctrMkNode("list",opt,-1,"/arch/val",_("Значення
архіваторів"),RWRWR_,"root",SARH_ID,5,"tp","br","idm","1","s_com","add,del","br_
pref","val_","idSz","20");
      return;
    }
    //> Процес управління на сторінці
```

```
    string a_path = opt->attr("path");
    if(a_path == "/br/mess_" || a_path == "/arch/mess")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))
      {
          vector<string> list;
          messList(list);
          for( unsigned i_a=0; i_a < list.size(); i_a++ )
            opt->childAdd("el")->setAttr("id",list[i_a])-
>setText(messAt(list[i_a]).at().name());
      }
      if(ctrChkNode(opt,"add",RWRWR_,"root",SARH_ID,SEC_WR))
      {
          string vid = TSYS::strEncode(opt->attr("id"),TSYS::oscdID);
          messAdd(vid); messAt(vid).at().setName(opt->text());
      }
      if(ctrChkNode(opt,"del",RWRWR_,"root",SARH_ID,SEC_WR))     messDel(opt-
>attr("id"),true);
    }
    else if(a_path == "/br/val_" || a_path == "/arch/val")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))
      {
          vector<string> list;
          valList(list);
          for(unsigned i_a=0; i_a < list.size(); i_a++)
            opt->childAdd("el")->setAttr("id",list[i_a])-
>setText(valAt(list[i_a]).at().name());
      }
      if(ctrChkNode(opt,"add",RWRWR_,"root",SARH_ID,SEC_WR))
      {
          string vid = TSYS::strEncode(opt->attr("id"),TSYS::oscdID);
          valAdd(vid); valAt(vid).at().setName(opt->text());
      }
      if(ctrChkNode(opt,"del",RWRWR_,"root",SARH_ID,SEC_WR))     valDel(opt-
>attr("id"),true);
    }
    else TModule::cntrCmdProc(opt);
}

//************************************************
//* Повідомлення архіватора                      *
//************************************************

//************************************************
//* TMArchivator                                 *
//************************************************
TMArchivator::TMArchivator(const string &iid, const string &idb, TElem *cf_el) :
    TConfig( cf_el ), run_st(false),
    m_id(cfg("ID").getSd()), m_name(cfg("NAME").getSd()),
m_dscr(cfg("DESCR").getSd()), m_addr(cfg("ADDR").getSd()),
    m_cat_o(cfg("CATEG").getSd()), m_start(cfg("START").getBd()),
m_level(cfg("LEVEL").getId()), m_db(idb)
{
    m_id = iid;
}

TCntrNode &TMArchivator::operator=( TCntrNode &node )
{
    TMArchivator *src_n = dynamic_cast<TMArchivator*>(&node);
    if( !src_n ) return *this;

    //> Конфігурація копіювання
    string tid = id();
    *(TConfig*)this = *(TConfig*)src_n;
    cfg("MODUL").setS(owner().modId());
    m_id = tid;
    m_db = src_n->m_db;
```

```
    if( src_n->startStat() && toStart() && !startStat() )
        start( );

    return *this;
}

void TMArchivator::postEnable( int flag )
{
    cfg("MODUL").setS(owner().modId());
}

void TMArchivator::preDisable( int flag )
{
    if( startStat() )   stop( );
}

void TMArchivator::postDisable(int flag)
{
    try
    {
      if( flag )
          SYS->db().at().dataDel(fullDB(),SYS-
>archive().at().nodePath()+tbl(),*this,true);
    }catch(TError err)
    { mess_warning(err.cat.c_str(),"%s",err.mess.c_str()); }
}

TTipArchivator &TMArchivator::owner( )    { return *(TTipArchivator*)nodePrev();
}

string TMArchivator::workId( )          { return owner().modId()+"."+id(); }

string TMArchivator::name()         { return (m_name.size())?m_name:m_id; }

string TMArchivator::tbl( )         { return
owner().owner().subId()+"_mess_proc"; }

void TMArchivator::load_( )
{
    if( !SYS->chkSelDB(DB()) ) return;
    SYS->db().at().dataGet(fullDB(),SYS->archive().at().nodePath()+tbl(),*this);
}

void TMArchivator::save_( )
{
    SYS->db().at().dataSet(fullDB(),SYS->archive().at().nodePath()+tbl(),*this);
}

void TMArchivator::categ( vector<string> &list )
{
    list.clear();
    string c_vl;
    for( int i_off = 0; (c_vl=TSYS::strSepParse(m_cat_o,0,';',&i_off)).size(); )
      list.push_back(c_vl);
}

bool TMArchivator::chkMessOK( const string &icateg, TMess::Type ilvl )
{
    vector<string> cat_ls;

    categ(cat_ls);

    if(ilvl >= level())
      for(unsigned i_cat = 0; i_cat < cat_ls.size(); i_cat++)
          if(TRegExp(cat_ls[i_cat], "p").test(icateg))
            return true;
   return false;
}
```

```
TVariant TMArchivator::objFuncCall( const string &iid, vector<TVariant> &prms,
const string &user )
{
    // bool status() - беремо статус архівування.
    if(iid == "status") return startStat();
    // int end() - беремо дані  архівування, час закінчення.
    if(iid == "end") return (int)end();
    // int begin() - беремо дані  архівування, час початку.
    if(iid == "begin") return (int)begin();

    //> Виклик функцій конфігурації
    TVariant cfRez = objFunc(iid, prms, user);
    if(!cfRez.isNull()) return cfRez;

    return TCntrNode::objFuncCall(iid, prms, user);
}

void TMArchivator::cntrCmdProc( XMLNode *opt )
{
    //> Беремо сторінку інформації
    if(opt->name() == "info")
    {
      TCntrNode::cntrCmdProc(opt);
      ctrMkNode("WSCADA_cntr",opt,-1,"/",_("Повідомлення архіватора:
")+name(),RWRWR_,"root",SARH_ID);
      if(ctrMkNode("area",opt,-1,"/prm",_("Архіватор")))
      {
          if(ctrMkNode("area",opt,-1,"/prm/st",_("Стан")))
          {
            ctrMkNode("fld",opt,-
1,"/prm/st/st",_("Running"),RWRWR_,"root",SARH_ID,1,"tp","bool");
            ctrMkNode("fld",opt,-1,"/prm/st/db",_("Архіватор
БД"),RWRWR_,"root","root",4,

"tp","str","dest","select","select","/db/list","help",TMess::labDB());
            ctrMkNode("fld",opt,-
1,"/prm/st/end",_("End"),R_R_R_,"root","root",1,"tp","time");
            ctrMkNode("fld",opt,-
1,"/prm/st/beg",_("Begin"),R_R_R_,"root","root",1,"tp","time");
          }
          if(ctrMkNode("area",opt,-1,"/prm/cfg",_("Конфігурація")))
          {
            ctrMkNode("fld",opt,-
1,"/prm/cfg/id",cfg("ID").fld().descr(),R_R_R_,"root",SARH_ID,1,"tp","str");
            ctrMkNode("fld",opt,-
1,"/prm/cfg/nm",cfg("NAME").fld().descr(),RWRWR_,"root",SARH_ID,2,"tp","str","le
n","50");
            ctrMkNode("fld",opt,-
1,"/prm/cfg/dscr",cfg("DESCR").fld().descr(),RWRWR_,"root",SARH_ID,3,"tp","str",
"cols","90","rows","3");
            ctrMkNode("fld",opt,-
1,"/prm/cfg/addr",cfg("ADDR").fld().descr(),RWRWR_,"root",SARH_ID,1,"tp","str");
            ctrMkNode("fld",opt,-
1,"/prm/cfg/lvl",cfg("LEVEL").fld().descr(),RWRWR_,"root",SARH_ID,2,"tp","dec",
                "help",_("Отримуємо повідомлення для рівня більшого і
дорівнюючого цьому."));
            ctrMkNode("fld",opt,-
1,"/prm/cfg/cats",cfg("CATEG").fld().descr(),RWRWR_,"root",SARH_ID,2,"tp","str",
                "help",_("Шаблон категорії повідомлень або регулярне вираження у
процесі архівування, відокремлюються символом';'.\n"
                  "Використовуйте тимчасові символи для групового
виділення:\n  '*' - будь-які підрядки;\n  '?' - будь-які символи.\n"
                  "Регулярне вираження складається з символів'/'
(/mod_(System|LogicLev)/)."));
            ctrMkNode("fld",opt,-1,"/prm/cfg/start",_("To
start"),RWRWR_,"root",SARH_ID,1,"tp","bool");
          }
      }
```

```
      if(run_st && ctrMkNode("area",opt,-
1,"/mess",_("Messages"),R_R___,"root",SARH_ID))
      {
          ctrMkNode("fld",opt,-
1,"/mess/tm",_("Time"),RWRW__,"root",SARH_ID,1,"tp","time");
          ctrMkNode("fld",opt,-1,"/mess/size",_("Розмір
(s)"),RWRW__,"root",SARH_ID,1,"tp","dec");
          ctrMkNode("fld",opt,-1,"/mess/cat",_("Зразок
категорії"),RWRW__,"root",SARH_ID,2,"tp","str","help",
            _("Шаблон категорії повідомлень або регулярне вираження.\n"
                  "Використовуйте тимчасові символи для групового виділення:\n
'*' - будь-які підрядки;\n  '?' - будь-які символи.\n"
                "Регулярне вираження складається з символів'/'
(/mod_(System|LogicLev)/)."));
          ctrMkNode("fld",opt,-
1,"/mess/lvl",_("Level"),RWRW__,"root",SARH_ID,4,"tp","dec","min","0","max","7",
            "help",_("Отримуємо повідомлення для рівня більшого і дорівнюючого
цьому."));
          if(ctrMkNode("table",opt,-
1,"/mess/mess",_("Messages"),R_R___,"root",SARH_ID))
          {
            ctrMkNode("list",opt,-
1,"/mess/mess/0",_("Час"),R_R___,"root",SARH_ID,1,"tp","time");
            ctrMkNode("list",opt,-
1,"/mess/mess/0a",_("mcsec"),R_R___,"root",SARH_ID,1,"tp","dec");
            ctrMkNode("list",opt,-
1,"/mess/mess/1",_("Категорія"),R_R___,"root",SARH_ID,1,"tp","str");
            ctrMkNode("list",opt,-
1,"/mess/mess/2",_("Рівень"),R_R___,"root",SARH_ID,1,"tp","dec");
            ctrMkNode("list",opt,-
1,"/mess/mess/3",_("Повідомлення"),R_R___,"root",SARH_ID,1,"tp","str");
          }
      }
      return;
    }
    //> Процес управління на сторінці
    string a_path = opt->attr("path");
    if(a_path == "/prm/st/st")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt->setText(
startStat() ? "1" : "0" );
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))     atoi(opt-
>text().c_str()) ? start() : stop();
    }
    else if(a_path == "/prm/st/db")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt->setText(
DB() );
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))     setDB( opt-
>text() );
    }
    else if(a_path == "/prm/st/end" && ctrChkNode(opt))          opt->setText(
TSYS::int2str(end()) );
    else if(a_path == "/prm/st/beg" && ctrChkNode(opt))          opt->setText(
TSYS::int2str(begin()) );
    else if(a_path == "/prm/cfg/id" && ctrChkNode(opt))          opt->setText(
id() );
    else if(a_path == "/prm/cfg/nm" )
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt->setText(
name() );
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))     setName( opt-
>text() );
    }
    else if(a_path == "/prm/cfg/dscr")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt->setText(
dscr() );
```

```
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))     setDscr( opt-
>text() );
    }
    else if(a_path == "/prm/cfg/addr")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt->setText(
addr() );
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))     setAddr( opt-
>text() );
    }
    else if(a_path == "/prm/cfg/lvl")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt-
>setText(TSYS::int2str(level()));
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))     setLevel(
atoi(opt->text().c_str()) );
    }
    else if(a_path == "/prm/cfg/start")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt->setText(
toStart() ? "1" : "0" );
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))     setToStart(
atoi(opt->text().c_str()) );
    }
    else if(a_path == "/prm/cfg/cats")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt-
>setText(m_cat_o);
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))     { m_cat_o =
opt->text(); modif(); }
    }
    else if(a_path == "/mess/tm")
    {
      if(ctrChkNode(opt,"get",RWRW__,"root",SARH_ID,SEC_RD))
      {
          opt->setText(TBDS::genDBGet(nodePath()+"messTm","0",opt-
>attr("user")));
          if(!atoi(opt->text().c_str()))  opt-
>setText(TSYS::int2str(time(NULL)));
      }
      if(ctrChkNode(opt,"set",RWRW__,"root",SARH_ID,SEC_WR))
          TBDS::genDBSet(nodePath()+"messTm",(atoi(opt-
>text().c_str())>=time(NULL))?"0":opt->text(),opt->attr("user"));
    }
    else if(a_path == "/mess/size")
    {
      if(ctrChkNode(opt,"get",RWRW__,"root",SARH_ID,SEC_RD))     opt-
>setText(TBDS::genDBGet(nodePath()+"messSize","10",opt->attr("user")));
      if(ctrChkNode(opt,"set",RWRW__,"root",SARH_ID,SEC_WR))
      TBDS::genDBSet(nodePath()+"messSize",opt->text(),opt->attr("user"));
    }
    else if(a_path == "/mess/cat")
    {
      if(ctrChkNode(opt,"get",RWRW__,"root",SARH_ID,SEC_RD))     opt-
>setText(TBDS::genDBGet(nodePath()+"messCat","",opt->attr("user")));
      if(ctrChkNode(opt,"set",RWRW__,"root",SARH_ID,SEC_WR))
      TBDS::genDBSet(nodePath()+"messCat",opt->text(),opt->attr("user"));
    }
    else if(a_path == "/mess/lvl")
    {
      if(ctrChkNode(opt,"get",RWRW__,"root",SARH_ID,SEC_RD))     opt-
>setText(TBDS::genDBGet(nodePath()+"messLev","0",opt->attr("user")));
      if(ctrChkNode(opt,"set",RWRW__,"root",SARH_ID,SEC_WR))
      TBDS::genDBSet(nodePath()+"messLev",opt->text(),opt->attr("user"));
    }
    else if(a_path == "/mess/mess" && run_st &&
ctrChkNode(opt,"get",R_R___,"root",SARH_ID))
    {
      vector<TMess::SRec> rec;
```

```
      time_t end = atoi(TBDS::genDBGet(nodePath()+"messTm","0",opt-
>attr("user")).c_str());
      if( !end ) end = time(NULL);
      time_t beg = end - atoi(TBDS::genDBGet(nodePath()+"messSize","10",opt-
>attr("user")).c_str());
      string cat = TBDS::genDBGet(nodePath()+"messCat","",opt->attr("user"));
      char   lev = atoi(TBDS::genDBGet(nodePath()+"messLev","0",opt-
>attr("user")).c_str());

      get( beg, end, rec, cat, lev );

      XMLNode *n_tm     = ctrMkNode("list",opt,-
1,"/mess/mess/0","",R_R___,"root",SARH_ID);
      XMLNode *n_tmu    = ctrMkNode("list",opt,-
1,"/mess/mess/0a","",R_R___,"root",SARH_ID);
      XMLNode *n_cat    = ctrMkNode("list",opt,-
1,"/mess/mess/1","",R_R___,"root",SARH_ID);
      XMLNode *n_lvl    = ctrMkNode("list",opt,-
1,"/mess/mess/2","",R_R___,"root",SARH_ID);
      XMLNode *n_mess   = ctrMkNode("list",opt,-
1,"/mess/mess/3","",R_R___,"root",SARH_ID);
      for(int i_rec = rec.size()-1; i_rec >= 0; i_rec--)
      {
          if(n_tm)      n_tm->childAdd("el")-
>setText(TSYS::int2str(rec[i_rec].time));
          if(n_tmu)     n_tmu->childAdd("el")-
>setText(TSYS::int2str(rec[i_rec].utime));
          if(n_cat)     n_cat->childAdd("el")->setText(rec[i_rec].categ);
          if(n_lvl)     n_lvl->childAdd("el")-
>setText(TSYS::int2str(rec[i_rec].level));
          if(n_mess)    n_mess->childAdd("el")->setText(rec[i_rec].mess);
      }
    }
    else TCntrNode::cntrCmdProc(opt);
}
```

tmodule.cpp – робота з модулями

```
// Петченко Даниіл Павлович
//SCADA_system системний файл: tmodule.cpp

#include <sys/types.h>
#include <sys/stat.h>
#include <stdarg.h>
#include <unistd.h>
#include <dlfcn.h>
#include <string.h>
#include <libintl.h>

#include "tsys.h"
#include "terror.h"
#include "tmess.h"
#include "tsubsys.h"
#include "tmodule.h"

using namespace WSCADA;

//************************************************
//* TModule                                      *
//************************************************
const char *TModule::l_info[] =
    {"Модуль","Ім'я","Тип","Джерело","Версія","Автор","Дескриптор","Ліцензія"};

TModule::TModule( const string &id ) : mId(id)
{
    lc_id = string("oscd_")+mId;
    bindtextdomain(lc_id.c_str(),LOCALEDIR);

    //> Переведення динамічного рядка
#if 0
    char mess[][100] = { _("Автор"), _("Ліцензія") };
#endif
}

TModule::~TModule( )
{
    //> Очищення списку експортуємих функцій
    for(unsigned i = 0; i < m_efunc.size(); i++)
        delete m_efunc[i];
}

string TModule::modName()
{
    return mName;
}

void TModule::postEnable( int flag )
{
    if(flag&TCntrNode::NodeRestore) return;

    mess_info(nodePath().c_str(),_("З'єднання з модулем!"));
}

TSubSYS &TModule::owner( )    { return *(TSubSYS*)nodePrev(); }

void TModule::modFuncList( vector<string> &list )
{
    list.clear();
    for(unsigned i = 0; i < m_efunc.size(); i++)
      list.push_back(m_efunc[i]->prot);
}

bool TModule::modFuncPresent( const string &prot )
{
```

```
    for(unsigned i = 0; i < m_efunc.size(); i++)
      if(m_efunc[i]->prot == prot)
          return true;
    return false;
}

TModule::ExpFunc &TModule::modFunc( const string &prot )
{
    for(unsigned i = 0; i < m_efunc.size(); i++)
      if(m_efunc[i]->prot == prot) return *m_efunc[i];
    throw TError(nodePath().c_str(),_("Функція '%s' не представлена у
модулі!"),prot.c_str());
}

void TModule::modFunc( const string &prot, void (TModule::**offptr)() )
{
    *offptr = modFunc(prot).ptr;
}

void TModule::modInfo( vector<string> &list )
{
    for(unsigned i_opt = 0; i_opt < sizeof(l_info)/sizeof(char *); i_opt++)
      list.push_back(l_info[i_opt]);
}

string TModule::modInfo( const string &name )
{
    string info;

    if(name == l_info[0])     info = mId;
    else if(name == l_info[1]) info = mName;
    else if(name == l_info[2]) info = mType;
    else if(name == l_info[3]) info = mSource;
    else if(name == l_info[4]) info = mVers;
    else if(name == l_info[5]) info = mAuthor;
    else if(name == l_info[6]) info = mDescr;
    else if(name == l_info[7]) info = mLicense;

    return info;
}

void TModule::cntrCmdProc( XMLNode *opt )
{
    //> Беремо сторінку інформації
    if(opt->name() == "info")
    {
      TCntrNode::cntrCmdProc(opt);
      ctrMkNode("WSCADA_cntr",opt,-1,"/",_("Module: ")+modId(),R_R_R_);
      ctrMkNode("branches",opt,-1,"/br","",R_R_R_);
      if(TUIS::icoPresent(owner().subId()+"."+modId())) ctrMkNode("img",opt,-
1,"/ico","",R_R_R_);
      if(ctrMkNode("area",opt,-1,"/help",_("Help")))
          if(ctrMkNode("area",opt,-1,"/help/m_inf",_("Модуль інформації")))
          {
            vector<string> list;
            modInfo(list);
            for(unsigned i_l = 0; i_l < list.size(); i_l++)
                ctrMkNode("fld",opt,-
1,(string("/help/m_inf/")+list[i_l]).c_str(),_(list[i_l].c_str()),R_R_R_,"root",
"root",1,"tp","str");
          }
      return;
    }

    //> Процес управління на сторінці
    string a_path = opt->attr("path");
    if(a_path == "/ico" && ctrChkNode(opt))
    {
      string itp;
```

```
      opt-
>setText(TSYS::strEncode(TUIS::icoGet(owner().subId()+"."+modId(),&itp),TSYS::ba
se64));
      opt->setAttr("tp",itp);
    }
    else if(a_path.substr(0,11) == "/help/m_inf" && ctrChkNode(opt))
      opt->setText(modInfo(TSYS::pathLev(a_path,2)));
    else TCntrNode::cntrCmdProc(opt);
}

const char *TModule::I18N( const char *mess )
{
    const char *rez = Mess->I18N(mess,lc_id.c_str());
    if( !strcmp(mess,rez) ) rez = _(mess);
    return rez;
}
```

tparamcontr.cpp – параметри управління системою

```
// Петченко Даниіл Павлович
//SCADA_system системний файл: tparamcontr.cpp

#include "tbds.h"
#include "tsys.h"
#include "tmess.h"
#include "tdaqs.h"
#include "tcontroller.h"
#include "ttipdaq.h"
#include "ttiparam.h"
#include "tparamcontr.h"

using namespace WSCADA;

//************************************************
//* TParamContr                                  *
//************************************************
TParamContr::TParamContr( const string &name, TTipParam *tpprm ) :
TConfig(tpprm), m_en(false), tipparm(tpprm)
{
    setId(name);
    setName(name);
}

TParamContr::~TParamContr( )
{
    nodeDelAll();
}

TCntrNode &TParamContr::operator=( TCntrNode &node )
{
    TParamContr *src_n = dynamic_cast<TParamContr*>(&node);
    if(!src_n) return *this;

    //> Перевіряємо тип параметрів й змінюємо їх, якщо вони змінні, або нижчі
    if(type().name != src_n->type().name && owner().owner().tpPrmToId(src_n-
>type().name) >= 0)
    {
      if(enableStat()) disable();
      setType(src_n->type().name);
    }

    //> Конфігурація копіювання
    string tid = id();
    *(TConfig*)this = *(TConfig*)src_n;
    setId(tid);

    //> Дозволяємо нові параметри
    if(src_n->enableStat() && toEnable( ) && !enableStat())enable();

    return *this;
}

TController &TParamContr::owner( ) { return *(TController*)nodePrev(); }

string TParamContr::name( )   { string nm = cfg("NAME").getS(); return nm.size()
? nm : id(); }

void TParamContr::setName( const string &inm ) { cfg("NAME").setS(inm); }

string TParamContr::descr( ) { return cfg("DESCR").getS(); }

void TParamContr::setDescr( const string &idsc ){ cfg("DESCR").setS(idsc); }

void TParamContr::postEnable(int flag)
{
```

```
    TValue::postEnable(flag);

    if(!vlCfg())  setVlCfg(this);
    if(!vlElemPresent(&SYS->daq().at().errE()))
      vlElemAtt(&SYS->daq().at().errE());
}

void TParamContr::preDisable(int flag)
{
    //> Видаляємо або зупиняємо архівування
    vector<string> a_ls;
    vlList(a_ls);

    for(unsigned i_a = 0; i_a < a_ls.size(); i_a++)
      if(!vlAt(a_ls[i_a]).at().arch().freeStat())
      {
          string arh_id = vlAt(a_ls[i_a]).at().arch().at().id();
          if(flag) SYS->archive().at().valDel(arh_id,true);
          else SYS->archive().at().valAt(arh_id).at().stop();
      }

    if(enableStat())    disable();
}

void TParamContr::postDisable(int flag)
{
    if(flag)
    {
      //> Видаляємо параметри з БД
      try
      {
          SYS->db().at().dataDel(owner().DB()+"."+owner().cfg(type().db).getS(),

owner().owner().nodePath()+owner().cfg(type().db).getS(),*this,true);
      }catch(TError err) { mess_err(err.cat.c_str(),"%s",err.mess.c_str()); }
    }
}

void TParamContr::load_( )
{
    if(!SYS->chkSelDB(owner().DB())) return;

    cfgViewAll(true);
    SYS->db().at().dataGet(owner().DB()+"."+owner().cfg(type().db).getS(),

owner().owner().nodePath()+owner().cfg(type().db).getS(),*this);
}

void TParamContr::save_( )
{
    SYS->db().at().dataSet( owner().DB()+"."+owner().cfg(type().db).getS(),

owner().owner().nodePath()+owner().cfg(type().db).getS(),*this );

    //> Зберігаємо архіви
    vector<string> a_ls;
    vlList(a_ls);
    for(unsigned i_a = 0; i_a < a_ls.size(); i_a++)
      if(!vlAt(a_ls[i_a]).at().arch().freeStat())
          vlAt(a_ls[i_a]).at().arch().at().save();
}

bool TParamContr::cfgChange( TCfg &cfg )  { modif( ); return true; }

TParamContr & TParamContr::operator=( TParamContr & PrmCntr )
{
    TConfig::operator=(PrmCntr);

    return *this;
```

```
}

void TParamContr::enable()
{
    m_en = true;
}

void TParamContr::disable()
{
    m_en = false;
}

void TParamContr::vlGet( TVal &val )
{
    if( val.name() == "err" )
    {
      if( !enableStat() ) val.setS(_("1:Параметри заборонені."),0,true);
      else if( !owner().startStat( ) ) val.setS(_("2:Контролер
зупинено."),0,true);
      else val.setS("0",0,true);
    }
}

void TParamContr::setId( const string &vl )
{
    cfg("SHIFR").setS(vl);
}

void TParamContr::setType( const string &tpId )
{
    if(enableStat() || tpId == type().name ||
!owner().owner().tpPrmPresent(tpId))     return;

    setNodeMode(TCntrNode::Disable);

    try
    {
      //> Чекаємо поки роз'єднаються інші
      while(nodeUse(true) > 1) usleep(1000);
      //> Видаляємо з БД
      postDisable(true);

      //> Створюємо тимчасову структуру
      TConfig tCfg(&type());
      tCfg = *(TConfig*)this;

      //> Встановлюємо нову конфігурацію структури
      tipparm = &owner().owner().tpPrmAt(owner().owner().tpPrmToId(tpId));
      setElem(tipparm);

      //> Відновлюємо конфігурацію
      *(TConfig*)this = tCfg;
    }catch(...) { }
    setNodeMode(TCntrNode::Enable);
    setVlCfg(this);
    modif();
}
TVariant TParamContr::objFuncCall( const string &iid, vector<TVariant> &prms,
const string &user )
{
    //> Виклик функцій конфігурації
    TVariant cfRez = objFunc(iid, prms, user);
    if(!cfRez.isNull()) return cfRez;
    return TValue::objFuncCall(iid, prms, user);
}
void TParamContr::cntrCmdProc( XMLNode *opt )
{
    string a_path = opt->attr("path");
    //> Сервіс управління процесами
```

```
    if(a_path.substr(0,6) == "/serv/")    { TValue::cntrCmdProc(opt); return; }
    //> Беремо сторінку інформації
    if(opt->name() == "info")
    {
      TValue::cntrCmdProc(opt);
      ctrMkNode("WSCADA_cntr",opt,-1,"/",_("Параметр:
")+name(),RWRWR_,"root",SDAQ_ID);
      if(ctrMkNode("area",opt,0,"/prm",_("Parameter")))
      {
          if(ctrMkNode("area",opt,-1,"/prm/st",_("Стан")))
          {
            if(!enableStat() && owner().owner().tpPrmSize() > 1)
                ctrMkNode("fld",opt,-
1,"/prm/st/type",_("Тип"),RWRWR_,"root",SDAQ_ID,4,"tp","str","dest","select","se
lect","/prm/tpLst",
                  "help",_("Змінюємо тип керівництва до останіх даних для
специфічних конфігурацій."));
            else ctrMkNode("fld",opt,-
1,"/prm/st/type",_("Тип"),R_R_R_,"root",SDAQ_ID,1,"tp","str");
            if(owner().enableStat())
                ctrMkNode("fld",opt,-
1,"/prm/st/en",_("Дозволено"),RWRWR_,"root",SDAQ_ID,1,"tp","bool");
          }
          if(ctrMkNode("area",opt,-1,"/prm/cfg",_("Configuration")))
            TConfig::cntrCmdMake(opt,"/prm/cfg",0,"root",SDAQ_ID,RWRWR_);
      }
        return;
    }
    //> Процес управління на сторінці
    if(a_path == "/prm/st/type")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SDAQ_ID,SEC_RD))     opt-
>setText(type().name);
      if(ctrChkNode(opt,"set",RWRWR_,"root",SDAQ_ID,SEC_WR))     setType(opt-
>text());
    }
    else if(a_path == "/prm/st/en")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SDAQ_ID,SEC_RD))     opt-
>setText(enableStat()?"1":"0");
      if(ctrChkNode(opt,"set",RWRWR_,"root",SDAQ_ID,SEC_WR))
      {
          if(!owner().enableStat()) throw TError(nodePath().c_str(),_("Контролер
не запустився!"));
          else atoi(opt->text().c_str())?enable():disable();
      }
    }
    else if(a_path.substr(0,8) == "/prm/cfg")
TConfig::cntrCmdProc(opt,TSYS::pathLev(a_path,2),"root",SDAQ_ID,RWRWR_);
    else if(a_path == "/prm/tmplList" && ctrChkNode(opt))
    {
        vector<string> lls, ls;
      SYS->daq().at().tmplLibList(lls);
      for(unsigned i_l = 0; i_l < lls.size(); i_l++)
      {
          SYS->daq().at().tmplLibAt(lls[i_l]).at().list(ls);
          for(unsigned i_t = 0; i_t < ls.size(); i_t++)
            opt->childAdd("el")->setText(lls[i_l]+"."+ls[i_t]);
      }
    }
    else if(a_path == "/prm/tpLst" && ctrChkNode(opt))
      for(unsigned i_tp = 0; i_tp < owner().owner().tpPrmSize(); i_tp++)
          opt->childAdd("el")->setAttr("id",owner().owner().tpPrmAt(i_tp).name)-
>setText(owner().owner().tpPrmAt(i_tp).descr);
    else TValue::cntrCmdProc(opt);
}
```

main.cpp – головна програма

```
// Петченко Даниіл Павлович
//SCADA_system файл системи: main.cpp

#include <getopt.h>
#include "terror.h"
#include "tmess.h"
#include "tsys.h"

using namespace WSCADA;

int main(int argc, char *argv[], char *envp[] )
{
    int rez = 0;

    //Перевірка початку роботи режиму основного процесу
    int next_opt;
    optind=opterr=0;
    struct option long_opt[] = { {"Режим основного процесу" ,0,NULL,'d'}, {NULL
,0,NULL,0  } };
    while((next_opt=getopt_long(argc,argv,"",long_opt,NULL)) != -1)
      if( next_opt == 'd' )
      {
          printf("Початок роботи режиму основного процесу!\n");
          int pid = fork();
          if( pid == -1 )
          {
            printf("Помилка: неможливо створити новий процес!\n");
            return -1;
          }
          if( pid != 0 )      return 0;

          //Готується оточення режиму основного процесу
          setsid();
          break;
      }

    try
    {
      SYS = new TSYS(argc,argv,envp);

      SYS->load();
      if( (rez=SYS->stopSignal()) > 0 ) return rez;
      rez = SYS->start();

      delete SYS;
    }catch(TError err) { mess_err(err.cat.c_str(),"%s",err.mess.c_str()); }

    printf("SCADA_system система коректно працює з даними %d.\n",rez);

    return rez;
}
```

tsys.cpp – встановлення та запуск системи

```
// Петченко Даниіл Павлович
//SCADA_system системний файл: tsys.cpp

#include <features.h>
#include <ieee754.h>
#include <syscall.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <sys/stat.h>
#include <sys/time.h>
#include <sys/utsname.h>
#include <errno.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>
#include <getopt.h>
#include <stdio.h>
#include <signal.h>
#include <stdarg.h>
#include <stdlib.h>
#include <langinfo.h>
#include <zlib.h>

#include "terror.h"
#include "tmess.h"
#include "tsys.h"

using namespace WSCADA;

//Поточна зміна доступу
TMess *WSCADA::Mess;
TSYS  *WSCADA::SYS;
bool TSYS::finalKill = false;
pthread_key_t TSYS::sTaskKey;

TSYS::TSYS( int argi, char ** argb, char **env ) : argc(argi), argv((const char
**)argb), envp((const char **)env),
    mUser("root"), mConfFile("/etc/WSCADA.xml"), mId("EmptySt"),
mName(_("Порожня Станція")), mIcoDir("./icons/"), mModDir("./"),
    mWorkDB("<cfg>"), mSaveAtExit(false), mSavePeriod(0), rootModifCnt(0),
mStopSignal(-1), mMultCPU(false)
{
    finalKill = false;
    SYS = this;         //Ініціалізуємо значення глобальних зміних доступу
    mSubst = grpAdd("sub_",true);
    nodeEn();
    pthread_key_create(&sTaskKey, NULL);

    Mess = new TMess();

    if(getenv("USER")) mUser = getenv("USER");

    //> Ініціалізуємо системний годинник
    clkCalc();

#if __GLIBC_PREREQ(2,4)
    //> Multi CPU дозволяють перевірку
    cpu_set_t cpuset;
    CPU_ZERO(&cpuset);
    CPU_SET(1,&cpuset);
    mMultCPU = !pthread_setaffinity_np(pthread_self(), sizeof(cpu_set_t),
&cpuset);
#endif

    //> Встановлюємо сигнальні програми обробки
    signal(SIGINT,sighandler);
```

```
    signal(SIGTERM,sighandler);
    //signal(SIGCHLD,sighandler);
    signal(SIGALRM,sighandler);
    signal(SIGPIPE,sighandler);
    //signal(SIGFPE,sighandler);
    //signal(SIGSEGV,sighandler);
    signal(SIGABRT,sighandler);
}

TSYS::~TSYS( )
{
    finalKill = true;

    //Видаляємо всі вузли в команді управління
    del("ModSched");
    del("UI");
    del("Special");
    del("Archive");
    del("DAQ");
    del("Protocol");
    del("Transport");
    del("Security");
    del("BD");

    delete Mess;
    pthread_key_delete(sTaskKey);
}

string TSYS::host( )
{
    utsname ubuf; uname(&ubuf);
    return ubuf.nodename;
}

string TSYS::workDir( )
{
    char buf[STR_BUF_LEN];
    return getcwd(buf,sizeof(buf));
}

void TSYS::setWorkDir( const string &wdir )
{
    if(workDir() == wdir) return;
    if(chdir(wdir.c_str()) != 0)
      mess_warning(nodePath().c_str(),_("Змініть робочу директорію'%s' Помилка:
%s. Можливо поточний каталог вже встановлено правильно'%s'."),
          wdir.c_str(),strerror(errno),workDir().c_str());
    modif( );
}

XMLNode *TSYS::cfgNode( const string &path, bool create )
{
    string s_el, ndNm;

    XMLNode *t_node = &rootN;
    if(t_node->name() != "SCADA_system")
    {
      if(!create) return NULL;
      t_node->setName("SCADA_system");
    }

    for(int l_off = 0, nLev = 0; true; nLev++)
    {
        s_el = TSYS::pathLev(path,0,true,&l_off);
        if(s_el.empty()) return t_node;
        bool ok = false;
        for(unsigned i_f = 0; !ok && i_f < t_node->childSize(); i_f++)
            if(t_node->childGet(i_f)->attr("id") == s_el)
            {
```

```
                t_node = t_node->childGet(i_f);
                ok = true;
              }
        if(!ok)
        {
          if(!create)   return NULL;
          ndNm = "prm";
          switch(nLev)
          {
            case 0: ndNm = "station";     break;
            case 1: if(s_el.compare(0,4,"sub_") == 0) ndNm = "node";   break;
            case 2: if(s_el.compare(0,4,"mod_") == 0) ndNm = "node";   break;
          }
          if(ndNm == "prm") t_node = t_node->childIns(0,ndNm)-
>setAttr("id",s_el);
          else t_node = t_node->childAdd(ndNm)->setAttr("id",s_el);
        }
    }
    return t_node;
}

string TSYS::int2str( int val, TSYS::IntView view )
{
    char buf[STR_BUF_LEN];
    if(view == TSYS::Dec)     snprintf(buf,sizeof(buf),"%d",val);
    else if(view == TSYS::Oct)      snprintf(buf,sizeof(buf),"%o",val);
    else if(view == TSYS::Hex)      snprintf(buf,sizeof(buf),"%x",val);

    return buf;
}

string TSYS::uint2str( unsigned val, IntView view )
{
    char buf[STR_BUF_LEN];
    if(view == TSYS::Dec)     snprintf(buf,sizeof(buf),"%u",val);
    else if(view == TSYS::Oct)      snprintf(buf,sizeof(buf),"%o",val);
    else if(view == TSYS::Hex)      snprintf(buf,sizeof(buf),"%x",val);

    return buf;
}

string TSYS::ll2str( int64_t val, IntView view )
{
    char buf[STR_BUF_LEN];
    if(view == TSYS::Dec)     snprintf(buf,sizeof(buf),"%lld",val);
    else if(view == TSYS::Oct)      snprintf(buf,sizeof(buf),"%llo",val);
    else if(view == TSYS::Hex)      snprintf(buf,sizeof(buf),"%llx",val);

    return buf;
}

string TSYS::real2str( double val, int prec, char tp )
{
    char buf[STR_BUF_LEN];
    if(tp == 'g') snprintf(buf,sizeof(buf),"%.*g",prec,val);
    else if(tp == 'e') snprintf(buf,sizeof(buf),"%.*e",prec,val);
    else snprintf(buf,sizeof(buf),"%.*f",prec,val);

    return buf;
}

string TSYS::time2str( time_t itm, const string &format )
{
    struct tm tm_tm;
    localtime_r(&itm,&tm_tm);
    char buf[100];
    int ret = strftime(buf, sizeof(buf), format.empty()?"%d-%m-%Y
%H:%M:%S":format.c_str(), &tm_tm);
    return (ret > 0) ? string(buf,ret) : string("");
```

```
}

string TSYS::time2str( double utm )
{
    if(utm < 1e-6) return "0";
    int lev = 0;
    int days = (int)floor(utm/(24*60*60*1e6));
    int часи = (int)floor(utm/(60*60*1e6))%24;
    int mins = (int)floor(utm/(60*1e6))%60;
    double usec = utm - 1e6*(days*24*60*60 + часи*60*60 + mins*60);

    string rez;
    if(days)            { rez += TSYS::int2str(days)+_("day"); lev =
vmax(lev,6); }
    if(часи)            { rez += (rez.size()?"
":"")+TSYS::int2str(часи)+_("годин"); lev = vmax(lev,5); }
    if(mins && lev < 6) { rez += (rez.size()?"
":"")+TSYS::int2str(mins)+_("хвилин"); lev = vmax(lev,4); }
    if((1e-6*usec) > 0.5 && lev < 5)      { rez += (rez.size()?"
":"")+TSYS::real2str(1e-6*usec,3)+_("секунд"); lev = vmax(lev,3); }
    else if((1e-3*usec) > 0.5 && !lev)    { rez += (rez.size()?"
":"")+TSYS::real2str(1e-3*usec,4)+_("мікросекунд"); lev = vmax(lev,2); }
    else if(usec > 0.5 && !lev)           { rez += (rez.size()?"
":"")+TSYS::real2str(usec,4)+_("us"); lev = vmax(lev,1); }
    else if(!lev) rez += (rez.size()?" ":"")+TSYS::real2str(1e3*usec,4)+_("ns");
    return rez;
}

string TSYS::cpct2str( double cnt )
{
    if(cnt > 0.2*pow(2,80))   return
TSYS::real2str(cnt/pow(2,80),3,'g')+_("YiB");
    if(cnt > 0.2*pow(2,70))   return
TSYS::real2str(cnt/pow(2,70),3,'g')+_("ZiB");
    if(cnt > 0.2*pow(2,60))   return
TSYS::real2str(cnt/pow(2,60),3,'g')+_("EiB");
    if(cnt > 0.2*pow(2,50))   return
TSYS::real2str(cnt/pow(2,50),3,'g')+_("PiB");
    if(cnt > 0.2*pow(2,40))   return
TSYS::real2str(cnt/pow(2,40),3,'g')+_("TiB");
    if(cnt > 0.2*pow(2,30))   return
TSYS::real2str(cnt/pow(2,30),3,'g')+_("GiB");
    if(cnt > 0.2*pow(2,20))   return
TSYS::real2str(cnt/pow(2,20),3,'g')+_("MiB");
    if(cnt > 0.2*pow(2,10))   return
TSYS::real2str(cnt/pow(2,10),3,'g')+_("KiB");
    return TSYS::real2str(cnt,3,'g')+_("B");
}

string TSYS::addr2str( void *addr )
{
    char buf[sizeof(void*)*2+3];
    snprintf(buf,sizeof(buf),"%p",addr);

    return buf;
}

void *TSYS::str2addr( const string &str )
{
    return (void *)strtoul(str.c_str(),NULL,16);
}

string TSYS::strNoSpace( const string &val )
{
    int beg = -1, end = -1;

    for(unsigned i_s = 0; i_s < val.size(); i_s++)
      if(val[i_s] != ' ' && val[i_s] != '\n' && val[i_s] != '\t')
      {
```

```
            if(beg < 0) beg = i_s;
            end = i_s;
        }

    return (beg>=0) ? val.substr(beg,end-beg+1) : "";
}

string TSYS::strMess( const char *fmt, ... )
{
    char str[STR_BUF_LEN];
    va_list argptr;

    va_start(argptr,fmt);
    vsnprintf(str,sizeof(str),fmt,argptr);
    va_end(argptr);

    return str;
}

string TSYS::optDescr( )
{
    utsname buf;
    uname(&buf);
    return TSYS::strMess(_(
      "**************************************************************************
**\n"
      "********** %s v%s (%s-%s). *********\n"
      "**************************************************************************
**\n\n"
      "==========================================================================
==\n"
      "======================== Загальносистемні опції
======================\n"
      "==========================================================================
==\n"
      "-h, --help             Інформаційні повідомлення про опції системи.\n"
      "    --Config=<path>    Шлях до конфігураційного файлу.\n"
      "    --Station=<id>     Ідентифікатор станції.\n"
      "    --режим основного процесу          Початок роботи режиму основного
процесу.\n"
      "    --MessLev=<level>  Повідомлення процесів<level> (0-7).\n"
      "    --log=<direct>     Повідомлення про :\n"
      "                         <direct> & 1 - події у системі;\n"
      "                         <direct> & 2 - ввод/вивід у системі;\n"
      "                         <direct> & 4 - помилки у системі;\n"
      "                         <direct> & 8 - архівація у системі;\n"
      "---------- Конфігураційний файл параметрів робочої станції'%s' ---------
--\n"
      "StName     <nm>  Ім’я робочої станції.\n"
      "WorkDB     <Type.Name> Робоча база даних (тип та ім’я).\n"
      "Workdir    <path>      Робоча директорія.\n"
      "IcoDir     <path>      Директорія іконок.\n"
      "ModDir     <path>      Директорія модулів.\n"
      "MessLev    <level>     Повідомлення <рівні> (0-7).\n"
      "LogTarget  <direction> Повідомлення про :\n"
      "                           <direct> & 1 - події у системі;\n"
      "                           <direct> & 2 - ввод/вивід у системі;\n"
      "                           <direct> & 4 - помилки у системі;\n"
      "                           <direct> & 8 - архівація у системі;\n"
      "Lang2CodeBase <lang>   Базова мова для переведення тексту, двосимвольний
код .\n"
      "SaveAtExit <true>      Збереження змін у системі перед виходом.\n"
      "SavePeriod <sec> Збереження періоду роботи  у системі .\n\n"),
      PACKAGE_NAME,VERSION,buf.sysname,buf.release,nodePath().c_str());
}

bool TSYS::cfgFileLoad( )
{
    bool cmd_help = false;
```

```
    //================ Редагування параметрів=========================
    int next_opt;
    const char *short_opt="h";
    struct option long_opt[] =
    {
      {"help"     ,0,NULL,'h'},
      {"Config"   ,1,NULL,'f'},
      {"Station"  ,1,NULL,'s'},
      {NULL       ,0,NULL,0  }
    };

    optind=opterr=0;
    do
    {
      next_opt=getopt_long(argc,(char * const *)argv,short_opt,long_opt,NULL);
      switch(next_opt)
      {
          case 'h':
            fprintf(stdout,"%s",optDescr().c_str());
            Mess->setMessLevel(7);
            cmd_help = true;
            break;
          case 'f': mConfFile = optarg; break;
          case 's': mId = optarg; break;
          case -1 : break;
      }
    } while(next_opt != -1);

    //Завантажуємо кофігураційний файл
    int hd = open(mConfFile.c_str(),O_RDONLY);
    if(hd < 0) mess_err(nodePath().c_str(),_("Кофігураційний файл '%s' Помилка:
%s"),mConfFile.c_str(),strerror(errno));
    else
    {
      string s_buf;
      int cf_sz = lseek(hd,0,SEEK_END);
      if(cf_sz > 0)
      {
          lseek(hd,0,SEEK_SET);
          char *buf = (char *)malloc(cf_sz+1);
          read(hd,buf,cf_sz);
          buf[cf_sz] = 0;
          s_buf = buf;
          free(buf);
      }
      close(hd);

      try
      {
          ResAlloc res(nodeRes(),true);
          rootN.load(s_buf,true);
          if(rootN.name() == "SCADA_system")
          {
            XMLNode *stat_n = NULL;
            for(int i_st = rootN.childSize()-1; i_st >= 0; i_st--)
                if(rootN.childGet(i_st)->name() == "station")
                {
                  stat_n = rootN.childGet(i_st);
                  if(stat_n->attr("id") == mId) break;
                }
            if(stat_n && stat_n->attr("id") != mId)
            {
                mess_warning(nodePath().c_str(),_("Робоча станція '%s' не
представлена у  кофігураційному файлі Використайте '%s' конфігурацію робочої
станції!"),
                  mId.c_str(), stat_n->attr("id").c_str());
                mId     = stat_n->attr("id");
            }
```

```
            if(!stat_n) rootN.clear();
          } else rootN.clear();
          if(!rootN.childSize()) mess_err(nodePath().c_str(),_("Помилка
конфігурації '%s'!"),mConfFile.c_str());
          rootModifCnt = 0;
      }
      catch(TError err) { mess_err(nodePath().c_str(),_("Завантажуємо
кофігураційний файл Помилка: %s"),err.mess.c_str() ); }
    }

    return cmd_help;
}

void TSYS::cfgFileSave( )
{
    ResAlloc res(nodeRes(),true);
    if(!rootModifCnt) return;
    int hd = open(mConfFile.c_str(), O_CREAT|O_TRUNC|O_WRONLY, 0664);
    if(hd < 0) mess_err(nodePath().c_str(),_("Кофігураційний файл '%s' Помилка:
%s"),mConfFile.c_str(),strerror(errno));

    string rezFile = rootN.save(XMLNode::XMLHeader);
    int rez = write(hd, rezFile.data(), rezFile.size());
    if(rez != (int)rezFile.size()) mess_err(nodePath().c_str(),_("Помилка запису
конфігурації. %s"),mConfFile.c_str(),((rez<0)?strerror(errno):""));
    rootModifCnt = 0;
    rootFlTm = time(NULL);
}

void TSYS::cfgPrmLoad( )
{
    //Системні параметри
    mName =
TBDS::genDBGet(nodePath()+"StName",name(),"root",TBDS::UseTranslate);
    mWorkDB = TBDS::genDBGet(nodePath()+"WorkDB",workDB(),"root",TBDS::OnlyCfg);
    setWorkDir(TBDS::genDBGet(nodePath()+"Workdir").c_str());
    setIcoDir(TBDS::genDBGet(nodePath()+"IcoDir",icoDir()));
    setModDir(TBDS::genDBGet(nodePath()+"ModDir",modDir()));
    setSaveAtExit(atoi(TBDS::genDBGet(nodePath()+"SaveAtExit","0").c_str()));
    setSavePeriod(atoi(TBDS::genDBGet(nodePath()+"SavePeriod","0").c_str()));
}

void TSYS::load_()
{
    static bool first_load = true;

    bool cmd_help = cfgFileLoad();
    mess_info(nodePath().c_str(),_("Load!"));
    cfgPrmLoad();
    Mess->load(); //Завантажуємо повідомлення

    if( first_load )
    {
      //> Створюємо підсистему
      add( new TBDS() );
      add( new TSecurity() );
      add( new TTransportS() );
      add( new TProtocolS() );
      add( new TDAQS() );
      add( new TArchiveS() );
      add( new TSpecialS() );
      add( new TUIS() );
      add( new TModSchedul() );

      //> Завантажуємо модулі
      modSchedul().at().load();
      if( !modSchedul().at().loadLibS() )
      {
```

```
            mess_err(nodePath().c_str(),_("Жоден модуль не завантажений. Ваша
конфігурація перервана!"));
            stop();
        }

        //> Завантажуємо базу даних першої  підсистеми
        db().at().load();
        if( !cmd_help ) modSchedul().at().modifG();     // Для перевантаження
спроби від бази даних

        //> Друге завантаження для завантаження від родової БД
        Mess->load();
        cfgPrmLoad();
    }

    //> Пряме завантаження підсистем та модулів
    vector<string> lst;
    list(lst);
    for( unsigned i_a=0; i_a < lst.size(); i_a++ )
      try { at(lst[i_a]).at().load(); }
      catch(TError err)
      {
          mess_err(err.cat.c_str(),"%s",err.mess.c_str());
          mess_err(nodePath().c_str(),_("Помилка завантаження підсистеми
'%s'."),lst[i_a].c_str());
      }

    if( cmd_help ) stop();
    first_load = false;
}

void TSYS::save_( )
{
    char buf[STR_BUF_LEN];

    mess_info(nodePath().c_str(),_("Save!"));

    //> Системні параметри
    getcwd(buf,sizeof(buf));
    TBDS::genDBSet(nodePath()+"StName",mName,"root",TBDS::UseTranslate);
    TBDS::genDBSet(nodePath()+"Workdir",buf);
    TBDS::genDBSet(nodePath()+"IcoDir",icoDir());
    TBDS::genDBSet(nodePath()+"ModDir",modDir());
    TBDS::genDBSet(nodePath()+"SaveAtExit",TSYS::int2str(saveAtExit()));
    TBDS::genDBSet(nodePath()+"SavePeriod",TSYS::int2str(savePeriod()));

    Mess->save(); //Завантажуємо повідомлення
}

int TSYS::start( )
{
    vector<string> lst;
    list(lst);

    mess_info(nodePath().c_str(),_("Start!"));
    for(unsigned i_a=0; i_a < lst.size(); i_a++)
      try { at(lst[i_a]).at().subStart(); }
      catch(TError err)
      {
          mess_err(err.cat.c_str(),"%s",err.mess.c_str());
          mess_err(nodePath().c_str(),_("Помилка запуску
підсистеми'%s'."),lst[i_a].c_str());
      }

    cfgFileScan( true );

    mess_info(nodePath().c_str(),_("Завершення запуску!"));

    unsigned int i_cnt = 1;
```

```
    mStopSignal = 0;
    while(!mStopSignal)
    {
      //> CPU підрахунок частоти
      if(!(i_cnt%(10*1000/STD_WAIT_DELAY)))    clkCalc( );

      //> Кофігураційний файл змін періодичних перевірок
      if(!(i_cnt%(10*1000/STD_WAIT_DELAY)))    cfgFileScan( );

      //> Періодична загальна перевірка бібліотек
      if(modSchedul( ).at().chkPer() && !(i_cnt%(modSchedul(
).at().chkPer()*1000/STD_WAIT_DELAY)))
          modSchedul( ).at().libLoad(modDir(),true);

      //> Періодичний запис змін до  БД
      if(savePeriod() && !(i_cnt%(savePeriod()*1000/STD_WAIT_DELAY))) save();

      //> Кофігураційний файл зберігає необхідні зміни
      if(!(i_cnt%(10*1000/STD_WAIT_DELAY)))    cfgFileSave();

      //> Викликаємо підсистему кожні 10 сек.
      if(!(i_cnt%(10*1000/STD_WAIT_DELAY)))
          for(unsigned i_a=0; i_a < lst.size(); i_a++)
            try { at(lst[i_a]).at().perSYSCall(i_cnt/(1000/STD_WAIT_DELAY)); }
            catch(TError err) { mess_err(err.cat.c_str(),"%s",err.mess.c_str());
}

      usleep(STD_WAIT_DELAY*1000);
      i_cnt++;
    }

    mess_info(nodePath().c_str(),_("Stop!"));
    if(saveAtExit() || savePeriod())     save();
    cfgFileSave();
    for(int i_a=lst.size()-1; i_a >= 0; i_a--)
      try { at(lst[i_a]).at().subStop(); }
      catch(TError err)
      {
          mess_err(err.cat.c_str(),"%s",err.mess.c_str());
          mess_err(nodePath().c_str(),_("Помилка остановки
підсистеми'%s'."),lst[i_a].c_str());
      }

    return mStopSignal;
}

void TSYS::stop( )
{
    mStopSignal = SIGUSR1;
}

bool TSYS::chkSelDB( const string& wDB,  bool isStrong )
{
    if(selDB().empty() && !isStrong) return true;
    if(SYS->selDB() == TBDS::realDBName(wDB)) return true;
    return false;
}

void TSYS::sighandler( int signal )
{
    switch(signal)
    {
      case SIGINT:
          SYS->mStopSignal=signal;
          break;
      case SIGTERM:
          mess_warning(SYS->nodePath().c_str(),_("Отриманий сигнал переривання
роботи. Сервер зупиняється!"));
          SYS->mStopSignal=signal;
```

```
            break;
        case SIGFPE:
            mess_warning(SYS->nodePath().c_str(),_("Виключення плаваючої крапки
спіймане!"));
              exit(1);
            break;
        case SIGCHLD:
        {
            int status;
            pid_t pid = wait(&status);
            if(!WIFEXITED(status) && pid > 0)
              mess_info(SYS->nodePath().c_str(),_("Вивільнено процес-
потомок%d!"),pid);
            break;
        }
        case SIGPIPE:
            //mess_warning(SYS->nodePath().c_str(),_("Сигнал переривання
PIPE!"));
            break;
        case SIGSEGV:
            mess_emerg(SYS->nodePath().c_str(),_("Сигнал помилки від сегменту!"));
            break;
        case SIGABRT:
            mess_emerg(SYS->nodePath().c_str(),_("SCADA_system перервала
роботу!"));
            break;
        case SIGALRM:     break;
        default:
            mess_warning(SYS->nodePath().c_str(),_("Невизначений
сигнал%d!"),signal);
    }
}

void TSYS::cfgFileScan( bool first )
{
    struct stat f_stat;

    if(stat(cfgFile().c_str(),&f_stat) != 0) return;
    bool up = false;
    if(rootCfgFl != cfgFile() || rootFlTm != f_stat.st_mtime) up = true;
    rootCfgFl = cfgFile();
    rootFlTm = f_stat.st_mtime;

    if(up && !first)
    {
      modifG();
      setSelDB("<cfg>");
      load();
      setSelDB("");
    }
}

int64_t TSYS::curTime( )
{
    timeval cur_tm;
    gettimeofday(&cur_tm,NULL);
    return (int64_t)cur_tm.tv_sec*1000000 + cur_tm.tv_usec;
}

bool TSYS::eventWait( bool &m_mess_r_stat, bool exempl, const string &loc,
time_t tm )
{
    time_t t_tm, s_tm;

    t_tm = s_tm = time(NULL);
    while( m_mess_r_stat != exempl )
    {
      time_t c_tm = time(NULL);
      //Контролюємо перерву
```

```
      if( tm && ( c_tm > s_tm+tm) )
      {
          mess_crit(loc.c_str(),_("Timeouted !!!"));
          return true;
      }
      //Створюємо повідомлення
      if( c_tm > t_tm+1 )  //1sec
      {
          t_tm = c_tm;
          mess_info(loc.c_str(),_("Чекаємо подію..."));
      }
      usleep(STD_WAIT_DELAY*1000);
    }
    return false;
}

string TSYS::strSepParse( const string &path, int level, char sep, int *off )
{
    int an_dir = off ? *off : 0;
    int t_lev = 0;
    size_t t_dir;

    if(an_dir >= (int)path.size()) return "";
    while(true)
    {
      t_dir = path.find(sep,an_dir);
      if( t_dir == string::npos )
      {
          if( off ) *off = path.size();
          return (t_lev == level) ? path.substr(an_dir) : "";
      }
      else if( t_lev == level )
      {
          if( off ) *off = t_dir+1;
          return path.substr(an_dir,t_dir-an_dir);
      }
      an_dir = t_dir+1;
      t_lev++;
    }
    return "";
}

string TSYS::strParse( const string &path, int level, const string &sep, int
*off, bool mergeSepSymb )
{
    int an_dir = off ? *off : 0;
    int t_lev = 0;
    size_t t_dir;

    if(an_dir >= (int)path.size() || sep.empty()) return "";
    while(true)
    {
      t_dir = path.find(sep,an_dir);
      if( t_dir == string::npos )
      {
          if( off ) *off = path.size();
          return (t_lev == level) ? path.substr(an_dir) : "";
      }
      else if( t_lev == level )
      {
          if( off ) *off = t_dir+sep.size();
          return path.substr(an_dir,t_dir-an_dir);
      }
      if( mergeSepSymb && sep.size() == 1 )
          for(an_dir = t_dir; an_dir < (int)path.size() && path[an_dir] ==
sep[0]; ) an_dir++;
      else an_dir = t_dir+sep.size();
      t_lev++;
    }
```

```
    return "";
}

string TSYS::strLine( const string &str, int level, int *off )
{
    int an_dir = off ? *off : 0;
    int t_lev = 0, edLnSmbSz = 1;
    size_t t_dir;

    if(an_dir >= (int)str.size()) return "";
    while(true)
    {
      for(t_dir = an_dir; t_dir < str.size(); t_dir++)
          if(str[t_dir] == '\x0D' || str[t_dir] == '\x0A')
          { edLnSmbSz = (str[t_dir] == '\x0D' && ((t_dir+1) < str.size()) &&
str[t_dir+1] == '\x0A') ? 2 : 1; break; }
      if(t_dir >= str.size())
      {
          if(off) *off = str.size();
          return (t_lev==level) ? str.substr(an_dir) : "";
      }
      else if(t_lev == level)
      {
          if(off) *off = t_dir+edLnSmbSz;
          return str.substr(an_dir,t_dir-an_dir);
      }
      an_dir = t_dir+edLnSmbSz;
      t_lev++;
    }
    return "";
}

string TSYS::pathLev( const string &path, int level, bool encode, int *off )
{
    int an_dir = off ? *off : 0;
    int t_lev = 0;
    size_t t_dir;

    //> Перший роздільний прохід
    while(an_dir < (int)path.size() && path[an_dir]=='/') an_dir++;
    if(an_dir >= (int)path.size()) return "";
    //> Шлях рівня процесу
    while(true)
    {
      t_dir = path.find("/",an_dir);
      if( t_dir == string::npos )
      {
          if( off ) *off = path.size();
          return (t_lev == level) ? ( encode ?
TSYS::strDecode(path.substr(an_dir),TSYS::PathEl) : path.substr(an_dir) ) : "";
      }
      else if( t_lev == level )
      {
          if( off ) *off = t_dir;
          return encode ? TSYS::strDecode(path.substr(an_dir,t_dir-
an_dir),TSYS::PathEl) : path.substr(an_dir,t_dir-an_dir);
      }
      an_dir = t_dir;
      t_lev++;
      while(an_dir < (int)path.size() && path[an_dir]=='/') an_dir++;
    }
}

string TSYS::path2sepstr( const string &path, char sep )
{
    string rez, curv;
    int off = 0;
    while( !(curv=TSYS::pathLev(path,0,false,&off)).empty() )
      rez+=curv+sep;
```

```
    if(!rez.empty())    rez.resize(rez.size()-1);

    return rez;
}

string TSYS::sepstr2path( const string &str, char sep )
{
    string rez, curv;
    int off = 0;
    while( !(curv=TSYS::strSepParse(str,0,sep,&off)).empty() )
      rez+="/"+curv;

    return rez;
}

string TSYS::strEncode( const string &in, TSYS::Code tp, const string &symb )
{
    int i_sz;
    string sout;

    switch(tp)
    {
      case TSYS::PathEl:
          sout = in;
          for( i_sz = 0; i_sz < (int)sout.size(); i_sz++ )
            switch( sout[i_sz] )
            {
                case '/': sout.replace(i_sz,1,"%2f"); i_sz+=2; break;
                case '%': sout.replace(i_sz,1,"%25"); i_sz+=2; break;
            }
          break;
      case TSYS::HttpURL:
          sout = in;
          for( i_sz = 0; i_sz < (int)sout.size(); i_sz++ )
            switch( sout[i_sz] )
            {
                case '%': sout.replace(i_sz,1,"%25"); i_sz+=2; break;
                case ' ': sout.replace(i_sz,1,"%20"); i_sz+=2; break;
                case '\t': sout.replace(i_sz,1,"%09"); i_sz+=2; break;
                default:
                  if( sout[i_sz]&0x80 )
                  {
                      char buf[4];
                      snprintf(buf,sizeof(buf),"%%%02X",(unsigned
char)sout[i_sz]);
                      sout.replace(i_sz,1,buf);
                      i_sz+=2;
                      break;
                  }
            }
          break;
      case TSYS::Html:
          sout.reserve(in.size()+10);
          for( i_sz = 0; i_sz < (int)in.size(); i_sz++ )
            switch( in[i_sz] )
            {
                case '>':     sout+="&gt;";     break;
                case '<':     sout+="&lt;";     break;
                case '"':     sout+="&quot;";   break;
                case '&':     sout+="&amp;";    break;
                case '\'':    sout+="&apos;";   break;
                default:      sout+=in[i_sz];
            }
          break;
      case TSYS::JavaSc:
          sout.reserve(in.size()+10);
          for( i_sz = 0; i_sz < (int)in.size(); i_sz++ )
            switch( in[i_sz] )
            {
```

```
                case '\n':    sout+="\\n";      break;
                default:      sout+=in[i_sz];
              }
            break;
        case TSYS::SQL:
            sout.reserve(in.size()+10);
            for( i_sz = 0; i_sz < (int)in.size(); i_sz++ )
              switch( in[i_sz] )
              {
                case '\'':    sout+="\\'";      break;
                case '\"':    sout+="\\\"";     break;
                case '`':     sout+="\\`";      break;
                case '\\':    sout+="\\\\";     break;
                default:      sout+=in[i_sz];
              }
            break;
        case TSYS::Custom:
            sout.reserve(in.size()+10);
            for( i_sz = 0; i_sz < (int)in.size(); i_sz++ )
            {
              unsigned i_smb;
              for(i_smb = 0; i_smb < symb.size(); i_smb++)
                if(in[i_sz] == symb[i_smb])
                {
                  char buf[4];
                  sprintf(buf,"%%%02X",(unsigned char)in[i_sz]);
                  sout += buf;
                  break;
                }
              if(i_smb >= symb.size()) sout += in[i_sz];
            }
            break;
        case TSYS::base64:
        {
            sout.reserve(in.size()+in.size()/4+in.size()/57+10);
            const char *base64alph =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/";
            for( i_sz = 0; i_sz < (int)in.size(); i_sz+=3 )
            {
              if(i_sz && !(i_sz%57))  sout.push_back('\n');
              sout.push_back(base64alph[(unsigned char)in[i_sz]>>2]);
              if((i_sz+1) >= (int)in.size())
              {
                sout.push_back(base64alph[((unsigned char)in[i_sz]&0x03)<<4]);
                sout += "==";
              }
              else
              {
                sout.push_back(base64alph[(((unsigned
char)in[i_sz]&0x03)<<4)|((unsigned char)in[i_sz+1]>>4)]);
                if((i_sz+2) >= (int)in.size())
                {
                  sout.push_back(base64alph[((unsigned
char)in[i_sz+1]&0x0F)<<2]);
                  sout.push_back('=');
                }
                else
                {
                  sout.push_back(base64alph[(((unsigned
char)in[i_sz+1]&0x0F)<<2)|((unsigned char)in[i_sz+2]>>6)]);
                  sout.push_back(base64alph[(unsigned char)in[i_sz+2]&0x3F]);
                }
              }
            }
            break;
        }
        case TSYS::FormatPrint:
            sout = in;
            for(i_sz = 0; i_sz < (int)sout.size(); i_sz++)
```

```
            if(sout[i_sz] == '%') { sout.replace(i_sz,1,"%%"); i_sz++; }
          break;
      case TSYS::oscdID:
          sout.reserve(in.size());
          for(i_sz = 0; i_sz < (int)in.size(); i_sz++)
            switch(in[i_sz])
            {
                case ' ': case '/': case '\\': case '&': case '(':
                case ')': case '[': case ']': case '!': case '~':
                case '`': case '@': case '%': case '^': case '-':
                case '+': case '=': case '*': case '{': case '}':
                case ':': case ';': case '"': case '\'': case '<':
                case '>': case '?': case '.': case ',':
                  sout+="_";  break;
                default:      sout+=in[i_sz];
            }
          break;
      case TSYS::Bin:
      {
          string svl, evl;
          sout.reserve(in.size());
          for(int off = 0; (svl=TSYS::strSepParse(in,0,'\n',&off)).size(); )
            for(int offE = 0; (evl=TSYS::strSepParse(svl,0,' ',&offE)).size(); )
                sout+=(char)strtol(evl.c_str(),NULL,16);
          break;
      }
      case TSYS::Reverse:
          for(i_sz = in.size()-1; i_sz >= 0; i_sz--) sout += in[i_sz];
          break;
      case TSYS::ShieldSimb:
          sout.reserve(in.size());
          for(i_sz = 0; i_sz < (int)in.size(); i_sz++)
            if(in[i_sz] == '\\' && i_sz < ((int)in.size()-1))
            {
                switch(in[i_sz+1])
                {
                  case 'a':   sout += '\a';     break;
                  case 'b':   sout += '\b';     break;
                  case 'f':   sout += '\f';     break;
                  case 'n':   sout += '\n';     break;
                  case 'r':   sout += '\r';     break;
                  case 't':   sout += '\t';     break;
                  case 'v':   sout += '\v';     break;
                  case 'x': case 'X':
                      if((i_sz+3) < (int)in.size() && isxdigit(in[i_sz+2]) &&
isxdigit(in[i_sz+3]))
                      { sout +=
(char)strtol(in.substr(i_sz+2,2).c_str(),NULL,16); i_sz += 2; }
                      else sout += in[i_sz+1];
                          break;
                  default:
                          if((i_sz+3) < (int)in.size() && in[i_sz+1] >= '0' &&
in[i_sz+1] <= '7' &&
                                                          in[i_sz+2] >= '0' &&
in[i_sz+2] <= '7' &&
                                                          in[i_sz+3] >= '0' &&
in[i_sz+3] <= '7')
                          { sout +=
(char)strtol(in.substr(i_sz+1,3).c_str(),NULL,8); i_sz += 2; }
                          else sout += in[i_sz+1];
                }
                i_sz++;
            }else sout += in[i_sz];
          break;
    }
    return sout;
}

unsigned char TSYS::getBase64Code(unsigned char asymb)
```

```
{
    switch(asymb)
    {
      case 'A' ... 'Z': return asymb-(unsigned char)'A';
      case 'a' ... 'z': return 26+asymb-(unsigned char)'a';
      case '0' ... '9': return 52+asymb-(unsigned char)'0';
      case '+':         return 62;
      case '/':         return 63;
    }
    return 0;
}

string TSYS::strDecode( const string &in, TSYS::Code tp )
{
    unsigned i_sz;
    string sout;

    switch(tp)
    {
      case TSYS::PathEl: case TSYS::HttpURL: case TSYS::Custom:
          sout.reserve(in.size());
          for(i_sz = 0; i_sz < in.size(); i_sz++)
            switch(in[i_sz])
            {
                case '%':
                  if(i_sz+2 < in.size())
                  {
                      sout += (char)strtol(in.substr(i_sz+1,2).c_str(),NULL,16);
                      i_sz += 2;
                  }else sout += in[i_sz];
                  break;
                default: sout += in[i_sz];
            }
          break;
      case TSYS::base64:
          sout.reserve(in.size());
          for( i_sz = 0; i_sz < in.size(); )
          {
            if(in[i_sz] == '\n')    i_sz+=sizeof('\n');
            if((i_sz+3) < in.size())
                if( in[i_sz+1] != '=' )
                {
                  char w_code1 = TSYS::getBase64Code(in[i_sz+1]);

      sout.push_back((TSYS::getBase64Code(in[i_sz])<<2)|(w_code1>>4));
                  if( in[i_sz+2] != '=' )
                  {
                      char w_code2 = TSYS::getBase64Code(in[i_sz+2]);
                      sout.push_back((w_code1<<4)|(w_code2>>2));
                      if( in[i_sz+3] != '=' )

      sout.push_back((w_code2<<6)|TSYS::getBase64Code(in[i_sz+3]));
                  }
                }
            i_sz+=4;
          }
          break;
      case TSYS::Bin:
          sout.reserve(in.size());
          for( i_sz = 0; i_sz < in.size(); i_sz++ )
            sout += TSYS::strMess(((i_sz+1)%16)?"%0.2x ":"%0.2x\n",(unsigned
char)in[i_sz]);
          break;
      default: sout = in;     break;
    }

    return sout;
}
```

```
string TSYS::strCompr( const string &in, int lev )
{
    z_stream strm;

    if( in.empty() )    return "";

    strm.zalloc = Z_NULL;
    strm.zfree = Z_NULL;
    strm.opaque = Z_NULL;

    if( deflateInit(&strm,lev) != Z_OK ) return "";

    uLongf comprLen = deflateBound(&strm,in.size());
    char out[comprLen];

    strm.next_in = (Bytef*)in.data();
    strm.avail_in = (uInt)in.size();
    strm.next_out = (Bytef*)out;
    strm.avail_out = comprLen;

    if( deflate(&strm, Z_FINISH) != Z_STREAM_END )
    {
      deflateEnd(&strm);
      return "";
    }

    comprLen = strm.total_out;

    deflateEnd(&strm);

    return string(out,comprLen);
}

string TSYS::strUncompr( const string &in )
{
    int ret;
    z_stream strm;
    unsigned char out[STR_BUF_LEN];
    string rez;

    if( in.empty() )    return "";

    strm.zalloc = Z_NULL;
    strm.zfree = Z_NULL;
    strm.opaque = Z_NULL;

    if( inflateInit(&strm) != Z_OK )      return "";

    strm.avail_in = in.size();
    strm.next_in = (Bytef*)in.data();
    do
    {
      strm.avail_out = sizeof(out);
      strm.next_out = out;
      ret=inflate(&strm,Z_NO_FLUSH);
      if( ret == Z_STREAM_ERROR || ret == Z_NEED_DICT || ret == Z_DATA_ERROR ||
ret == Z_MEM_ERROR )
          break;
      rez.append((char*)out,sizeof(out)-strm.avail_out);
    } while( strm.avail_out == 0 );

    inflateEnd(&strm);

    if( ret != Z_STREAM_END ) return "";

    return rez;
}

float TSYS::floatLE(float in)
```

```
{
#if __BYTE_ORDER == __BIG_ENDIAN
    ieee754_double ieee754_be;
    union ieee754_le
    {
      float f;
      struct
        {
          unsigned int mantissa:23;
          unsigned int exponent:8;
          unsigned int negative:1;
      } ieee;
    } ieee754_le;

    ieee754_be.f = in;
    ieee754_le.ieee.mantissa  = ieee754_be.ieee.mantissa;
    ieee754_le.ieee.exponent  = ieee754_be.ieee.exponent;
    ieee754_le.ieee.negative  = ieee754_be.ieee.negative;

    return ieee754_le.f;
#endif

    return in;
}

float TSYS::floatLErev(float in)
{
#if __BYTE_ORDER == __BIG_ENDIAN
    ieee754_double ieee754_be;
    union ieee754_le
    {
      float f;
      struct
        {
          unsigned int mantissa:23;
          unsigned int exponent:8;
          unsigned int negative:1;
      } ieee;
    } ieee754_le;

    ieee754_le.f = in;
    ieee754_be.ieee.mantissa  = ieee754_le.ieee.mantissa;
    ieee754_be.ieee.exponent  = ieee754_le.ieee.exponent;
    ieee754_be.ieee.negative  = ieee754_le.ieee.negative;

    return ieee754_be.f;
#endif

    return in;
}

double TSYS::doubleLE(double in)
{
#if __BYTE_ORDER == __BIG_ENDIAN || __FLOAT_WORD_ORDER == __BIG_ENDIAN
    ieee754_double ieee754_be;
    union ieee754_le
    {
      double d;
      struct
        {
          unsigned int mantissa1:32;
          unsigned int mantissa0:20;
          unsigned int exponent:11;
          unsigned int negative:1;
      } ieee;
    } ieee754_le;

    ieee754_be.d = in;
    ieee754_le.ieee.mantissa0 = ieee754_be.ieee.mantissa0;
```

```
    ieee754_le.ieee.mantissa1 = ieee754_be.ieee.mantissa1;
    ieee754_le.ieee.exponent  = ieee754_be.ieee.exponent;
    ieee754_le.ieee.negative  = ieee754_be.ieee.negative;

    return ieee754_le.d;
#endif

    return in;
}

double TSYS::doubleLErev(double in)
{
#if __BYTE_ORDER == __BIG_ENDIAN || __FLOAT_WORD_ORDER == __BIG_ENDIAN
    ieee754_double ieee754_be;
    union ieee754_le
    {
      double d;
      struct
        {
          unsigned int mantissa1:32;
          unsigned int mantissa0:20;
          unsigned int exponent:11;
          unsigned int negative:1;
      } ieee;
    } ieee754_le;

    ieee754_le.d = in;
    ieee754_be.ieee.mantissa0 = ieee754_le.ieee.mantissa0;
    ieee754_be.ieee.mantissa1 = ieee754_le.ieee.mantissa1;
    ieee754_be.ieee.exponent  = ieee754_le.ieee.exponent;
    ieee754_be.ieee.negative  = ieee754_le.ieee.negative;

    return ieee754_be.d;
#endif

    return in;
}

long TSYS::HZ()
{
    return sysconf(_SC_CLK_TCK);
}

bool TSYS::cntrEmpty( )
{
    ResAlloc res( nodeRes(), false );
    return mCntrs.empty();
}

double TSYS::cntrGet( const string &id )
{
    ResAlloc res( nodeRes(), false );
    map<string,double>::iterator icnt = mCntrs.find(id);
    if( icnt == mCntrs.end() )      return 0;
    return icnt->second;
}

void TSYS::cntrSet( const string &id, double vl )
{
    ResAlloc res( nodeRes(), true );
    mCntrs[id] = vl;
}

void TSYS::taskCreate( const string &path, int priority, void
*(*start_routine)(void *), void *arg, int wtm, pthread_attr_t *pAttr, bool
*startSt )
{
    int detachStat = 0;
    pthread_t procPthr;
```

```
    pthread_attr_t locPAttr, *pthr_attr;
    map<string,STask>::iterator ti;

    ResAlloc res(taskRes, true);
    for(time_t c_tm = time(NULL); mTasks.find(path) != mTasks.end(); )
    {
      if(time(NULL) >= (c_tm+wtm)) throw TError(nodePath().c_str(),_("Завдання
'%s' вже присутнє!"),path.c_str());
      res.release();
      usleep(10000);
      res.request(true);
    }
    STask &htsk = mTasks[path];
    htsk.path = path;
    htsk.task = start_routine;
    htsk.taskArg = arg;
    htsk.flgs = 0;
    res.release();

    if(pAttr) pthr_attr = pAttr;
    else
    {
      pthr_attr = &locPAttr;
      pthread_attr_init(pthr_attr);
    }
    pthread_attr_setinheritsched(pthr_attr, PTHREAD_EXPLICIT_SCHED);
    struct sched_param prior;
    prior.sched_priority = 0;

    int policy = SCHED_OTHER;
#if __GLIBC_PREREQ(2,4)
    if(priority < 0)    policy = SCHED_BATCH;
#endif
    if(priority > 0 /*&& SYS->user() == "root"*/)    policy = SCHED_RR;
    pthread_attr_setschedpolicy(pthr_attr, policy);
    prior.sched_priority =
vmax(sched_get_priority_min(policy),vmin(sched_get_priority_max(policy),priority
));
    pthread_attr_setschedparam(pthr_attr,&prior);

    try
    {
      pthread_attr_getdetachstate(pthr_attr,&detachStat);
      if(detachStat == PTHREAD_CREATE_DETACHED) htsk.flgs |= STask::Detached;
      int rez = pthread_create(&procPthr, pthr_attr, taskWrap, &htsk);
      if(rez == EPERM)
      {
          mess_warning(nodePath().c_str(),_("No permission for create real-time
policy. Default thread is created!"));
          policy = SCHED_OTHER;
          pthread_attr_setschedpolicy(pthr_attr, policy);
          prior.sched_priority = 0;
          pthread_attr_setschedparam(pthr_attr,&prior);
          rez = pthread_create(&procPthr, pthr_attr, taskWrap, &htsk);
      }
      if(!pAttr) pthread_attr_destroy(pthr_attr);

      if(rez) throw TError(nodePath().c_str(), _("Завдання створило
помилку%d."), rez);

      //> Чекаємо закінчення ініціалізації структури потоку для не відривних
завдань
      while(!(htsk.flgs&STask::Detached) && !htsk.thr) pthread_yield();
      //> Чекаємо запуск статусу
      for(time_t c_tm = time(NULL); !(htsk.flgs&STask::Detached) && startSt &&
!(*startSt); )
      {
          if(time(NULL) >= (c_tm+wtm)) throw
TError(nodePath().c_str(),_("Завдання '%s' запуск відкладений!"),path.c_str());
```

```
            usleep(STD_WAIT_DELAY *1000);
        }
    }
    catch(TError)
    {
      res.request(true);
      mTasks.erase(path);
      res.release();
      throw;
    }
}

void TSYS::taskDestroy( const string &path, bool *endrunCntr, int wtm, bool
noSignal )
{
    ResAlloc res(taskRes, false);
    map<string,STask>::iterator it = mTasks.find(path);
    if(it == mTasks.end()) return;
    pthread_t thr = it->second.thr;
    res.release();

    if(endrunCntr) *endrunCntr = true;
    if(!noSignal) pthread_kill(thr, SIGALRM);

    //> Чекаємо завершення завдання та повторюємо відправлення SIGALRM
    time_t t_tm, s_tm;
    t_tm = s_tm = time(NULL);
    while(!(it->second.flgs&STask::FinishTask))
    {
        if(!noSignal) pthread_kill(thr, SIGALRM);
      time_t c_tm = time(NULL);
        //Контролюємо перерву
        if(wtm && (c_tm > (s_tm+wtm)))
        {
            mess_crit((nodePath()+path+": stop").c_str(),_("Timeouted !!!"));
          throw TError(nodePath().c_str(),_(«завдання '%s' is not
stopped!"),path.c_str());
        }
      //Створюємо повідомлення
        if(c_tm > t_tm+1)  //1sec
        {
            t_tm = c_tm;
            mess_info((nodePath()+path+": stop").c_str(),_("Чекаємо подію..."));

        }
        usleep(STD_WAIT_DELAY*1000);
    }

    if(!(it->second.flgs&STask::Detached)) pthread_join(thr, NULL);

    res.request(true);
    mTasks.erase(it);
}

void *TSYS::taskWrap( void *stas )
{
    //> Беремо тимчасову структуру завдання
    STask *tsk = (STask *)stas;
    pthread_setspecific(TSYS::sTaskKey, tsk);

    //> Запам'ятовуємо параметри виклику
    void *(*wTask) (void *) = tsk->task;
    void *wTaskArg = tsk->taskArg;

    //> Отримуємо поточні політику і пріорітет
    int policy;
    struct sched_param param;
    pthread_getschedparam(pthread_self(), &policy, &param);
    tsk->policy = policy;
```

```
    tsk->prior = param.sched_priority;

#if __GLIBC_PREREQ(2,4)
    //> Отримуємо і завантажуємо CPU установлення
    if(SYS->multCPU() && !(tsk->flgs & STask::Detached))
    {
      tsk->cpuSet = TBDS::genDBGet(SYS->nodePath()+"CpuSet:"+tsk->path);
      cpu_set_t cpuset;
      CPU_ZERO(&cpuset);
      string sval;
      bool cpuSetOK = false;
      for(int off = 0; (sval=TSYS::strParse(tsk->cpuSet,0,":",&off)).size();
cpuSetOK = true)
          CPU_SET(atoi(sval.c_str()),&cpuset);
      if(cpuSetOK) pthread_setaffinity_np(pthread_self(), sizeof(cpu_set_t),
&cpuset);
    }
    else if(SYS->multCPU() && (tsk->flgs & STask::Detached)) tsk->cpuSet = "NA";
#endif

    //> Закінчуємо установки та ініціалізуємо індікатор закінчення
    tsk->tid = syscall(SYS_gettid);
    tsk->thr = pthread_self();

    //> Викликаємо робочі завдання
    void *rez = NULL;
    try { rez = wTask(wTaskArg); }
    catch(TError err)
    {
      mess_err(err.cat.c_str(),err.mess.c_str());
      mess_err(SYS->nodePath().c_str(),_("Завдання %u несподівано закінчено
виключенням."),tsk->thr);
    }

    //> Відмічаємо закінчення завдання
    tsk->flgs |= STask::FinishTask;

    //> Переміщуємо об’єкти завдання окремо
    if(tsk->flgs & STask::Detached) SYS->taskDestroy(tsk->path, NULL);

    return rez;
}

void TSYS::taskSleep( int64_t per, time_t cron )
{
    struct timespec sp_tm;
    STask *stsk = (STask*)pthread_getspecific(sTaskKey);

    if(!cron)
    {
      if(!per) per = 1000000000;
      clock_gettime(CLOCK_REALTIME,&sp_tm);
      int64_t end_tm = (int64_t)sp_tm.tv_sec*1000000000+sp_tm.tv_nsec;
      int64_t pnt_tm = (end_tm/per + 1)*per;
      do
      {
          sp_tm.tv_sec = pnt_tm/1000000000; sp_tm.tv_nsec = pnt_tm%1000000000;
          if(clock_nanosleep(CLOCK_REALTIME,TIMER_ABSTIME,&sp_tm,NULL))
      return;
          clock_gettime(CLOCK_REALTIME,&sp_tm);
      }while(((int64_t)sp_tm.tv_sec*1000000000+sp_tm.tv_nsec) < pnt_tm);

      if(stsk)
      {
          stsk->tm_beg = stsk->tm_per;
          stsk->tm_end = end_tm;
          stsk->tm_per = (int64_t)sp_tm.tv_sec*1000000000+sp_tm.tv_nsec;
      }
    }
```

```
    else
    {
      time_t end_tm = time(NULL);
      while(time(NULL) < cron && usleep(1000000) == 0) ;
      if(stsk)
      {
          stsk->tm_beg = stsk->tm_per;
          stsk->tm_end = 1000000000ll*end_tm;
          stsk->tm_per = 1000000000ll*time(NULL);
      }
    }
}

time_t TSYS::cron( const string &vl, time_t base )
{
    string cronEl, tEl;
    int vbeg, vend, vstep, vm;

    time_t ctm = base?base:time(NULL);
    struct tm ttm;
    localtime_r(&ctm,&ttm);
    ttm.tm_sec = 0;

reload:
    bool isReload = false;

    //> Хвилини check
    cronEl = TSYS::strSepParse(vl,0,' ');
    vm = 200;
    for(int eoff = 0; (tEl=TSYS::strSepParse(cronEl,0,',',&eoff)).size(); )
    {
      vbeg = vend = -1; vstep = 0;
      sscanf(tEl.c_str(),"%d-%d/%d",&vbeg,&vend,&vstep);
      if(vbeg < 0) { sscanf(tEl.c_str(),"*/%d",&vstep); vbeg=0; vend=59; }
      if(vend < 0) vm = vmin(vm,vbeg+((ttm.tm_min>=vbeg)?60:0));
      else if((vbeg=vmax(0,vbeg)) < (vend=vmin(59,vend)))
      {
          if(ttm.tm_min < vbeg) vm = vmin(vm,vbeg);
          else if((vstep>1 && ttm.tm_min >= (vbeg+((vend-vbeg)/vstep)*vstep)) ||
(vstep <= 0 && ttm.tm_min >= vend))
            vm = vmin(vm,vbeg+60);
          else if(vstep>1 ) vm = vmin(vm, vbeg + vstep*(((ttm.tm_min+1)-
vbeg)/vstep + ((((ttm.tm_min+1)-vbeg)%vstep)?1:0)));
          else vm = vmin(vm, ttm.tm_min+1);
      }
      if(vm == ttm.tm_min+1) break;
    }
    ttm.tm_min = vm;
    mktime(&ttm);

    //> Перевіряємо час
    cronEl = TSYS::strSepParse(vl,1,' ');
    vm = 200;
    for(int eoff = 0; (tEl=TSYS::strSepParse(cronEl,0,',',&eoff)).size(); )
    {
      vbeg = vend = -1; vstep = 0;
      sscanf(tEl.c_str(),"%d-%d/%d",&vbeg,&vend,&vstep);
      if(vbeg < 0) { sscanf(tEl.c_str(),"*/%d",&vstep); vbeg=0; vend=23; }
      if(vend < 0) vm = vmin(vm,vbeg+((ttm.tm_hour>vbeg)?24:0));
      else if((vbeg=vmax(0,vbeg)) < (vend=vmin(23,vend)))
      {
          if(ttm.tm_hour < vbeg) vm = vmin(vm,vbeg);
          else if((vstep>1 && ttm.tm_hour > (vbeg+((vend-vbeg)/vstep)*vstep)) ||
(vstep <= 0 && ttm.tm_hour > vend))
            vm = vmin(vm,vbeg+24);
          else if(vstep>1 ) vm = vmin(vm, vbeg + vstep*((ttm.tm_hour-vbeg)/vstep
+ (((ttm.tm_hour-vbeg)%vstep)?1:0)));
          else vm = vmin(vm, ttm.tm_hour);
      }
```

```
      if(vm == ttm.tm_hour) break;
    }
    isReload = (vm != 200 && ttm.tm_hour!=vm);
    ttm.tm_hour = vm;
    mktime(&ttm);
    if(isReload)  { ttm.tm_min = -1; goto reload; }

    //> Перевіряємо день
    cronEl = TSYS::strSepParse(vl,2,' ');
    string cronElw = TSYS::strSepParse(vl,4,' ');
    vm = 200;
    if(cronEl != "*")
      for(int eoff = 0; (tEl=TSYS::strSepParse(cronEl,0,',',&eoff)).size(); )
      {
          vbeg = vend = -1; vstep = 0;
          sscanf(tEl.c_str(),"%d-%d/%d",&vbeg,&vend,&vstep);
          if(vbeg < 0) { sscanf(tEl.c_str(),"*/%d",&vstep); vbeg=1; vend=31; }
          if(vend < 0) vm = vmin(vm,vbeg+((ttm.tm_mday>vbeg)?31:0));
          else if((vbeg=vmax(1,vbeg)) < (vend=vmin(31,vend)))
          {
            if(ttm.tm_mday < vbeg) vm = vmin(vm,vbeg);
            else if((vstep>1 && ttm.tm_mday > (vbeg+((vend-vbeg)/vstep)*vstep))
|| (vstep <= 0 && ttm.tm_mday > vend))
                vm = vmin(vm,vbeg+31);
            else if(vstep>1 ) vm = vmin(vm, vbeg + vstep*((ttm.tm_mday-
vbeg)/vstep + (((ttm.tm_mday-vbeg)%vstep)?1:0)));
            else vm = vmin(vm, ttm.tm_mday);
          }
          if(vm == ttm.tm_mday) break;
        }
    if(cronEl == "*" || (cronElw != "*" && !cronElw.empty()))
      for(int eoff = 0; (tEl=TSYS::strSepParse(cronElw,0,',',&eoff)).size(); )
      {
          vbeg = vend = -1; vstep = 0;
          sscanf(tEl.c_str(),"%d-%d/%d",&vbeg,&vend,&vstep);
          if(vbeg < 0) { sscanf(tEl.c_str(),"*/%d",&vstep); vbeg=0; vend=6; }
          if(vend < 0) vm = vmin(vm, ttm.tm_mday - ttm.tm_wday +
vbeg+((ttm.tm_wday>vbeg)?7:0));
          else if((vbeg=vmax(0,vbeg)) < (vend=vmin(6,vend)))
          {
            if(ttm.tm_wday < vbeg) vm = vmin(vm, ttm.tm_mday - ttm.tm_wday +
vbeg);
            else if((vstep>1 && ttm.tm_wday > (vbeg+((vend-vbeg)/vstep)*vstep))
|| (vstep <= 0 && ttm.tm_wday > vend))
              vm = vmin(vm, ttm.tm_mday - ttm.tm_wday + vbeg+7);
            else if(vstep>1) vm = vmin(vm, ttm.tm_mday - ttm.tm_wday + vbeg +
vstep*((ttm.tm_wday-vbeg)/vstep + (((ttm.tm_wday-vbeg)%vstep)?1:0)));
            else vm = vmin(vm, ttm.tm_mday);
          }
          if(vm == ttm.tm_mday) break;
      }
    isReload = (vm!=200 && ttm.tm_mday!=vm);
    if(vm <= 31) ttm.tm_mday = vm;
    else { ttm.tm_mday = vm-31; ttm.tm_mon++; }
    mktime(&ttm);
    if(isReload)  { ttm.tm_min = -1; ttm.tm_hour = 0; goto reload; }

    //> Перевіряємо місяць
    cronEl = TSYS::strSepParse(vl,3,' ');
    vm = 200;
    for(int eoff = 0; (tEl=TSYS::strSepParse(cronEl,0,',',&eoff)).size(); )
    {
      vbeg = vend = -1; vstep = 0;
      sscanf(tEl.c_str(),"%d-%d/%d",&vbeg,&vend,&vstep);
      if(vbeg < 0) { sscanf(tEl.c_str(),"*/%d",&vstep); vbeg=1; vend=12; }
      if(vend < 0) vm = vmin(vm,vbeg+(((ttm.tm_mon+1)>vbeg)?12:0));
      else if((vbeg=vmax(1,vbeg)) < (vend=vmin(12,vend)))
      {
          if((ttm.tm_mon+1) < vbeg) vm = vmin(vm,vbeg);
```

```
          else if((vstep>1 && (ttm.tm_mon+1) > (vbeg+((vend-vbeg)/vstep)*vstep))
|| (vstep <= 0 && (ttm.tm_mon+1) > vend))
            vm = vmin(vm,vbeg+12);
          else if(vstep>1) vm = vmin( vm, vbeg + vstep*(((ttm.tm_mon+1)-
vbeg)/vstep + ((((ttm.tm_mon+1)-vbeg)%vstep)?1:0)));
          else vm = vmin(vm, ttm.tm_mon+1);
      }
      if(vm == (ttm.tm_mon+1)) break;
    }
    isReload = (vm!=200 && ttm.tm_mon!=(vm-1));
    ttm.tm_mon = vm-1;
    mktime(&ttm);
    if(isReload)  { ttm.tm_min = -1; ttm.tm_hour = 0; ttm.tm_mday = 1; goto
reload; }

    return mktime(&ttm);
}

TVariant TSYS::objFuncCall( const string &iid, vector<TVariant> &prms, const
string &user )
{
    // int message(string cat, int level, string mess) - форматуємо системне
повідомлення  <mess> з категорією <cat>, рівнем <level>
    //  cat - повідомлення категорії
    //  level - повідомлення рівня
    //  mess - повідомлення тексту
    if(iid == "message" && prms.size() >= 3)   { message(
prms[0].getS().c_str(), (TMess::Type)prms[1].getI(), "%s",
prms[2].getS().c_str() ); return 0; }
    // int messDebug(string cat, string mess) - форматуємо системне повідомлення
<mess> з категорією <cat> і відповідний рівень
    //  cat - повідомлення категорії
    //  mess - повідомлення тексту
    if(iid == "messDebug" && prms.size() >= 2) { mess_debug(
prms[0].getS().c_str(), "%s", prms[1].getS().c_str() ); return 0; }
    if(iid == "messInfo" && prms.size() >= 2)  { mess_info(
prms[0].getS().c_str(), "%s", prms[1].getS().c_str() ); return 0; }
    if(iid == "messNote" && prms.size() >= 2)  { mess_note(
prms[0].getS().c_str(), "%s", prms[1].getS().c_str() ); return 0; }
    if(iid == "messWarning" && prms.size() >= 2){ mess_warning(
prms[0].getS().c_str(), "%s", prms[1].getS().c_str() ); return 0; }
    if(iid == "messErr" && prms.size() >= 2)   { mess_err(
prms[0].getS().c_str(), "%s", prms[1].getS().c_str() ); return 0; }
    if(iid == "messCrit" && prms.size() >= 2)  { mess_crit(
prms[0].getS().c_str(), "%s", prms[1].getS().c_str() ); return 0; }
    if(iid == "messAlert" && prms.size() >= 2) { mess_alert(
prms[0].getS().c_str(), "%s", prms[1].getS().c_str() ); return 0; }
    if(iid == "messEmerg" && prms.size() >= 2) { mess_emerg(
prms[0].getS().c_str(), "%s", prms[1].getS().c_str() ); return 0; }
    // string system(string cmd, bool noPipe = false) - викликаємо консольні
команди <cmd> для повернення у ОС  результатів з каналів
    //  cmd - текс команди
    //  noPipe - результат блокують для другорядного виклику
    if(iid == "system" && prms.size() >= 1)
    {
      if(prms.size() >= 2 && prms[1].getB()) return
system(prms[0].getS().c_str());
      FILE *fp = popen(prms[0].getS().c_str(),"r");
      if(!fp) return string("");

      char buf[STR_BUF_LEN];
      string rez;
      for(int r_cnt = 0; (r_cnt=fread(buf,1,sizeof(buf),fp)); )
          rez.append(buf,r_cnt);

      pclose(fp);
      return rez;
    }
    // string fileRead( string file ) - Повертаємо <file> контент у рядку.
```

```
    if(iid == "fileRead" && prms.size() >= 1)
    {
      char buf[STR_BUF_LEN];
      string rez;
        int hd = open(prms[0].getS().c_str(),O_RDONLY);
      if(hd != -1)
      {
          for(int len = 0; (len=read(hd,buf,sizeof(buf))) > 0; )
rez.append(buf,len);
          close(hd);
      }
      return rez;
    }
    // int fileWrite( string file, string str, bool append = false ) - Записуємо
<str> до <file>, переміщуємо представлення, або <append>.
    //        Return wrote bytes count.
    if(iid == "fileWrite" && prms.size() >= 2)
    {
      int wcnt = 0, wflags = O_WRONLY|O_CREAT|O_TRUNC;
      string val = prms[1].getS();
      if(prms.size() >= 3 && prms[2].getB()) wflags = O_WRONLY|O_CREAT|O_APPEND;
        int hd = open(prms[0].getS().c_str(), wflags, 0664);
      if(hd != -1)
      {
          wcnt = write(hd,val.data(),val.size());
          close(hd);
      }
      return wcnt;
    }
    // XMLNodeObj XMLNode(string name = "") - створюємо XML об’єкт вузлів з
іменем <name>
    //  name - XML ім’я вузлу
    if(iid == "XMLNode") return new XMLNodeObj((prms.size()>=1) ? prms[0].getS()
: "");
    // string cntrReq(XMLNodeObj req, string stat = "") - запит інтерфейсу
управління до системи через XML
    //  req - запити XML вузлів
    //  stat - переміщуємо SCADA system-робочу станцію для запиту
    if(iid == "cntrReq" && prms.size() >= 1)
    {
      XMLNode req;
      if(!dynamic_cast<XMLNodeObj*>(prms[0].getO())) return string(_("1:Запит не
об’єктний!"));
      ((XMLNodeObj*)prms[0].getO())->toXMLNode(req);
      string path = req.attr("path");
      if(prms.size() < 2 || prms[1].getS().empty())
      {
          req.setAttr("user",user);
          cntrCmd(&req);
      }
      else
      {
          req.setAttr("path","/"+prms[1].getS()+path);
          transport().at().cntrIfCmd(req,"cntrReq");
          req.setAttr("path",path);
      }
      ((XMLNodeObj*)prms[0].getO())->fromXMLNode(req);
      return string("0");
    }
    // string sleep(int tm, int ntm = 0) – викликаємо завдання переходу до
сплячого режиму через <tm> секунд та <ntm> наносекунд.
    //  tm - чекаємо цей час у секундах
    //  ntm - чекаємо цей час у наносекундах
    if(iid == "sleep" && prms.size() >= 1)
    {
      struct timespec sp_tm;
      sp_tm.tv_sec = prms[0].getI();
      sp_tm.tv_nsec = (prms.size() >= 2) ? prms[1].getI() : 0;
      int rez = clock_nanosleep(CLOCK_REALTIME,0,&sp_tm,NULL);
```

```
      return rez;
    }
    // int time(int usec) – повертаємо абсолютний час у секундах від 1/1/1970 та
в мікросекундах, якщо <usec> задано
    //  usec - microсекунди of time
    if(iid == "time")
    {
      if(prms.empty()) return (int)time(NULL);
      int64_t tm = curTime();
      prms[0].setI(tm%1000000); prms[0].setModify();
      return (int)(tm/1000000);
    }
    // int localtime(int fullsec, int sec, int min, int hour, int mday, int
month, int year, int wday, int yday, int isdst)
    //      - повертаємо повну дату, базуємо на абсолютному часі у секундах
<fullsec> від 1.1.1970
    //  fullsec – час джерела у секундах від 1.1.1970
    //  sec - секунди
    //  min - хвилини
    //  hour - часи
    //  mday - дні місяця
    //  month - місяці
    //  year - роки
    //  wday - дні неділі
    //  yday - дні року
    //  isdst - відмітка про літній час
    if(iid == "localtime" && prms.size() >= 2)
    {
      time_t tm_t = prms[0].getI();
      struct tm tm_tm;
      localtime_r(&tm_t,&tm_tm);

      prms[1].setI(tm_tm.tm_sec); prms[1].setModify();
      if(prms.size() >= 3)    { prms[2].setI(tm_tm.tm_min); prms[2].setModify();
}
      if(prms.size() >= 4)    { prms[3].setI(tm_tm.tm_hour);
prms[3].setModify(); }
      if(prms.size() >= 5)    { prms[4].setI(tm_tm.tm_mday);
prms[4].setModify(); }
      if(prms.size() >= 6)    { prms[5].setI(tm_tm.tm_mon); prms[5].setModify();
}
      if(prms.size() >= 7)    { prms[6].setI(1900+tm_tm.tm_year);
prms[6].setModify(); }
      if(prms.size() >= 8)    { prms[7].setI(tm_tm.tm_wday);
prms[7].setModify(); }
      if(prms.size() >= 9)    { prms[8].setI(tm_tm.tm_yday);
prms[8].setModify(); }
      if(prms.size() >= 10)   { prms[9].setI(tm_tm.tm_isdst);
prms[9].setModify(); }
      return 0;
    }
    // string strftime(int sec, string form = "%Y-%m-%d %H:%M:%S") – перетворює
абсолютний час  <sec> у рядок формату <form>
    //  sec – час у секундах від 1.1.1970
    //  form – вихідний форматований рядок
    if(iid == "strftime" && !prms.empty())
    {
      time_t tm_t = prms[0].getI();
      struct tm tm_tm;
      localtime_r(&tm_t,&tm_tm);
      char buf[1000];
      int rez = strftime(buf, sizeof(buf), (prms.size()>=2) ?
prms[1].getS().c_str() : "%Y-%m-%d %H:%M:%S", &tm_tm);
      return (rez>0) ? string(buf,rez) : "";
    }
    // int strptime(string str, string form = "%Y-%m-%d %H:%M:%S") - повертає
час  у секундах від of 1/1/1970,
    //      базується на запису рядку часу <str>, відповідно до вказаного
шаблону  <form>
```

```
    //  str - джерело часу у рядку
    //  form - рядки часу у форматі POSIX-функцій "strptime"
    if(iid == "strptime" && !prms.empty())
    {
      struct tm stm;
      stm.tm_isdst = -1;
      strptime(prms[0].getS().c_str(), (prms.size()>=2) ? prms[1].getS().c_str()
: "%Y-%m-%d %H:%M:%S", &stm);
      return (int)mktime(&stm);
    }
    // int cron(string cronreq, int base = 0) - повертає час , планується у
форматі стандартного Cron <cronreq>,
    //      початок від основного часу<base> або від поточного, якщо основа не
вказана
    //  cronreq - розповсюджений у стандартному форматі Cron
    //  base - основний час
    if(iid == "cron" && !prms.empty())
      return (int)cron(prms[0].getS(), (prms.size()>=2) ? prms[1].getI() : 0);
    // string strFromCharCode(int char1, int char2, int char3, ...) - створює
рядок з кодових символів
    //  char1, char2. char3 - кодові символи
    if(iid == "strFromCharCode")
    {
      string rez;
      for(unsigned i_p = 0; i_p < prms.size(); i_p++)
          rez += (unsigned char)prms[i_p].getI();
      return rez;
    }
    // string strCodeConv( string src, string fromCP, string toCP ) - Текстовий
рядок перекодовує з кодової сторінки <fromCP> до кодової сторінки <toCP>.
    //  src - source text;
    //  fromCP - з кодової сторінки , порожньої для внутрішньої кодової сторінки
;
    //  toCP - до кодової сторінки , порожньої для внутрішньої кодової сторінки
.
    if(iid == "strCodeConv" && prms.size() >= 3)
      return Mess->codeConv((prms[1].getS().size() ? prms[1].getS() : Mess-
>charset()),
                  (prms[2].getS().size() ? prms[2].getS() : Mess->charset()),
prms[0].getS());

    return TCntrNode::objFuncCall(iid,prms,user);
}

void TSYS::cntrCmdProc( XMLNode *opt )
{
    char buf[STR_BUF_LEN];

    //Беремо сторінку інформації
    if(opt->name() == "info")
    {
      TCntrNode::cntrCmdProc(opt);
      snprintf(buf,sizeof(buf),_("%s station:
\"%s\""),PACKAGE_NAME,name().c_str());
      ctrMkNode("WSCADA_cntr",opt,-1,"/",buf,R_R_R_);
      if(ctrMkNode("branches",opt,-1,"/br","",R_R_R_))
          ctrMkNode("grp",opt,-
1,"/br/sub_",_("Subsystem"),R_R_R_,"root","root",1,"idm","1");
      if(TUIS::icoPresent(id())) ctrMkNode("img",opt,-1,"/ico","",R_R_R_);
      if(ctrMkNode("area",opt,-1,"/gen",_("Station"),R_R_R_))
      {
          ctrMkNode("fld",opt,-
1,"/gen/id",_("ID"),R_R_R_,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-
1,"/gen/stat",_("Station"),RWRWR_,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-
1,"/gen/prog",_("Program"),R_R_R_,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-
1,"/gen/ver",_("Version"),R_R_R_,"root","root",1,"tp","str");
```

```
          ctrMkNode("fld",opt,-1,"/gen/host",_("Ім'я
хосту"),R_R_R_,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-1,"/gen/user",_("Користувач
системи"),R_R_R_,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-1,"/gen/sys",_("Операційна
система"),R_R_R_,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-1,"/gen/frq",_("Частота
(MHZ)"),R_R_R_,"root","root",1,"tp","real");
          ctrMkNode("fld",opt,-1,"/gen/clk_res",_("Значення годинника реального
часу"),R_R_R_,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-1,"/gen/in_charset",_("Внутрішній набір
символів"),R_R___,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-1,"/gen/config",_("Кофігураційний
файл"),R_R___,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-1,"/gen/workdir",_("Робоча
директорія"),RWRW__,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-1,"/gen/icodir",_("Директорія
іконок"),RWRW__,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-1,"/gen/moddir",_("Директорія
модулів"),RWRW__,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-1,"/gen/wrk_db",_("Робоча база
даних"),RWRWR_,"root","root",4,"tp","str","dest","select","select","/db/list",
            "help",_("Адрес робочої бази даних у форматі [<БД module>.<БД
name>].\n Змінюємо ці поля якщо необхідно зберегти або завантажити усю систему з
іншої  БД."));
          ctrMkNode("fld",opt,-1,"/gen/saveExit",_("Збереження змін у системі
перед виходом"),RWRWR_,"root","root",2,"tp","bool",
            "help",_("Обираємо автоматичне збереження системи до  БД перед
виходом."));
          ctrMkNode("fld",opt,-1,"/gen/savePeriod",_("Збереження періоду роботи
у системі "),RWRWR_,"root","root",2,"tp","dec",
            "help",_("Використовуємо нульовий період (секунди) для періодичного
збереження  змін частин системи у БД."));
          ctrMkNode("fld",opt,-
1,"/gen/lang",_("Language"),RWRWR_,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-1,"/gen/baseLang",_("Базова мова тексту
змінних"),RWRWR_,"root","root",5,"tp","str","len","2","dest","sel_ed","select","
/gen/baseLangLs",
            "help",_("Мультимовність для змінного тексту підтримки для вибору
базової мови."));
          if(ctrMkNode("area",opt,-1,"/gen/mess",_("Повідомлення"),R_R_R_))
          {
            ctrMkNode("fld",opt,-1,"/gen/mess/lev",_("Найменший
рівень"),RWRWR_,"root","root",3,
              "tp","dec","len","1","help",_("Повідомлення найменшого рівня для
відображення процесів, які відбуваються у системі."));
            ctrMkNode("fld",opt,-1,"/gen/mess/log_sysl",_("To
syslog"),RWRWR_,"root","root",1,"tp","bool");
            ctrMkNode("fld",opt,-1,"/gen/mess/log_stdo",_("To
stdout"),RWRWR_,"root","root",1,"tp","bool");
            ctrMkNode("fld",opt,-1,"/gen/mess/log_stde",_("To
stderr"),RWRWR_,"root","root",1,"tp","bool");
            ctrMkNode("fld",opt,-1,"/gen/mess/log_arch",_("До
архіву"),RWRWR_,"root","root",1,"tp","bool");
          }
      }
      if(ctrMkNode("area",opt,-1,"/subs",_("Підсистема")))
          ctrMkNode("list",opt,-
1,"/subs/br",_("Підсистеми"),R_R_R_,"root","root",3,"idm","1","tp","br","br_pref
","sub_");
      if(ctrMkNode("area",opt,-1,"/tasks",_("Tasks"),R_R___))
          if(ctrMkNode("table",opt,-
1,"/tasks/tasks",_("Завдання"),RWRW__,"root","root",2,"key","path",
            "help",!multCPU()?"":_("Для CPU встановлюємо рядок номерів
процесорів використання, відокремлений символом':'.\n"
                                   "CPU починаємо з 0.")))
          {
            ctrMkNode("list",opt,-
1,"/tasks/tasks/path",_("Path"),R_R___,"root","root",1,"tp","str");
```

```
            ctrMkNode("list",opt,-
1,"/tasks/tasks/thrd",_("Потоk"),R_R___,"root","root",1,"tp","str");
            ctrMkNode("list",opt,-
1,"/tasks/tasks/tid",_("TID"),R_R___,"root","root",1,"tp","dec");
            ctrMkNode("list",opt,-
1,"/tasks/tasks/stat",_("Статус"),R_R___,"root","root",1,"tp","str");
            ctrMkNode("list",opt,-
1,"/tasks/tasks/plc",_("Політика"),R_R___,"root","root",1,"tp","str");
            ctrMkNode("list",opt,-
1,"/tasks/tasks/prior",_("Prior."),R_R___,"root","root",1,"tp","dec");
#if __GLIBC_PREREQ(2,4)
            if(multCPU())
                ctrMkNode("list",opt,-1,"/tasks/tasks/cpuSet",_("CPU
встановлення"),RWRW__,"root","root",1,"tp","str");
#endif
          }
      if( !cntrEmpty() && ctrMkNode("area",opt,-1,"/cntr",_("Країна")) )
          if( ctrMkNode("table",opt,-
1,"/cntr/cntr",_("Counters"),R_R___,"root","root") )
          {
            ctrMkNode("list",opt,-
1,"/cntr/cntr/id","ID",R_R___,"root","root",1,"tp","str");
            ctrMkNode("list",opt,-
1,"/cntr/cntr/vl",_("Value"),R_R___,"root","root",1,"tp","real");
          }
      if( ctrMkNode("area",opt,-1,"/hlp",_("Help"),R_R___) )
          ctrMkNode("fld",opt,-1,"/hlp/g_help",_("Опції
допомоги"),R_R___,"root","root",3,"tp","str","cols","90","rows","10");
      return;
    }

    //Процес управління на сторінці
    string a_path = opt->attr("path");
    if(a_path == "/ico" && ctrChkNode(opt))
    {
      string itp;
      opt->setText(TSYS::strEncode(TUIS::icoGet(id(),&itp),TSYS::base64));
      opt->setAttr("tp",itp);
    }
    else if(a_path == "/gen/host" && ctrChkNode(opt))opt->setText(host());
    else if(a_path == "/gen/sys" && ctrChkNode(opt))
    {
      utsname ubuf; uname(&ubuf);
      opt->setText(string(ubuf.sysname)+"-"+ubuf.release);
    }
    else if(a_path == "/gen/user" && ctrChkNode(opt))opt->setText(mUser);
    else if(a_path == "/gen/prog" && ctrChkNode(opt))opt->setText(PACKAGE_NAME);
    else if(a_path == "/gen/ver" && ctrChkNode(opt))  opt->setText(VERSION);
    else if(a_path == "/gen/id" && ctrChkNode(opt))   opt->setText(id());
    else if(a_path == "/gen/stat")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root","root",SEC_RD))opt->setText(name());
      if(ctrChkNode(opt,"set",RWRWR_,"root","root",SEC_WR))setName(opt->text());
    }
    else if(a_path == "/gen/frq" && ctrChkNode(opt))  opt-
>setText(TSYS::real2str((float)sysClk()/1000000.,6));
    else if(a_path == "/gen/clk_res" && ctrChkNode(opt))
    {
      struct timespec tmval;
      clock_getres(CLOCK_REALTIME,&tmval);
      opt->setText(TSYS::time2str(1e-3*tmval.tv_nsec));//
TSYS::real2str((float)tmval.tv_nsec/1000000.,4));
    }
    else if(a_path == "/gen/in_charset" && ctrChkNode(opt))opt->setText(Mess-
>charset());
    else if(a_path == "/gen/config" && ctrChkNode(opt))          opt-
>setText(mConfFile);
    else if(a_path == "/gen/wrk_db" )
    {
```

```
      if(ctrChkNode(opt,"get",RWRWR_,"root","root",SEC_RD))opt-
>setText(mWorkDB);
      if(ctrChkNode(opt,"set",RWRWR_,"root","root",SEC_WR))setWorkDB(opt-
>text());
    }
    else if(a_path == "/gen/saveExit")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root","root",SEC_RD))opt->setText(
int2str(saveAtExit()) );
      if(ctrChkNode(opt,"set",RWRWR_,"root","root",SEC_WR))setSaveAtExit(
atoi(opt->text().c_str()) );
    }
    else if(a_path == "/gen/savePeriod")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root","root",SEC_RD))opt->setText(
int2str(savePeriod()) );
      if(ctrChkNode(opt,"set",RWRWR_,"root","root",SEC_WR))setSavePeriod(
atoi(opt->text().c_str()) );
    }
    else if(a_path == "/gen/workdir")
    {
      if(ctrChkNode(opt,"get",R_R___,"root","root",SEC_RD))opt-
>setText(workDir());
      if(ctrChkNode(opt,"set",R_R___,"root","root",SEC_WR))setWorkDir(opt-
>text().c_str());
    }
    else if(a_path == "/gen/icodir")
    {
      if(ctrChkNode(opt,"get",R_R___,"root","root",SEC_RD))opt-
>setText(icoDir());
      if(ctrChkNode(opt,"set",R_R___,"root","root",SEC_WR))setIcoDir(opt-
>text().c_str());
    }
    else if(a_path == "/gen/moddir")
    {
      if(ctrChkNode(opt,"get",R_R___,"root","root",SEC_RD))opt-
>setText(modDir());
      if(ctrChkNode(opt,"set",R_R___,"root","root",SEC_WR))setModDir(opt-
>text().c_str());
    }
    else if(a_path == "/gen/lang")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root","root",SEC_RD))opt->setText(Mess-
>lang());
      if(ctrChkNode(opt,"set",RWRWR_,"root","root",SEC_WR))Mess->setLang(opt-
>text());
    }
    else if(a_path == "/gen/baseLang")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root","root",SEC_RD))opt->setText(Mess-
>lang2CodeBase());
      if(ctrChkNode(opt,"set",RWRWR_,"root","root",SEC_WR))Mess-
>setLang2CodeBase(opt->text());
    }
    else if(a_path == "/gen/baseLangLs" && ctrChkNode(opt))
    {
      opt->childAdd("el")->setText(Mess->lang2Code());
      if(!Mess->lang2CodeBase().empty() && Mess->lang2CodeBase() != Mess-
>lang2Code())
          opt->childAdd("el")->setText(Mess->lang2CodeBase());
      opt->childAdd("el")->setText("");
    }
    else if(a_path == "/gen/mess/lev")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root","root",SEC_RD))opt-
>setText(TSYS::int2str(Mess->messLevel()));
      if(ctrChkNode(opt,"set",RWRWR_,"root","root",SEC_WR))Mess-
>setMessLevel(atoi(opt->text().c_str()));
    }
```

```
    else if(a_path == "/gen/mess/log_sysl")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root","root",SEC_RD))opt->setText((Mess-
>logDirect()&0x01)?"1":"0");
      if(ctrChkNode(opt,"set",RWRWR_,"root","root",SEC_WR))Mess->setLogDirect(
atoi(opt->text().c_str())?Mess->logDirect()|0x01:Mess->logDirect()&(~0x01) );
    }
    else if(a_path == "/gen/mess/log_stdo")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root","root",SEC_RD))opt->setText((Mess-
>logDirect()&0x02)?"1":"0");
      if(ctrChkNode(opt,"set",RWRWR_,"root","root",SEC_WR))Mess->setLogDirect(
atoi(opt->text().c_str())?Mess->logDirect()|0x02:Mess->logDirect()&(~0x02) );
    }
    else if(a_path == "/gen/mess/log_stde")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root","root",SEC_RD))opt->setText((Mess-
>logDirect()&0x04)?"1":"0");
      if(ctrChkNode(opt,"set",RWRWR_,"root","root",SEC_WR))Mess->setLogDirect(
atoi(opt->text().c_str())?Mess->logDirect()|0x04:Mess->logDirect()&(~0x04) );
    }
    else if(a_path == "/gen/mess/log_arch")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root","root",SEC_RD))opt->setText((Mess-
>logDirect()&0x08)?"1":"0");
      if(ctrChkNode(opt,"set",RWRWR_,"root","root",SEC_WR))Mess->setLogDirect(
atoi(opt->text().c_str())?Mess->logDirect()|0x08:Mess->logDirect()&(~0x08) );
    }
    else if((a_path == "/br/sub_" || a_path == "/subs/br") &&
ctrChkNode(opt,"get",R_R_R_,"root","root",SEC_RD))
    {
      vector<string> lst;
      list(lst);
      for(unsigned i_a=0; i_a < lst.size(); i_a++)
          opt->childAdd("el")->setAttr("id",lst[i_a])-
>setText(at(lst[i_a]).at().subName());
    }
    else if(a_path == "/tasks/tasks")
    {
      if(ctrChkNode(opt,"get",RWRW__,"root","root"))
      {
          XMLNode *n_path    = ctrMkNode("list",opt,-
1,"/tasks/tasks/path","",R_R___,"root","root");
          XMLNode *n_thr     = ctrMkNode("list",opt,-
1,"/tasks/tasks/thrd","",R_R___,"root","root");
          XMLNode *n_tid     = ctrMkNode("list",opt,-
1,"/tasks/tasks/tid","",R_R___,"root","root");
          XMLNode *n_stat    = ctrMkNode("list",opt,-
1,"/tasks/tasks/stat","",R_R___,"root","root");
          XMLNode *n_plc     = ctrMkNode("list",opt,-
1,"/tasks/tasks/plc","",R_R___,"root","root");
          XMLNode *n_prior   = ctrMkNode("list",opt,-
1,"/tasks/tasks/prior","",R_R___,"root","root");
          XMLNode *n_cpuSet  = (multCPU() ? ctrMkNode("list",opt,-
1,"/tasks/tasks/cpuSet","",RWRW__,"root","root") : NULL);

          ResAlloc res(taskRes,false);
          for(map<string,STask>::iterator it = mTasks.begin(); it !=
mTasks.end(); it++)
          {
            if(n_path)  n_path->childAdd("el")->setText(it->first);
            if(n_thr)   n_thr->childAdd("el")->setText(TSYS::uint2str(it-
>second.thr));
            if(n_tid)   n_tid->childAdd("el")->setText(TSYS::int2str(it-
>second.tid));
            if(n_stat)
            {
                int64_t tm_beg = 0, tm_end = 0, tm_per = 0;
                for(int i_tr = 0; tm_beg == tm_per && i_tr < 2; i_tr++)
```

```
                { tm_beg = it->second.tm_beg; tm_end = it->second.tm_end; tm_per
= it->second.tm_per; }
                XMLNode *cn = n_stat->childAdd("el");
                if(it->second.flgs&STask::FinishTask) cn->setText(_("Закінчено.
"));
                if(tm_beg && tm_beg < tm_per)
                  cn->setText(cn->text()+TSYS::strMess(_("Останій: %s.
Завантажено: %3.1f% (%s з %s)"),
                    time2str((time_t)(1e-9*tm_per),"%d-%m-%Y
%H:%M:%S").c_str(), 100*(double)(tm_end-tm_beg)/(double)(tm_per-tm_beg),
                    time2str(1e-3*(tm_end-tm_beg)).c_str(), time2str(1e-
3*(tm_per-tm_beg)).c_str()));
            }
            if(n_plc)
            {
                string plcVl = _("Стандартний");
                if(it->second.policy == SCHED_RR) plcVl = _(" Кругова система
");
#if __GLIBC_PREREQ(2,4)
                if(it->second.policy == SCHED_BATCH) plcVl = _("Style
\"batch\"");
#endif
                n_plc->childAdd("el")->setText(plcVl);
            }
            if(n_prior) n_prior->childAdd("el")->setText(TSYS::int2str(it-
>second.prior));
            if(n_cpuSet)      n_cpuSet->childAdd("el")->setText(it-
>second.cpuSet);
          }
      }
#if __GLIBC_PREREQ(2,4)
      if(multCPU() && ctrChkNode(opt,"set",RWRW__,"root","root",SEC_WR) && opt-
>attr("col") == "cpuSet")
      {
          ResAlloc res(taskRes,true);
          map<string,STask>::iterator it = mTasks.find(opt->attr("key_path"));
          if(it == mTasks.end()) throw TError(nodePath().c_str(),_("Не
представлене завдання '%s'."));
          if(it->second.flgs & STask::Detached) return;

          it->second.cpuSet = opt->text();

          cpu_set_t cpuset;
          CPU_ZERO(&cpuset);
          string sval;
          for(int off = 0; (sval=TSYS::strParse(it-
>second.cpuSet,0,":",&off)).size(); )
            CPU_SET(atoi(sval.c_str()),&cpuset);
          int rez = pthread_setaffinity_np(it->second.thr, sizeof(cpu_set_t),
&cpuset);
          res.release();
          TBDS::genDBSet(nodePath()+"CpuSet:"+it->first,opt->text());
          if(rez == EINVAL && opt->text().size()) throw
TError(nodePath().c_str(),_("Не встановлено жодних дозволених процесорів."));
          if(rez && opt->text().size()) throw TError(nodePath().c_str(),_("CPU
установки для потоку помилкові."));
      }
#endif
    }
    if(!cntrEmpty() && a_path == "/cntr/cntr" &&
ctrChkNode(opt,"get",R_R___,"root","root"))
    {
      XMLNode *n_id     = ctrMkNode("list",opt,-
1,"/cntr/cntr/id","",R_R___,"root","root");
      XMLNode *n_vl     = ctrMkNode("list",opt,-
1,"/cntr/cntr/vl","",R_R___,"root","root");

      ResAlloc res( nodeRes(), false );
```

```
      for(map<string,double>::iterator icnt = mCntrs.begin(); icnt !=
mCntrs.end(); icnt++)
      {
          if(n_id)      n_id->childAdd("el")->setText(icnt->first);
          if(n_vl)      n_vl->childAdd("el")->setText(TSYS::real2str(icnt-
>second));
      }
    }
    else if(a_path == "/hlp/g_help" &&
ctrChkNode(opt,"get",R_R___,"root","root",SEC_RD)) opt->setText(optDescr());
    else TCntrNode::cntrCmdProc(opt);
}
```