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 першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки захищеної WebSCADA системи”

Виконав здобувач вищої освіти
IV курсу, групи КБ-20-3СК
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Самборський Д.Д.
«____» _____ 2023 р.

Керівник проекту
кандидат технічних наук
_____ Смірнова Т.В.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Освітній ступінь *бакалавр*
Галузь знань ._12 _____*“*Інформаційні технології*”*_____
Спеціальність *125 “*Кібербезпека*”*
Освітньо-професійна (освітньо-наукова) програма _*“*Кібербезпека*”*_

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

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Самборському Данилу Денисовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки захищеної WebSCADA системи*

2. Керівник роботи *Смірнова Тетяна Віталіївна, канд. техн. наук*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 13-02 від 5.01.2023 року

3. Строк подання студентом роботи до захисту *23.05.2023 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи кібербезпеки захищеної WebSCADA системи*

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

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи кібербезпеки в промислову експлуатацію.
6. Висновки

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

Структурна схема системи кібербезпеки	*1 аркуш*
Функціональна схема системи кібербезпеки	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Дата видачі завдання « 17 » січня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2023 р.	
3.	Розробка моделі компонента	20.03.2023 р.	
4.	Розробка структур даних	25.03.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2023 р.	
6.	Програмування алгоритмів	10.04.2023 р.	
7.	Оформлення ПЗ	17.04.2023 р.	
8.	Попередній захист роботи	23.05.2023 р.	

Дата видачі завдання
« 17 » січня 2023 р.

Підпис керівника
_____ Смірнова Т.В.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2023 р.

Підпис здобувача
_____ Самборський Д.Д.
(прізвище та ініціали)

АНОТАЦІЯ

Самборський Д.Д. Програмне забезпечення системи кібербезпеки захищеної WebSCADA системи. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки захищеної WebSCADA системи.

Метою розробки є програмне забезпечення системи кібербезпеки захищеної WebSCADA системи.

Результат роботи – програмна реалізація системи кібербезпеки захищеної WebSCADA системи.

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: кібербезпека, захищена WebSCADA

ABSTRACT

Samborskyi D.D. Software of the cyber security system of the protected WebSCADA system. 125 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this final qualification work for the first (bachelor) level of higher education, software is developed, which is intended for the cyber security system of the protected WebSCADA system.

The purpose of the development is the software of the cyber security system of the protected WebSCADA system.

The result of the work is the software implementation of the cyber security system of the protected WebSCADA system.

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: cybersecurity protected by WebSCADA

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. Інтеграція контролерних засобів в автоматизовані системи управління технологічними процесами визначила нові погляди на оцінку їхньої ефективності й надійності за економічними критеріями.

Виходячи з того, що зазначений клас апаратури має істотні відмінності від тих комплексів, що раніше застосовувалися в промисловості, виникла необхідність у розробці таких систем.

Аналіз літератури [1-4], присвяченої рішенню даного завдання показує, що теоретичні підходи, що застосовувалися раніше, вимагають коректування. Це викликано насамперед наявністю принципу децентралізації в структурі комп'ютерно-інтегрованої системи управління технологічним процесом.

Будь-яка система управління є незалежною розробкою й має унікальні функції й архітектурні рішення.

Загальним же для сучасних систем є наявність декількох ієрархічних рівнів – багаторівність.

Як правило, це дворівневі системи, тому що саме на цих рівнях реалізується безпосереднє управління технологічними процесами. Специфіка кожної конкретної системи управління визначається використовуваної на кожному рівні програмно-апаратною платформою.

Нижній рівень – рівень об'єкта (контролерний), включає різні датчики для збору інформації про хід технологічного процесу, електроприводи й виконавчі механізми для реалізації регулюючих і керуючих впливів. Датчики поставляють інформацію контролерам, які можуть виконувати наступні функції:

– збір і обробка інформації про параметри технологічного процесу;

– управління електроприводами й іншими виконавчими механізмами;

– рішення завдань автоматичного логічного управління й ін.

До апаратно-програмних засобів контролерного рівня управління пред'являються тверді вимоги по надійності, часу реакції на виконавчі пристрої, датчики й т.д.

Програмувальні логічні контролери повинні гарантовано відгукуватися на зовнішні події, що надходять від об'єкта, за час, певний для кожної події.

Верхній рівень – диспетчерський пункт, включає, насамперед, одну або кілька станцій управління, що представляють собою автоматизоване робоче місце диспетчера/оператора.

Тут же може бути розміщений сервер бази даних, робочі місця (комп'ютери) для фахівців і т.д.

Часто як робочі станції використовуються ЕОМ. На верхньому рівні як програмний сегмент АСУ ТП застосовують SCADA-системи, які надають широкий спектр можливостей по контролі за технічним станом вузлів і елементів автоматизації, а також дають можливість оперативного втручання для зниження збиток внаслідок їхньої відмови.

Зазначені рівні об'єднані за допомогою так званих польових шин, причому кожному ієрархічному рівню управління відповідає певний тип шини із властивими їй характеристиками надійності й ефективності.

Необхідно також відзначити, що аналогові сигнали від первинних датчиків і перетворювачів і сигнали управління виконавчими механізмами циркулюють тільки на польовому рівні й перетворяться в цифрові сигнали й назад безпосередньо на технологічному об'єкті управління.

Сучасні розподілені системи управління характеризуються наявністю промислової мережі, що складає з багатьох вузлів, обмін між якими виробляється цифровим способом.

Використання промислової мережі дозволяє розташувати вузли, у якості яких виступають контролери й інтелектуальні пристрої введення-виводу, максимально наблизивши їх до датчиків і виконавчих механізмів, скоротивши до мінімуму довжину ліній передачі аналогових сигналів.

З огляду на високу надійність контролерних і комунікаційних засобів, основним джерелом зниження надійності системи в цілому можна віднести до засобів виміру технологічних параметрів і виконавчих механізмів системи, а також до комунікаційних каналів. Однак при цьому виникла проблема обліку надійності програмного забезпечення: загальносистемного й прикладного.

Таким чином, мережна структура побудови комп'ютерно-інтегрованої системи управління обумовила необхідність застосування нових підходів.

Розвиток сучасних мережних технологій обумовило появи такого напрямку SCADA-систем, як WebSCADA. Ці системи, на відміну від традиційних, дозволяють віддалено, через комп'ютерну мережу, як дротову, так і бездротову, управляти АСУ ТП.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки захищеної WebSCADA системи.

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

– Огляд існуючих систем захищеної WebSCADA системи.

– Дослідження системи кібербезпеки захищеної WebSCADA системи.

– Програмна реалізація системи кібербезпеки захищеної WebSCADA системи.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі захищеної WebSCADA системи.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки захищеної WebSCADA системи, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Під терміном WebSCADA, як правило, розуміється реалізація людино-машинного інтерфейсу (HMI) SCADA-систем на основі WEB-технологій.

Це дозволяє здійснювати контроль і управління SCADA-системою через стандартний браузер, що виступає в цьому випадку в ролі тонкого клієнта.

Архітектура таких систем містить у собі WebSCADA-сервер і клієнтські термінали – ПК, КПК або мобільні телефони з Web–браузером. Підключення клієнтів до WebSCADA-сервера через Internet/Intranet дозволяє їм взаємодіяти із прикладним завданням автоматизації як із простий WEB або WAP-сторінкою. Однак на даному етапі розвитку WebSCADA ще не досягло рівня широкого промислового впровадження, тому що існують складності із захистом переданої інформації. Крім цього, реалізація функцій управління через незахищені канали зв'язку суперечить міркуванням безпеки будь-якого промислового об'єкта. У зв'язку із цим, у більшості випадків WEB інтерфейси використовуються як віддалені клієнти для контролю й збору даних.

1.2 Область застосування

Областю застосування розроблювальної системи є SCADA. SCADA (Supervisory Control And Data Acquisition – диспетчерське управління й збір даних) – програмний пакет призначений для розробки або забезпечення роботи в реальному часі систем збору, обробки, відображення й архівування інформації про об'єкт моніторингу або управління. SCADA може бути частиною АСУ ТП, АСКУЕ, системи екологічного моніторингу, наукового експерименту, автоматизації будинку й т.д. SCADA-системи використовуються у всіх галузях

господарства, де потрібно забезпечувати операторський контроль за технологічними процесами в реальному часі. Дане програмне забезпечення встановлюється на комп'ютери й, для зв'язку з об'єктом, використовує драйвери вводу-виводу або OPC/DDE сервери. Програмний код може бути як написаний мовою програмування (наприклад на C++), так і згенерований у середовищі проектування.

Іноді SCADA-системи комплектуються додатковим ПЗ для програмування промислових контролерів. Такі SCADA-системи називаються інтегрованими й до них додають термін SoftLogic.

Термін SCADA має двояке тлумачення. Найбільше широко поширене розуміння SCADA як додатка, тобто програмного комплексу, що забезпечує виконання зазначених функцій, а також інструментальних засобів для розробки цього програмного забезпечення. Однак, часто під SCADA-системою мають на увазі програмно-апаратний комплекс. Подібне розуміння терміна SCADA більш характерно для розділу телеметрія.

Значення терміна SCADA перетерпіло зміни разом з розвитком технологій автоматизації й управління технологічними процесами. В 80-е роки під SCADA-системами частіше розуміли програмно-апаратні комплекси збору даних реального часу. З 90-х років термін SCADA більше використовується для позначення тільки програмної частини людино-машинного інтерфейсу АСУ ТП.

Основні завдання розв'язувані SCADA-системами

SCADA-системи вирішують наступні завдання:

– Обмін даними з ПЗО (пристрій зв'язку з об'єктом, тобто із промисловими контролерами й платами введення/виводу) у реальному часі через драйвери.

– Обробка інформації в реальному часі.

– Логічне управління.

– Відображення інформації на екрані монітора в зручній і зрозумілій для людини формі.

– Ведення бази даних реального часу з технологічною інформацією.

– Аварійна сигналізація й управління тривожними повідомленнями.

– Підготовка й генерування звітів про хід технологічного процесу.

– Здійснення мережної взаємодії між SCADA ПК.

– Забезпечення зв'язку із зовнішніми додатками (СУБД, електронні таблиці, текстові процесори й т.д.). У системі управління підприємством такими додатками найчастіше є додатки, які відносяться до рівня MES.

SCADA-системи дозволяють розробляти АСУ ТП у клієнт-серверній або в розподіленій архітектурі.

Основні компоненти SCADA

SCADA-система звичайно містить наступні підсистеми:

– Драйвери або сервери вводу-виводу – програми, що забезпечують зв'язок SCADA із промисловими контролерами, лічильниками, АЦП і іншими пристроями вводу-виводу інформації.

– Система реального часу – програма, що забезпечує обробку даних у межах заданого тимчасового циклу з урахуванням пріоритетів.

– Людино-машинний інтерфейс – інструмент, що представляє дані про хід процесу людині операторові, що дозволяє операторові контролювати процес і управляти ім. Програма-Редактор для розробки людино-машинного інтерфейсу.

– Система логічного управління – програма, що забезпечує виконання користувальницьких програм (скриптів) логічного управління в SCADA-системі. Набір редакторів для їхньої розробки.

– База даних реального часу – програма, що забезпечує збереження історії процесу в режимі реального часу.

– Система управління тривогами – програма, що забезпечує автоматичний контроль технологічних подій, віднесення їх до категорії нормальних, попереджуючих або аварійних, а також обробку подій оператором або комп'ютером.

– Генератор звітів – програма, що забезпечує створення користувальницьких звітів технологічних подіях. Набір редакторів для їхньої розробки.

– Зовнішні інтерфейси – стандартні інтерфейси обміну даними між SCADA і іншими додатками. Звичайно OPC, DDE, ODBC, DLL і т.д.

Концепції систем

Термін SCADA звичайно відноситься до централізованих систем контролю й управління всією системою, або комплексами систем, здійснюваного за участю людини. Більшість керуючих впливів виконується автоматично RTU або ПЛК. Безпосереднє управління процесом звичайно забезпечується RTU або PLC, а SCADA управляє режимами роботи. Наприклад, PLC може управляти потоком охолодної води усередині частини виробничого процесу, а SCADA система може дозволити операторам змінювати уставку для потоку, міняти маршрути руху рідини, заповнювати ті або інші ємності, а так само стежити за тривожними повідомленнями (алармами), такими як – втрата потоку й висока температура, які повинні бути відображені, записані, і на які оператор повинен вчасно реагувати. Цикл управління зі зворотним зв'язком проходить через RTU або ПЛК, у той час як SCADA система контролює повне виконання циклу.

Збір даних починається в RTU або на рівні PLC і включає показання вимірювального приладу. Далі дані збираються й форматуються таким способом, щоб оператор диспетчерської, використовуючи HMI міг прийняти контролюючі рішення – коректувати або перервати стандартне управління засобами RTU/ПЛК. Дані можуть також бути записані в архів для побудови трендів і іншої аналітичної обробки накопичених даних.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки захищеної WebSCADA системи, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

WEB-SCADA «Систел»

WEB-SCADA «Систел» – це програмне забезпечення для віддаленого перегляду інформації засобами Web-оглядача.

Комплекс програмного забезпечення, що реалізує Web-інтерфейс, можна розділити на три частини:

– Web-сервер.

– Сервер ТМ і джерела даних.

– Web-клієнти.

«WEB-сервер» встановлює канал зв'язку з «Сервером ТМ», що забезпечує збір телеінформації від джерел даних.

Користувач через Web-оглядач відправляє WEB-серверу запит на документ, що цікавить, з даними (наприклад, мнемосхему) і одержує як результат певним чином сформовану динамічну HTML-сторінку.

Джерелами даних є додатки, які займаються прийомом інформації від телемеханічних або яких-небудь інших пристроїв і передачею її по TCP-каналах на «Сервер ТМ».

Як джерела даних виступають:

– оперативні дані;

– архівні дані;

– мнемосхеми енергооб’єктів, підготовлені за допомогою графічного редактора GRED.

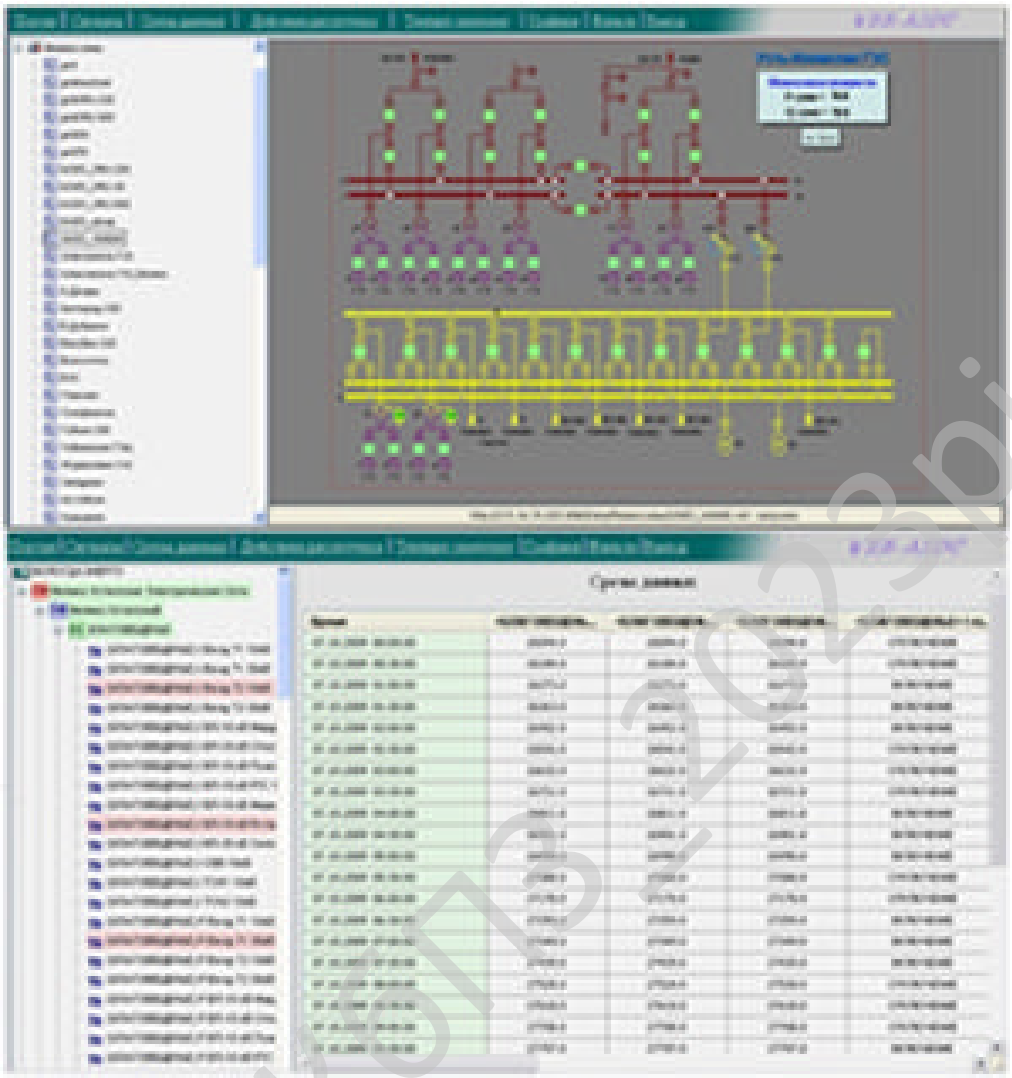

Рисунок 2.1 – Робочі вікна додатка WEB-SCADA «Систел»

Клієнтські додатки, що є джерелами або використовують дані, описуються в конфігураційній базі даних «Сервера ТМ», де для них визначаються такі параметри, як порт TCP-з'єднання, сигнатура й т.п.

Кожний елемент даних так само прописується в БД «Сервера ТМ».

Установлюючи TCP-з'єднання з «Сервером ТМ», WEB-сервер забезпечує передачу інформації із запитів Інтернет-оглядача клієнтського робочого місця.

Як виконавча система клієнтських робочих місць використовуються Інтернет-оглядачі – Internet Explorer, Opera, Mozilla Firefox.

Основна функція ПЗ Web-SCADA – подання інформації про режими роботи й стан об'єктів електричних мереж у вигляді таблиць, графіків, мнемосхем.

Для формування даних використовуються оперативні й архівні дані, які надалі відображаються у вигляді графіків, а також мнемосхеми енергооб’єктів, які створюються за допомогою графічного редактора GRED.

Додатки, що беруть участь у формуванні даних для архівів використовують бази даних Сервера ТМ. Інформація, до якої відбувається обіг при роботі програми, представлена в архівах бази даних (СУБД MS SQL Server), які створюються при роботі Сервера ТМ.

WebSCADA – система моніторингу й диспетчеризації промислових об'єктів ТОВ "Автоматизація Виробництв"

Система призначена для рішення завдань диспетчеризації й моніторингу об'єктів, автоматизованих за допомогою програмно-технічних засобів. Система виконана в клієнт-серверній архітектурі. Серверна частина системи може бути встановлена на глобальному Інтернет-сервері або локальному сервері. Ніяких спеціальних програм на комп'ютері клієнта (користувача) установлювати не потрібно, досить мати лише Інтернет-браузер і вихід у мережу Інтернет.

Короткий опис основних функціональних можливостей системи:

– Розробка мнемосхем у середовищі WebSCADA. Розробка мнемосхем може бути здійснена безпосередньо в редакторі WebSCADA. Дана можливість опирається на бібліотеку примітивів(технологічні об'єкти, датчики), які мають ряд параметрів, що набудовуються, є каталог примітивів, розбитий у групи. Є можливість створення користувальницьких примітивів користувачем системи. Динамічні(змінювані) дані на мнемосхеми довантажуються на мнемосхему за технологією динамічної підзавантаження з оптимізацією трафіку

– Графічний моніторинг технологічного процесу. Для наочного подання процесу, що відбувається, використовується графічний моніторинг. На графіку більш наочно можна бачити зміни й статистику роботи різних пристроїв, блоків, реєстрації датчиків і т.п.

– Засоби оповіщення. Для найбільш швидкого реагування на різні ситуації, що відбуваються в технологічному процесі, використовуються засоби оповіщення: SMS-повідомленя і відправлення електронної пошти. Це може бути корисним при відмовах системи, помилках і інших позаштатних ситуаціях. Також можна одержувати статистику від системи поштою за певний період її роботи, якщо така необхідність є.

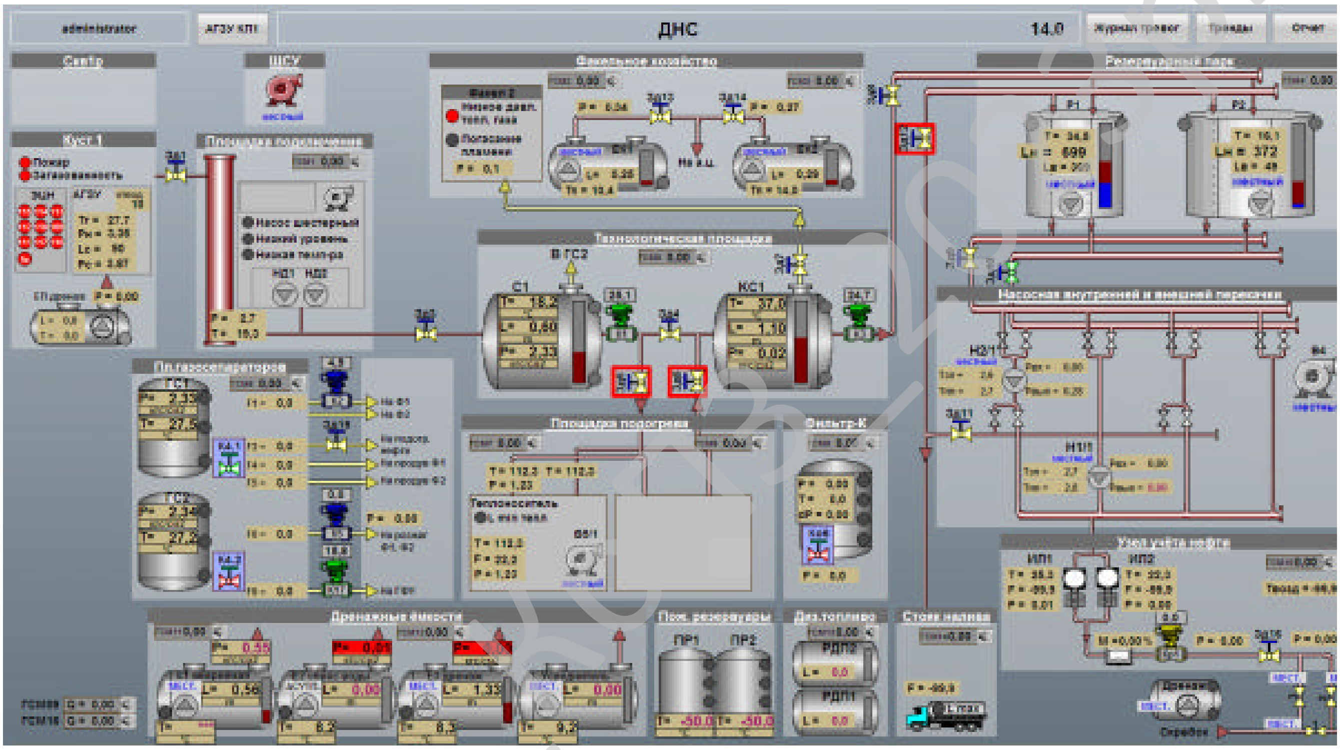

Рисунок 2.2 – Інтерфейс користувача системи ТОВ "Автоматизація Виробництв"

– Планувальник. У систему закладена можливість задавати параметри системи на певний час (годинники:мінути день/місяць/рік). Може задаватися інтервал, на якому необхідно мати певні параметри. У результаті система працює згідно закладеного в неї плану (програми).

– Система логістики. Система веде логі (логістику) по всьому технологічному процесі. У користувачів є можливість у будь-який момент відобразити ці параметри на екрані (або вивести їх у файл звіту). По певних

характеристиках можна побудувати графік для наочного моніторингу процесу. Система логістики фіксує всі зміни, які вносить кожний користувач, тобто всі дії фіксуються.

– Правовий доступ до системи. Передбачений 3-х рівневий (SuperAdmin, Admin і User) багатокористувальницький доступ користувачів у систему. Права на групи Admin і User визначаються при реєстрації користувачів. Кожна група має певні можливості й доступ до розділів системи.

Netbiter WEBSCADA

Надамо опис системи по функціям.

Віддалений контроль устаткування через Ethernet і GSM/GPRS:

– Забезпечується прямий доступ до даних про стан різного встаткування (у тому числі й польових пристроях).

– Обслуговуючий персонал одержує оперативний доступ до інформації про процес і автоматично сповіщається при аварійних ситуаціях.

– Керівництво (власник) одержує простий спосіб віддаленого контролю.

Состав системи Netbiter:

1. Спеціалізований захищений сервер NetBiter.net:

– Управління проектом.

2. Ethernet-шлюз:

– Забезпечення доступу до даних через провідні мережі Ethernet і бездротові GSM/ GPRS.

– Вбудований Modbus Master для одержання даних від зовнішніх пристроїв Modbus Slave.

– Вбудовані канали аналогового й дискретного вводу/виводу (у деяких моделях).

3. Модулі вводу/виводу

– Забезпечення необхідної кількості аналогових і дискретних каналів вводу/виводу потрібного типу.

Моніторинг і управління встаткуванням:

– Захищений доступ.

– Парольна система розмежування прав.

– Швидкий огляд усього встаткування в проекті, що набудовується.

– Віддалений доступ до поточних даних устаткування, можливість посилати команди управління.

– Довгострокове архівне зберігання важливих файлів (резервні копії, конфігураційні файли та ін.).

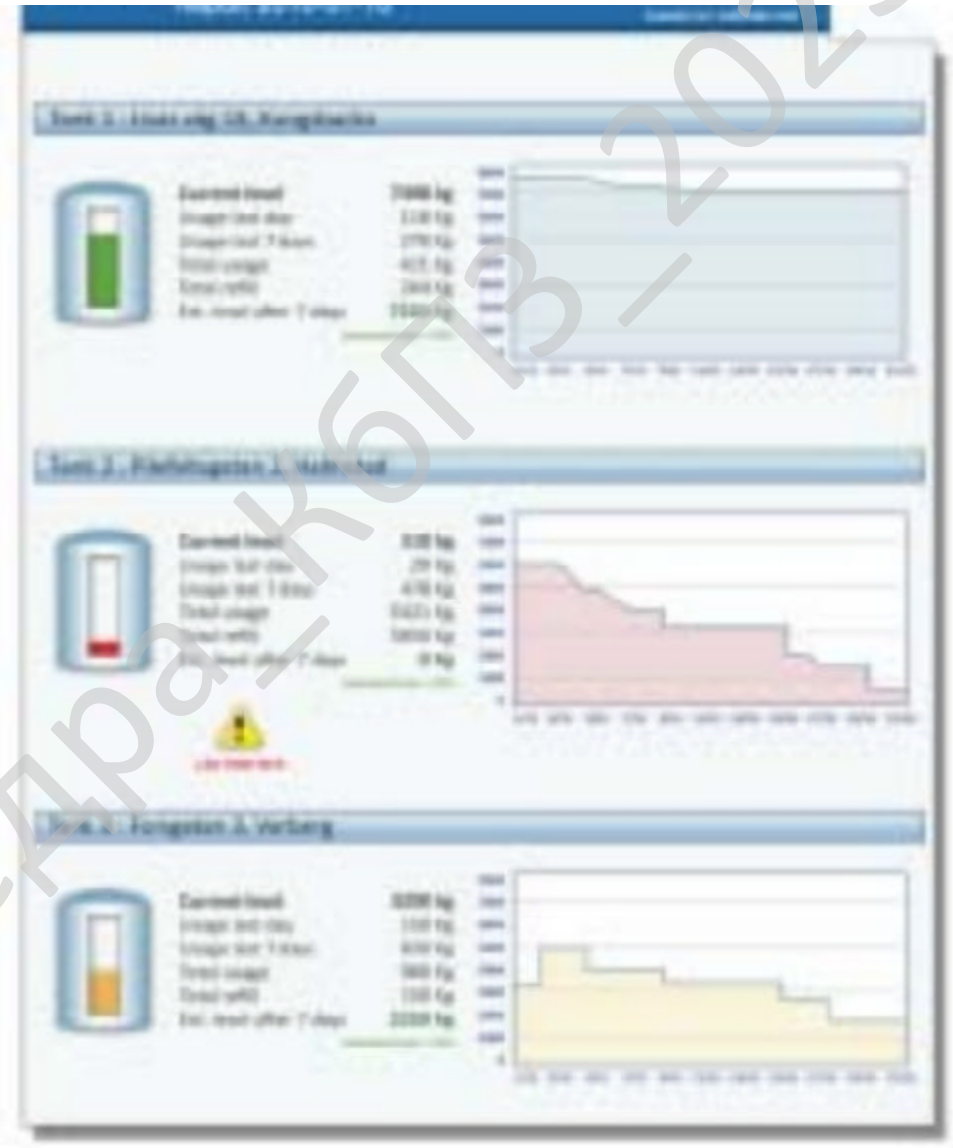

Рисунок 2.3 – Інтерфейс користувача Netbiter WebSCADA

Тривожне оповіщення:

– Два типи оповіщень: звичайні й з обов'язковим підтвердженням одержання.

– Розсилання оповіщень по SNMP, SMS або email.

– Ведення архіву тривожних оповіщень.

Міжсерверний обмін даними:

– Заснований на стандартному протоколі SOAP.

– Доступ до даних сервера Netbiter® другими системами, такими як:

– Користувальницький web/intranet сервер.

– Системи управління підприємством.

– SCADA /HMI системи.

– Інші продукти, що використовують OPC технології.

– Передача зведень по протоколу RSS.

2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки 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 часові 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 застосунку 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 першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи кібербезпеки захищеної WebSCADA системи.

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

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

б) вибрати та обґрунтувати методику побудови системи кібербезпеки контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати

поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

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

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Розроблювальна, у результаті виконання дипломного проектування, захищена WebSCADA система – це апаратно-програмний комплекс для рішення завдань диспетчерського управління енергооб'єктами різного рівня складності – від підстанцій до центрів управління мережами.

Для спрощення розробки програмної складової АСУТП, у цей час використовуються програми MMI (Man-Machine Interface – інтерфейс людина-машина) і SCADA. Застосування цих пакетів дозволяє вести автоматизовану розробку ПЗ АСУТП, здійснювати контроль і управління технологічним процесом, одержувати й обробляти інформацію про процес у зручному виді [5].

Таким системам управління необхідно оперативно вирішувати завдання автоматизації й на основі повної інформації про технологічний режим роботи підприємства планувати роботу системи, робити аналіз і резервування отриманої інформації. Основою ефективності системи є одержання достовірної інформації з локальних джерел контролю й підтримка процесів ведення контролю системи в цілому. Методом безперервного динамічного опитування системи із централізованим управлінням одержують дані реального часу із всіх ділянок виробництва. Ця й інша інформація, що супроводжує технологічний процес, накопичується у вигляді великих обсягів даних, які також необхідно резервувати й ураховувати.

SCADA-пакети складаються з декількох програмних блоків: модулі доступу й управління, сигналізації, бази даних реального часу, бази даних і модулі введення-виводу й аварійних ситуацій.

SCADA – системи застосовуються, в основному, у газовій, нафтопереробній промисловості й на великих електростанціях.

Всі SCADA-системи мають наступні загальні характеристики:

– практично однакові функціональні можливості;

– технологія програмування систем SCADA близька до інтуїтивного сприйняття автоматизуємого процесу;

– потужні засоби об’єктно-орієнтованого програмування, що робить одержувані продукти легкими в освоєнні й доступними для широкого кола користувачів.

Перевага SCADA – систем полягає в тому, що користувач може створювати сучасні системи автоматизації й модернізувати їх самотужки.

Однак комплексні системи складні в реалізації, впроваджуються довго й проблема одного модуля веде за собою масу проблем у всіх інших частинах механізму. Локальні завдання теж створюють масу складностей, тому що не завжди добре стикуються один з одним, можуть працювати на різних платформах, мати різний інтерфейс і ідеологію. Фактором, що негативно впливає на вибір і впровадження SCADA, є вартість, істотна для малих виробництв. SCADA – системи прагнуть до максимальної продуктивності й мінімізації ціни.

Застосовувані інформаційні технології – багатопоточність, об'єктна архітектура – уможливлюють застосування розроблювальної, у результаті виконання дипломного проектування, захищеної WebSCADA системи в будь-яких галузях енергетики й промисловості:

– електро-, тепло-, гідроенергетика;

– добувна промисловість;

– хімічна промисловість;

– водоканал;

– комунальне господарство.

Відмінними рисами комплексу є відкритість, модульний принцип побудови й масштабованість, що забезпечує побудови розподілених систем різної складності на всіх рівнях управління мережних компаній і можливість інтеграції з іншими автоматизованими системами управління.

Варіанти побудови комплексу й состав програмно-апаратних засобів залежать від структури АСДУ, рівня її ієрархії й виконуваних функцій. Підсистеми захищеної WebSCADA є функціонально-закінченими й можуть використовуватися незалежно в складі різних систем диспетчерського управління. Кожний варіант захищеної WebSCADA передбачає можливість подальшого нарощування й розширення розв'язуваних завдань і виконуваних функцій.

Повна конфігурація захищеної WebSCADA включає:

1. Підсистему прийому й обробки інформації:

– основний і резервний сервери телемеханіки (промислові комп'ютери, ПЗ Windows Server 2022);

– засоби синхронізації часу серверів;

– комунікаційне встаткування локальних і технологічних обчислювальних мереж;

– резервне живлення;

– програмний комплекс «Сервер збору й обробки телеінформації для систем диспетчерського управління».

2. Підсистему зберігання інформації:

– основний і резервний сервери БД (промислові комп'ютери, ПЗ Windows Server 2022, СУБД MS SQL Sever, PostgreSQL);

– пристрій зберігання даних – зовнішній дисковий накопичувач;

– резервне живлення.

3. Підсистему відображення інформації:

– робочі станції захищеної WebSCADA (персональні комп'ютери робочих місць оперативного й технічного персоналу, ПЗ ОС Windows XP Professional/7);

– «Програмний комплекс для побудови автоматизованих робочих місць диспетчерського персоналу з убудованим графічним редактором «Gred» – АРМ Диспетчера, АРМ Адміністратора, АРМ Керівника;

– резервне живлення робочих станцій АРМ.

Функції:

1. Обмін інформацією в різних телемеханічних протоколах і по TCP/IP – прийом і передача телеметричних даних, команд управління.

2. Ретрансляція телемеханічної, технологічної й релейної інформації з каналів телемеханіки або безпосередньо із сервера ТМ по декількох напрямках, у тому числі з використанням індивідуальних для кожного напрямку протоколів.

3. Сумісність із різними засобами збору даних на рівні протоколів стандарту МЕК 870-5-101/104.

4. Обробка даних – контроль якості, доразрахунки, достоверизація даних, робота з дублерами, блокуваннями.

5. Людино-машинний інтерфейс:

– перетворення й вивід даних на АРМ оперативного й диспетчерського персоналу;

– оповіщення про події, візуальна й звукова сигналізація подій, фіксуємих системою;

– телеуправління з АРМ;

– подання інформації у вигляді мнемосхем, таблиць, графіків, списків в екранних формах і у вигляді твердих копій;

– формування звітної інформації за допомогою системи генерації звітних форм, інтегрованої з електронними таблицями MS Excel;

– засоби відображення колективного користування – диспетчерські щити, відеостіни та ін.

6. Синхронізація часу від джерел точного часу (NTP-протокол, GPS, ГЛОНАСС).

7. Розвинена система WWW і WAP АРМів, що дозволяє здійснювати віддалений моніторинг стану об'єкта управління.

8. Створення архівів заданої глибини, проріджування й видалення інформації в архівах у міру її старіння, забезпечення доступу до архівних даних з боку клієнтських додатків.

9. Повноцінне «гаряче» резервування програмно-апаратних засобів з метою підвищення надійності комплексу при експлуатації.

3.2 Розробка структурної схеми

Структурна схема системи зображена на рисунку 3.1.

З неї ми бачимо, що структурна схема системи кібербезпеки захищеної захищеної WebSCADA системи складається з наступних блоків:

– WEB-сервер.

– Джерела даних.

– Сервер ТМ (сервер телемеханіки).

– Архіви.

– Мнемосхеми.

– WEB-апплети.

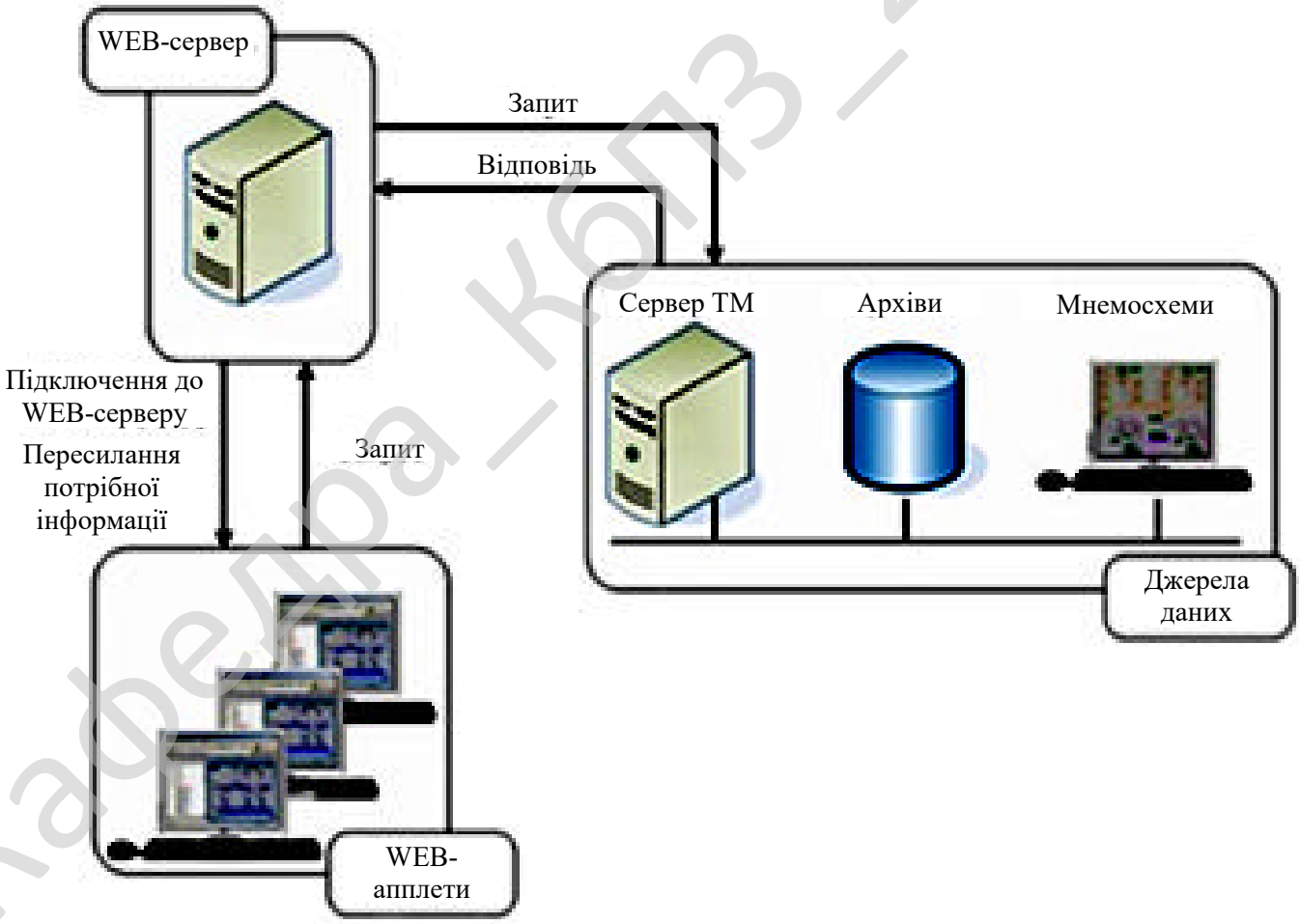

Рисунок 3.1 – Структурна схема системи

Розглянемо ці структурні блоки більш детально.

WEB-сервер

Web-сервер дозволяє створювати додатків SCADA, до яких можна одержати доступ з інтернету, використовуючи стандартний web-браузер. У такий спосіб забезпечується істотне скорочення вартості систем контролю й управління віддаленими підприємствами й технологічними процесами.

Web-сервер SCADA забезпечує зв'язок TCP між віддаленими клієнтами й сервером. Користувач може звернутися до інформації підприємства, просто зв'язуючись із Web-сервером SCADA, установленим на основній контролюючій станції. Щоб здійснювати віддалений диспетчерський моніторинг необхідно вказати IP адресу сервера в рядку web-браузера (Internet Explorer, Firefox...). При цьому буде відображена домашня сторінка для завантаження додатка й клієнтської конфігурації:

– клієнт може відображати дані й у числовому й у графічному форматі (індикатори, рядок стану, шаблони, і т.д.) з автоматичним відновленням;

– під час періоду спостереження доступні онлайн графіки (тренди);

– велика кількість клієнтів може звернутися одночасно без яких або проблем.

Web-сервер гарантує повну безпеку даних і абсолютну незалежність додатка від додатка Web-сервера, що працює в захищеному середовищі.

Джерела даних

Головна вимога до SCADA-систем – коректна робота в режимі реального часу. Причому головним пріоритетом при передачі й обробці володіють сигнали, що надходять від технологічного процесу або на нього й впливають на його протікання.

Джерела даних в SCADA – системах можуть бути наступними:

– Драйвери зв'язку з контролерами. Дуже важлива надійність драйверів зв'язку. Драйвери повинні мати засобу захисту й відновлення даних при збоях,

автоматичне повідомлення оператора й системи про втрату зв'язку, подачу сигналу тривоги при необхідності.

– Реляційні бази даних. SCADA-системи підтримують протоколи, незалежні від типу бази даних, завдяки чому як джерело даних може виступати більшість популярних СУБД: Access, Oracle і т.д. Такі підходи дозволяють оперативно змінювати настроювання технологічного процесу й аналізувати його хід поза системами реального часу, різними, спеціально створеними для цього програмами.

– Додатки, що містять стандартний інтерфейс DDE (Dynamic Data Exchange) або OLE-технологію (Object Linking and Embedding), що дозволяє зв'язувати й вбудовувати об'єкти. Це дає можливість використовувати як джерело даних у тому числі й деякі стандартні офісні додатки, наприклад Microsoft.

Введення поступаючих і вивід переданих даних організовано як система спеціальних функціональних блоків. Вхідні блоки одержують інформацію й приводять її до виду, придатного для подальшого аналізу й обробки. Блоки обробки реалізують алгоритми контролю й управління. Вихідні блоки передають керуючий сигнал від системи до об'єкта. Для зв'язку з об'єктами використовуються широко розповсюджені інтерфейси RS-232, RS-422, RS-485, Ethernet. SCADA-система може бути інтегрована із самими різними мережами: іншими SCADA-системами, офісними мережами підприємства, що реєструють і сигналізують мережами.

SCADA-системи завершують цеховий рівень автоматизації, зв'язаний, насамперед, з одержанням і візуалізацією інформації від програмувальних контролерів розподілених систем управління. Інформація, що поставляється на даний рівень, недоступна на рівні управління виробництвом. Тому важливо відзначити, що деякі фірми розробляють системи управління виробництвом і забезпечують обмін між цими рівнями [3].

Системи, описані вище, працюють під управлінням операційних систем Windows і підтримують всі сучасні технології Microsoft.

Microsoft запропонувала засіб передачі даних між процесами OLE (Object Linking and Embedding – зв'язування 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. Цей стандарт дозволяє поєднувати на рівні об'єктів різні системи управління й контролю, що функціонують у розподіленому гетерогенному середовищі; а так само усуває необхідність використання різного нестандартного встаткування й відповідних комунікаційних програмних драйверів. OPC-інтерфейс допускає різні варіанти обміну й одержання «сирих» даних з фізичних пристроїв з розподіленої системи управління.

Сервер ТМ

Сервер збору й обробки телеінформації для систем диспетчерського управління (Сервер ТМ) призначений для організації й управління роботою пунктів диспетчерського й технологічного контролю й управління різних рівнів ієрархії в автоматизованих системах диспетчерського управління (АСДУ), і є потужною системою збору, зберігання й обробки інформації.

Серверна частина комплексу функціонально складається із сервера телемеханіки (ТМ) і сервера баз даних (БД), що включають наступні програмні модулі: прийому-передачі, обробки даних, резервування, архівування, доразрахунку, авторизації користувачів, захисту від несанкціонованих прав доступу й обробки подій.

Сервер ТМ забезпечує:

– обмін інформацією із центральною приймально-передавальною станцією (ЦППС) і іншими пристроями телемеханіки;

– обміну даними між Сервером ТМ і робочими станціями клієнтських АРМів;

– роботу з архівною інформацією;

– прийом і попередню обробку прийнятих даних;

– обробку запитів клієнтів, робота з архівною інформацією, веденням бази даних реального часу (БД РЧ), оповіщення про події;

– організацію розрахунків;

– рішення серверних завдань;

– роботу з макетами формату центрального диспетчерського управління (ЦДУ);

– спостереження за станом каналів зв'язку й функціонуванням пристроїв телемеханіки;

– ведення й зберігання різних архівів і іншої нормативно-довідкової інформації (НДІ);

– роботу в режимі «гарячого» резервування;

– побудова на основі однорангових і ієрархічних мереж розподілених багаторівневих систем;

– засоби настроювання й контролю:

– спостереження за станом каналів зв'язку й функціонуванням пристроїв телемеханіки;

– синхронізацію даних;

– ретрансляцію даних;

– міжмашиний обмін даними.

Сервер БД здійснює запис подійної і регулярної інформації, організує зберігання накопиченої архівної інформації, обслуговує клієнтів, забезпечує прив'язку архівної й конфігураційної інформації, підтримує резервне копіювання.

Архіви

Архівний сервер здійснює повний менеджмент архівів для систем автоматизації з підвищеними вимогами до обсягів або гнучкості зберігання даних, а також їхньої доступності для обробки зовнішніми системами.

Передбачено індивідуальна для кожного елемента проекту настроювання місця зберігання даних або повідомлень (у класичному файловому архіві захищеної WebSCADA або будь-якому зовнішньому SQL-сервері), а також спосіб

їхнього зберігання (список рівнів) і попередньої обробки перед переміщенням в архів. Архівний сервер підтримує пряме одержання даних з архівів контролерів, OPC HDA серверів, а також надає доступ до накопиченої інформації через OPC HDA або SQL.

Можливості архівного сервера:

– Модуль доступний як розширення будь-якого базового комплекту захищеної WebSCADA, так і у вигляді окремого базового комплекту.

– Зберігання архівів даних і повідомлень в MS SQL Server 2005 (підтримка зберігання архівів в інших SQL серверах можлива за запитом). Одна робоча станція виділяється як архівний сервер, інші – архівні клієнти. Архівний сервер зберігає функціонал робочої станції, що дозволяє використовувати його навіть в однокомп'ютерних системах, а також формувати сервісні мнемосхеми. Архівний сервер має модуль для обслуговування багатьох потоків запису даних і обробки запитів на читання даних.

– Буферизація запису (кеш) даних на кожній робочій станції забезпечує безрозривний запис архівів навіть у випадку тимчасової відсутності зв'язку із сервером.

– Окремий рівень зберігання для кожного часового інтервалу (мінута, година й т.п.) і можливістю попередньої обробки (усереднення, інтегрування й т.п. – функціонал може бути доданий по запиті) перед записом для тих самих даних з метою швидкої зміни масштабу трендів (при використанні архівного сервера).

– Збереження набору настроювань архівування параметрів у вигляді шаблона, прив'язаного до шкали (з можливістю перевизначення для окремого параметра).

– Одержання повідомлень архівного сервера всіма комп'ютерами мережі.

Мнемосхеми

Мнемосхема – спосіб подання технологічної інформації в графічному виді. Мнемосхема – сукупність сигнальних пристроїв і сигнальних зображень

устаткування й внутрішніх зв'язків контрольованого об'єкта, розташовуваних на диспетчерських пультах, операторських панелях або виконаних на персональному комп'ютері. Інформація, що виводиться на мнемосхему може бути представлена у вигляді аналогового, дискретного й релейного сигналу, а також графічно. На мнемосхемах відбивається основне устаткування, сигнали, стан регулювальних органів. Допоміжний і довідковий матеріал повинен бути розташований у додаткових формах відображення, з можливостями максимально швидкого добування цих допоміжних форм на екран.

Візуалізація техпроцесу – спосіб відображення інформації про стан технологічного встаткування й параметри технологічного процесу на моніторі комп'ютера або операторської панелі в системі автоматичного керування в промисловості, що передбачає також графічні способи керування техпроцесом. Система візуалізації повинна враховувати вимоги, пропоновані до людино-машинного інтерфейсу. Візуалізація техпроцесу реалізується в ряді екранів або вікон, які можуть являти собою ієрархічну систему. В основі системи відображення лежить мнемосхема техпроцеса, статичне зображення у візуально простій і інтуїтивно зрозумілій формі яка показує елементи встаткування, можливо, оброблювані матеріали й продукцію, і їхня взаємодія, порядок обробки. Статична мнемосхема пожвавлюється – анімується, відображаючи реальний стан устаткування й сировини. При цьому використовуються різні методи:

– Зміна кольору об'єкта залежно від його стану. Наприклад, відповідно до вимог ергономіки, небезпечні або аварійні об'єкти офарблюються в червоний колір. Можна також використовувати миготливе фарбування.

– Зміна графічного образа залежно від стану об'єкта. Наприклад, повний або порожній контейнер, положення ручки рубильника.

– Використання мультиплікації, тобто послідовності швидко, що переміняються кадрів.

– Переміщення об'єктів по екрані.

– Зміна розміру об'єкта.

Сучасні засоби обчислювальної техніки й монітори мають багаті графічні можливості, широке використання яких може ввійти в суперечність із вимогами ергономіки. Так, для відображення стану системи не можна використовувати багато колірної палітри. Подання інформації повинне обмежуватися простими й однозначно сприйманими кольорами (червоний, зелений, жовтий, білий, чорний, сірий). Не можна використовувати й занадто дрібний шрифт у написах. Надмірне захоплення динамічними картинами, наприклад, мультиплікацією відволікає й стомлює оператора. Погіршує сприйняття й використання фотореалістичних зображень об'єктів. Для керування техпроцесом на мнемосхемі розташовуються елементи графічного інтерфейсу, найчастіше типові для сучасного програмного забезпечення: вікна введення-виводу, кнопки, повзунки (слайдери). Крім цього можуть використовуватися події, наприклад, клацання кнопками миші, на елементах зображення. На додаток до динамізованої мнемосхеми використовуються спеціальні або розташовані поверх мнемосхеми вікна, у яких відображаються залежності параметрів техпроцеса від часу, а також текстові повідомлення про стан системи й дії оператора. Сучасні засоби проектування операторських систем керування SCADA, як правило містять убудовані редактори, що дозволяють здійснити всі завдання проектування візуалізації.

Залежно від складності проекту автоматизації SCADA-додаток може містити одну й більше мнемосхему. При реалізації проектів застосовуються сучасні TFT-монітори з великою діагоналлю й, відповідно, з більшим розрішенням екрана. Ці міри дозволюють створювати мнемосхеми, що відображають весь техпроцес на одному екрані, що безсумнівно позначається на продуктивності роботи оператора.

Розробку мнемосхем можна розбити на такі кроки:

– малювання в графічному редакторі фонових картинок;

– створення вікон для мнемосхем із шаблона;

– "набрасуваннння" візуальних компонентів (написи, повзунки, поршні, контейнери картинок і т.д.) на форму;

– настроювання властивостей візуальних компонентів;

– "прив'язка" тегів з конфігурації проекту до візуальних компонентів;

– програмування, при необхідності, додатковій функціональності додатка.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно. Модульна інтегрована захищеної WebSCADA дозволяє вибрати ті компоненти, які необхідні користувачеві.

Компоненти верхнього рівня

1. Середовище розробки. Головними функціями Середовища розробки є:

– Конфігурування системи контролю й керування.

– Створення й верифікація Бази Даних реального часу.

– Розробка графічного інтерфейсу користувача (відеокадрів бази даних).

– Вибір і настроювання алгоритмів керування.

– Імітація роботи системи контролю й керування.

– Створення ієрархічної (об'єктної) структури змінних БД.

– Створення й редагування користувальницьких скриптів.

– Надання програмного інтерфейсу (API) для створення прикладних програм користувача.

2. Сервер БД. Забезпечує виконання функцій обробки змінних БД, ведення оперативних трендів, програмного перезапуску, обробки подій, статистики, OPC-сервера, драйвера ODBC. OPC (OLE for Process Control) – набір повсюдно прийнятих специфікацій, що надають універсальний механізм обміну даними в системах контролю й керування. OPC технологія забезпечує незалежність споживачів від наявності або відсутності драйверів або протоколів, що дозволяє вибирати встаткування й програмне забезпечення, що найбільше повно відповідає реальним потребам бізнесу.

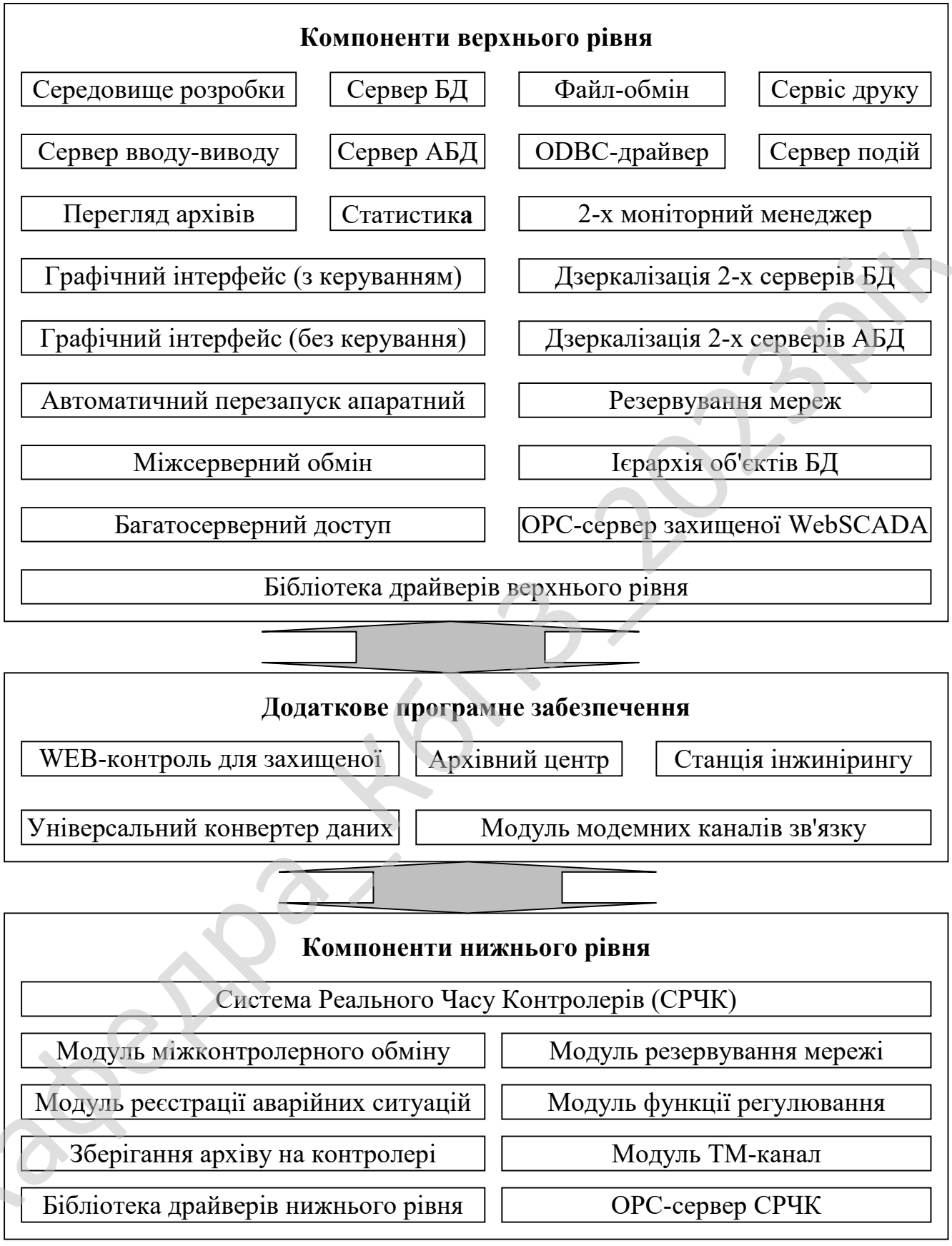

Рисунок 3.2 – Функціональна схема системи

3. Сервер АБД. Забезпечує виконання всіх функцій Сервера БД (див. Сервер БД), а також функцій обробки й зберігання архівів трендів, протоколу подій, друкованих документів від року й більше.

4. Сервер вводу-виводу. Забезпечує організацію зв'язку різних УСО з захищеної WebSCADA через драйвери верхнього рівня, що підключаються, (не СРВ РС-сумісних контролерів), а також виконання стандартних обробок.

5. Графічний інтерфейс (з керуванням). Під керуванням розуміється можливість оператора за допомогою динамічних елементів (кнопка, поле введення й т.п.) внесення змін у роботу АСУ ТП (наприклад, зміна завдання або коефіцієнтів настроювання регуляторів, включення/вимикання виконавчих механізмів, зміна границь сигналізації, керування каналами зв'язку й резервуванням і т.д.)

6. Графічний інтерфейс (без керування). Користувач не може вносити які-небудь зміни в хід технологічного процесу, крім квітування світлової й звукової сигналізації.

7. Перегляд архівів. Обов'язковий модуль для комплексів, що працюють із архівними даними. Забезпечує доступ до архівів трендів і до архівів протоколів подій.

8. Дзеркалізація 2-х серверів БД. Дзеркалізація (періодичне резервне копіювання в режимі on-line) даних з основного в резервний сервер. Мережа, по якій здійснюється дзеркалізація, призначається Користувачем на етапі генерації. Основний і резервний сервери БД установлюються на двох різних АРМ. Дзеркалізуєма інформація: оперативна БД, тренди, друковані документи, протокол подій.

9. Дзеркалізація 2-х серверів АБД. Аналогічно дзеркалізації 2-х серверів БД. Крім того, дзеркалізуються архіви історичних трендів, архіви друкованих документів і архіви протоколів подій.

10. Автоматичний перезапуск апаратний. Використовується при установці в комп'ютер (АРМ) спеціальної плати автоматичного перезапуску. Разом із

платою забезпечується автоматичний перезапуск операційної системи у випадках зависання й зациклення.

11. Сервіс друку. Забезпечує функції автоматичної видачі звітних документів і протоколу подій на пристрої друку.

12. Файл-обмін. Забезпечує прийом/передачу файлів даних з/у систему захищеної WebSCADA. Даними для прийому/передачі можуть бути наступні: поточні значення змінних, атрибути змінних (параметри настроювання). Обмін файлами даних здійснюється як з ініціативи зовнішньої системи, так і з ініціативи системи захищеної WebSCADA. Дані передаються у двійковому форматі.

13. Резервування мереж. Забезпечує виконання наступних функцій резервування: мережа контролера – сервер БД/АБД; мережа сервер БД/АБД – клієнти сервера БД/АБД (у тому числі при резервуванні серверів і клієнтів).

14. 2-х моніторний менеджер. Забезпечує функціонування АРМ із двома моніторами. Потрібна установка в системний блок комп'ютера графічного адаптера, що підтримує двомоніторний режим роботи. Реалізує функції менеджера вікон, що забезпечує просте й зручне автоматичне керування вікнами на двох моніторах.

15. Ієрархія об'єктів БД. Створення ієрархічної структури даних РВ відповідно до конструкторського, технологічного або іншого угруповання параметрів. Формування узагальнених ознак сигналізації й вибірок. Підтримує існуючі системи кодувань і дозволяє створювати власні.

16. Статистика. Дозволяє провести вибірку змінних для будь-якого часового інтервалу, за який є тренди, і застосувати бібліотечні функції статистичної обробки. Можливість роботи з архівними даними. Отриманий результат може бути використаний у графічному інтерфейсі й при формуванні звітних документів

17. Сервер подій. Нова система подій створена для організації більш гнучкого й ефективного відстеження реакцій системи на виникаючі події в контрольованому об'єкті. Система подій надає Користувачеві величезні можливості по вибірках (фільтрам) найрізноманітніших видів і категорій подій,

починаючи від комбінацій символів у позиції змінної (підтримка систем класифікації типу AKS, KKS і будь-яких інших), вибіркою подій по одному агрегату або пристрою й закінчуючи всіма подіями для однієї змінної. Для обробки, формування й візуалізації подій розроблені й істотно модифіковані багато компонентів як середовища розробки, так і середовища виконання захищеної WebSCADA.

18. Міжсерверний обмін. Міжсерверний обмін призначений для прямого обміну інформацією між серверами різних АСУ ТП на базі захищеної WebSCADA. Обмін виробляється паспортами обраних змінних і пов'язаними з ними подіями. Дозволяє будувати системи контролю й керування зі складними багатошаровими архітектурами й функціоналом.

19. Багатосерверний доступ. Багатосерверний доступ – це можливість легкого перемикання доступу клієнтських станцій до декількох Серверів бази даних через інтерфейс Користувача. Клієнтські додатки одержують можливості по керуванню й діагностиці декількох Серверів бази даних: відображення узагальненої звукової сигналізації із заданих Серверів БД; зміна графічного проекту й підключення клієнта до іншого Сервера БД по команді із графічного інтерфейсу.

20. OPC-сервер захищеної WebSCADA (специфікація DA+HDA). Модуль призначений для передачі оперативних і архівних даних з захищеної WebSCADA у системи сторонніх виробників відповідно до стандартів OPC DA і OPC HDA.

21. Бібліотека драйверів верхнього рівня. Безкоштовна бібліотека драйверів для різних контролерів, модулів введення/виводу й інших цифрових приладів.

22. ODBC-драйвер. Відкритий інтерфейс доступу до БД захищеної WebSCADA являє собою бібліотеку функцій, що дозволяє прикладній програмі звертатися до бази даних захищеної WebSCADA, використовуючи структуровану мову запитів SQL.

Додаткове програмне забезпечення

1. WEB-контроль для захищеної WebSCADA. Забезпечує перегляд поточної інформації у вигляді мнемосхем, друкованих документів, трендів, протоколу подій на будь-якому персональному комп'ютері за допомогою MS Internet Explorer. Взаємодіє із серверами БД і АБД. Підтримка списку улюблених кадрів. Захист від несанкціонованого доступу. Можлива робота як у режимі без функцій керування, так і з функціями керування.

2. Універсальний конвертер даних. Конвертує дані (у т.ч. друковані документи, тренди, протоколи повідомлень) у формат MS Excel і XML. Дозволяє робити вибірку по параметрах і за часом.

3. Архівний центр. Призначений для збору архівних і інших даних від різних АСУ ТП, побудованих на базі пакета захищеної WebSCADA для Windows, з подальшою можливістю їхнього перегляду. Має власний протокол подій, колірну й звукову сигналізацію, має можливість дубльованого копіювання даних. Для перегляду архівів на комп'ютері Архівного центра необхідно мати або Середовище Розробки (поставляється безкоштовно), або Універсальний конвертер даних.

4. Модуль модемних каналів зв'язку. Дозволяє будь-яким драйверам захищеної WebSCADA і OPC-серверам, використовувати GSM- і GPRS-канали стільникових мереж, лінії телефонної мережі, що комутирується, загального користування й інші модемні канали зв'язку.

5. Станція інжинірингу. Конфігурування й настроювання контролерів; діагностика роботи контролерів у режимі реального часу. Віддалене програмування контролерів під керуванням СРЧК. Крім того, станція інжинірингу дозволяє робити діагностику каналів зв'язку, а також виконувати ряд додаткових сервісних функцій.

Компоненти нижнього рівня

1. Система Реального Часу Контролерів (СРЧК).

До складу базової версії середовища виконання контролерів входять:

– Модулі зв'язку зі станцією оператора.

– Модуль зв'язку зі станцією інжинірингу.

– Драйвери плат введення/виводу.

– Модуль сервера бази даних реального часу.

– Служба автовідновлення програм.

– Модуль візуалізації.

– Передача керування резервному процесорному модулю у випадку виходу з ладу основного процесорного модуля.

– Дзеркалізація даних з основного процесорного модуля контролера на резервний і забезпечення автоматичного відновлення інформації про зміни конфігурації на резервному процесорному модулі.

– Забезпечення ненаголошеного переходу регуляторів при переході з основного на резервний процесорний модуль.

– Передача керування резервному контролеру у випадку виходу з ладу процесорного модуля або модулів введення/виводу основного контролера.

– Дзеркалізація даних з основного контролера на резервний, діагностика каналів передачі даних .

– Забезпечення ненаголошеного переходу регуляторів при переході з основного на резервний контролер або процесорний модуль.

– Синхронізація контролерів при старті (або команді оператора) шляхом повної дзеркалізації даних СРЧК по дубльованим мережам Ethernet.

– Збереження безперервності інформаційного обміну при відмові одного із процесорних модулів або каналів введення/виводу.

– Забезпечення ненаголошеного переходу регуляторів при відмові одного з контролерів кластера.

2. Модуль міжконтролерного обміну. Модуль міжконтролерного обміну призначений для створення розподілених систем з використанням обміну даними між декількома контролерами. Модуль виконує наступні функції:

– обмін оперативними й розрахунковими значеннями бази даних;

– обмін значеннями атрибутів змінні бази даних;

– підтримка резервування мережі Ethernet для міжконтролерного обміну;

– діагностика каналів зв'язку міжконтролерного обміну.

3. Модуль резервування мережі. Модуль робить діагностику мережі Ethernet для зв'язку із сервером БД/АБД, реєстрацію несправностей мережі й забезпечує можливість автоматичного переходу на резервну мережу у випадку відмови одного з каналів зв'язку.

4. Модуль реєстрації аварійних ситуацій (РАС). Модуль РАС використовується для реєстрації аварійних ситуацій у контролері, тобто для збереження заздалегідь заданих параметрів у певному інтервалі часу при виникненні аварійної ситуації. Модуль забезпечує період реєстрації від 50 мс.

Модуль виконує наступні функції:

– Підтримка двох режимів реєстрації РАС:

а) режим реєстрації значень параметрів із заданою періодичністю;

б) режим реєстрації значень параметрів по зміні.

– Формування перед- і післяаварійних протоколів на енергонезалежному накопичувачі в контролері. Для перегляду протоколів РАС використовується спеціалізована програма верхнього рівня.

При цьому модуль має можливості:

– контролювати кілька РАС;

– задавати кілька умов формування РАС;

– набудовувати період і глибину реєстрації.

5. Модуль функції регулювання. Модуль забезпечує підтримку регуляторів. Дозволяє створювати контури ПІД-регулювання (у тому числі каскадного й багатозв’язного). Користувальницькі алгоритми, розгалужена система сигналізацій, функції динамічних і статичних балансувань, компенсації люфтів, а також безліч інших додаткових настроювань дозволяють реалізувати найрізноманітніші системи регулювання.

6. Зберігання архіву на контролері. Дозволяє формувати й зберігати архіви в пам'яті контролера. Архіви можуть зберігатися як в оперативній, так і в енергонезалежній пам'яті контролера залежно від розв'язуваного завдання.

7. Модуль ТМ-канал. ТМ-канал дозволяє організувати інформаційний обмін по повільних і нестійких каналах зв'язку. Ідеально підходить для зв'язку із пристроями по радіоканалі або GSM/ GPRS-каналу низької якості й т.п.

8. Бібліотека драйверів нижнього рівня. Дозволяє організувати інформаційний обмін між контролером під керуванням СРЧК і різноманітними приладами, модулями введення/виводу, панелями операторів, а також контролерами під керуванням власних ОС, таких як Omron, CIU 850 серії й т.д. Багато хто із драйверів мають додаткового функціонала, наприклад, дозволяють використовувати резервуємі канали зв'язку, зчитувати архіви приладів і т.д.

9. OPC-сервер СРЧК (підтримка специфікації DA+HDA). Призначений для передачі оперативних і архівних даних відповідно до стандарту OPC з контролера під керуванням СРЧК у будь-які SCADA/HMI-системи сторонніх виробників (захищеної WebSCADA підтримує прямий інформаційний обмін з контролером), може бути використаний для одночасної передачі даних у кілька інформаційних систем верхнього рівня.

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

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання дипломного проектування, наведена на рисунку 3.3. З нього видно, що процеси, які відбуваються у системі взаємодіють наступним чином. Спершу запускається процес завантаження графічного інтерфейсу.

Він взаємодіє з наступними процесами:

– Процес керування роботою пристроїв.

– Процес роботи архівного центру.

– Процес контролю поточних параметрів пристрою.

– Процес завантаження бази даних апаратних засобів.

Рисунок 3.3 – Діаграма взаємодії процесів

Процес керування роботою пристроїв взаємодіє з наступними процесами:

– Процес налаштування параметрів зв'язку з пристроями.

– Процес призначення команд, який у свою чергу взаємодіє з процесом налагодження команд.

– Процес автоматичного керування.

– Процес керування з мнемосхеми.

Процес роботи архівного центру взаємодіє з наступними процесами:

– Процес журналювання подій.

– Формування звітів.

Процес контролю поточних параметрів взаємодіє з наступними процесами:

– Процес обробки даних та обчислення.

– Процес обробки змін параметрів.

– Процес сигналізації виключних ситуацій.

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

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

Спершу відбувається виведення основного вікна програми.

Після цього відбувається перевірка, чи необхідно додати в БД нові пристрої.

Якщо необхідно, тоді відбувається додавання в БД нових пристроїв.

Після цього відбувається перевірка, чи є підключені пристрої.

Якщо є, тоді відбувається виведення на екран поточних параметрів підключених пристроїв.

Якщо необхідно керувати роботою пристроїв, тоді відбуваються наступні дії:

– Налаштовуються параметри зв’язку з пристроями.

– Призначаються команди пристроям.

– Відбувається налагодження команд.

– Відбувається запуск призначених команд.

Якщо треба обробити вимірювання, тоді запускається підпрограма обробки змін параметрів пристроїв.

Якщо необхідно створити звіти, то відбувається створення звітів з використанням архівних даних.

Після цього користувач обирає, працювати йому далі з програмою, або ні.

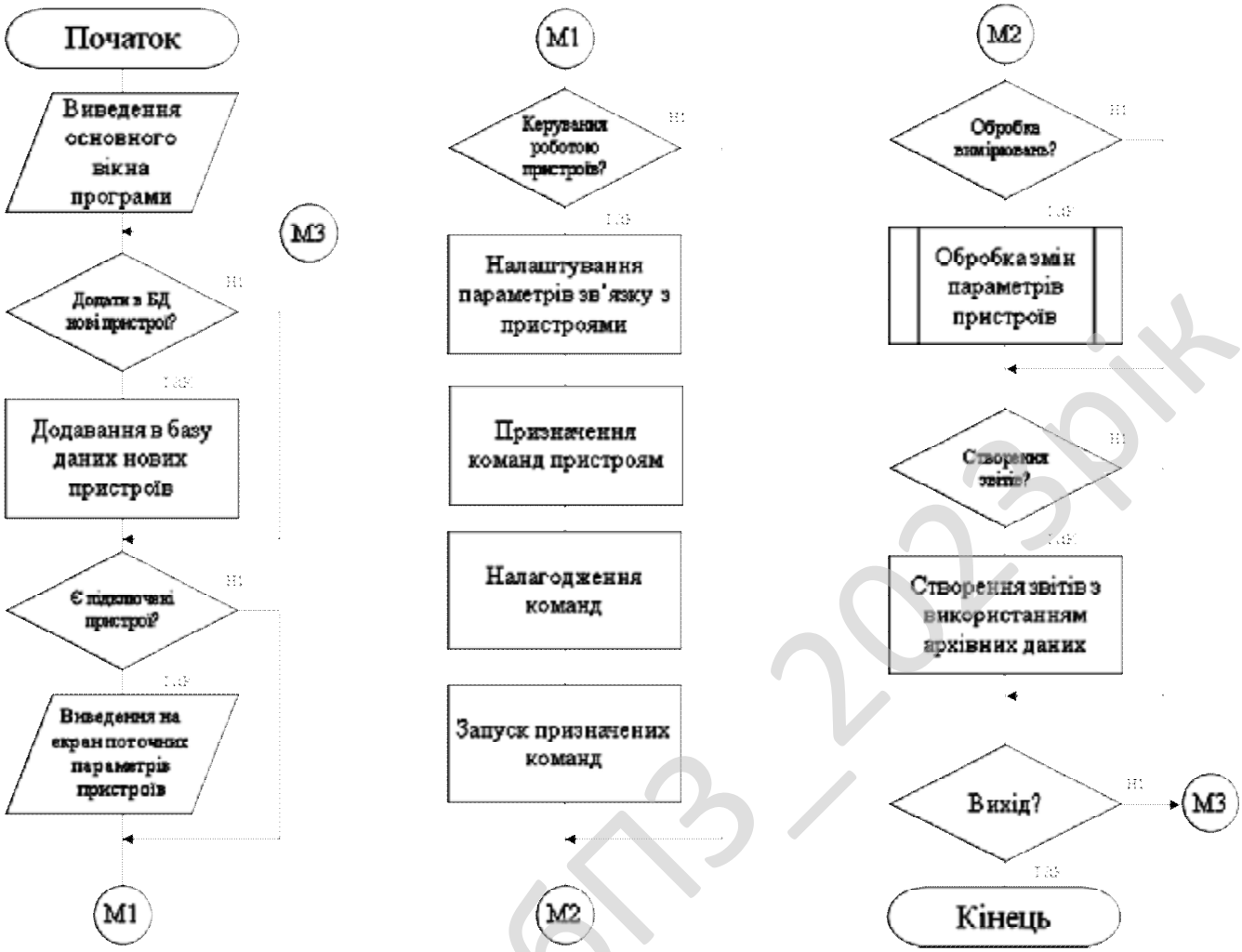

Рисунок 4.1 – Блок-схема основної програми

На рисунку 4.2 зображена блок-схема алгоритму роботи підпрограми обробки змін параметрів пристроїв. Вона працює наступним чином.

Спершу відбувається зчитування значень параметрів.

Після цього параметри відображаються на мнемосхемі.

Якщо значення параметрів є недопустимим, тоді включається сигналізація виключних ситуацій.

Якщо необхідно провести розрахунки, тоді виконуються наступні дії:

– Вибирається формула та додаються у неї потрібні параметри у список використовуємих змінних.

– Здійснюються обчислення.

– Виводяться результати.

Якщо необхідно провести архівацію параметрів, тоді здійснюється архівація параметрів.

Якщо ж необхідно провести відображення графіків, тоді виконуються наступні дії:

– Обирається параметр.

– Обирається вид графіку.

– Відображаються зміни

Після цього підпрограма завершує свою роботу.

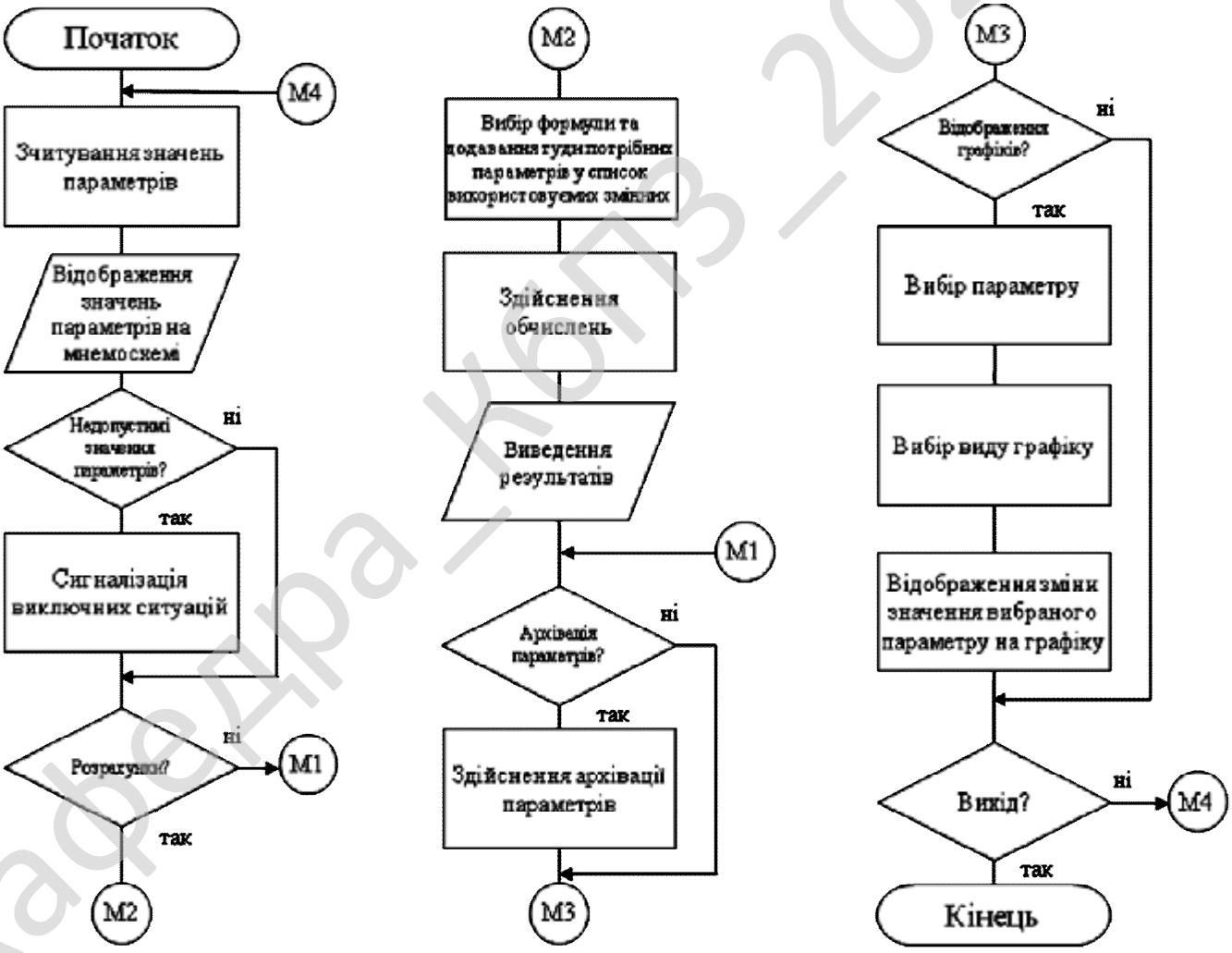

Рисунок 4.2 – Блок-схема алгоритму роботи підпрограми обробки змін параметрів пристроїв

Опишемо елементи системи з розбиттям по функціям.

Модульність

Для додання гнучкості й високого ступеня масштабованості система захищеної WebSCADA побудована за модульним принципом. Тісна інтеграція модулів з ядром поліпшує стабільність системи в цілому, завдяки повторному використанню налагодженого коду. Однак сам процес розробки власного коду модулів захищеної WebSCADA накладає більшу відповідальність, можливі помилки вводять елемент нестабільності в систему. Можливість створення розподілених конфігурацій згладжує цю небезпеку. Модулі системи захищеної WebSCADA зберігаються в динамічних бібліотеках. Кожна динамічна бібліотека може містити безліч модулів різного типу. Наповнення динамічних бібліотек модулями визначається функціональною зв'язністю самих модулів. Динамічні бібліотеки допускають гарячу заміну, що дозволяє в процесі функціонування робити відновлення окремих частин системи. Метод зберігання коду модулів у динамічних бібліотеках є основним для системи захищеної WebSCADA, оскільки підтримується практично всіма сучасними операційними системами (ОС). Однак це не виключає можливості розробки інших методів зберігання коду модулів.

На основі модулів реалізовані наступні функціональні частини системи захищеної WebSCADA:

– бази даних;

– архіви (повідомлень і значень);

– протоколи комунікаційних інтерфейсів;

– комунікаційні інтерфейси, транспорти;

– джерела даних і збір даних;

– інтерфейси користувача (GUI, TUI, WebGUI, speach, signal ... .);

– додаткові модулі, спеціальні.

Керування модулями здійснюється підсистемою «Керування модулями». Функціями підсистеми є: підключення, відключення, відновлення модулів, а також інші операції, пов'язані з модулями й бібліотеками модулів.

Підсистеми

Архітектурно система захищеної WebSCADA 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 захищеної WebSCADA містить 9 підсистем з них 7 підсистем є модульними. 9 підсистем системи захищеної WebSCADA є базовими й присутні в будь-якій конфігурації. До списку 9 підсистем можуть додаватися нові підсистеми за допомогою модулів. Підсистеми системи захищеної WebSCADA:

– Архіви (модульна).

– Бази даних (модульна).

– Безпека.

– Інтерфейси користувача (модульна).

– Керування модулями.

– Збір даних (модульна).

– Транспортні протоколи (модульна).

– Спеціальні (модульна).

– Транспорти (модульна).

PLC і інші джерела динамічних даних. Підсистема "Збір даних"

Для забезпечення підтримки джерел динамічних даних, будь те PLC-Контролери, плати УСО, віртуальні джерела й т.д., призначена підсистема «Збір даних». У функції цієї підсистеми входить надання отриманих даних у структурованому виді й забезпечення керування цими даними, наприклад, модифікація даних.

Підсистема «Збір даних» є модульною й, як наслідок, містить модульні об'єкти типів джерел динамічних даних. захищеної WebSCADA підтримує наступні типи джерел даних:

– Плати збору даних від "Diamond systems".

– Збір даних операційної системи (ОС).

– Блоковий обчислювач.

– Обчислювач на BuilderC.

– Транспортер дані підсистеми "Збір даних" від однієї захищеної WebSCADA станції до іншої.

– Доступ до логічних контролерів за допомогою протоколу "ModBUS".

– Збір даних мережних пристроїв за допомогою протоколу SNMP.

– Джерело даних логічного рівня системи захищеної WebSCADA.

– Доступ до високоінтелектуальних логічних контролерів за допомогою протоколу MPI і комунікаційного процесора CIF50PB, фірми Hilscher GMBH.

Кожний тип джерела виконаний у вигляді окремого модуля, що може бути підключений/відключений. Кожний тип джерела може містити окремі джерела (контролери).

Окремо взятий контролер може містити параметри заданих модулем типів. Наприклад, параметри аналогового типу; основною інформацією, що вони надають, є значення цілого або дійсного типу. Структурно параметр являє собою список атрибутів, які й містять дані. Атрибути можуть бути чотирьох базових типів: символьний рядок(текст), цілий, дійсний й логічний тип.

Структури контролерів, параметрів і їхніх типів утримуються в підсистемі "Збір даних", а об'єкти модулів виконують їхнє заповнення відповідно до власної специфіки. Джерело динамічних даних може бути віддаленим, тобто бути підключений на віддаленій системі захищеної WebSCADA. Для зв'язку з такими джерелами даних використовується транспортний тип контролерів (Transporter). Функцією даного типу джерела даних є відбиття джерел даних віддаленої захищеної WebSCADA станції на локальну станцію.

Бази даних. Підсистема "Бази даних"

Для зберігання даних системи повсюдно використовуються бази даних (БД). З метою систематизації доступу й керування базами даних у системі захищеної WebSCADA передбачена підсистема "Бази даних". Для забезпечення підтримки різних БД/СУБД підсистема виконана модульною.

У ролі модульних об'єктів, що втримуються в підсистемі, виступає тип БД/СУБД, тобто модуль підсистеми «Бази даних» практично містить реалізацію доступу до певного типу БД. Наприклад, модулі: DBF, MySQL, SQLite.

Об'єкт типу БД/СУБД у свою чергу містить список об'єктів окремих БД даного типу, а об'єкт БД містить список об'єктів таблиць, які й містять дані в табличній формі.

Практично всі дані системи захищеної WebSCADA зберігаються в тій або іншій БД. Інструментарій системи дозволяє легко переносити дані з одного типу БД в інший, і, як наслідок, оптимально підбирати тип БД під конкретну область застосування системи захищеної WebSCADA. Перенос інформації з однієї БД в іншу може бути виконаний двома способами. Перший – це зміна адреси робочої БД і збереження всієї системи на неї, другий – це пряме копіювання інформації між БД. Крім копіювання підтримується й функція прямого редагування вмісту таблиць БД.

Для організації централізованого доступу розподіленої системи до єдиного БД передбачаються два способи. Перший це використання мережних СУБД, наприклад, MySQL. Другий спосіб це використання транспортного типу БД на локальних системах для доступу до одній центрального БД . Функцією транспортної БД є пересилання запитів до БД на віддалену захищеної WebSCADA систему.

Дані можуть зберігатися також у конфігураційному файлі системи. Реалізовано механізм повного відбиття структури БД на структуру конфігураційного файлу. Тобто стандартну конфігурацію можна розміщати в конфігураційному файлі. Суть такого механізму в тому, що дані системи за

замовчуванням, наприклад, при старті без БД можна описувати в конфігураційному файлі. Надалі ці дані можуть перевизначатися в БД. Крім цього для випадків неможливості запуску який або БД взагалі можна всі дані зберігати в конфігураційному файлі.

Для доступу до баз даних використовується механізм реєстрації БД. Зареєстровані в системі БД доступні всім підсистемам системи захищеної WebSCADA і можуть використовуватися в їхній роботі. Завдяки цьому механізму можна забезпечити розподіленість зберігання даних. Наприклад, різні бібліотеки можуть зберігатися й поширюватися незалежно, а підключення бібліотеки буде полягати в простої реєстрації потрібної БД.

Надалі планується реалізація дублювання БД шляхом зв'язування зареєстрованих БД. Цей механізм дозволить значно підвищити надійність системи захищеної WebSCADA у цілому шляхом резервування механізму зберігання даних.

Архіви. Підсистема "Архіви"

Будь-яка SCADA система надає можливість архівування зібраних даних, тобто формування історії зміни (динаміки) процесів. Архіви умовно можна розділити на два типи: архіви повідомлень і архіви значень.

Особливістю архівів повідомлень є те, що архівуються так звані події. Характерною ознакою події є час виникнення цієї події. Архіви повідомлень звичайно використовуються для архівування повідомлень у системі, тобто ведення логів і протоколів. Залежно від джерела повідомлення можуть класифікуватися за різними критеріями. Наприклад, це можуть бути протоколи аварійних ситуацій, протоколи дій операторів, протоколи збоїв зв'язку й ін.

Особливістю архівів значень є їхня періодичність, обумовлена проміжком часу між двома суміжними значеннями. Архіви значень застосовуються для архівування історії безперервних процесів. Оскільки процес безперервний, те й архівувати його можна тільки шляхом введення поняття квантування опитування значень, оскільки інакше ми одержуємо архіви нескінченних розмірів, через

безперервність самої природи процесу. Крім цього практично ми можемо одержувати значення з періодом обмеженим самими джерелами даних. Наприклад, досить якісні джерела даних у промисловості рідко дозволяють одержувати дані із частотою більше 1 кГц. І це без обліку самих датчиків, що мають ще менш якісні характеристики.

Для рішення завдань архівування потоків даних у системі захищеної WebSCADA передбачена підсистема "Архіви". Підсистема "Архіви" дозволяє вести як архіви повідомлень, так і архіви значень. Підсистема "Архіви" є модульною. Модульним об'єктом, що втримується в підсистемі "Архіви", виступає тип архіватора. Тип архіватора визначає спосіб зберігання даних, тобто сховище (файлова система, СУБД, мережа й т.д.). Кожний модуль підсистеми "Архіви" може реалізовувати як архівування повідомлень, так і архівування значень. Підсистема "Архіви" може містити безліч архівів, що обслуговуються різними модулями підсистеми.

Повідомлення в системі захищеної WebSCADA характеризується датою, рівнем важливості, категорією й текстом повідомлення. Дата повідомлення вказує на час створення повідомлення. Рівень важливості вказує на ступінь важливості повідомлення. Категорія визначає адресу або умовний ідентифікатор джерела повідомлення. Звичайно, категорія містить повний шлях до джерела повідомлення в системі. Текст повідомлення, властиво, і несе значеннєве навантаження повідомлення.

У процесі архівування повідомлення пропускаються через фільтр. Фільтр працює за рівнем важливості й категорії повідомлення. Рівень повідомлення у фільтрі вказує, що потрібно пропускати повідомлення із зазначеним або більше високим рівнем важливості. Для фільтрування по категорії застосовуються шаблони або регулярні вираження, які визначають які повідомлення пропускати. Кожний архіватор містить власні настроювання фільтра. Отже можна легко створювати різні спеціалізовані архіватори для архіву повідомлень. Наприклад, архіватори повідомлень можна спеціалізувати на:

– логи, для зберігання відладочної інформації й іншій робочій інформації сервера;

– різні протоколи (протокол дій клієнтів, протокол порушень і виключень, протокол подій ...).

Архів значень у системі захищеної WebSCADA виступає як незалежний компонент, що включає буфер, оброблюваний архіваторами. Основним параметром архіву значення є джерело даних. У ролі джерела даних можуть виступати атрибути параметрів системи захищеної WebSCADA, а також інші зовнішні джерела даних (пасивний режим). Іншими джерелами даних можуть бути мережні архіватори віддалених захищеної WebSCADA систем, середовище програмування системи захищеної WebSCADA і ін.

Ключовим компонентом архівування значень безперервних процесів є буфер значень. Буфер значень призначений для проміжного зберігання масиву значень, отриманих з певною періодичністю (квантом часу). Буфер значень використовується як для безпосереднього зберігання більших масивів значень в архівах значень як перед безпосереднім «скиданням» на фізичні носії, так і для маніпуляцій з кадрами значень, тобто у функціях покадрового запиту значень і їхнього приміщення в буфери архівів.

Для організації виділених архіваторов у розподілених системах можна використовувати транспортний тип архіватора . Функцією транспортного типу архіватора є відбиття віддаленого центрального архіватора на локальній системі. Як наслідок архіватори транспортного типу виконують передачу даних між локальною системою й архіватором віддаленої системи, приховуючи від підсистем локальної системи реальну природу архіватора.

Комунікації. Підсистеми "Транспорти" і "Транспортні протоколи"

Оскільки система захищеної WebSCADA заставляється як високо-маштабуєма система, то підтримка комунікацій повинна бути досить гнучкою. Для задоволення високого ступеня гнучкості комунікації в системі захищеної

WebSCADA реалізовані в підсистемах "Транспорти" і "Транспортні протоколи", які є модульними.

Підсистема «Транспорти» призначена для обміну неструктурованими даними між системою захищеної WebSCADA і зовнішніми системами. У ролі зовнішніх систем можуть виступати й віддалені системи кібербезпеки захищеної захищеної WebSCADA системи. Під неструктурованими даними розуміється масив символів певної довжини. Модульним об'єктом, що втримується в підсистемі «Транспорти», виступає тип транспорту. Тип транспорту визначає механізм передачі неструктурованих даних. Наприклад, це можуть бути:

– сокети (TCP/UDP/UNIX);

– канали;

– поділювана пам'ять.

Підсистема "Транспорти" включає підтримку вхідних і вихідних транспортів. Вхідний транспорт призначений для обслуговування зовнішніх запитів і відправлення відповідей. Вихідний транспорт, навпаки, призначений для відправлення повідомлень і очікування відповіді. Отже, що входить транспорт містить конфігурацію даної станції як сервера, а вихідний транспорт містить конфігурацію віддаленого сервера. Модуль підсистеми "Транспорти" реалізує підтримку як вхідного, так і вихідного транспортів.

Підсистема "Транспортні протоколи" призначена для структуризації даних, отриманих від підсистеми "Транспорти". По суті, підсистема "Транспортні протоколи" є продовженням підсистеми "Транспорти" і виконує функції перевірки структури й цілісності отриманих даних. Так, для вказівки протоколу, у зв'язуванні з яким повинен працювати транспорт, передбачене спеціальне конфігураційне поле. Модульним об'єктом, що втримується в підсистемі "Протоколи", є сам протокол. Наприклад, транспортними протоколами можуть бути:

– HTTP (Hyper Text Transfer Protocol);

– SelfSystem (захищеної WebSCADA системний протокол).

Повний ланцюжок зв'язку можна записати в такий спосіб:

– повідомлення передається в транспорт;

– транспорт передає повідомлення пов'язаному з ним протоколу шляхом створення нового об'єкта протоколу;

– протокол перевіряє цілісність даних;

– якщо прийшли всі дані, то повідомити транспорт про припинення очікування даних і передати йому відповідь інакше повідомити, що потрібно очікувати ще;

– транспорт, одержавши підтвердження, відсилає відповідь і видаляє об'єкт протоколу;

– якщо підтвердження ні, то транспорт продовжує очікування даних, і у випадку їхнього надходження передає їхньому збереженому об'єкту протоколу.

Підтримуються протоколи й для вихідних транспортів. Вихідний протокол бере на себе функцію спілкування із транспортом і реалізацію особливостей протоколу. Внутрішня сторона доступу до протоколу реалізується потоковим образом із власною структурою для кожного протокольного модуля. Такий механізм дозволяє виконувати прозорий доступ до зовнішньої системи, за допомогою транспорту, просто вказуючи ім'я протоколу, за допомогою якого обслуговувати передачу.

Завдяки стандартному API-доступу до транспортів системи захищеної WebSCADA можна легко міняти спосіб обміну даними, не зачіпаючи самих систем, що обмінюються. Наприклад, у випадку локального обміну можна використовувати більше швидкий транспорт на основі поділюваної пам'яті, а у випадку обміну через інтернет і локальну мережу використовувати TCP або UDP сокети.

Інтерфейси користувача. Підсистема "Інтерфейси користувача"

SCADA-Системи, як клас, припускають наявність інтерфейсів користувача. В захищеної WebSCADA для надання користувальницьких інтерфейсів передбачена підсистема "Користувальницькі інтерфейси". Під користувальницьким інтерфейсом системи захищеної WebSCADA розуміється не

тільки середовище візуалізації, з якого повинен працювати кінцевий користувач, але й усе, що має відношення до користувача, наприклад:

–середовища візуалізації;

–конфігуратори;

–сигналізатори.

Підсистема "Користувальницькі інтерфейси" є модульною. Модульним об'єктом підсистеми виступає властиво конкретний інтерфейс користувача. Модульність підсистеми дозволяє створювати різні інтерфейси користувачів на різних GUI/TUI бібліотеках і використовувати найбільш оптимальне з рішень у конкретно взятому випадку, наприклад, для середовищ виконання програмувальних логічних контролерів можна використовувати конфігуратори й візуалізатори на основі Web-технологій (WebCfg, WebUI), а у випадку стаціонарних робочих станцій використовувати ті ж конфігуратори й візуалізатори, але на основі бібліотек типу QT, GTK.

Безпека системи. Підсистема "Безпека"

Система захищеної WebSCADA є розгалуженою системою, що складається з десятка підсистем і може включати безліч модулів. Отже, надання всім необмеженого доступу до цих ресурсів є принаймні небезпечним. Тому для розмежування доступу в системі захищеної WebSCADA передбачена підсистема "Безпеки". Основними функціями підсистеми "Безпеки" є:

– зберігання облікових записів користувачів і груп користувачів;

– автентифікація користувачів;

– перевірка прав доступу користувача до того або іншого ресурсу.

Керування бібліотеками модулів і модулями. Підсистема "Керування модулями"

Система захищеної WebSCADA побудована за модульним принципом, що має на увазі наявність безлічі модулів, якими необхідно управляти. Для виконання функції керування модулями системи захищеної WebSCADA передбачена підсистема "Керування модулями". Всі модулі на справжній момент

поставляються в систему за допомогою поділюваних бібліотек(контейнерів). Кожний контейнер може містити безліч модулів різного типу. Підсистема "Керування модулями" реалізує контроль за станом контейнерів і дозволяє виконувати гаряче додавання, видалення й відновлення контейнерів і модулів, що втримуються в них.

Непередбачені можливості. Підсистема "Спеціальні"

Зрозуміло, передбачити всіх можливих функцій неможливо, тому в системі захищеної WebSCADA передбачена підсистема "Спеціальні". Підсистема "Спеціальні" є модульною й призначена для додавання в систему захищеної WebSCADA непередбачених функцій шляхом модульного розширення. Наприклад, за допомогою підсистеми «Спеціальні» можуть бути реалізовані:

– тести системи захищеної WebSCADA і її модулів;

– бібліотеки функцій користувальницького програмування.

Користувальницькі функції. Об'єктна модель і середовище програмування системи

Будь-яка сучасна SCADA система повинна містити механізми, що надають можливість програмувати на користувальницькому рівні, тобто містити середовище програмування. Система захищеної WebSCADA містить таке середовище. За допомогою середовища програмування системи захищеної WebSCADA можна реалізовувати:

– Алгоритми керування технологічними процесами.

– Великі динамічні моделі реального часу технологічних, хімічних, фізичних і інших процесів.

– Адаптивні механізми керування по моделях.

– Користувальницькі процедури керування внутрішніми функціями системи, її підсистемами й модулями.

– Гнучке формування структур параметрів на рівні користувача, з метою створення параметрів нестандартної структури й заповнення її по алгоритму користувача.

– Допоміжні обчислення.

Середовище програмування системи захищеної WebSCADA являють собою комплекс засобів, що організують обчислювальне оточення користувача. До складу комплексу засобів входять:

– об'єктна модель системи захищеної WebSCADA;

– модулі бібліотек функцій;

– обчислювальні контролери підсистеми «Збір даних» і інші обчислювачі.

Модулі бібліотек функцій надають безліч функцій певної спрямованості, що розширюють об'єктну модель системи. Бібліотеки можуть реалізуватися як набором функцій фіксованого типу, так і функціями, що допускають вільну модифікацію й доповнення.

Бібліотеки функцій фіксованого типу можуть надаватися стандартними модулями системи, органічно доповнюючи об'єктну модель. Функції таких бібліотек будуть являти собою інтерфейс доступу до засобів модуля на рівні користувача. Наприклад, «Середовище візуального подання даних» можуть надавати функції для видачі різних повідомлень. Використовуючи ці функції, користувач може реалізовувати інтерактивні алгоритми взаємодії із системою.

Бібліотеки функцій вільного типу надають середовище написання користувальницьких функцій на одній з мов програмування. У рамках модуля бібліотек функцій можуть надаватися механізми створення бібліотек функцій. Так, можна створювати бібліотеки апаратів технологічних процесів, а в наслідку використовувати їх шляхом зв'язування. Різні модулі бібліотек функцій можуть надавати реалізації різних мов програмування.

На основі функцій, надаваних об'єктною моделлю, будуються обчислювальні контролери. Обчислювальні контролери виконують зв'язування функцій з параметрами системи й механізмом обчислення.

Загальносистемне API користувальницького програмування

API користувальницького програмування являє собою дерево об'єктів системи захищеної WebSCADA, кожний об'єкт якого може представляти власний

перелік властивостей і функцій. Властивості й функції об'єктів можуть використовуватися користувачем у процедурах на мовах користувальницького програмування захищеної WebSCADA. Крапкою входу для доступу до об'єктів системи захищеної WebSCADA з мови користувальницького програмування BuilderCLikeCalc є зарезервоване слово "SYS" кореневого об'єкта захищеної WebSCADA. Наприклад, для доступу до функції вихідного транспорту потрібно записати: SYS.Transport.Serial.out_ModBus.messIO(mess);.

API об'єктів, надаваних модулями, описується у власній документації модуля.

Загальносистемні користувальницькі 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 vall, 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 <vall> 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 SYS.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 SYS.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 захищеної WebSCADA 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 messlnfo(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 = SYS.XMLNode("get").setAttr("path","/%2fgen%2fid");
SYS.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 charl, 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 захищеної WebSCADA (SYS.*)

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 захищеної WebSCADA. 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 захищеної WebSCADA. Один символ роздільника вказується в <sep> для одержання шляху через роздільник, наприклад, "DAQ.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" (SYS.Security)

Функції об'єкта підсистеми (SYS.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 (SYS.Security["usr_User"]) і групи (SYS.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" (SYS.BD)

Функції об'єкта БД (SYS.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=SYS.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";  SYS.messDebug("TEST  DB","Row  "+i_rw+":
"+rec);
if(i_rw)   SYS.messDebug("TEST DB:   ","Row "+i_rw+":   'NAME'"+DBTbl[i_rw]
["NAME"]);
}
```

Функції об'єкта Таблиці (SYS.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 = SYS.XMLNode("field");
req.childAdd("user").setAttr("type","str").setAttr("key","1").setText("root
");
req.childAdd("id").setAttr("type","str").setAttr("key","1").setText("/Lang2
CodeBase");
req.childAdd("val").setAttr("type","str");
SYS.BD.MySQL.GenDB.SYS.fieldGet(req);
SYS.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" (SYS.DAQ)

Функції об'єкта підсистеми (SYS.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;
SYS.DAQ.funcCall("BuilderCLikeCalc.BuilderCScript",args,"y=2*x;");
SYS.messDebug("TEST Calc","TEST Calc rezult:   "+args.y);
```

Функції об'єкта контролера (SYS.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 (SYS.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 (SYS.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 (SYS.DAQ[tmplb_Lib"]) і шаблона (SYS.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>.

Модуль DAQ.BuilderCLikeCalc

Об'єкт "Бібліотека функцій"

(SYS.DAQ.BuilderCLikeCalc["lib_Lfunc"])

ElTp funcID}(ElTp prml, ...) – виклик функції бібліотеки funcID}. Повертає результат викликуваної функції.

Об'єкт "Користувальницька функція"

(SYS.DAQ.BuilderCLikeCalc["lib_Lfunc"]["func"])

ElTp call(ElTpprml, ...) – виклик даної функції з параметрами <prm{N}>. Повертає результат викликуваної функції.

Модуль DAQ.ModBus

Об'єкт "Контролер" (this.nodePrev())

string messIO(string pdu) – відправлення PDU <pdu> через транспорт об'єкта контролера за допомогою ModBus протоколу. PDU результату запиту міститься замість запиту в <pdu> , а помилка вертається в результаті функції.

Підсистема "Архіви" (SYS.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 (SYS.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 (SYS.Archive["va_Archive"]) і архіватора значень (SYS.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" (SYS.Transport)

Функції об'єкта вхідного транспорту (SYS.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 (SYS.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=SYS.Transport.Serial.out_ttyUSB0.messIO(SYS.strFromCharCode(0x4B,0x0 0,
0x37,0x40),0.2);
while(true)
{
trez = SYS.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 = SYS.XMLNode("TCP");
req.setAttr("id","test").setAttr("reqTm",1).setAttr("node",1).setAttr("req
Try",2).setText(SYS.strFromCharCode(0x03,0x0 0,0x0 0,0x0 0,0x05));
 SYS.Transport.Sockets.out_testModBus.messIO(req,"ModBus");
test = Special.FLibSYS.strDec4Bin(req.text());
```

Підсистема "Користувальницькі інтерфейси" (SYS.UI)

Модуль UI.VCAEngine

Об'єкт "Сеанс" (this.ownerSess())

– string user( ) – поточний користувач сеансу.

– string alrmSndPlay( ) – шлях віджета для якого на даний момент відтворюється повідомлення про порушення.

– int alrmQuittance(int quittmpl, string wpath = "") – квітирування порушень <wpath> із шаблоном <quit_tmpl>. Якщо <wpath> порожній рядок то виробляється глобальне квітирування.

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)7              ~
```

– 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);
```

Об'єкт "Віджет", примітива "Документ" (this)

string getArhDoc(integer nDoc) – одержати текст документа архіву на глибині <nDoc> (0-0-{aSize-1}).

Підсистема "Спеціальні" (SYS.Special)

Модуль Special.FLibSYS

Об'єкт "Бібліотека функцій" (SYS.Special.FLibSYS)

ElTp {funcID}(ElTp prml, ...) – виклик функції бібліотеки {funcID}. Повертає результат викликуваної функції.

Об'єкт "Користувальницька функція"

(SYS.Special.FLibSYS["funcID"])

ElTp call(ElTp prml, ...) – виклик даної функції з параметрами <prm{N}>. Повертає результат викликуваної функції.

Модуль Special.FLibMath

Об'єкт "Бібліотека функцій" (SYS.Special.FLibMath)

ElTp {funcID}(ElTp prml, ...) – виклик функції бібліотеки {funcID}. Повертає результат викликуваної функції.

Об'єкт "Користувальницька функція"

(SYS.Special.FLibMath["funcID"])

ElTp call(ElTp prml, ...) – виклик даної функції з параметрами <prm{N}>. Повертає результат викликуваної функції.

Модуль Special.FLibComplex1

Об'єкт "Бібліотека функцій" (SYS.Special.FLibComplex1)

ElTp {funcID}(ElTp prml, ...) – виклик функції бібліотеки {funcID}. Повертає результат викликуваної функції.

Об'єкт "Користувальницька функція"

(SYS.Special.FLibComplex1["funcID"])

ElTp call(ElTp prml, ...) – виклик даної функції з параметрами <prm{N}>. Повертає результат викликуваної функції.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою CRYPTON – алгоритм симетричного блочного шифрування (розмір блоку 128 біт, ключ довжиною до 256 біт), розроблений південнокорейським криптологом Чьо Лім Хун з південнокорейської компанії Future Systems, яка з кінця 1980-х років працює на ринку забезпечення мереж і захисту інформації. Алгоритм був розроблений в 1998 році в якості шифру – учасника конкурсу AES. Як зізнавався автор, конструкція алгоритму спирається на алгоритм SQUARE[1]. В алгоритмі Crypton немає традиційних для блочних шифрів мережі Фейстеля. Основу даного шифру становить так звана SP-мережа (повторювана циклова функція, що складається із замін-перестановок, орієнтована на розпаралелену нелінійну обробку всього блоку даних). Крім високої швидкості, перевагами таких алгоритмів є полегшення дослідження стійкості шифру до методів диференціального та лінійного криптоаналізу, що є на сьогодні основними інструментами розтину блочних шифрів. На конкурс AES була представлена версія алгоритму Crypton v0.5. Однак, як казав Чьо Лім Хун, йому не вистачало часу для розробки повної версії. І вже на першому етапі конкурсу AES в ході

аналізу алгоритмів, версія Crypton v0.5 була замінена на версію Crypton v1.0. Відмінність нової версії від первинної полягала в зміні таблиці замін та в модифікації процесу розширення ключа.

Як і інші учасники конкурсу AES, Crypton призначений для шифрування 128-бітових блоків даних[2]. При шифруванні використовуються ключі шифрування для декількох фіксованих розмірів – від 0 до 256 біт з кратністю 8 бітів. Структура алгоритму Crypton – структура «Квадрата» – багато в чому схожа на структуру алгоритму Square, створеного в 1997 році. Криптографічні перетворення для алгоритмів з даною структурою можуть бути виконані як для цілих рядків і стовпців масиву, так і над окремими його байтами. (Варто зазначити, що алгоритм Square був розроблений авторами майбутнього переможця конкурсу AES – авторами алгоритму Rijndael – Вінсентом Ріджменом і Джоан Дейменом.)

Шифрування

Алгоритм Crypton являє 128-бітовий блок шифруємих даних у вигляді байтового масиву 4×4, над якими в процесі шифрування проводиться кілька раундів перетворень. У кожному раунді передбачається послідовне виконання наступних операцій:

– Таблична заміна γ;

– Лінійне перетворення π;

– Байтова перестановка τ;

– Операція σ.

Таблична заміна γ

Алгоритм Crypton використовує 4 таблиці замін. Кожна з яких заміщає 8-бітне вхідне значення на вихідне такого ж розміру.

Лінійне перетворення π

Тут використовується 4 спеціальні константи. Ці константи об'єднані в маскуючі послідовності

Байтова перестановка τ

Дана перестановка перетворює найпростішим чином рядок даних у стовпець.

Операція σ

Дана операція є побітовим складанням всього масиву даних з ключем раунду. Зауважимо, саме 12 раундів шифрування рекомендується автором алгоритму Чьо Хун Лімом, проте сувора кількість раундів не встановлена.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображене головне вікно програми. З нього видно, що інтерфейс користувача складається з наступних блоків:

– Блок меню.

– Блок клавіш швидкого доступу.

– Блок відображення елементів захищеної WebSCADA.

– Блок контролю та параметрів.

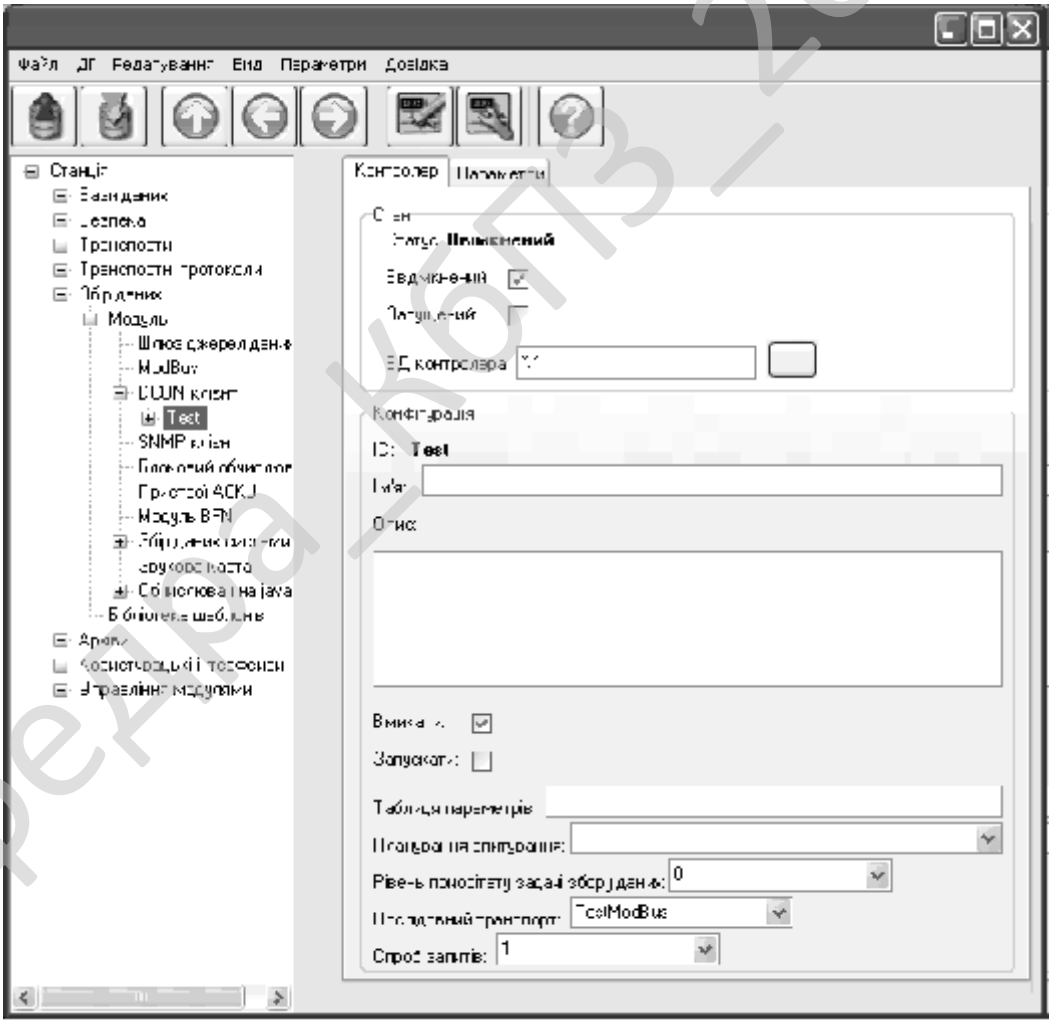

Рисунок 5.1 – Головне вікно програми

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.

Блок відображення елементів захищеної WebSCADA складається з наступних елементів:

– Робоча станція.

– Бази даних.

– Безпека.

– Транспорти.

– Транспортні протоколи.

– Збір даних.

– Архіви.

– Користувальницькі інтерфейси.

– Управління модулем.

Блок контролю та параметрів складається з наступних елементів:

– Статус.

– БД контролера.

– Конфігурація.

– Ідентифікатор процесу та пристрою.

– Таблиця параметрів.

– Планування опитування.

– Рівень пріоритету задачі збору даних.

– Послідовний транспорт.

– Кількість спроб запитів.

На рисунку 5.2 зображена довідка, з якої можливо отримати інформацію про автора, керівника, тему та місце виконання дипломного проекту.

Рисунок 5.2 – Довідка

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки захищеної WebSCADA системи.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем захищеної WebSCADA системи.

– Досліджена система захищеної WebSCADA системи.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки захищеної WebSCADA системи.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання захищеної WebSCADA системи.

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 призначені для системи кібербезпеки захищеної WebSCADA системи. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне

програмне забезпечення, що спеціально розроблене для даної конкретної системи кібербезпеки й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм CRYPTON.

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

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Смірнова Т.В., Дрєєв О.М., «Інформаційна технологія оптимізації технологічного процесу відновлення та зміцнювання поверхонь валів зі сталі як хмарний сервіс» у *Інформаційна безпека та інформаційні технології : монографія / за заг. ред. В. С. Пономаренка*. – Х. : Вид. Рожко С.Г. 2019. С. 92-107.

2. Смірнова Т.В., Поліщук Л.І., «Дослідження хмарних технологій як сервісів для системи інженерних розрахунків» у *Кібербезпека та інформаційні технології: монографія.* – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. С. 106-121.

3. Smirnova, T., Kuznetsov, A., Oleshko, I., Chernov, K., Bagmut, M., «Biometric authentication using convolutional neural networks». *Lecture Notes in Networks and Systems,* 2021, vol. 152, pp. 85–98. (Scopus). Режим доступу: https://www.scopus.com/record/display.uri?eid=2-s2.0-85090914783&origin=resultslist

4. 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. (Scopus). Режим доступу: https://www.scopus.com/record/display.uri?eid=2-s2.0-85118101973&origin=AuthorNamesList&txGid=9fba77a9424db54ff3b099e4400c22bb

5. 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. (Scopus). Режим доступу: https://www.scopus.com/record/display.uri?eid=2-s2.0-85133613188&origin=resultslist&sort=plf-

f&featureToggles=FEATURE_NEW_DOC_DETAILS_EXPORT:1

6. Smirnova T., Anas M. Al-Oraiqat, Drieiev O., Smirnov O., Polishchuk L., Sheroz Khan, Yassin M. Y. Hasan, Aladdein M. Amro, Hazim S. AlRawashdeh «Method for Determining Treated Metal Surface Quality Using Computer Vision Technology». *Sensors (Basel, Switzerland)* Volume 22, Issue 16, 6223, 2022. (Scopus). Режим доступу: https://www.scopus.com/record/display.uri?eid=2-s2.0-85137126823&origin=SingleRecordEmailAlert&dgcid=raven_sc_author_ru_ru_email&txGid=ee77c55f56deb4fdf2c6ce0e37730eb9

7. Смірнова Т.В., Дрєєв О.М., Смірнов О.А., «Експертна система оптимізації процесу відновлення та зміцнення поверхонь деталей типу «вал» електродуговим напиленням», *Системи управління, навігації та зв'язку*, № 2 (54). с. 149-154, 2019. Режим доступу: http://nbuv.gov.ua/UJRN/suntz_2019_2_32

8. 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. Режим доступу: http://nbuv.gov.ua/j-pdf/znpkntu_2019_1_23.pdf

9. Смірнова Т.В., Дрєєв О.М., Смірнов О.А., Солових Є.К., «Методи оптимізації технологічних процесів відновлення сталевих покриттів», *Shipbuilding & marine infrastructure / Суднобудування і морська інфраструктура* № 1 (11). с. 48-57, 2019. Режим доступу: http://smi.nuos.mk.ua/archive/2019/1/6.pdf

10. Smirnova T., Ageev M., Lopata L., Dudan A., Lopata A., «Of combined electric arc coatings», *Problems of Tribologyy*, Vol. 24 № 3/93. P. 51-61, 2019. Режим доступу: http://tribology.khnu.km.ua/index.php/ProbTrib/article/view/727/1207

11. Смирнова Т.В., Смирнов А.А., Дреев А.Н., Дудан А.В., «Оптимизация технологического процесса восстановления и упрочнения поверхностей с заданными характеристиками в виде облачного сервиса», *Вестник Полоцкого государственного университета. Серия В, Промышленность. Прикладные науки.*

Республика Беларусь - 2020. - № 3. - С. 50-61. Режим доступу: http://elib.psu.by:8080/handle/123456789/24988

12. Смірнова Т.В., Поліщук Л.І., Смірнов О.А., Буравченко К.О., Макевнін А.О., «Дослідження хмарних технологій як сервісів», *Кібербезпека: освіта, наука, техніка*. № 3(7). С. 43-62. 2020. Режим доступу: https://www.csecurity.kubg.edu.ua/index.php/journal/article/view/132/132 (Категорія Б)

13. Смірнова Т.В., «Формалізація та реалізація структури технологічного процесу електродугового напилення для оптимізаційної експертної системи», *Технічні науки та технології*. № 1(19). С. 104-113. 2020. Режим доступу: http://tst.stu.cn.ua/article/view/204062 (Категорія Б)

14. Smirnova T., Smirnov I., Lopata A., Lopata L., «Improvement of functional properties of gas-thermal coatings by electro-contact treatment», *Problems of Tribology*, Vol. 25, № 1/95, P. 41-48. 2020. Режим доступу: http://tribology.khnu.km.ua/index.php/ProbTrib/article/view/742/1222

15. Смірнова Т.В. «Формування евристичних правил, бази знань та формалізація структури й правил технологічного процесу для оптимізаційної хмарної інформаційної системи», *Системи управління, навігації та зв'язку*, № 2 (60). с. 101-104, 2020. Режим доступу: http://journals.nupp.edu.ua/sunz/article/view/1839/1515 (Категорія Б)

16. Смірнова, Т.В., Смірнов, С.А., Минайленко, Р.М., Доренський, О.П., Сисоєнко С.В., «Хмарна автоматизована система інтелектуальної підтримки прийняття рішень для технологічних процесів». *Вісник Черкаського державного технологічного університету. Технічні науки.* №4, 2020, С. 84-92. Режим доступу: https://er.chdtu.edu.ua/handle/ChSTU/1888 (Фахове видання. Категорія «Б»)

17. Смирнова Т.В., Дудан А.В., Смирнов А.А., «Формализация структуры технологического процесса электродугового напыления». *Сборник научных трудов «Актуальные вопросы машиноведения».* Объединенный институт

машиностроения Национальной Академии Наук Беларуси. №9. С. 308-312, 2020. Режим доступу: http://oim.by/ru/izdaniya/sbornik.html

18. Смірнова Т.В., Пархоменко Д.О., Голубець Р.О., Щербань А.В., Багдасарян Е.К., «Формалізація проблеми підтримки технологічних процесів у хмарних сервісах». *Системи озброєння і військова техніка.* 2021. № 3(67). С. 105-112. https://doi.org/10.30748/soivt.2021.67.14 (Фахове видання. Категорія «Б»)

19. Смірнова Т.В., Столяренко М.П., Янков М.О., Грудік В.В., Моторін Ю.Ю., «Модель реалізації структури технологічного процесу у хмарному сервісі». *Збірник наукових праць Харківського національного університету Повітряних Сил.* 2021. № 4(70). С. 132-142. https://doi.org/10.30748/zhups.2021.70.19. (Фахове видання. Категорія «Б»)

20. Смірнова Т.В., Буравченко К.О., Кравченко С.С., Горбов В.О., Смірнов О.А., «Хмарна система підтримки прийняття рішень технологічного процесу відновлення поверхонь конструкцій і деталей машин». *Сучасні інформаційні системи*. 2021. Т. 5, № 4. С. 79-95 Режим доступу: http://ais.khpi.edu.ua/article/view/247293 (Фахове видання. Категорія «Б»)

21. Смірнова Т.В., Гнатюк С.О., Бердибаєв Р.Ш., Бурмак Ю.А., Оспанова Д.М., «Удосконалений модуль криптографічного захисту інформації в сучасних інформаційно-комунікаційних системах та мережах». *Кібербезпека: освіта, наука, техніка*. 2021. № 2(14). С. 176-185. Режим доступу: https://csecurity.kubg.edu.ua/index.php/journal/article/view/329 (Фахове видання. Категорія «Б»)

22. Смірнова Т.В., Бурмак Ю.А., Улічев О.С., Усік П.С., Доренський О.П., «Стійка функція шифрування удосконаленого модуля криптографічного захисту інформації в інформаційно-комунікаційних системах» *Кібербезпека: освіта, наука, техніка*. 2021. № 1(13). С. 183-201. Режим доступу: https://csecurity.kubg.edu.ua/index.php/journal/article/view/346 (Фахове видання. Категорія «Б»)

23. Смірнова Т.В., Якименко Н.М., Улічев О.С., Конопліцька-Слободенюк О.К., Смірнов С.А., «Дослідження лінійних перетворень запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Кібербезпека: освіта, наука, техніка*. 2022. № 3(15). С. 85-92. Режим доступу: https://csecurity.kubg.edu.ua/index.php/journal/article/view/337 (Фахове видання. Категорія «Б»)

24. Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., Смірнов О.А., «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. Режим доступу: http://journals.nupp.edu.ua/sunz/article/view/2615 (Фахове видання. Категорія «Б»)

25. Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., Смірнов О.А. «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. Режим доступу: http://journals.nupp.edu.ua/sunz/article/view/2449/1918 (Фахове видання. Категорія «Б»)

26. Смірнова Т.В., Якименко Н.М., Смірнов О.А., 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»,* 2022. № 2 (307). С. 46-52. Режим доступу: http://journals.khnu.km.ua/vestnik/?cat=65 (Фахове видання. Категорія «Б»)

27. Смірнова Т.В., Моторін Ю.Ю., Буравченко К.О., Бочуля Т.В., Коваленко О.В. «Вибір оптимальної технології побудови хмарної інформаційно-

комунікаційної системи автоматизації виробничих процесів». *Вимірювальна та обчислювальна техніка в технологічних процесах,* № 1 (2022). С. 15-26. 2022. Режим доступу: http://vottp.khmnu.edu.ua/index.php/vottp/article/view/30/36 (Фахове видання. Категорія «Б»)

28. Смірнова Т.В., Янков М.О., Грудік В.В., Горбов В.О., Коваленко А.С. «Планування радіопокриття та моделювання поширення радіосигналів мобільних мереж 5G для автоматизації виробничих процесів». *Електронне моделювання*, № 3, т. 44. С. 113-122. 2022. Режим доступу: https://www.emodel.org.ua/uk/archive-ukr/2022/44-3-u/c-113-122 (Фахове видання. Категорія «Б»)

29. Смірнова Т.В., Буравченко К.О., Щербань А.В., Багдасарян Е.К., Коваленко А.С. «Проектування та оптимізація структурованих кабельних систем для автоматизації виробничих процесів підприємства» *Сучасні інформаційні системи*. 2022. Т. 6, № 1. С. 129-133. Режим доступу: http://ais.khpi.edu.ua/article/view/254256/251522 (Фахове видання. Категорія «Б»)

30. Смірнова Т.В. «Метод забезпечення надійності підключення вузлів до інформаційно-комунікаційної системи підприємства на базі 5G» *Сучасні інформаційні системи*. 2022. Т. 6, № 2. С. 82-87. Режим доступу: http://ais.khpi.edu.ua/article/view/260767/257189 (Фахове видання. Категорія «Б»)

31. Смірнова Т.В., Верховець О.С., Буравченко К.О., Смірнов С.А., Гермак В.С. «Алгоритмічне забезпечення для планування інформаційно-комунікаційної мережі підприємства на базі технологій 5G». *Збірник наукових праць Національного університету кораблебудування імені адмірала Макарова*. № 1 (488). 2022. С. 81-88. Режим доступу: http://znp.nuos.mk.ua/archives/2022/1/11.pdf (Фахове видання. Категорія «Б»).

32. Смірнова Т.В., Гнатюк С.О., Юдін О.Ю., Сидоренко В.М., Жаксигулова Д.Д., «Експериментальне дослідження моделі розрахунку кількісного критерію оцінювання захищеності інформаційно-телекомунікаційних систем критичної інфраструктури держави» *Кібербезпека: освіта, наука, техніка*. № 4(16). 2022. С. 6-18. Режим доступу:

https://csecurity.kubg.edu.ua/index.php/journal/article/view/359/298 (Фахове видання. Категорія «Б»)

33. Смірнова Т.В., Гнатюк С.О., Сидоренко В.М., Юдін О.Ю., «Метод розрахунку критичності галузевих інформаційно-телекомунікаційних систем». *Наукоємні технології* № 2(54), 2022. С. 94-104. Режим доступу: https://jrnl.nau.edu.ua/index.php/SBT/article/view/16757 (Фахове видання. Категорія «Б»)

34. 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. Режим доступу: https://jrnl.nau.edu.ua/index.php/PIU/article/view/16844 (Фахове видання. Категорія «Б»)

35. Смірнова Т.В., Охріменко Т.О., Бредніков А.В., Макаренко О.І., «Обґрунтування необхідності впровадження інформаційних систем управління підприємствами нафтопереробної промисловості». *Інформаційно-аналітичні системи обробки даних,* Том 23, № 4, 2021. С. 17-27. Режим доступу: http://drsp.ipri.kiev.ua/article/view/265648 (Фахове видання. Категорія «Б»)

36. Смірнова Т.В., Смірнов О.А., Дрєєв О.М., С.А. Смірнов, «Використання хмарних експертних систем в сфері інформаційного забезпечення обробки поверхні деталей», *Комп'ютерна інженерія і кібербезпека: досягнення та інновації*, м. Кропивницький. 27-29 листопада, 2018, с. 111-113

37. Смірнова Т.В., Дрєєв О.М., Смірнов О.А. «Захист даних у інформаційній технології відновлення поверхонь деталей у вигляді хмарної платформи як послуги», *Проблеми кібербезпеки інформаційно-телекомунікаційних систем*, м. Київ, 11-12 квітня, 2019, с. 221-224.

38. Смірнова Т.В., Дрєєв О.М., Солових Є.К., «Хмарний сервіс інформаційної технології оптимізації технологічного процесу відновлення та зміцнювання поверхонь зі сталі», *Інформаційна безпека та інформаційні*

технології, Information Security and Information Technologies, м. Харків, 24-25 квітня, 2019, с. 36

39. Смірнова Т.В., Дрєєв О.М., Смірнов О.А., «Побудова хмарної експертної системи оптимізації технологічного процесу електродугового напилення сталевих покриттів», *Міжнародний форум з інформаційних систем і технологій INFOS-2019,* м. Харків, 24-27 квітня, 2019, с. 95-98.

40. Смирнова Т.В., Смирнов А.А., Соловых Е.К. «Разработка математической модели и программного обеспечения для оптимизации режима электроконтактной обработки газотермических покрытий», *Комплексне забезпечення якості технологічних процесів та систем,* том 2, м. Чернігів, 14 - 16 травня, 2019, с. 78-79.

41. Смірнова Т.В., Дрєєв О.М., Солових Є.К., Смірнов О.А., «Експертна система інформаційної технології оптимізації абстрактного технологічного процесу як хмарного сервісу», *Інформаційні технології: Наука, техніка, технологія, освіта, здоров'я. MicroCAD-2019,* м. Харків, 15-17 травня, 2019, с. 195.

42. Смірнова Т.В., Дрєєв О.М., Смірнов О.А., «Формування абстрактних експертних систем на основі досліджень відомих експертних систем», *XXI Міжнародний науково-практичний семінар «Комбінаторні конфігурації та їх застосування»*, м. Кропивницький, 17-18 травня, 2019, с. 143-147.

43. Смірнова Т.В., Смірнов О.А., «Захист даних у інформаційній технології відновлення поверхонь деталей у вигляді хмарної платформи як послуги», *Інформація, комунікація, суспільство – 2019*, см.т. Чинадієво, 16-18 травня, 2019, с. 25-26

44. Смирнова Т.В., Дреев А.Н., Смирнов А.А., «Информационная технология оптимизации технологического процесса восстановления и упрочнения поверхностей в виде облачного сервиса», *Modern Information, Measurement And Control Systems: Problems And Perspectives (MIMCS 2019)*, Baku, Azerbaijan, 01-02 July, 2019, p. 282.

45. Смірнова Т.В., Дрєєв О.М., Смірнов О.А., «Огляд відомих експертних систем оптимізації технологічних процесів», *Стратегія якості в промисловості і освіті,*м. Варна, Болгарія, 3-6 червня, 2019, с.442-444

46. Смірнова Т.В., Дрєєв О.М., Смірнов О.А., «Формалізація та узагальнення інформаційної моделі технологічних операцій зміцнення та відновлення сталевих поверхонь», *Інтелектуальні системи та інформаційні технології*, м.Одеса, 19-24 серпня, 2019, с. 211-213.

47. Смірнова Т.В., Дрєєв О.М., Смірнов О.А., «Формування інформаційної моделі технологічного процесу». *V Всеукраїнська науково-практична Інтернет-конференція «Електронні та мехатронні системи: теорія, інновації, практика»*, м. Полтава, 8 листопада, 2019, с. 87-91.

48. Smirnova T.V., Chernovol M.S., Ageev M. «Study of the spraying process and the influence of its factors on the properties of electric arc spraying coatings». *Materials of the 20th international scientific and technical seminar «Modern questions of production and repair in industry and in transport»*, *Tbilisi, Georgia*, 23-28 *March* 2020, с. 201-205.

49. Смірнова Т.В., Солових Є.К., Смірнов О.А., «Дослідження стандартів забезпечення кібербезпеки хмарних технологій як сервісів», *X міжнародна науково-технічна конференція «ITSec: Безпека інформаційних технологій»*, 19-24 березня 2020 р. – К.: НАУ, 2020. с. 28-29. Режим доступу: http://bit.nau.edu.ua/wp-content/uploads/2020/05/ITSec-2020_Zbirnyk.pdf

50. Смірнова Т.В., Поліщук Л.І., Смірнов О.А., «Аналіз хмарних технологій як сервісів», *XIII всеукраїнська науково-практична конференція «Комп'ютерні інтелектуальні системи та мережі (КІСМ-2020)»*. м. Кривий Ріг. 24-26 березня, 2020, С. 210-212.

51. Смірнова Т.В., Поліщук Л.І., Дрєєв О.М., «Застосування сервісу CAEaaS як системи інженерних розрахунків з використанням хмарних технологій», *II Міжнародна науково-практична конференція «Інформаційна*

безпека та інформаційні технології, Information Security and Information Technologies», м. Кропивницький, 2-3 квітня, 2020, с. 52.

52. Смірнова Т.В., Дрєєв О.М., Смірнов О.А., Солових Є.К., «Інформаційна структура технологічного процесу електродугового напилення». *Всеукраїнська науково-технічна конференція «Стан, досягнення та перспективи інформаційних систем і технологій»*, м. Одеса, 21-22 квітня, 2020, с. 184-186.

53. Смірнова Т.В., Дрєєв О.М., Смірнов О.А., Солових Є.К., «Реалізація інформаційної структури технологічного процесу електродугового напилення», *XXII Міжнародний науково-практичний семінар «Комбінаторні конфігурації та їх застосування»*, м. Кропивницький, 15-16 травня, 2020, с. 158-162.

54. Смірнова Т.В., Щербань А.В., Моторін Ю.Ю., Смірнов О.А., «Перспективні напрямки забезпечення кібербезпеки сервісу CAEaaS», *VI Всеукраїнська науково-практична конференція «Перспективні напрями захисту інформації»*, м. Одеса, 2-6 вересня 2020, с. 58-59

55. Смірнова Т.В., Дрєєв О.М., Смірнов О.А. «Хмарна інформаційна система оцінювання шорсткості з використанням дискретного частотного аналізу макрофотографій». *IV міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології",* м. Кропивницький. 15-16 квітня 2021р. – Кропивницький: ЦНТУ. – 2021. – С. 30. Режим доступу: http://www.kntu.kr.ua/doc/science/zahody/vikl/2021/1-tez.pdf

56. Смірнова Т.В., Смірнов О.А., Буравченко К.О. «Системи підтримки технологічних процесів з використанням хмарних технологій». *IV Міжнародна науково-практична конференція «Інтеграція інформаційних систем і інтелектуальних технологій в умовах трансформації інформаційного суспільства»,* м. Полтава, 21-22 жовтня 2021 р., – Полтава: ПДАУ. – 2021. – С. 56-60. Режим доступу: https://doi.org/10.32782/978-966-289-562-9

57. Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Стійка функція шифрування удосконаленого модуля криптографічного захисту інформації». *І міжнародна науково-практична конференція «Цифрова трансформація*

суспільства (Digital Society – 2022)», м. Кропивницький. 21-22 квітня 2022 р. – Кропивницький: ЦНТУ. – 2022. – С. 52-55. Режим доступу: http://dspace.kntu.kr.ua/jspui/bitstream/123456789/11933/1/%d0%a1onference_abstract_book_Digital_Society_2022.pdf

58. Смірнова Т.В., Смірнов С.А., Смірнов О.А. «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування». *Дванадцята міжнародна науково-практична конференція «Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління»,* м. Баку – Харків – Жиліна. 27-28 квітня 2022 року. – Харків: ХПІ. – 2022. – С. 130. Режим доступу: https://nure.ua/wp-content/uploads/conf-2022-akov/telecom_2022_volume_1.pdf

59. Смірнова Т.В., Смірнов С.А., Смірнов О.А. «Оптимальна технологія побудови хмарної інформаційно-комунікаційної системи автоматизації виробничих процесів». *Міжнародна науково-практична конференція «Комбінаторні конфігурації та їх застосування»,* м. Кропивницький - Запоріжжя. 13-14 травня 2022р. – Кропивницький: ЦНТУ. – 2022. – С. 30. Режим доступу:

60. Смірнова Т.В., Смірнов С.А., Смірнов О.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування». *V міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології",* м. Кропивницький. 19-20 травня 2022р. – Кропивницький: ЦНТУ. – 2022. – С. 26-27. Режим доступу: http://dspace.kntu.kr.ua/jspui/bitstream/123456789/11994/3/%d0%97%d0%b1%d1%96%d1%80%d0%bd%d0%b8%d0%ba_%d1%82%d0%b5%d0%b7%202022.pdf

61. Смірнова Т.В., Одарченко Р.С., Смірнов О.А. «Проектування приватних мереж 5G для потреб виробничих підприємств». *VI Міжнародна науково-практична конференція «Інформаційні технології в освіті, науці і техніці» (ІТОНТ-2022*), м. Черкаси, 23-25 червня 2022 р. – Черкаси: ЧДТУ, 2022. – С.65-67.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки захищеної WebSCADA системи.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 13-02 від 5.01.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи кібербезпеки захищеної WebSCADA системи.

4 Джерела розробки

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

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

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

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи кібербезпеки захищеної WebSCADA системи;

– 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 Перелік документів, що розробляються

– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Пояснювальна записка – 91 аркуш.

8 Етапи розробки

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

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

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2023 р.

11.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 12.06.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Смірнова Т.В.

Програмне забезпечення системи кібербезпеки захищеної WebSCADA системи

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 68

Літера: РП

Кропивницький – 2023 року

main.cpp – головна програма

```
//WebSCADA 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("WebSCADA система коректно працює з даними %d.\n",rez);

    return rez;
}
```

tsys.cpp – встановлення та запуск системи

```
//WebSCADA 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() != "WebSCADA")
    {
      if(!create) return NULL;
      t_node->setName("WebSCADA");
    }

    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() == "WebSCADA")
          {
            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(),_("WebSCADA 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 WebSCADA-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",_("Поток"),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);
}
```

resalloc.cpp – Розподіл ресурсів у системі

```
//WebSCADA 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

```
//WebSCADA 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

```
//WebSCADA 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

```
//WebSCADA 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);
}
```