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

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

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи моніторингу стану SSD диску на основі технології Life Left”

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

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

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _12 _____*"Інформаційні технології"*_____

Спеціальність 123 *"Комп'ютерна інженерія"*

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

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

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

Ергашеву Саідмурод Нозімжон угли
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи моніторингу стану SSD диску на основі технології Life Left*

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

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

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

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи моніторингу стану SSD диску на основі технології Life Left*

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

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

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

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

6. Консультанти розділів роботи

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

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

КАЛЕНДАРНИЙ ПЛАН

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

Дата видачі завдання
« 6 » вересня 2023 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2023 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Ергашев С.Н.у. Дослідження та програмна реалізація системи моніторингу стану SSD диску на основі технології Life Left. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи моніторингу стану SSD диску на основі технології Life Left.

Метою розробки є дослідження та програмна реалізація системи моніторингу стану SSD диску на основі технології Life Left.

Об'єктом дослідження є процес моніторингу стану SSD диску на основі технології Life Left.

Предметом дослідження є методи моніторингу стану SSD диску на основі технології Life Left.

Методи дослідження базуються на методах архітектури комп'ютерів, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи моніторингу стану SSD диску на основі технології Life Left.

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 Visual C++.

Ключові слова: комп'ютерна інженерія, SSD диск, Life Left

ABSTRACT

Erhashev S.N.u. Research and software implementation of the SSD disk status monitoring system based on Life Left technology. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this graduation thesis for the second (master's) level of higher education, software is developed, which is intended for the monitoring system of the state of the SSD disk based on the Life Left technology.

The goal of the development is the research and software implementation of the SSD disk status monitoring system based on Life Left technology.

The object of the research is the process of monitoring the state of the SSD disk based on the Life Left technology.

The subject of the study is the methods of monitoring the state of the SSD disk based on the Life Left technology.

Research methods are based on computer architecture methods, mathematical statistics methods, and software development methods.

The result of the work is the software implementation of the SSD disk status monitoring system based on Life Left technology.

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 Visual C++ environment.

Keywords: computer engineering, SSD disk, Life Left

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

API	–	прикладний програмний інтерфейс
HDD	–	жорсткий диск
LBA	–	адресація секторів
MFC	–	Microsoft Foundation Class library
RO	–	тільки читання
SMART	–	Self-Monitoring, Analysis and Reporting Technology
БМГ	–	блок магнітних головок
ПЗ	–	програмне забезпечення

ВСТУП

Актуальність теми. Розвиток технологій дозволяє змінити думку про ненадійність SSD-дисків. Звичайно, їхній технологічний ресурс може бути трохи нижчий, ніж в SAS або SATA. Але їхній незаперечний плюс – блискавична швидкість роботи – перекриває не тільки це, але і їхні невеликі обсяги й високу вартість.

SSD-диски дуже добре показують себе в умовах, коли швидкість роботи додатків більше важлива, ніж дисковий простір. Наприклад, інтернет-магазини або великі інтернет-портали, які виросли з ресурсів віртуального хостингу й віртуальних серверів. Різні ігрові сервера також не вимогливі до обсягу дискового простору, але вкрай залежні від продуктивності.

А під резервні копії можна з успіхом використовувати зовнішній диск для резервних копій.

У жорстких дисків є певний ресурс, при виробітку якого він виходить із ладу й стає непрацездатний.

Якщо у випадку з SATA-диском проблем з діагностикою не виникає, то в SSD – ще досить молодої технології, виникають проблеми й невідповідність інформації. Не всі виробники дотримуються якихось загальноприйнятих стандартів, тому параметри S.M.A.R.T. можуть відрізнятися не тільки в показниках, але й у наявності самих параметрів.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи моніторингу стану SSD диску на основі технології Life Left.

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

– Огляд існуючих систем моніторингу стану SSD диску на основі технології Life Left.

– Дослідження системи моніторингу стану SSD диску на основі технології Life Left.

– Програмна реалізація системи моніторингу стану SSD диску на основі технології Life Left.

Об'єктом дослідження є процес моніторингу стану SSD диску на основі технології Life Left.

Предметом дослідження є методи моніторингу стану SSD диску на основі технології Life Left.

Методи дослідження базуються на методах архітектури комп'ютерів, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод моніторингу стану SSD диску на основі технології Life Left.

– Розроблено вітчизняний продукт моніторингу стану SSD диску на основі технології Life Left, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі моніторингу стану SSD диску на основі технології Life Left.

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

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

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи моніторингу стану SSD диску на основі технології Life Left, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Основним показником здоров'я диску можна вважати параметр Media_Wearout_Indicator, що відображає зношування диску. Його значення можуть змінюватися від 100 і до 0. Також існує якесь граничне значення (в основному в районі 20-10) при досягненні якого диск переходить у режим «тільки для читання» і його краще замінити.

Але як бути, якщо цей параметр відсутній при запиті S.M.A.R.T.?

Крім основного параметра, є не менш важливі:

– Wear_Leveling_Count – цей параметр аналогічний Media_Wearout_Indicator.

– Erase Fail Count – параметр, що вказує кількість невдалих спроб очищення комірок пам'яті. При збільшенні лічильника з'являється ймовірність передчасного строку виходу диску з ладу

– SSD Life Left – параметр, у якому вказується значення здоров'я диску у відсотках. У процесі експлуатації він змінюється від 100% до 0 по алгоритму, що заклали виробники SSD-диску.

– Percentage_of_the_Rated_Lifetime_Used – лічильник, зворотний попередньому. Указує на відсоток зношування й змінюється від 0 до 100%.

– Grown_Failing_Block_Count – виробники жорстких дисків не прийшли до єдиної думки, що відображати цим параметром, але однаково збільшення, так само як і високі значення цього лічильника – причина більш пильно спостерігати за станом диску.

Але всі ці параметри не можуть дати повної картини й точно пророчити строк виходу з ладу жорсткого диску. Тому рекомендую вам регулярно стежити за його станом і робити резервні копії.

1.2 Область застосування

Система S.M.A.R.T. (або SMART) працює трохи інакше при використанні разом з накопичувачами SSD. Справді, такий параметр, як «лічильник невдалих спроб розкручування пластин» у випадку з SSD безглуздий. У той же час, кількість циклів запису в кожну комірку пам'яті у випадку з SSD – обмежено кінцевим значенням, тому цей параметр має сенс підраховувати й порівнювати із граничним значенням.

Розглянемо носій інформації компанії SanDisk. Програма CrystalDiskInfo обчислює ступінь зношування SSD накопичувачів аналізом змінних Reallocated Sectors Count, Current Pending Sectors Count, Uncorrectable Sector Count, а також змінної, специфічної для дисків типу SSD – Percentage of the Rated Lifetime Used (або, для деяких моделей, параметра SSD Life Left).

Зношування диску (англ. Wear Leveling Count). Лічильник має ненульове значення на початку, і зменшується згодом. При досягненні якогось заданого виробником граничного значення, диск зізнається повністю зношеним і непридатним до подальшої експлуатації. Зверніть увагу на цей параметр – він покаже, скільки залишилося жити вашому диску.

Спроби очищення комірки пам'яті (англ. Erase Fail Count). При передчасному виході осередків з ладу цей лічильник збільшується. Велика кількість таких осередків указує на високу ймовірність того, що диск вийде з ладу передчасно – задовго до досягнення закладеного виробником числа циклів перезапису.

Залишок життя диску (англ. SSD Life Left). Виробники обчислюють цю змінну у відсотках: значення 100 (100%) указує на повністю здоровий пристрій, а значення 1 (1%) означає, що накопичувач повністю зношений. Іноді замість цього параметра використовується зворотний йому лічильник – Percentage of the Rated Lifetime Used.

Зношування диску (англ. Percentage of the Rated Lifetime Used). Одиниця означає новий диск, 100 – сто відсотків зношування, диск можна викидати.

У теорії, строк життя SSD, що залишився, пророчити досить легко простим читанням змінних S.M.A.R.T. На жаль, у житті все не так просто. SSD накопичувачі будь-яких виробників (наприклад, Sandisk, Transcend, і т.д.) виходять із ладу передчасно й зненацька: учора працював – а сьогодні вже немає. На поточному рівні розвитку технологій це, на жаль, неминуче зло. У позитиві можна сказати тільки те, що ситуація поліпшується згодом, і ймовірність несподіваного виходу з ладу нових моделей нижче, ніж у попередніх поколінь накопичувачів. Ну а поки ви можете використовувати наші програми для відновлення інформації у випадку не запланованої втрати даних.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи моніторингу стану SSD диску на основі технології Life Left, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

На ринку представлена досить велика кількість утиліт, які забезпечують моніторинг дисків шляхом контролю параметрів S.M.A.R.T. і температури. Деякі утиліти обмежуються їхнім зчитуванням і відображенням, інші інтерпретують отримані значення й видають власний вердикт про стан диску у вигляді когось умовного відсотка його здоров'я, іноді – ще й з рекомендаціями користувачеві про те, що варто почати в тій або іншій ситуації.

Всі відомі програми S.M.A.R.T.-моніторингу, як правило, без проблем розпізнають і сканують внутрішні жорсткі диски. Із зовнішніми накопичувачами справа є складнішою – повноцінно працювати з такими пристроями можуть далеко не всі утиліти (навіть при офіційно заявленої розроблювачем підтримці такого типу дисків). Більше того, більшість розроблювачів взагалі замовчує про те, які конкретно зовнішні накопичувачі в їхніх дітищах (у тому числі платних) підтримуються. Крім того, навіть за умови розпізнавання обраною утилітою конкретного зовнішнього накопичувача зовсім не факт, що програма зможе визначити стан «здоров'я» диску, оскільки не всі контролери твердих USB-дисків підтримують команди S.M.A.R.T. Що стосується SSD-накопичувачів, те їхнє розпізнавання вкупі з діагностикою, як правило, особливих проблем не викликає – правда, за умови, що в утиліті, що сподобалася, реалізована підтримка твердотільних накопичувачів.

Hard Disk Sentinel

Hard Disk Sentinel – визнане рішення для S.M.A.R.T.-моніторингу стану жорстких дисків (внутрішніх і зовнішніх) і твердотільних накопичувачів.

Програма представлена в декількох комерційних редакціях; для широкого кола користувачів інтерес представляють базова редакція Standard і розширена Professional, а також портативна редакція Portable для ноутбуків. Головна відмінність професійної редакції від редакції Standard – наявність функціонала для резервування даних (періодично або у випадку виявлення проблем з диском). Крім того, є безкоштовна редакція для DOS, що дозволяє контролювати температуру й атрибути S.M.A.R.T. жорстких дисків IDE/SATA, підключених прямо або через зовнішні контролери.

Програма стежить за атрибутами S.M.A.R.T. і температурою, скануючи жорсткі диски в автоматичному режимі через зазначену кількість мінут і на вимогу, і відображає на вкладці «Огляд» рівень продуктивності й «здоров'я» обраного диску з описом його поточного стану й перерахуванням проблем, що виникали за час роботи, (рисунок 2.1). Додатково на цій вкладці показуються загальний час роботи диску й приблизна оцінка часу, що залишилося, його життя, а також температури всіх контрольованих дисків, їхня ємність і кількість вільного простору. Про температури можна одержати й більше докладну інформацію (вкладка «Температура»), наприклад подивитися динаміку зміни середніх і максимальних температур. Крім того, короткий вердикт про стан диску відображається в системному треї (рисунок 2.2) і на іконках дисків у провіднику.

Що стосується значень S.M.A.R.T.-параметрів, то по них теж приводяться вичерпні відомості (вкладка «S.M.A.R.T.») – це забезпечує зручність відстеження змін, що мали місце, (рисунок 2.3). При бажанні можна навіть провести онлайн-порівняння значень S.M.A.R.T. обраного диску зі значеннями дисків такої ж моделі. Всю отриману в ході моніторингу інформацію нескладно зберегти у вигляді текстового або HTML-звіту й при необхідності відіслати по зазначеній електронній адресі. У випадку виявлення неполадок або перевищення температури програма може попередити користувача звуковим сигналом або повідомленням і відразу (при відповідних налаштуваннях) запустити процес резервного копіювання даних.

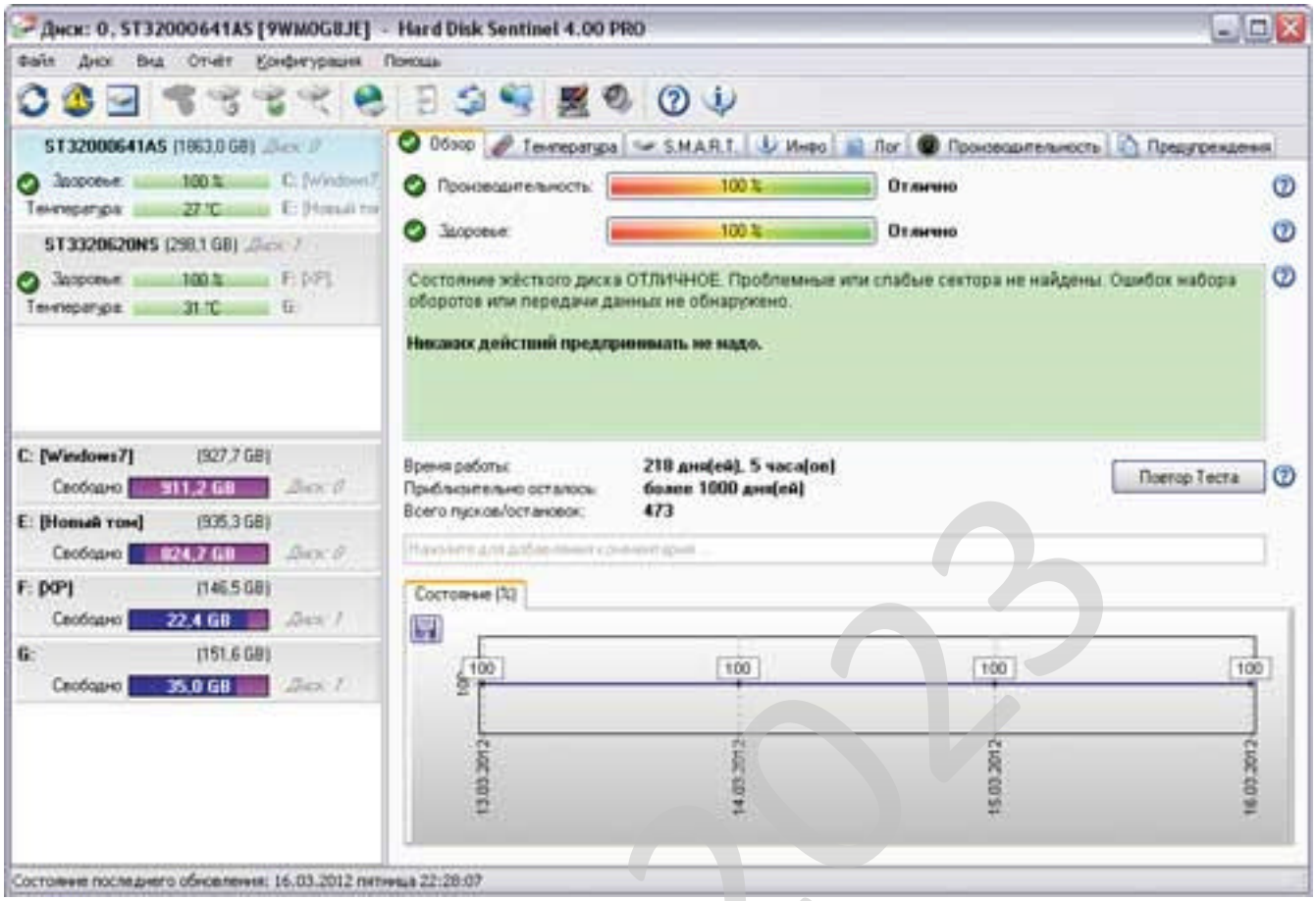

Рисунок 2.1 – Огляд дисків в Hard Disk Sentinel

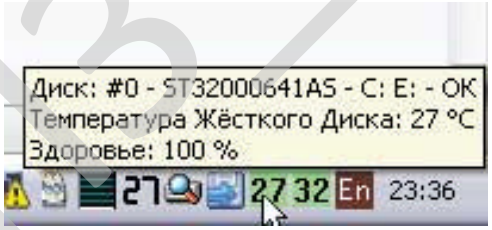

Рисунок 2.2 – Коротка інформація про стан диску в треї (Hard Disk Sentinel)

Додатково утиліта показує докладну інформацію про жорсткі диски (виробник, модель, серійний номер і т.д.) і вимірює швидкість передачі даних у реальному часі. Крім того, вона може застосовуватися для тестування диску на працездатність (тест підведення головки диску, тест дискової поверхні й ін.).

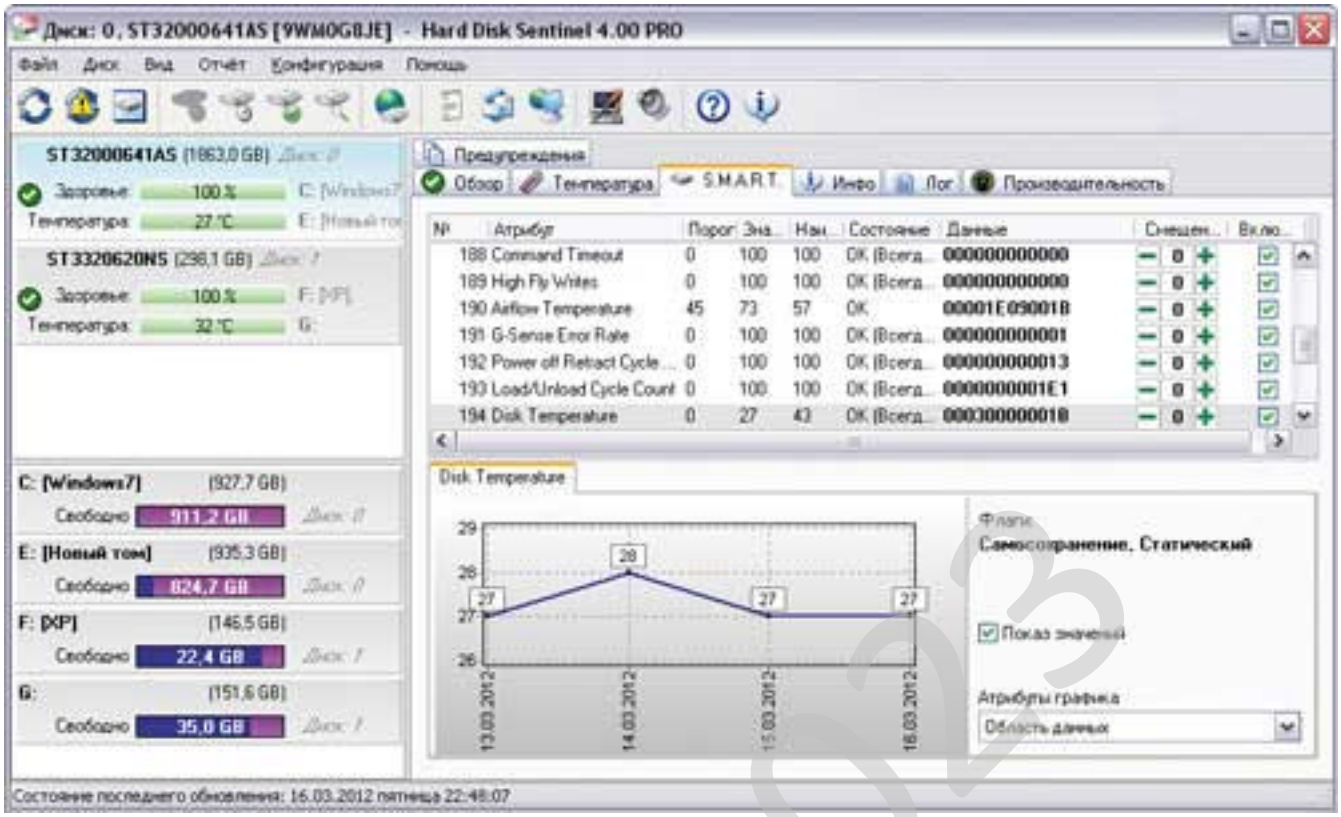

Рисунок 2.3 – Моніторинг S.M.A.R.T.-параметрів в Hard Disk Sentinel

Hard Drive Inspector

Hard Drive Inspector – зручне рішення для S.M.A.R.T.-моніторингу зовнішніх і внутрішніх жорстких дисків, а також SSD. Програма представлена в декількох редакціях: редакція Professional позиціонується як інструмент контролю жорстких дисків, а для роботи із твердотільними накопичувачами призначена редакція SSD. У програмі підтримується два робітники режиму – спрощений і розширений. У спрощеному режимі, що орієнтований на новачків, відображається тільки найважливіша інформація про поточний стан дисків. Розширений режим забезпечує доступ до широкого переліку технічних даних, проаналізувавши які професіонали зможуть одержати більше детальне подання про стан дисків.

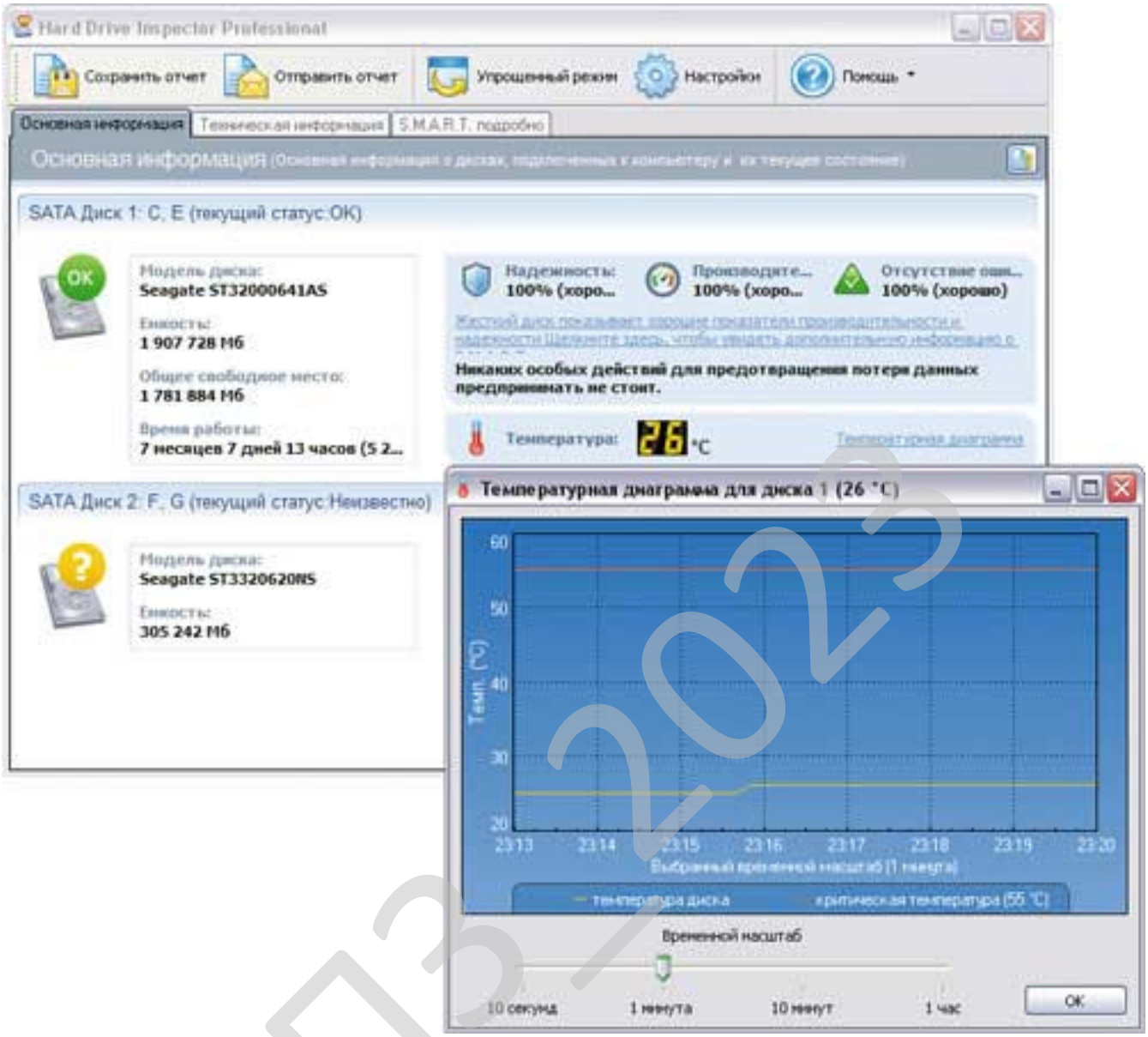

Рисунок 2.4 – Базова інформація про поточний стан диску в Hard Drive Inspector

Перевірка S.M.A.R.T.-атрибутів проводиться автоматично через зазначені проміжки часу. Під час аналізу всіх життєво важливих параметрів диску виробляється розрахунок значень умовних індикаторів його стану: «надійність», «продуктивність» і «відсутність помилок», які відображаються на вкладці «Основна інформація» разом із числовим значенням температури й температурною діаграмою (рисунок 2.4). Ця інформація супроводжується

технічними даними про модель диску, ємності, загальному вільному місці й часі роботи в годинниках (днях). У розширеному режимі додатково надається всебічна інформація про параметри дисків (розмір буфера, назва прошивання, список підтримуваних режимів передачі даних і т.д.), відображаються значення S.M.A.R.T.-параметрів із прапорами (рисунок 2.5).

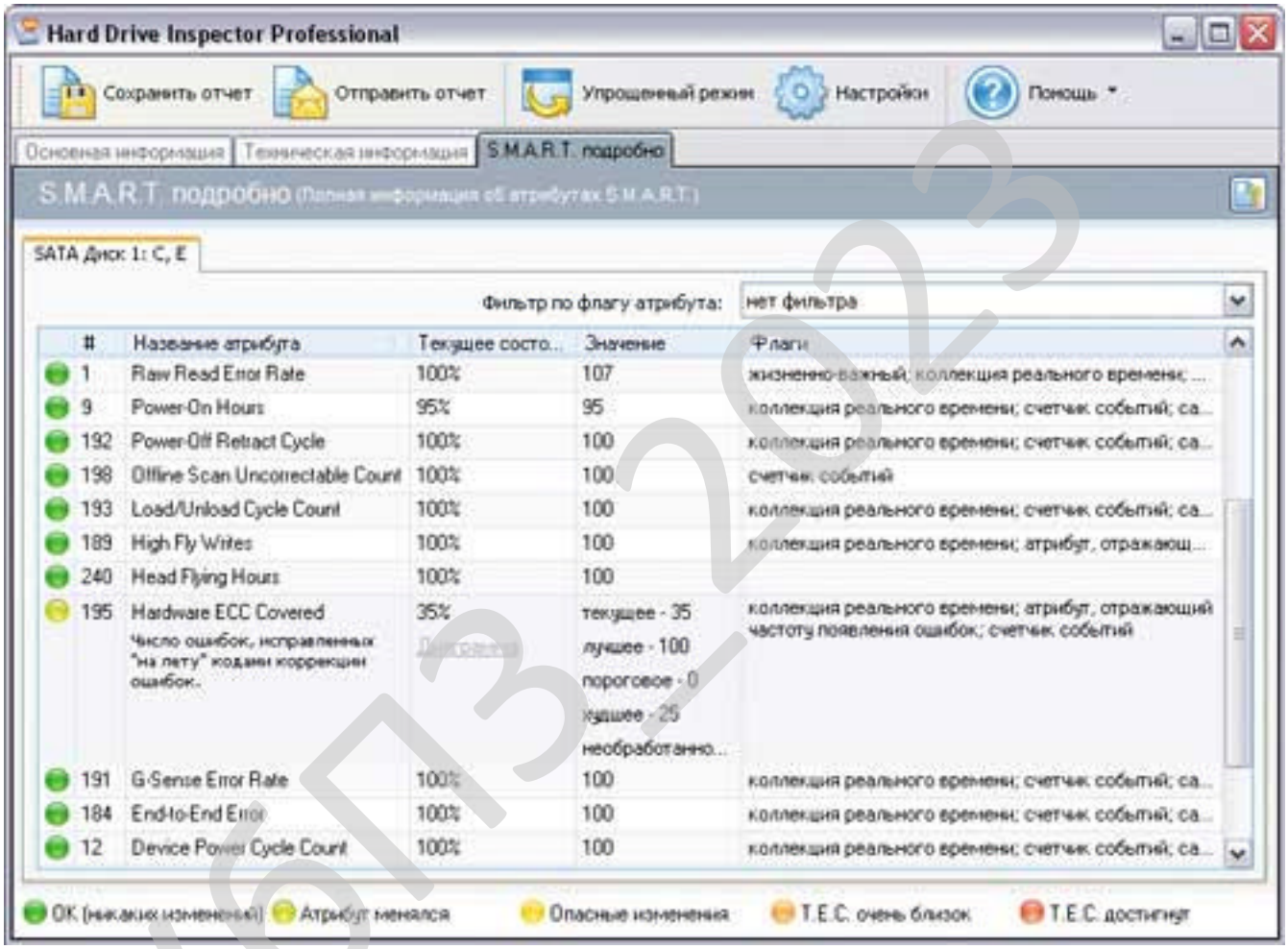

Рисунок 2.5 – S.M.A.R.T.-атрибути «докладно» в Hard Drive Inspector

При критичних змінах S.M.A.R.T.-параметрів програма може (після відповідних налаштувань) інформувати про це користувача самими різними способами – відобразивши на екрані повідомлення, подавши звуковий сигнал, відправивши повідомлення на зазначену електронну адресу та ін. Можливий навіть запуск сторонньої програми, що дозволяє зробити невідкладні дії для

збереження даних відразу після виявлення небезпеки (наприклад, запустити процедуру резервного копіювання даних).

Крім цього утиліта може застосовуватися для автоматичного керування шумом, виробленим жорсткими диском (дозволяє знизити рівень шуму за рахунок невеликого падіння продуктивності), і поліпшеного керування живленням (забезпечує скорочення енергоспоживання жорсткого диску – теж за рахунок деякого падіння продуктивності).

Active SMART

Active SMART – програма моніторингу жорстких дисків шляхом контролю параметрів S.M.A.R.T. і температури. Утиліта забезпечує автоматичну перевірку стану диску при завантаженні системи, здійснює безперервний моніторинг із зазначеним інтервалом часу й може проводити швидке сканування дисків на вимогу (актуально на малопотужних ПК). Статус контрольованого диску показується в треї, а отримані в ході аналізу дані відображаються у вигляді зведеної таблиці (рисунок 2.6) і докладних звітів по кожному з контрольованих S.M.A.R.T.-параметрів (із вказівкою значень, порога, дати T.E.C. і графіка змін значення атрибута) – рисунок 2.7. Для аналізу ситуації також надаються графіка змін температури диску в реальному часі й журнал подій, у якому фіксується вся історія S.M.A.R.T.-подій диску. Передбачено підтримку різних видів оповіщень (спливаюче повідомлення, звуковий сигнал, лист на зазначену поштову адресу й мережне повідомлення) при настанні критичних подій по контрольованих пристроях (у тому числі по окремих атрибутах S.M.A.R.T.).

Додатково програма надає загальну інформацію про вінчестер (модель, ємність, серійний номер, список всіх доступних дискових розділів із вказівкою вільного місця, що залишилося, на кожному з них, підтримувані й включені дискові режими передачі й ін.) і дозволяє з'ясувати, якими типами даних заповнений диск.

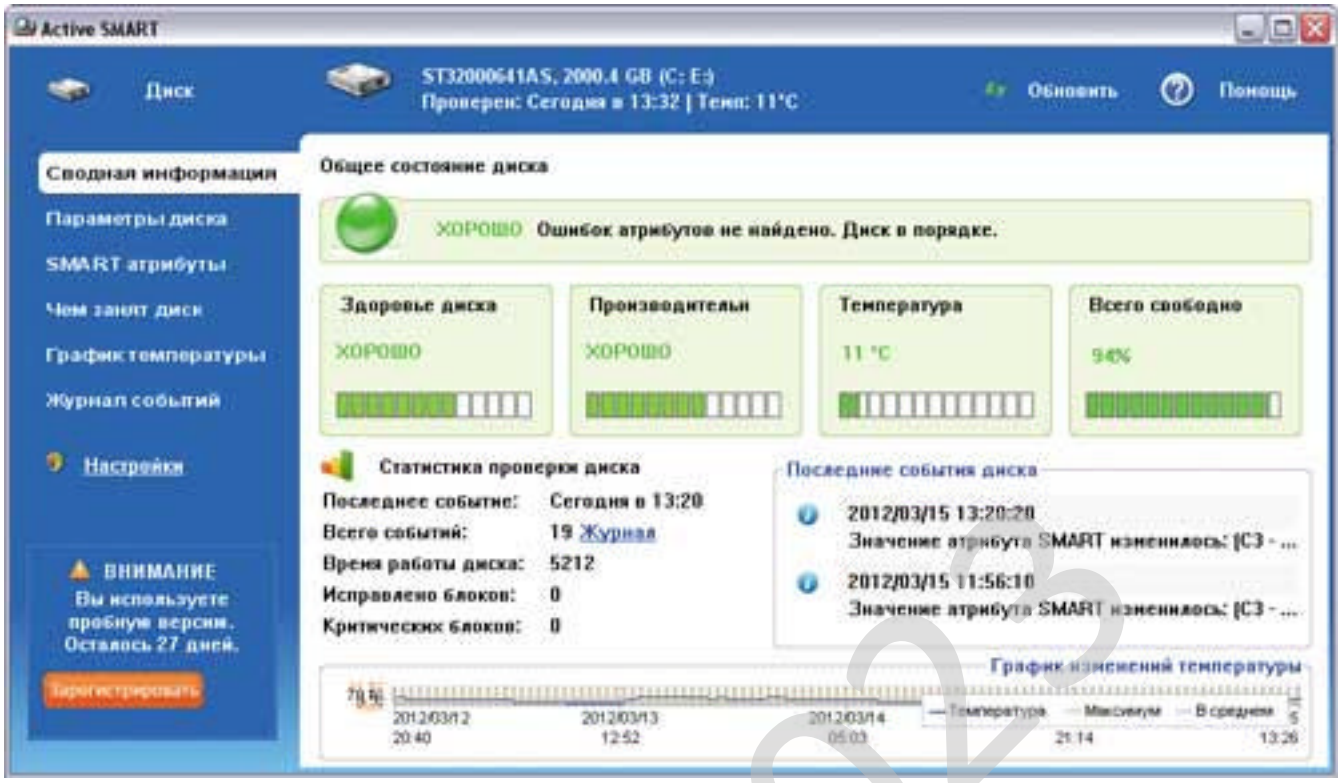

Рисунок 2.6 – Зведена інформація про стан диску в Active SMART

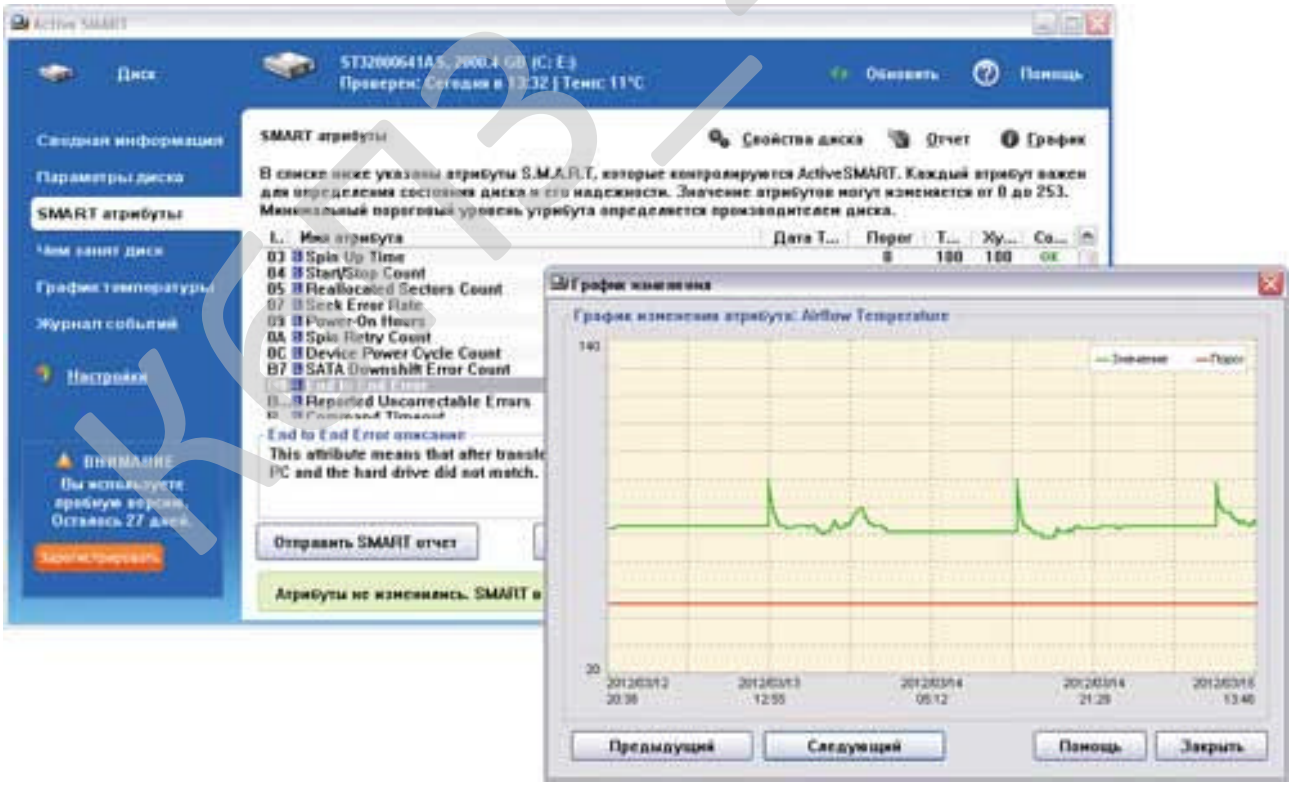

Рисунок 2.7 – Значення S.M.A.R.T.-атрибутів із графіком зміни обраного атрибута (Active SMART)

HDDlife

HDDLife – зручна в застосуванні утиліта, призначена для контролю стану жорстких дисків і SSD (з версії 4.0) в інтуїтивно зрозумілій формі.

Рисунок 2.8 – Вердикт про стан «здоров'я» диску, виданий HDDlife Pro

Програма здійснює безперервний моніторинг параметрів S.M.A.R.T. для всіх установлених у системі дисків і за результатами аналізу видає власний висновок про стан диску у вигляді когось умовного відсотка його здоров'я (рисунок 2.8). Даний підхід ідеальний для новачків, яким конкретні S.M.A.R.T.-значення, швидше за все, ні про що не скажуть, а відсоток «здоров'я» дозволить зорієнтуватися в ситуації. Одержати подання про стан «здоров'я» дисків можна декількома способами. Так, уже по зовнішньому вигляді індикатора в системному треї й іконок дисків у провіднику відразу видно, наскільки «здоровий» контрольований диск. Крім того, через головне вікно утиліти доступні більше докладні дані про стан пристрої. У випадку виникнення критичних ситуацій (зниження рівня «здоров'я» до критичного, досягнення критичної температури та ін.) передбачена система оповіщень. У ролі такого оповіщення може виступати хінт-повідомлення в системному треї, звуковий сигнал або текстове повідомлення, відправлене по комп'ютерній мережі або по електронній пошті.

Додатково утиліта відображає рівень продуктивності жорсткого диску, а також показує дані про робочу температуру пристрої, його ємності, обсязі вільного простору й відпрацьованому диском часу.

CrystalDiskInfo

CrystalDiskInfo – простий інструмент для S.M.A.R.T.-моніторингу стану жорстких дисків (включаючи багато зовнішні HDD) і твердотільних накопичувачів. Програма відрізняється компактним дистрибутивом, безкоштовна й має всього необхідного функціонала для організації контролю дисків.

Сканування дисків виробляється автоматично через зазначене число мінут або на вимогу. Температури контрольованих пристроїв відображаються в області повідомлень (потрібне включення відповідної опції), а детальна інформація про встановлені в комп'ютері накопичувачах, включаючи значення базових S.M.A.R.T.-параметрів, температуру й вердикт програми про стан пристроїв, – у головному вікні утиліти (рисунок 2.9). Крім того, на графіку можна побачити, як змінювалися ті або інші значення параметрів із часом. Для деяких параметрів

передбачений функціонал по налаштуванню граничних значень і відправленню повідомлень по електронній пошті у випадку перевищення параметром установленого порога.

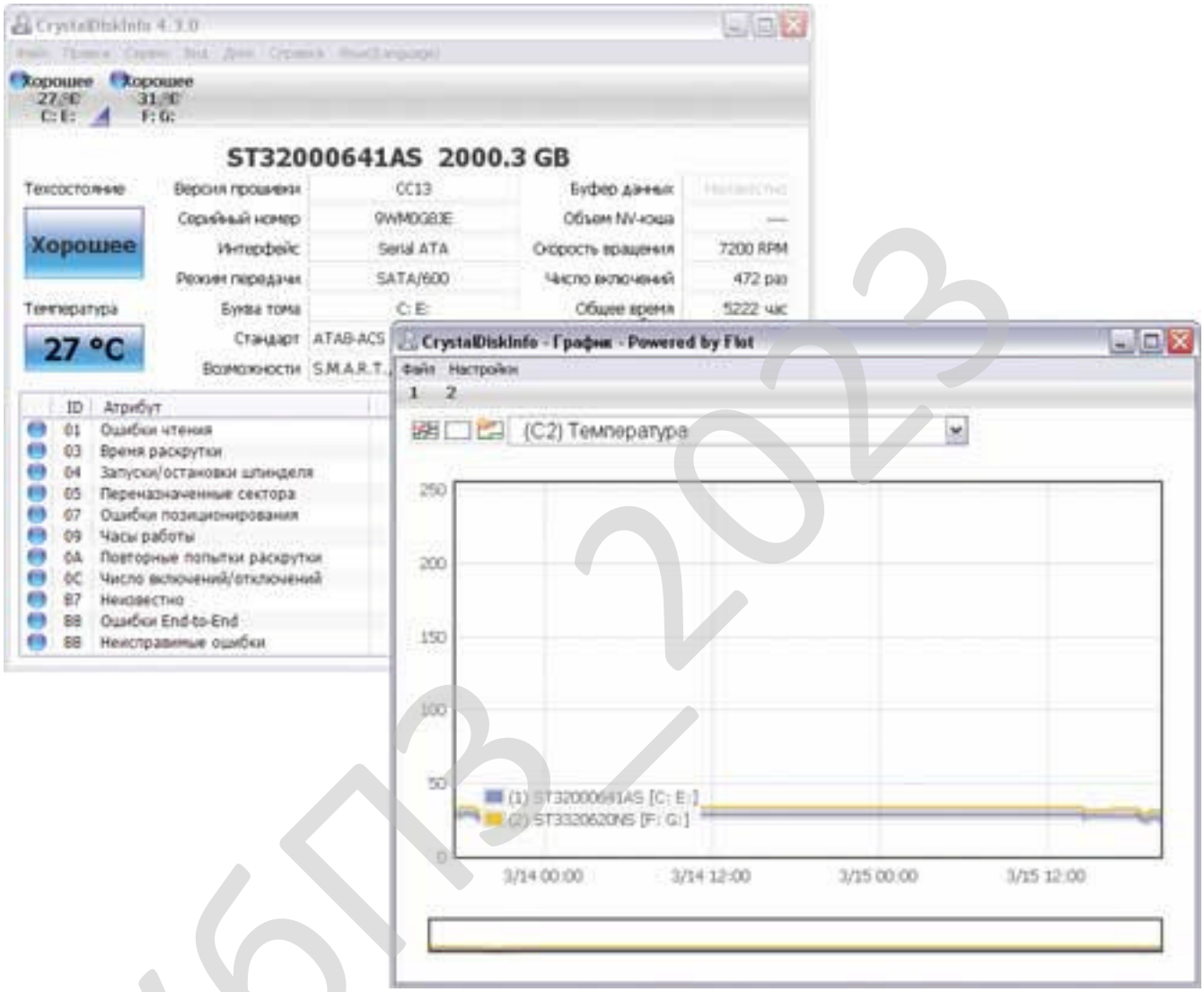

Рисунок 2.9 – Моніторинг дисків в CrystalDiskInfo

Додатково в програму включений інструментарій для автоматичного керування шумозаглушенням дисків (AAM) і розширеного керування живленням (APM).

Acronis Drive Monitor

Acronis Drive Monitor – утиліта для контролю «стану здоров'я» і температур жорстких дисків (включаючи зовнішні накопичувачі) і SSD через систему самодіагностики S.M.A.R.T. Програма безкоштовна, має мінімального функціонала для організації моніторингу дисків і здатна інтегруватися із продуктами резервного копіювання компанії Acronis, але відрізняється дуже громіздким дистрибутивом і не має російськомовної локалізації.

Утиліта сканує диски самостійно по передвстановленному розроблювачами розкладу (моніторинг окремих дисків може бути відмінний) і відображає результуючу інформацію на вкладці Disks (рисунок 2.10). Даних мінімум – це рівень «здоров'я» диску у відсотках (Health), індикатор кількості днів експлуатації (Power On Time) і температура (Disc temperature; значення критичних температур прописуються в налаштуваннях). Теоретично дані про ситуацію відображаються й у системному треї, однак інформативність їх досить сумнівна (відсутній список дисків, дані по їхніх температурах і рівню «здоров'я»). Інформація щодо значень параметрів S.M.A.R.T. представлена в традиційному текстовому виді, ніяких графіків (зокрема, графіка температури) не передбачено. Для контрольованих дисків ведеться протокол критичних подій, фіксуємих у журналі. При виявленні проблем утиліта інформує про це користувача (шляхом видачі текстового повідомлення на екран або відправлення його по електронній пошті) і може автоматично створити завдання на резервне копіювання даних. Останнє реалізується тільки при наявності на комп'ютері одного з відповідних продуктів Acronis.

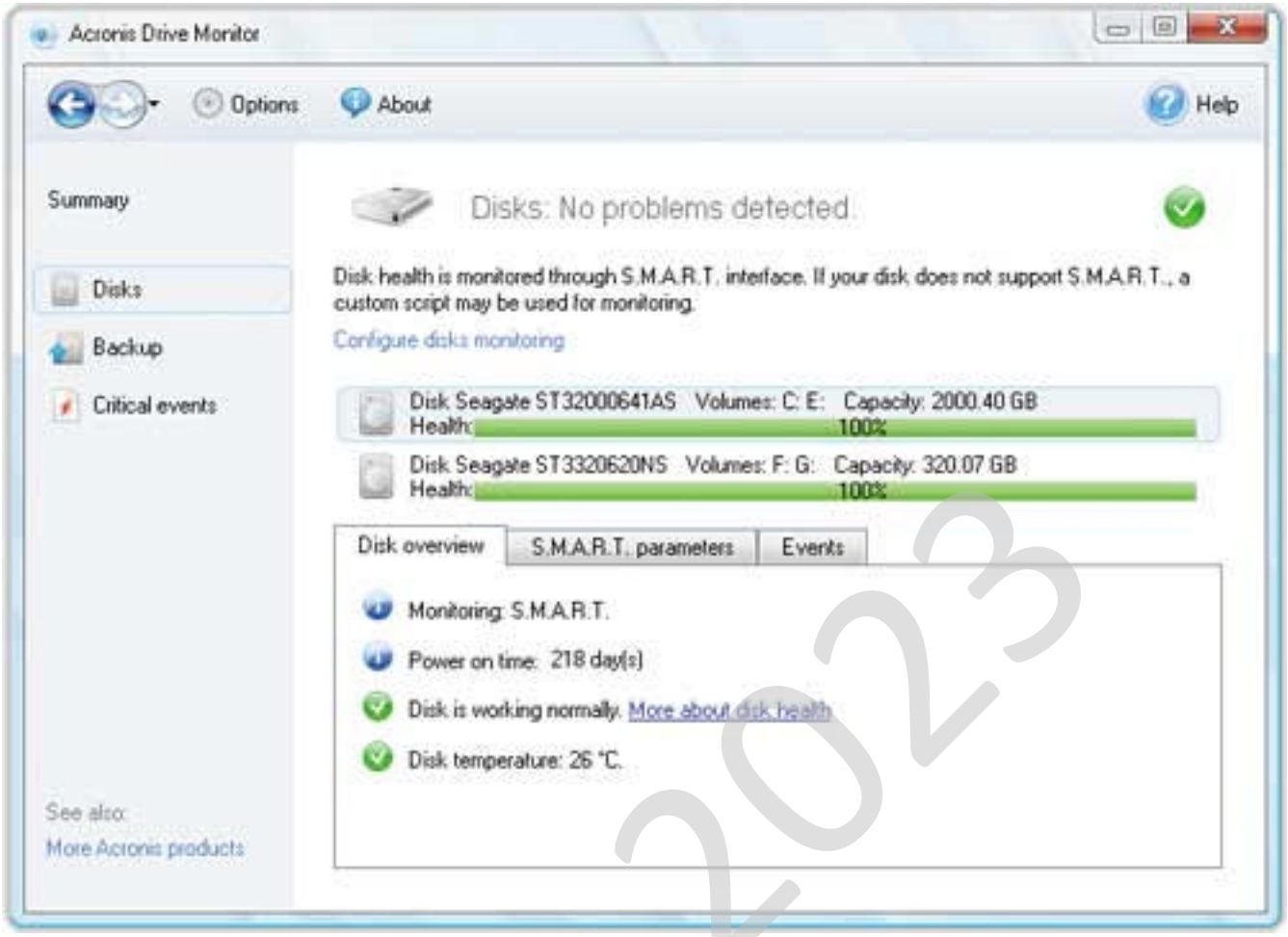

Рисунок 2.10 – Результуюча інформація про диски в Acronis Drive Monitor

Висновок

Про майбутній вихід з ладу жорсткого диску або твердотільного накопичувача будь-якому користувачеві хочеться знати заздалегідь. Це цілком реально, якщо довірити моніторинг даних пристроїв однієї з розглянутих у статті утиліт, що дозволить завчасно одержати інформацію про можливий збій диску й надасть деякий ліміт часу на те, щоб попередити втрату даних. Яку з утиліт вибрати – справа смаку, однак, на наш погляд, з безкоштовних рішень помітно виділяється програма CrystalDiskInfo, а серед розглянутих комерційних інструментів найбільше вражає Hard Disk Sentinel.

Вибір рішення варто робити з обліком того, які диски передбачається контролювати, оскільки не всі утиліти підтримують SSD і не все справляються з

контролем зовнішніх накопичувачів. Крім того, варто звернути увагу на особливості подання в утилітах накопичених даних. У більшості рішень на суд користувача виноситься значний набір значень S.M.A.R.T.-параметрів, які не дуже зрозумілі широкому користувачеві, але професіоналам можуть повідати багато про що. Разом з тим на ринку представлені й продукти з іншим підходом відображення S.M.A.R.T.-параметрів – у них замість «незрозумілих» абстрактних цифр приводиться деяка умовна величина, що відбиває загальний стан диску у відсотках (наприклад, «здоров'я» диску в HDDlife), що дозволить правильно оцінити ситуацію з диском навіть новачкові.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Для реалізації програми мною була використана мова програмування Visual C++. У зв'язку з тим, що сьогодні рівень складності програмного забезпечення дуже високий, розробка застосунків Windows з використанням тільки якої-небудь мови програмування значно утрудняється. Програміст повинен затратити масу часу на рішення стандартних завдань по створенню багатовіконного інтерфейсу. Реалізація технології зв'язування й вбудовування об'єктів – OLE – зажадає від програміста ще більш складної роботи. Щоб полегшити роботу програміста практично всі сучасні компілятори з мови C++ містять спеціальні бібліотеки класів. Такі бібліотеки містять у собі практично весь програмний інтерфейс Windows і дозволяють користуватися при програмуванні засобами більш високого рівня, чим звичайні виклики функцій. За рахунок цього значно спрощується розробка застосунків, що мають складний інтерфейс користувача, полегшується підтримка технології OLE і взаємодія з базами даних. Сучасні інтегровані засоби розробки застосунків Windows дозволяють автоматизувати процес створення застосунка. Для цього використовуються генератори застосунків. Програміст відповідає на питання

генератора застосунків і визначає властивості застосунка – чи підтримує воно багатовіконний режим, технологію OLE, тривимірні органи керування, довідкову систему. Генератор застосунків, створить додаток, що відповідає вимогам, і надасть вихідні тексти. Користуючись їм як шаблоном, програміст зможе швидко розробляти свої застосунки. Подібні засоби автоматизованого створення застосунків включені в компілятор Microsoft Visual C++ і називаються MFC AppWizard. Заповнивши кілька діалогових панелей, можна вказати характеристики застосунка й одержати його тексти, постачені великими коментарями. MFC AppWizard дозволяє створювати одновіконні й багатовіконні застосунки, а також застосунки, що не мають головного вікна, – замість нього використовується діалогова панель. Можна також включити підтримку технології OLE, баз даних, довідкової системи. Звичайно, MFC AppWizard не всесильний. Прикладну частину застосунка програмістові прийдеться розробляти самостійно. Вихідний текст застосунка, створений MFC AppWizard, стане тільки основою, до якої потрібно підключити інше. Але працюючий шаблон застосунка – це вже половина всієї роботи. Вихідні тексти застосунків, автоматично отриманих від MFC AppWizard, можуть становити сотні рядків тексту. Набір його вручну був би дуже стомлюючий. Потрібно відзначити, що MFC AppWizard створює тексти застосунків тільки з використанням бібліотеки класів MFC (Microsoft Foundation Class library). Тому тільки вивчивши мову C++ і бібліотеку MFC, можна користуватися засобами автоматизованої розробки й створювати свої застосунки в найкоротший термін. Як уже згадувався, MFC – це базовий набір (бібліотека) класів, написаних мовою C++ і призначених для спрощення й прискорення процесу програмування для Windows. Бібліотека містить багаторівневу ієрархію класів, що нараховує близько 200 членів. Вони дають можливість створювати Windows-застосунки на базі об’єктно-орієнтованого підходу. З погляду програміста, MFC являє собою каркас, на основі якого можна писати програми для Windows. Бібліотека MFC розроблялася для спрощення завдань, що стоять перед програмістом. Як відомо, традиційний метод програмування під Windows

вимагає написання досить довгих і складних програм, що мають ряд специфічних особливостей. Зокрема, для створення тільки каркаса програми таким методом знадобиться близько 75 рядків коду. У міру ж збільшення складності програми її код може досягати воістину неймовірних розмірів. Однак та ж сама програма, написана з використанням MFC, буде приблизно в три рази менше, оскільки більшість приватних деталей приховано від програміста.

Одною з основних переваг роботи з MFC є можливість багаторазового використання того самого коду. В зв'язку з тим, що бібліотека містить багато елементів, загальних для всіх Windows-застосунків, немає необхідності щораз писати їх заново. Замість цього їх можна просто успадковувати (говорячи мовою об'єктно-орієнтованного програмування). Крім того, інтерфейс, забезпечуваний бібліотекою, практично незалежний від конкретних деталей, його що реалізують. Тому програми, написані на основі MFC, можуть бути легко адаптовані до нових версій Windows (на відміну від більшості програм, написаних звичайними методами). Ще однією істотною перевагою MFC є спрощення взаємодії із прикладним програмним інтерфейсом (API) Windows. Будь-який додаток взаємодіє з Windows через API, що містить кілька сотень функцій. Значний розмір API утрудняє спроби зрозуміти й вивчити його цілком. Найчастіше навіть складно простежити, як окремі частини API зв'язані один з одним! Але оскільки бібліотека MFC поєднує (шляхом інкапсуляції) функції API у логічно організовану безліч класів, інтерфейсом стає значно легше управляти.

Оскільки MFC являє собою набір класів, написаних мовою C++, тому програми, написані з використанням MFC, повинна бути в той же час програмами на C++. Для цього необхідно володіти відповідними знаннями. Для початку необхідно вміти створювати власні класи, розуміти принципи спадкування й уміти перевизначати віртуальні функції. Хоча програми, що використовують бібліотеку MFC, звичайно не містять занадто специфічних елементів з арсеналу C++, для їхнього написання проте потрібні солідні знання в даній області.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи моніторингу стану SSD диску на основі технології Life Left.

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

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

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

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

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

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

SSD (Solid State Drive) або твердотільний накопичувач – пристрій для постійного зберігання даних з використанням твердотільної (звичайно – флеш) пам'яті. Попросту говорячи, це просто флешка.

SSD логічно емулює звичайний жорсткий диск (HDD) і теоретично скрізь може застосовуватися замість нього.

Система зберігання даних у наші дні є основним «вузьким місцем» комп'ютера. Саме тому стільки надій сьогодні пов'язане з SSD, які можуть ефективно помножити продуктивність накопичувачів. Якщо ви встановите твердотільний накопичувач навіть у дешевий нетбук, то його чуйність збільшиться набагато сильніше, ніж якби ви подвоїли його оперативну пам'ять.

SSD – відносно нова технологія (принаймні, якщо порівнювати з жорсткими диском, яким здійснилося майже 60 років). Тому логічно порівнювати SSD і HDD.

Надійність SSD

Здавалося б, немає частин, що рухаються, – все повинне бути дуже надійно. Це не зовсім так. Будь-яка електроніка може зламатися, не виключення й SSD. Але найбільше джерело проблем – контролер і його прошивання. Через те, що контролер фізично розташований між інтерфейсом і мікросхемами пам'яті, імовірність його ушкодження в результаті збоїти або проблем з живленням дуже велика. При цьому самі дані, у більшості випадків зберігаються. Крім фізичних ушкоджень, при яких доступ до даних користувача неможливий, існують логічні ушкодження, при яких також порушується доступ до вмісту мікросхем пам'яті. Будь-яка, навіть незначна помилка, помилки в прошиванні, може привести до повної втрати даних. Структури даних дуже складні. Інформація «розмазується»

по декількох чипах, плюс чергування, роблять відновлення даних досить складним завданням.

У таких випадках відновити накопичувач допомагає прошивання контролера з низькорівневим форматуванням, коли заново створюються службові структури даних. Виробники намагаються постійно допрацьовувати мікропрограму, виправляти помилки, оптимізувати роботу контролера. По цьому, рекомендується періодично обновляти прошивання накопичувача для виключення можливих збоїв.

Безпека SSD

В SSD накопичувачі, як і в HDD, дані не віддаляються відразу після того, як файл був стертий з ОС. Навіть якщо переписати файл по верху нулями – фізично дані ще залишаються, і якщо чипи флеш-пам'яті дістати, і вважати на програматорі – можна знайти 4кб фрагменти файлів. Повне стирання даних варто чекати тоді, коли на диск буде записано даних рівне кількості вільного місця + обсяг резерву (приблизно 4 Гб для 60 Гб SSD). Якщо файл потрапить на «зношений» осередок, контролер ще не швидко перезапише її новими даними.

Основні принципи, особливості, відмінності у відновленні даних з SSD і USB Flash накопичувачів.

Відновлення даних з SSD накопичувачів досить трудомісткий і довгий процес у порівнянні з портативними flash накопичувачами. Процес пошуку правильного порядку, об'єднання результатів і вибору необхідного збирача (алгоритм/програма повністю емулююча роботу контролера SSD накопичувача) для створення образа диску не легке завдання.

Пов'язано це в першу чергу зі збільшенням числа мікросхем у складі SSD накопичувача, що в багато разів збільшує число можливих варіантів дій на кожному етапі відновлення даних, кожне з яких вимагає перевірки й спеціалізованих знань. Так само, у силу того, що до SSD пред'являються значно більше тверді вимоги по всіх характеристиках (надійність, швидкодія й т.д.), ніж до мобільних флеш накопичувачам, технології й методики роботи з даними,

застосовувані в них, досить складні, що вимагає індивідуального підходу до кожного рішення й наявності спеціалізованих інструментів і знань.

Оптимізація SSD:

1. Для того, щоб диск прослужив вам довго, потрібно все, що часто міняється (тимчасові файли, кеш браузера, індексування) необхідно перенести на HDD, відключити відновлення часу останнього доступу до папок і каталогів (fsutil behavior set disablelastaccess 1). Відключити в ОС дефрагментацію файлів.

2. Перед установкою на SSD Windows XP, при форматуванні диску рекомендується виконати «вирівнювання» розділів кратним ступеня двійки (наприклад, утилітою diskpart), інакше SSD прийдеться робити 2 читання замість одного. Крім цього в Windows XP є деякі проблеми з підтримкою секторів більше 512кб (в SSD за замовчуванням використовується 4кб) і проблеми, що випливають звідси, із продуктивністю. Windows Vista, Windows 7/10/11, останні версії Mac OS і Linux вирівнюють диски вже правильно.

3. Обновити прошивання контролера, якщо стара версія не знає команду TRIM. Установити останні драйвера на SATA контролери. Наприклад, якщо у вас контролер від Intel, ви можете на 10-20% збільшити продуктивність, включивши режим ACHI і встановивши Intel Matrix Storage Driver в операційній системі.

4. Не слід використовувати останні 10-20% вільного простору від розділу, тому що, це може негативно позначитися на продуктивності. Це особливо важливо, коли працює TRIM, оскільки йому необхідний простір для перегрупування даних: для приклада, схоже, працюють утиліти дефрагментації, адже їм теж потрібно не менш 10% відсотків від обсягу диску. Тому дуже важливо стежити за даним фактором, адже через невеликий обсяг SSD вони дуже швидко заповнюються.

Здоров'я SSD

На відміну від жорстких дисків, у світі SSD усе більш виразно. Флеш-пам'ять, на основі якої побудовані SSD диски, має точно відомий ресурс використання – 10000 перезаписів (спрощено говорячи, точне число залежить від

використовуваного в SSD типу пам'яті). Всі диски містять у собі мікропрограму, що стежить за рівномірним використанням всіх комірок пам'яті й відслідковує, скільки перезаписів був зроблений, який ресурс, що залишився, SSD диску. У підсумковому виді саме ці дані й повідомляються мікропрограмою диску в одному з параметрів S.M.A.R.T. с красномовною назвою SSD Life Left (Залишилося життя SSD) або Media wear out indicator (Показник зношування носія) – і саме цей параметр у зручному й зрозумілому для користувача виді відображає програма SSDLife.

Деякі виробники приводять загальний обсяг записаного на диск як один з показників терміну служби диску. Наприклад, Intel на диски X 25-M дає гарантію на загальний обсяг запису близько 37 Тбайт (20 Гбайт у день у плині 5 років – "The drive will have a minimum of 5 years of useful life under typical client workloads with up to 20GB host writes per day").

Переваги SSD перед HDD:

– висока швидкість читання будь-якого блоку даних не залежно фізичного від розташування (більше 200 Мб/с);

– низьке енергоспоживання при читанні даних з накопичувача (приблизно на 1 Ват нижче, ніж в HDD);

– знижене тепловиділення (внутрішнє тестування в компанії Intel показало, що ноутбуки з SSD нагріваються на 12.2° менше ніж аналогічні з HDD, також тестуванням установлено, що ноутбуки з SSD і 1 GB пам'яті в розповсюджених бенчмарках не уступають моделям з HDD і 4 GB пам'яті);

– безшумність і висока механічна надійність;

– більше швидкий запуск від включення до переходу в робочий стан, оскільки не потрібна розкручування шпинделя;

– краща здатність переносити екстремальні зовнішні умови – перевантаження, вібрації, перепади тиску й температури, що, крім спеціальних областей застосування, дуже добре для застосування в ноутбуках і іншій мобільній електроніці;

– відносно низька вага й розміри для SSD низкою ємності – незважаючи на те, що питома ємність на одиницю ваги й обсягу краще в "традиційних" HDD, для накопичувачів обсягом менш 256 Гб перевага у вазі й габаритах залишається за SSD;

– відмінна сумісність із існуючими інтерфейсами й розніманнями підключення, що виключає незручності, пов'язані з пошуком необхідних переходників.

Недоліки SSD

Головний недолік SSD – обмежена кількість циклів перезапису. Флеш-пам'ять дозволяє записувати дані приблизно 10 000 разів. Обмежене число циклів запису для SSD на базі флеш-пам'яті – звичайна флеш-пам'ять витримує 300.000 – 500.000 операцій стирання/запису в ту саму осередок, у деяких спеціальних типів флеш-пам'яті цей параметр декларується на рівні півтора мільйона операцій. Спеціальні файлові системи або алгоритми роботи контролера пристрою можуть зм'якшити цю проблему шляхом динамічного розподілу часто перезаписуваних кластерів рівномірно по диску (так зване "вирівнювання зношування"). У принципі, комплекс мер дозволяє сучасним SSD теоретично витримати до 20 років щоденної звичайної експлуатації в персональному комп'ютері.

Підпроблема сумісності SSD накопичувачів із застарілими версіями ОС сімейства Microsoft Windows, які не враховують специфіку SSD накопичувачів і додатково зношують їх, що зменшує цикл життя носія:

– Застосування в SSD-накопичувачах команди TRIM унеможливлює відновлення вилученої інформації recovery-утилітами

– Ємність – хоча зараз максимальна ємність випускаємої серійно SSD значно нижче такої у жорстких дисків, вона має тенденцію до швидкого збільшення.

– Менша швидкість запису (також для заснованих на флеш-пам'яті SSD) у силу конструктивних особливостей флеш-пам'яті, що допускає стирання тільки досить великими блоками, що дуже сильно знижує швидкість випадкового запису, і в меншому ступені – послідовного.

– Більша уразливість до ряду негативних факторів, включаючи раптове відключення живлення, магнітні поля й статична електрика.

– Ціна. У середньому вартість зберігання одного гігабайта на SSD поки ще трохи вище такої для звичайних жорстких дисків.

– Статистика повернень жорстких дисків і SSD-дисків. (Відмова диску має на увазі, що пристрій більше не функціонує. Але повернення може припускати безліч причин. Це створює певну проблему, адже в нас немає ніякої додаткової інформації із причин повернення дисків: вони могли бути мертві ще при надходженні в магазин, зламатися в плині строку експлуатації або всього лише мала місце якась несумісність із залізом, що перешкодило покупцеві використовувати накопичувач.)

Але, незважаючи не деякі недоліки SSD-дисків, їхні технічні характеристики постійно поліпшуються й саме перехід на SSD є тенденцією 2015 року.

Варто відзначити, що Intel збирається випустити 500-ю серію твердотільних накопичувачів King Cres у третьому кварталі 2015, а в четвертому кварталі ще й 700-ю серію Taylorsville (100 ГБ, 200 ГБ, 400 ГБ, 800 ГБ), 300-ю серію накопичувачів Jay Crest і Oak Crest. Незважаючи на те, що SSD накопичувачі коштують значно дорожче HDD дисків, саме перехід на SSD є тенденцією 2015 року.

Надійність твердих і SSD-дисків далека від ідеалу – рано або пізно вони виходять із ладу, що може викликати серйозні проблеми, аж до повної втрати даних, що зберігаються на них. Звичайно, у більшості випадків інформацію (повністю або частково) на зовні «мертвих» жорстких дискух можна відновити, але прийде звертатися до професіоналів (тому що необхідно спеціальне устаткування й відповідна технічна підготовка), та й обійдеться рішення даної проблеми в досить круглу суму. Тому краще спробувати попередити виникнення подібних ситуацій.

У цьому плані самим надійним є регулярне резервне копіювання інформації. Однак навіть при скрупульозному підході до завдання резервування можна не встигнути скопіювати важливу інформацію, адже вихід диску з ладу «за законом підлості» завжди відбувається зненацька. Разом з тим уникнути небезпеки втрати даних можна, якщо контролювати стан жорсткого диску за допомогою спеціалізованої утиліти S.M.A.R.T.-моніторингу. Кілька подібних програм ми й розглянемо в даній статті.

Двома словами про S.M.A.R.T.-моніторинг. Сьогодні всі сучасні HDD-диски й навіть твердотільні накопичувачі (SSD) підтримують технологію самодіагностики дисків – S.M.A.R.T. (Self-Monitoring Analysis and Reporting Technolodgy), що була спеціально розроблена для своєчасного виявлення прийдешнього виходу накопичувача з ладу. В основу цієї технології покладений безперервний моніторинг показань спеціальних сенсорів. Дані сенсори відображають поточні значення S.M.A.R.T.-параметрів, кожний з яких показує стан певної життєво важливої частини жорсткого диску (кількість помилок читання або записи, температуру, час роботи диску, продуктивність, швидкість пошуку інформації й т.п.). Значення параметрів при нормальній роботі диску можуть варіюватися в тих або інших інтервалах. При цьому для будь-якого параметра виробником визначене якесь граничне значення – мінімальне безпечне значення, що не може бути перевищене при нормальних умовах експлуатації.

Утиліти S.M.A.R.T.-моніторингу регулярно сканують жорсткі диски, витягають S.M.A.R.T.-інформацію із сенсорів і термодатчиків (датчиків температури, якими оснащені всі сучасні жорсткі диски), аналізують її й стежать за зміною стану всіх атрибутів. При виявленні критичних змін, які вказують на істотне падіння надійності диску, програми інформують користувача про те, що зберігання даних на жорсткому диску стало небезпечним. По запевненнях ряду розроблювачів, це відбувається не пізніше, ніж за день-два до виходу з ладу вінчестера, що забезпечує користувачеві деякий резерв часу, за який можна встигнути зробити копії всієї інформації, а може бути, навіть замінити жорсткий

диск. Варто помітити при цьому, що всі утиліти S.M.A.R.T.-моніторингу працюють у фоновому режимі, вимагають мінімуму апаратних ресурсів, і тому їхнє функціонування для користувача буде зроблено необтяжливим і ні в якій мері не перешкодить основному робочому процесу.

На жаль, застосування таких моніторингових утиліт не є панацеєю, оскільки вони не завжди можуть пророчити вихід накопичувача з ладу. Причина такого положення справ криється в тім, що жорсткі диски складаються з електронних і механічних компонентів. Зношування механічних частин відбувається поступово, і це процес контрольований, завдяки чому утиліти, як правило, вдало прогнозують вихід диску з ладу з вини «механіки». Вихід з ладу електронних компонентів найчастіше трапляється зовсім зненацька й тому практично непередбачений. Однак, відповідно до статистики компанії Seagate, близько 60% виходів жорстких дисків з ладу відбувається з вини механічних компонентів диску. Це значить, що ігнорувати систему S.M.A.R.T.-діагностики дисків у жодному разі не треба.

3.2 Розробка структурної схеми

SMART робить спостереження за основними характеристиками накопичувача, кожна з яких одержує оцінку. Характеристики можна розбити на дві групи:

– Параметри, що відбивають процес природного старіння жорсткого диску (число обертів шпинделя, число переміщень головок, кількість циклів включення-вимикання).

– Поточні параметри SSD диску (висота головок над поверхнею диску, число перепризначених секторів, час пошуку доріжки й кількість помилок пошуку).

Технологія SMART дозволяє здійснювати:

– моніторинг параметрів стану;

– сканування поверхні;

– сканування поверхні з автоматичною заміною сумнівних секторів на надійні.

Варто помітити, що технологія SMART дозволяє пророкувати вихід пристрою з ладу в результаті механічних несправностей, що становить близько 60 % причин, по яких вінчестери виходять із ладу. Пророчити наслідки стрибка напруги або ушкодження SSD диску в результаті удару SMART нездатний.

Слід зазначити, що SSD диски не можуть самі повідомляти про свій стан за допомогою технології SMART, для цього існують спеціальні програми. Таким чином, використання технології SMART немислимо без двох складових:

– ПЗ, убудованого в контролер SSD диску.

– Зовнішнього ПЗ, убудованого в хост.

S.M.A.R.T. являє собою набір міні-підпрограм, які є частиною мікрокоду SSD диску й визначають підтримувані діагностичні функції. Найпоширеніші серед них:

– набір атрибутів, що відбивають стан окремих параметрів SSD диску (до 30);

– внутрішні тести SSD диску (self– test);

– журнали S.M.A.R.T. (помилок, загального стану, дефектних секторів і т.п.).

У даний момент не існує офіційної документації або стандарту на технологію S.M.A.R.T. У зв'язку із цим, виробники не публікують повні характеристики й підтримувані функції S.M.A.R.T. у своїх SSD дискух. Обов'язковий мінімум описаний в останньому стандарті ATA/ ATAPI-6.

Розвиток технології S.M.A.R.T.

Історія технології S.M.A.R.T. не так уже й багата подробицями:

– SMART I передбачав моніторинг основних життєво важливих параметрів і запускався тільки після команди по інтерфейсу.

– в SMART II з'явилася можливість фонової перевірки поверхні, що виконувалася SSD диском автоматично під час "холостого ходу"; з'явилася функція журналювання помилок.

– в SMART III уперше з'явилася не тільки функція виявлення дефектів поверхні, але й можливість їхнього відновлення "прозоро" для користувача й багато інших нововведень.

Відомо, що першими розробили основи й запропонували цю технологію спільно Western Digital, Seagate і Quantum. Після цього їх уже підтримали такі компанії як IBM, Maxtor і Samsung. Hitachi взяла участь у розвитку технології S.M.A.R.T. уже на стадії розробки SMART II, першими запропонувавши методику повної самодіагностики SSD диску (extended self-test).

У цей час виробники жорстких дисків готуються прийняти до використання новий варіант технології S.M.A.R.T. – "1024 S.M.A.R.T.", характерною рисою якого буде помітно більший розмір журналів, повсюдне використання мультисекторних журналів, більше точні алгоритми аналізу показань убудованих у SSD диск сенсорів (термодатчики, сенсори ударів, і т.п.) і багато чого іншого. От кілька нових функцій:

– введення алгоритму аналізу температурного режиму SSD диску;

– введення обмеження по мінімальній і максимальній температурі в робочому стані;

– введення лічильника загальної кількості записаних секторів протягом життєвого циклу SSD диску;

– введення лічильника запусків внутрішніх алгоритмів відновлення (recovery counters).

Головним же плюсом можна вважати введення нових атрибутів, які дозволять контролювати стан і робочі характеристики по кожній з головок читання/запису:

– відносна стійкість (стабільність "польоту") головки;

– виправлення помилок читання (з "схованими" повторними спробами);

– автоматичний перерозподіл дефектних ділянок поверхні при операціях запису;

– лічильник-SSD диск G-List для обліку кількості прийнятих ударних навантажень;

– лічильник-SSD диск S-List для обліку загальної кількості "програмних" помилок.

Дані зберігаються в шістнадцатковому виді, називаному «raw value», а потім перераховуються в «value», значення, що символізує надійність щодо деякого еталонного значення. Звичайно «value» розташовується в діапазоні від 0 до 100 (деякі атрибути мають значення від 0 до 200 і від 0 до 253).

Висока оцінка говорить про відсутність змін даного параметра або повільному його погіршенні. Низька говорить про можливий швидкий збій.

Значення, менше, чим мінімальне, при якому виробником гарантується безвідмовна робота SSD диску, означає вихід вузла з ладу.

Програми, що відображають стан SMART-атрибутів, працюють за наступним алгоритмом:

– Перевіряють наявність підтримки технології SMART SSD диском.

– Подають у SSD диск команду запиту SMART-таблиць.

– Одержують таблиці в буфер додатка.

– Розбирають табличні структури, витягаючи з них номера атрибутів і їхні числові значення.

– Зіставляють стандартизовані номери атрибутів їхнім назвам (іноді – залежно від типу, моделі або фірми-виробника HDD).

– Виводять числові значення в зручному для сприйняття виді.

– Витягають із таблиць прапори атрибутів (ознаки, що характеризують призначення атрибута в рамках конкретної firmware SSD диску, наприклад, «життєво важливий» або «лічильник»).

– На підставі всіх таблиць, значень і прапорів виводять загальний стан пристрою.

На рисунку 3.1 зображена структурна схема розробленої системи моніторингу стану SSD диску на основі технології Life Left.

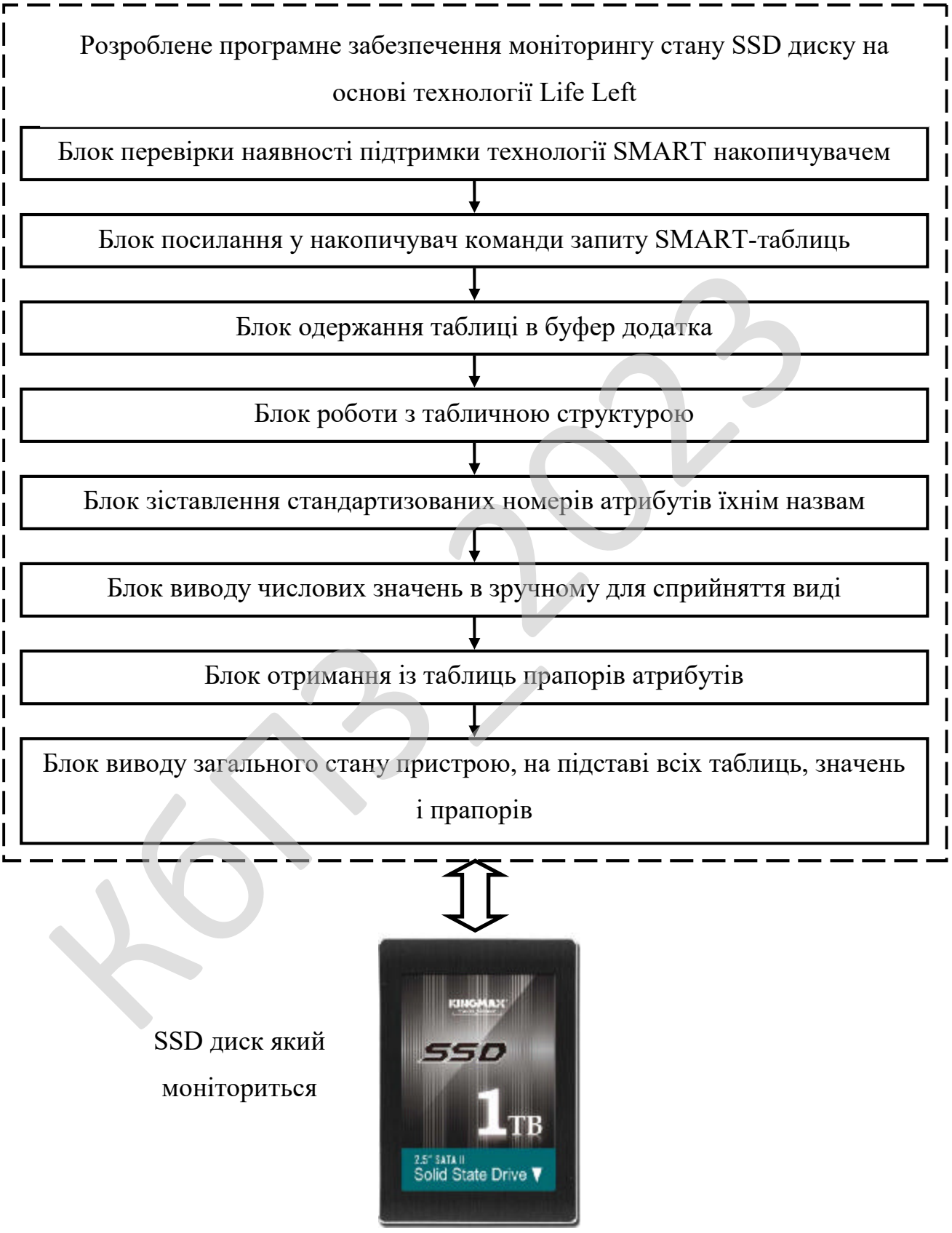

Рисунок 3.1 – Структурна схема системи

Структурна схема складається з наступних блоків:

– Блок перевірки наявності підтримки технології SMART SSD диском.

– Блок посилання у SSD диск команди запиту SMART-таблиць.

– Блок одержання таблиці в буфер додатка.

– Блок роботи з табличною структурою, що витягає з них номера атрибутів і їхні числові значення.

– Блок зіставлення стандартизованих номерів атрибутів їхнім назвам (іноді – залежно від типу, моделі або фірми-виробника HDD).

– Блок виводу числових значень в зручному для сприйняття виді.

– Блок отримання із таблиць прапорів атрибутів (ознаки, що характеризують призначення атрибута в рамках конкретної firmware SSD диску, наприклад, «життєво важливий» або «лічильник»).

– Блок виводу загального стану пристрою, на підставі всіх таблиць, значень і прапорів.

3.3 Розробка функціональної схеми

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

Функціональна схема складається з наступних блоків:

1. Блок читання журналу помилок:

– Каталог журналів S.M.A.R.T.

– Сумарний журнал помилок.

– Комплексний журнал помилок.

– Розширений комплексний журнал помилок.

– Журнал результатів самоконтролю.

– Розширений журнал результатів самоконтролю.

– Журнал параметрів продуктивності потоків.

– Журнал помилок потокового запису.

– Журнал помилок потокового читання.

– Журнал непоправних помилок.

– Користувальницькі журнали.

– Технічні журнали виробника.

2. Блок вбудованих функцій самоконтролю.

3. Блок вибору методів тесту:

– автономний (off-line);

– монопольний (captive).

4. Блок набору «активних» тестів.

5. Блок читання атрибутів:

– Частота появи помилок при читанні даних з SSD диску.

– Середня продуктивність (пропускна здатність) SSD диску.

– Час розкручування шпинделя.

– Кількість циклів запуск/останов шпинделя.

– Кількість перепризначених секторів.

– Запас каналу читання.

– Частота появи помилок позиціонування БМГ.

– Середня продуктивність операцій позиціонування БМГ.

– Кількість відпрацьованих годин у включеному стані.

– Кількість повторів спроб старту шпинделя SSD диску.

– Кількість повторів спроб рекалібровки SSD диску.

– Кількість повних циклів запуску/останова SSD диску.

– Частота появи "програмних" помилок при читанні даних з SSD диску.

– Кількість циклів виводу БМГ у спеціальну парковочну зону/у робоче положення.

– Температура.

– Кількість операцій перепризначення (ремаппінгу).

– Поточна кількість нестабільних секторів.

– Кількість нескоректованих помилок.

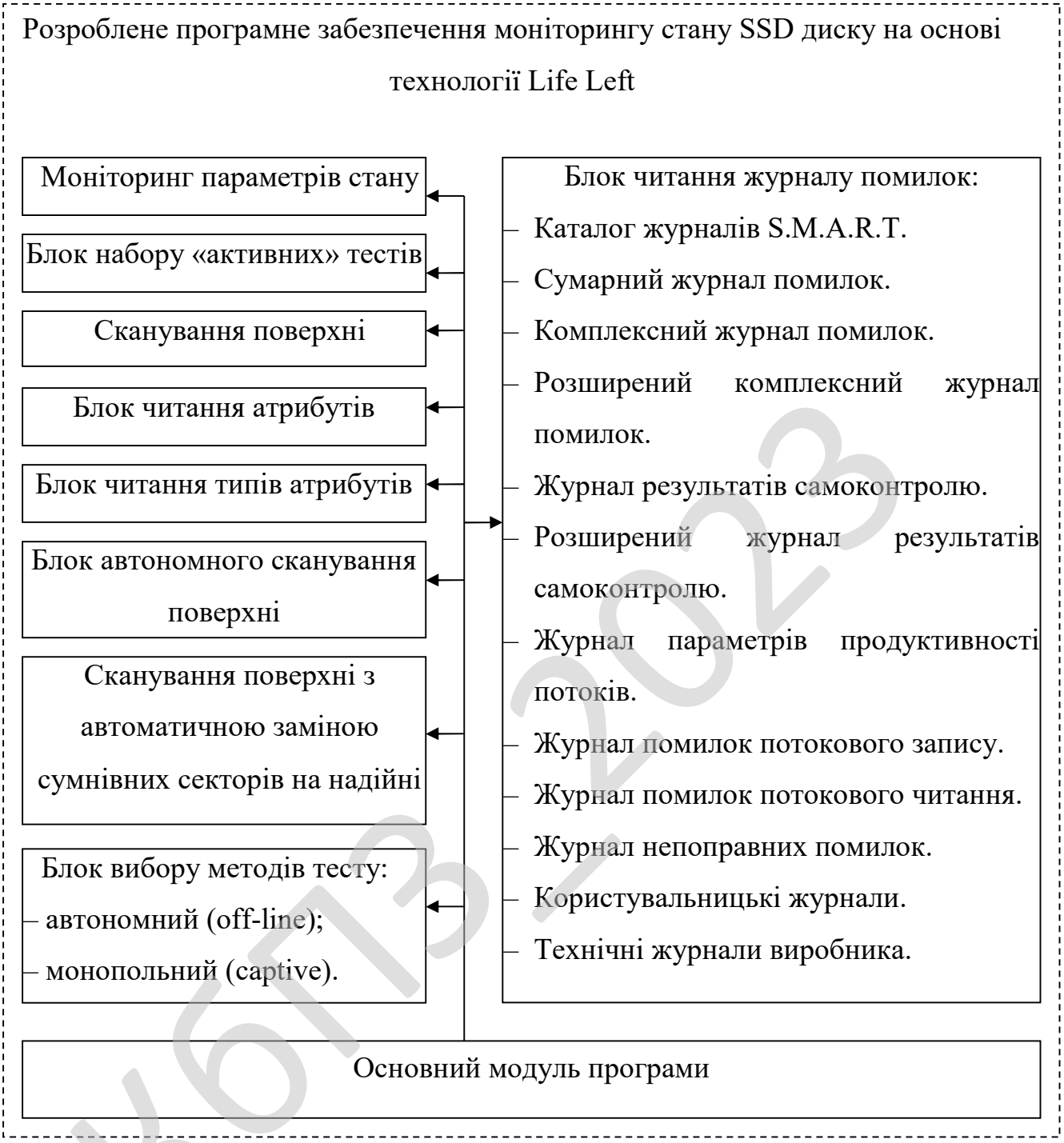

Рисунок 3.2 – Функціональна схема системи

– Загальна кількість помилок CRC у режимі UltraDMA.
– Частота появи помилок при записі даних.
– Зрушення пакета дисків щодо осі шпинделя.
– Частота появи помилок у результаті ударних навантажень.

– Навантаження на привод БМГ, викликана загальним наробітком годин SSD дискем.

– Навантаження на привод БМГ, викликана численними повтореннями операцій читання, запису, позиціонування головок і т.п.

– Навантаження на привод БМГ, викликане тертям у механічних частинах SSD диску.

– Загальна кількість циклів навантаження на привод БМГ.

– Загальний час навантаження на привод БМГ.

– Кількість зусиль обертаючого моменту привода.

– Кількість зафіксованих повторів включення/вимикання живлення SSD диску.

– Амплітуда тремтіння головок (GMR-head) у робочому стані.

6. Блок читання типів атрибутів:

– Pre-failure (PF). Якщо атрибут має цей тип, то поле threshold атрибута містить мінімально припустиме значення атрибута, нижче якого не гарантується працездатність SSD диску й різко збільшується ймовірність його виходу з ладу.

– On-line collection (OC). Указує, що значення даного атрибута обновляється (обчислюється) під час виконання on-line тестів S.M.A.R.T. або ж під час обох видів тестів (on-line/off-line). У противному випадку, значення атрибута обновляється тільки при виконанні off-line тестів.

– Performance related (PR). Указує на те, що значення цього атрибута прямо залежить від продуктивності SSD диску за окремими показниками (seek/throughput/etc. performance). Звичайно обновляється після виконання self-test'ов SMART.

– Error rate (ER). Указує на те, що значення атрибута відбиває відносну частоту помилок по даному параметрі (raw read/write, seek, etc.).

– Events count (EC). Указує на те, що атрибут є лічильником подій.

– Self-preserve (SP). Указує на те, що значення атрибута обновляється й зберігається автоматично (звичайно при кожному старті SSD диску й при виконанні тестів SMART).

7. Блок автономного сканування поверхні.

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

3.4 Розробка діаграми процесів

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

– Інтерфейс ПЗ.

– Налаштування системи.

– Журнал роботи ПЗ.

– Перевірка наявності підтримки технології Life Left.

– Запит Life Left даних.

– Отримання із таблиць прапорів атрибутів.

– Одержання таблиці в буфер додатка.

– Зіставлення номерів атрибутів їхнім назвам.

– Пошук несправностей.

– Аналіз загального стану пристрою.

– Виведення результатів.

Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі. Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування). Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи.

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

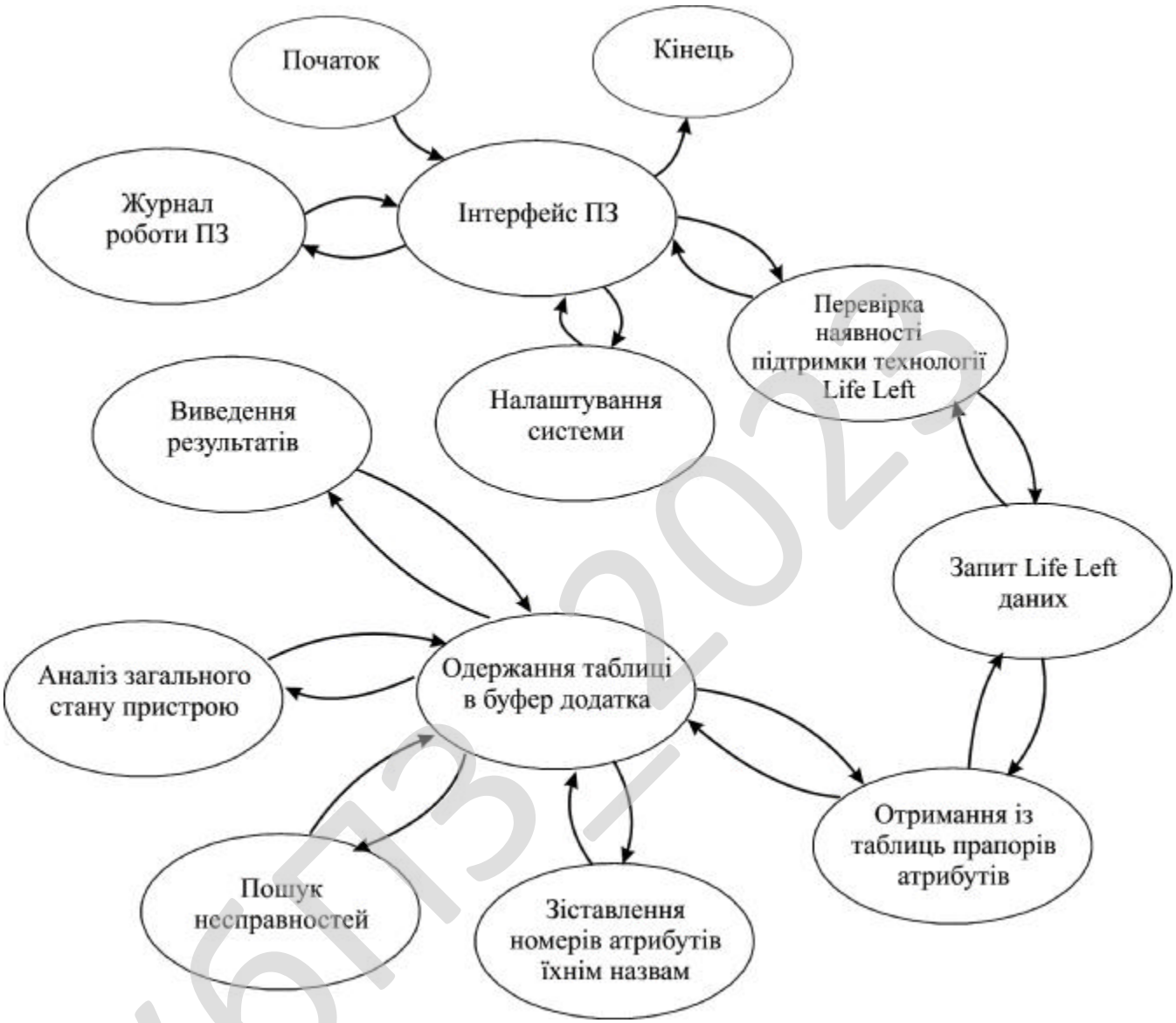

Рисунок 3.3 – Діаграма взаємодії процесів

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно розробка блок-схем. На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми. З якої видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ. При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю.

UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм. Різні види діаграм які

підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

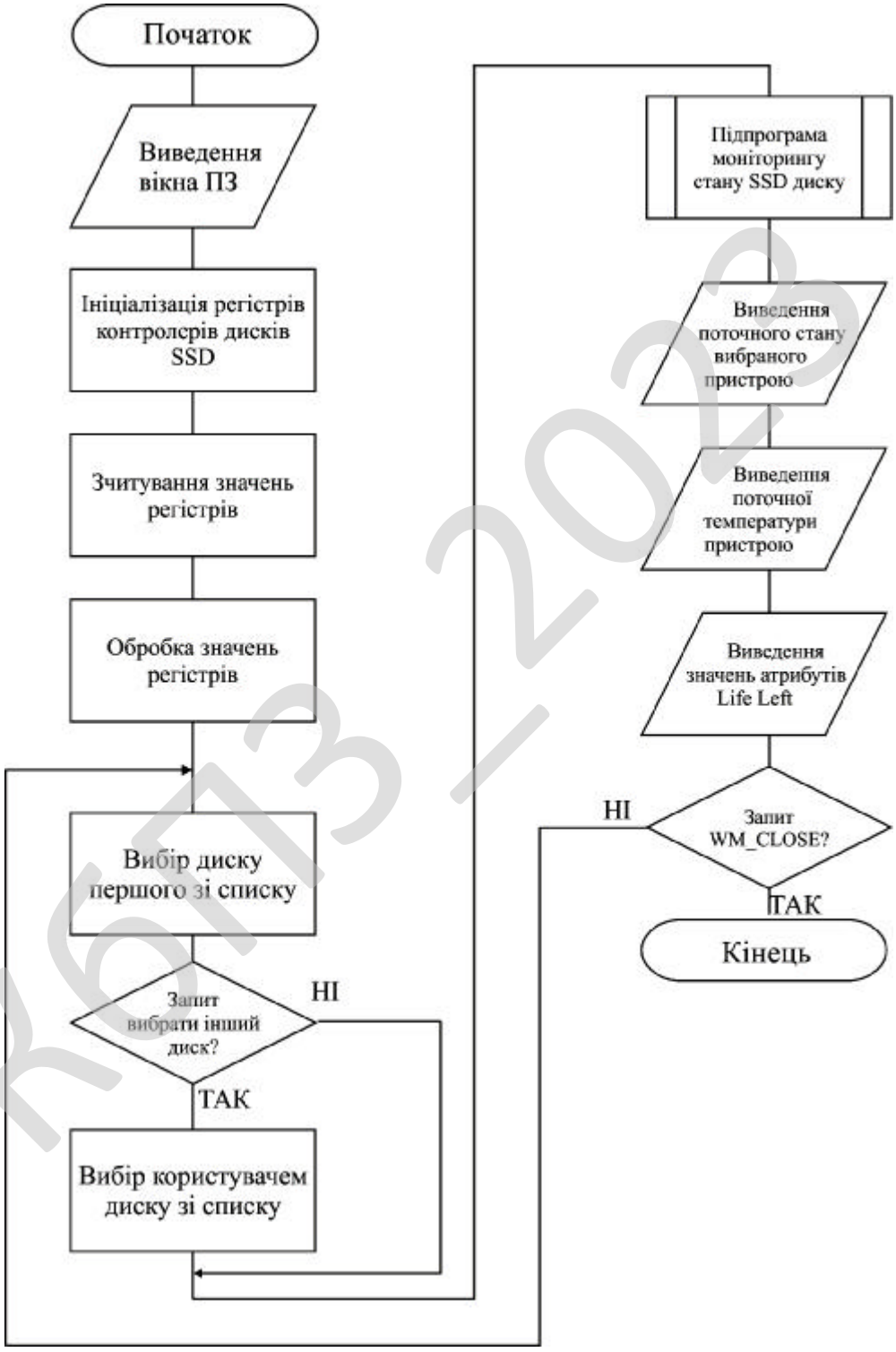

Рисунок 4.1 – Блок схема основної програми

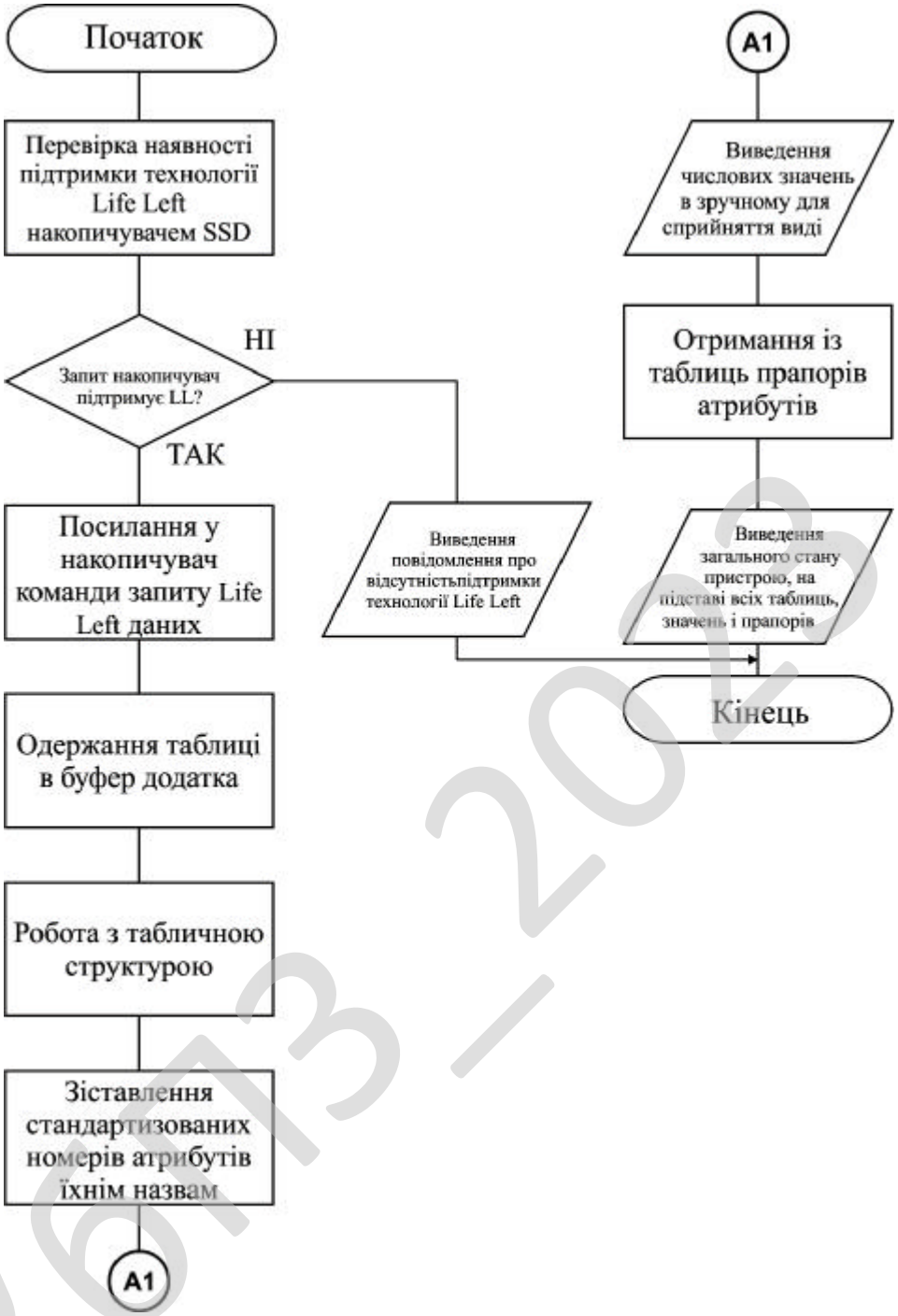

Рисунок 4.2 – Блок схема підпрограми

Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний код. Основною причиною використання мови UML є спілкування розробників між собою.

Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту.

UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об’єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки.

Практично усі CASE-засоби (програми автоматизації процесу аналізу і проектування) мають підтримку UML. Моделі розроблені в UML, дозволяють значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи.

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

– Керівникам проектів, які керують розподілом завдань і контролем за проектом.

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

– Бізнес-аналітикам, які досліджують реальну систему і здійснюють інжиніринг і реінжиніринг бізнесу компанії.

– Програмістам які реалізовують модулі інформаційної системи.

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності. Використання побудованої моделі при модифікаціях системи дає можливість усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

Опис алгоритмів функціонування системи

Опис функцій розробленого ПЗ моніторингу стану SSD диску на основі технології Life Left.

```
// читання та розшифровки таблиці атрибутів.
```

```
DWORD CAtaSmart::CAUpdateLifeLeft(DWORD index, SMART_ATTRIBUTE* pre,
SMART_ATTRIBUTE* cur)
{
     if(memcmp(pre, cur, sizeof(SMART_ATTRIBUTE) * MAX_ATTRIBUTE) == 0)
     {
          return SMART_STATUS_NO_CHANGE;
     }
     else
     {
          for(int i = 0; i < MAX_ATTRIBUTE; i++)
          {
               switch(cur[i].Id)
               {
               case 0x09: // Кількість відпрацьованих годин у включеному стані
                    {
                    DWORD preRawValue =
                    MAKELONG
                    (
                    MAKEWORD(pre[i].RawValue[0], pre[i].RawValue[1]),
                    MAKEWORD(pre[i].RawValue[2], pre[i].RawValue[3])
                    );
                    DWORD curRawValue = MAKELONG
                    (
                    MAKEWORD(cur[i].RawValue[0], cur[i].RawValue[1]),
                    MAKEWORD(cur[i].RawValue[2], cur[i].RawValue[3])
                    );
if(GetPowerOnHours(preRawValue, vars[index].DetectedTimeUnitType)
               != GetPowerOnHours(curRawValue,
               vars[index].DetectedTimeUnitType))
                    {
                    memcpy(pre, cur, sizeof(SMART_ATTRIBUTE) * MAX_ATTRIBUTE);
                    return SMART_STATUS_MAJOR_CHANGE;
                    }
if(GetPowerOnHours(preRawValue, vars[index].MeasuredTimeUnitType)
               != GetPowerOnHours(curRawValue,
               vars[index].MeasuredTimeUnitType))
               {
               memcpy(pre, cur, sizeof(SMART_ATTRIBUTE) * MAX_ATTRIBUTE);
               return SMART_STATUS_MAJOR_CHANGE;
               }
               }
               break;
```

```
                case 0x0C:
// Кількість зафіксованих повторів включення/вимикання живлення накопичувача
                    {
                    DWORD preRawValue = MAKELONG(
                    MAKEWORD(pre[i].RawValue[0], pre[i].RawValue[1]),
                    MAKEWORD(pre[i].RawValue[2], pre[i].RawValue[3])
                    );
                    DWORD curRawValue = MAKELONG(
                    MAKEWORD(cur[i].RawValue[0], cur[i].RawValue[1]),
                    MAKEWORD(cur[i].RawValue[2], cur[i].RawValue[3])
                    );
                    if(preRawValue != curRawValue)
                    {
                    memcpy(pre, cur, sizeof(SMART_ATTRIBUTE) * MAX_ATTRIBUTE);
                    return SMART_STATUS_MAJOR_CHANGE;
                    }
                    }
                    break;
                case 0xC2:
// Температура
                    if(pre[i].RawValue[0] != cur[i].RawValue[0]
                    || pre[i].CurrentValue != cur[i].CurrentValue)
                    {
                    memcpy(pre, cur, sizeof(SMART_ATTRIBUTE) * MAX_ATTRIBUTE);
                    return SMART_STATUS_MAJOR_CHANGE;
                    }
                    break;
                default:
                    break;
                }
            }
            return SMART_STATUS_MINOR_CHANGE;
        }
}
// Визначення виробника
          if(GetDiskInfo(i, -1, -1, INTERFACE_TYPE_UNKNOWN, VENDOR_UNKNOWN))
          {
               int index = (int)vars.GetCount() - 1;
               CString cmp;
               cmp = vars[index].Model;
               if(cmp.Find(_T("DW C")) == 0
// WDC
```

```
                || cmp.Find(_T("iHat")) == 0
// Hitachi
                || cmp.Find(_T("ASSM")) == 0
// SAMSUNG
                || cmp.Find(_T("aMtx")) == 0
// Maxtor
                || cmp.Find(_T("OTHS")) == 0
// TOSHIBA
                || cmp.Find(_T("UFIJ")) == 0
// FUJITSU
                )
                {
          vars[index].SerialNumber = vars[index].SerialNumberReverse;
          vars[index].FirmwareRev = vars[index].FirmwareRevReverse;
          vars[index].Model = vars[index].ModelReverse;
            vars[index].ModelSerial = vars[index].Model +
            vars[index].SerialNumber;
          vars[index].ModelSerial.Replace(_T("/"), _T(""));
          }
          }
// Розмір диску та розмір буферу
     asi.Cylinder = identify->LogicalCylinders;
     asi.Head = identify->LlogicalHeads;
     asi.Sector = identify->LogicalSectors;
     asi.Sector28 = identify->TotalAddressableSectors;
     asi.Sector48 = identify->MaxUserLba;
      asi.DiskSizeChs   = (DWORD)(((ULONGLONG)identify->LogicalCylinders *
      identify->LlogicalHeads * identify->LogicalSectors * 512) / 1000 / 1000  -
       50);
      asi.DiskSizeLba28 = (DWORD)(((ULONGLONG)identify->TotalAddressableSectors *
      512) / 1000 / 1000 - 50);
     if(asi.IsLba48Supported)
     {
      asi.DiskSizeLba48 = (DWORD)(((ULONGLONG)identify->MaxUserLba * 512) / 1000 /
       1000 - 50);
     }
     asi.BufferSize = identify->BufferSize * 512;
     if(asi.IsNvCacheSupported)
     {
          asi.NvCacheSize = identify->NvCacheSizeLogicalBlocks * 512;
     }
     if(asi.DiskSizeChs == 0)
```

```
    {
        asi.TotalDiskSize = 0;
    }
    else if(asi.DiskSizeLba48 > asi.DiskSizeLba28)
    {
        asi.TotalDiskSize = asi.DiskSizeLba48;
    }
    else if(asi.DiskSizeLba28 > asi.DiskSizeChs)
    {
        asi.TotalDiskSize = asi.DiskSizeLba28;
    }
    else
    {
        asi.TotalDiskSize = asi.DiskSizeChs;
    }
    // Перевірка помилок для контролеру External ATA
     if(asi.IsLba48Supported && (identify->TotalAddressableSectors < 268435455 &&
      asi.DiskSizeLba28 != asi.DiskSizeLba48))
    {
        asi.DiskSizeLba48 = 0;
    }
// перевірка підтримки технології Life Left
    switch(asi.CommandType)
    {
    case CMD_TYPE_PHYSICAL_DRIVE:
// РОЗШИФРУВАННЯ ЗНАЧЕНЬ
//        SendAtaCommandPd(physicalDriveId, 0xEF, 0x42, 0xFE);
//        SendAtaCommandPd(physicalDriveId, 0xEF, 0x05, 0xFE);
        if(GetSmartAttributePd(physicalDriveId, &asi))
        {
        GetSmartThresholdPd(physicalDriveId, &asi);
          asi.DiskStatus = CheckDiskStatus(asi.Attribute, asi.Threshold,
          asi.AttributeCount, asi.VendorId);
        asi.IsSmartEnabled = TRUE;
        }
        else if(ControlSmartStatusPd(physicalDriveId, ENABLE_SMART))
        {
    if(GetSmartAttributePd(physicalDriveId, &asi))
            {
    GetSmartThresholdPd(physicalDriveId, &asi);
     asi.DiskStatus = CheckDiskStatus(asi.Attribute, asi.Threshold,
     asi.AttributeCount, asi.VendorId);
```

```
     asi.IsSmartEnabled = TRUE;
          }
          }
          break;
     case CMD_TYPE_SCSI_MINIPORT:
// РОЗШИФРУВАННЯ ЗНАЧЕНЬ
SendAtaCommandScsi(scsiPort, scsiTargetId, 0xEC, 0x00, 0x00);
// ID
SendAtaCommandScsi(scsiPort, scsiTargetId, 0xEF, 0x05, 0x80);
// APM
SendAtaCommandScsi(scsiPort, scsiTargetId, 0xEF, 0x42, 0x80);
// AAM
// Підтримка функцій
/*-----------------------------------------------*/
DWORD CAtaSmart::CheckDiskStatus(SMART_ATTRIBUTE* attribute, SMART_THRESHOLD*
threshold, DWORD attributeCount, DWORD vendorId)
{
     int error = 0;
     int caution = 0;
     BOOL flagUnknown = TRUE;
     for(DWORD j = 0; j < attributeCount; j++)
     {
          if( attribute[j].Id != 0xBE
// Температура воздуха
          &&   threshold[j].ThresholdValue != 0
          &&   attribute[j].CurrentValue <= threshold[j].ThresholdValue)
          {
               error++;
          }
          switch(attribute[j].Id)
          {
          case 0x05:
// Кількість перепризначених секторів
          case 0xC4:
// Кількість операцій перепризначення (ремаппінгу).
          case 0xC5:
// Поточна кількість нестабільних секторів
          case 0xC6:
// Кількість нескоректованих помилок
               if(attribute[j].RawValue[0] == 0xFF
               && attribute[j].RawValue[1] == 0xFF
               && attribute[j].RawValue[2] == 0xFF
```

```
                && attribute[j].RawValue[3] == 0xFF)
                {
                     flagUnknown = FALSE;
                }
                else
                {
                     caution += attribute[j].RawValue[0]
                      + attribute[j].RawValue[1];
                     flagUnknown = FALSE;
                }
                break;
           case 0xBB:
// Визначення фірми-розробника
                if(vendorId == VENDOR_MTRON)
                {
                     if(attribute[j].CurrentValue == 0)
                     {
                          error = 1;
                     }
                     else if(attribute[j].CurrentValue < 10)
                     {
                          caution = 1;
                     }
                     else
                     {
                          flagUnknown = FALSE;
                     }
                }
                break;
           default:
                break;
           }
     }
     if(error > 0)
     {
          return DISK_STATUS_BAD;
     }
     else if(flagUnknown)
     {
          return DISK_STATUS_UNKNOWN;
     }
     else if(caution > 0)
```

```
    {
        return DISK_STATUS_CAUTION;
    }
    else
    {
        return DISK_STATUS_GOOD;
    }
}
```

Опис типів атрибутів

Кожний атрибут може мати деякий набір прапорів, що визначають його функціональні особливості. Нижче приводяться всі шість основних типів і їхні короткі описи:

– Pre-failure (PF). Якщо атрибут має цей тип, то поле threshold атрибута містить мінімально припустиме значення атрибута, нижче якого не гарантується працездатність накопичувача й різко збільшується ймовірність його виходу з ладу.

– On-line collection (OC). Указує, що значення даного атрибута обновляється (обчислюється) під час виконання on-line тестів S.M.A.R.T. або ж під час обох видів тестів (on-line/off-line). У противному випадку, значення атрибута обновляється тільки при виконанні off-line тестів.

– Performance related (PR). Указує на те, що значення цього атрибута прямо залежить від продуктивності накопичувача за окремими показниками (seek/throughput/etc. performance). Звичайно обновляється після виконання self-test'ов SMART.

– Error rate (ER). Указує на те, що значення атрибута відбиває відносну частоту помилок по даному параметрі (raw read/write, seek, etc.).

– Events count (EC). Указує на те, що атрибут є лічильником подій.

– Self-preserve (SP). Указує на те, що значення атрибута обновляється й зберігається автоматично (звичайно при кожному старті накопичувача й при виконанні тестів SMART).

Застосування атрибутів

– * – (використовується в програмі HDD Speed) – даний покажчик показує, що відповідний атрибут Life Left є критичним для нормального функціонування накопичувача.

– Raw Read Error Rate – Частота появи помилок при читанні даних з диска.

– Throughput Performance – Середня продуктивність (пропускна здатність) диска.

– Spin Up Time – Час розкручування шпинделя.

– Start/Stop Count – Кількість циклів запуск/останов шпинделя.

– Reallocated Sectors Count – Кількість перепризначених секторів.

– Read Channel Margin – Запас каналу читання.

– Seek Error Rate – Частота появи помилок позиціонування блоку магнітних головок (БМГ).

– Seek Time Performance – Середня продуктивність операцій позиціонування БМГ.

– Power-On Hours – Кількість відпрацьованих годин у включеному стані.

– Spin Retry Count – Кількість повторів спроб старту шпинделя диска.

– Recalibration Retries – Кількість повторів спроб рекалібровки накопичувача.

– Device Power Cycle Count – Кількість повних циклів запуску/останова жорсткого диска.

– Soft Read Error Rate – Частота появи "програмних" помилок при читанні даних з диска.

– Load/Unload Cycle Count – Кількість циклів виводу БМГ у спеціальну парковочну зону/у робоче положення.

– Temperature – Температура.

– Reallocation Event Count – Кількість операцій перепризначення (ремаппінгу).

– Current Pending Sector Count – Поточна кількість нестабільних секторів.

– Uncorrectable Sector Count – Кількість нескоректованих помилок.

– UltraDMA CRC Error Count – Загальна кількість помилок CRC у режимі UltraDMA.

– Write Error Rate (Multi Zone Error Rate) – Частота появи помилок при записі даних.

– Disk Shift – Зрушення пакета дисків щодо осі шпинделя.

– G-Sense Error Rate – Частота появи помилок у результаті ударних навантажень.

– Loaded Hours – Навантаження на привод БМГ, викликана загальним наробітком годин накопичувачем.

– Load/Unload Retry Count – Навантаження на привод БМГ, викликана численними повтореннями операцій читання, запису, позиціонування головок і т.п.

– Load Friction – Навантаження на привод БМГ, викликане тертям у механічних частинах накопичувача.

– Load/Unload Cycle Count – Загальна кількість циклів навантаження на привод БМГ.

– Load-in Time – Загальний час навантаження на привод БМГ.

– Torque Amplification Count – Кількість зусиль обертаючого моменту привода.

– Power-Off Retract Count – Кількість зафіксованих повторів включення/вимикання живлення накопичувача.

– GMR Head Amplitude – Амплітуда тремтіння головок (GMR-head) у робочому стані.

4.2 Захист розробленого програмного забезпечення

Tiny Encryption Algorithm (TEA) [1] – блочний алгоритм шифрування типу «Мережі Фейстеля». Алгоритм був розроблений на факультеті комп'ютерних

наук Кембриджського університету Девідом Вілером (David Wheeler) і Роджером Нідгемом (Roger Needham) та вперше представлений в 1994 році [2] на симпозіумі зі швидкими алгоритмами шифрування в Льовені (Бельгія).

Шифр не патентований, широко використовується в ряді криптографічних додатків і широкому спектрі апаратного забезпечення, завдяки вкрай низькими вимогами до пам'яті й простоті реалізації. Алгоритм має як програмну реалізацію на різних мовах програмування, так і апаратну реалізацію на інтегральних схемах типу FPGA.

Захист розробленого програмного забезпечення буде відбуватися за допомогою алгоритму TEA, який заснований на бітових операцій з 64-бітним блоком, має 128-бітний ключ шифрування. Стандартна кількість раундів мережі Фейстеля біля 64 (32 циклу), однак, для досягнення найкращої продуктивності або шифрування, число циклів можна варіювати від 8 (16 раундів) до 64 (128 раундів). Мережа Фейстеля несиметрична через використання в якості операції накладення додавання по модулю 232.

Перевагами шифру є його простота в реалізації, невеликий розмір коду й досить висока швидкість виконання, а також можливість оптимізації виконання на стандартних 32-бітних процесорах, так як в якості основних операцій використовуються операції виключна «АБО» (XOR), побітового зсуву й додавання по модулю 232. Оскільки алгоритм не використовує таблиць підстановки і раундова функція досить проста, алгоритму потрібно не менше 16 циклів (32 раундів) для досягнення ефективної дифузії, хоча повна дифузія досягається вже через 6 циклів (12 раундів).

Алгоритм має відмінну стійкість до лінійного криптоаналізу і досить гарну до диференціального криптоаналізу. Головним недоліком цього алгоритму шифрування є його вразливість до атак «на пов'язаних ключах» (англ. Related-key attack). Через простий розклад ключів кожен ключ має 3 еквівалентних ключа. Це означає, що ефективна довжина ключа складає всього 126 біт [3] [4], тому даний алгоритм не слід використовувати в якості геш-функції.

Опис алгоритму

Вихідний текст розбивається на блоки по 64 біта кожен. 128-бітний ключ K ділиться на чотири 32-бітних підключа K [0], K [1], K [2] і K [3]. На цьому підготовчий процес закінчується, після чого кожен 64-бітний блок шифрується протягом 32 циклів (64 раундів) за нижченаведеним алгоритмом. [5]

Припустимо, що на вхід n-го раунду ($1 \leq n \leq 64$) надходять права й ліва частини (L_n, R_n), тоді на виході n-го раунду будуть ліва й права частини (L_{n+1}, R_{n+1}), які обчислюються за такими правилами:

$$L_{n+1} = R_n.$$

Якщо $n = 2 * i - 1$ для $1 \leq i \leq 32$ (непарні раунди), то:

$$R_{n+1} = L_n (\{ [R_n 4] K [0] \} \{ R_n i * \delta \} \{ [R_n 5] K [1] \}).$$

Якщо $n = 2 * i$ для $1 \leq i \leq 32$ (парні раунди), то:

$$R_{n+1} = L_n (\{ [R_n 4] K [2] \} \{ R_n i * \delta \} \{ [R_n 5] K [3] \}).$$

Де

X Y – операція додавання чисел X і Y по модулю 232.

X Y – побітове виключне АБО» (XOR) чисел X і Y, яке в мові програмування Сі позначається як X ^ Y

X Y і X Y – операції побітового зсуву числа X на Y біт вліво й вправо відповідно.

Константа δ була виведена з Золотого перерізу:

$$\delta = (-1) * 2^{31} = 2654435769 = 9E3779B9_h.$$

У кожному раунді константа множиться на номер циклу i. Це було зроблено для запобігання простих атак, заснованих на симетрії раундів.

Також очевидно, що в алгоритмі шифрування TEA немає як такого алгоритму розкладу ключів. Замість цього в непарних раундах використовуються підключі K [0] та [1], у парних – K [2] і [3].

Так як це блочний шифроалгоритм, де довжина блоку 64-біт, а довжина даних може бути не кратна 64-біт, значення всіх байтів, які доповнюють блок до кратності в 64-біт, встановлюється в 0x01.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

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

1. Розділ Меню:

– Файл – де відбуваються типові операції.

– Диски – де обираються диски з якими відбувається робота.

– Функції – де визначені усі функції, які може реалізувати програма.

– Параметри – де визначені усі параметри, які може задати користувач.

– Довідка – де знаходяться дані про роботу програми та розробника програми.

2. Розділ функціональних полів: Поле вибору диску; Поле виводу версії прошивки; Поле виводу серійного номеру; Поле виводу типу інтерфейсу; Поле виводу стандарту; Поле виводу виду передачі; Поле виводу опції; Поле виводу об’єму буфера; Поле виводу об’єму NV-кешу; Поле виводу 28bit LBA; Поле виводу 48bit LBA; Поле виводу кількості підключень; Поле виводу загального часу роботи.

Рисунок 5.1 – Головне вікно програми

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

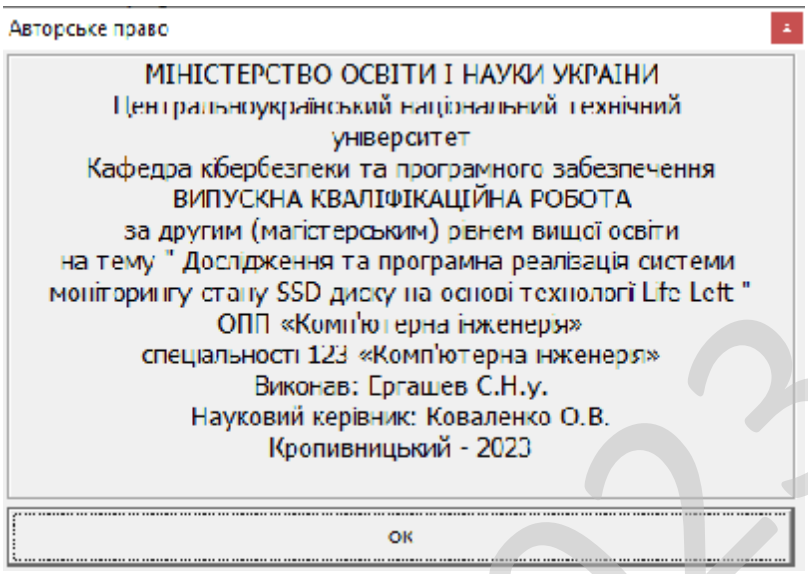

Рисунок 5.2 – Авторське право

Обрано умови розповсюдження – Shareware. Під умовно-безплатним програмним забезпеченням можна розуміти спосіб або метод розповсюдження комерційного ПЗ на ринку (тобто на шляху до кінцевого користувача), при якому випробувачеві пропонується обмежена за можливостями (не повнофункціональна або демонстраційна версія), терміном дії (тріал версія) або версія з вбудованим набридливим нагадуванням про необхідність оплати використання програми. В угоді про використання (ліцензії для кінцевого користувача, EULA) також може бути обумовлена заборона на комерційне або професійне (не тестове) її використання. Основний принцип умовно-безплатного ПЗ – «спробуй, перш ніж купити» (try before you buy). ПЗ що поширюється як умовно-безплатний, надається користувачам безоплатно. Звичайно користувач платить тільки за час завантаження файлів через Інтернет або за носій (CD диск, флешку, ключ). Протягом певного терміну, що становить зазвичай тридцять днів, він може користуватися програмою, тестувати її, освоювати її можливості.

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи моніторингу стану SSD диску на основі технології Life Left.

Метою розробки є дослідження та програмна реалізація системи моніторингу стану SSD диску на основі технології Life Left.

Об'єктом дослідження є процес моніторингу стану SSD диску на основі технології Life Left.

Предметом дослідження є методи моніторингу стану SSD диску на основі технології Life Left.

Методи дослідження базуються на методах архітектури комп'ютерів, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод моніторингу стану SSD диску на основі технології Life Left.

– Розроблено вітчизняний продукт моніторингу стану SSD диску на основі технології Life Left, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

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

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 24 днів (один місяць).

В магістерській роботі було проведено дослідження та виконана програмна реалізація системи моніторингу стану SSD диску на основі технології Life Left.

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

а) невеликий розмір;

б) невеликі системні потреби;

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

г) зручність у користуванні та надійність.

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт.	N	1
2. Кількість екземплярів програм, шт.	Ne	320
3. Запланований термін розробки, днів	Fpq	24 (1 місяць)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

1	2	3
7. Кількість макетів вхідної інформації	–	8
8. Кількість форм вихідної інформації.	–	6
9. Мова програмування (1-6)	–	2
10. Попередній досвід (1-6)	–	3
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	–	1
13. Рівень спрацьованості колективу (1-6)	–	2
14. Ступінь вимірності процесів (1-6)	–	3
15. Необхідна надійність програмного забезпечення (1-6)	–	3
16. Розмір бази даних (порівняно з розміром програми) (1-6)	–	4
17. Складність кінцевого програмного продукту (1-6)	–	5
18. Необхідний рівень забезпечення повторного використання (1-6)	–	2
19. Документованість відповідно до планованого життєвого циклу (1-6)	–	3
20. Вимоги до швидкодії ПП (1-6)	–	3
21. Обмеження на розміри основного сховища даних (1-6)	–	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	–	4
23. Професійний рівень аналітиків (1-6)	–	3
24. Професійний рівень програмістів (1-6)	–	4
25. Постійність складу команди розробників (1-6)	–	2
26. Досвід розробки додатків (1-6)	–	1
27. Досвід роботи з обчислювальною платформою (1-6)	–	2

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	3
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	32000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	50
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

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

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

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

де: A – коефіцієнт Боема, A = 2,45;

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

B – показник ступеня, що визначається співвідношенням:

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

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

$$B = 1{,}01+0{,}001(2{,}43+3{,}64+4{,}22+3{,}95+2{,}73) = 1{,}027.$$

$$T_{ном} = 2{,}45 \cdot 2{,}2^{1{,}027} = 5{,}5 \text{ люд-міс.}$$

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

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

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

$$T_{уточн} = 5{,}5 \cdot (1 \cdot 1{,}09 \cdot 1{,}30 \cdot 0{,}91 \cdot 1 \cdot 1 \cdot 1 \cdot 1{,}15 \cdot 1 \cdot 0{,}87 \cdot 1{,}10 \cdot$$

$$\cdot 1{,}22 \cdot 1{,}12 \cdot 1{,}10 \cdot 1 \cdot 1 \cdot 1{,}10) = 12{,}9 \text{ люд-міс.}$$

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

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

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

де: *C* – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4);

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

$$T_{РП} = 0{,}3 \cdot 2{,}66 \cdot 12{,}9^{0{,}33+0{,}2(1{,}027-1{,}01)} \cdot 50 = 94 \text{ люд/день.}$$

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

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	15	Д7
Робочий проект	94	Ф 7.1- 7.4
Впровадження	15	Д13
Всього	143	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

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

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

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

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

$$Ч = \frac{143 \cdot 1}{24\text{-}3} = 6,8 \text{ ставки.}$$

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

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

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

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	10	900	15
Монітор	60	10	600	10
Клавіатура	30	10	300	5
Маніпулятор «мишка»	30	10	300	5
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	2	240	4
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор-маршрутизатор	30	2	60	1
Кабельні господарства ЛОМ на 1 м.п.	2,5	300	750	12,5
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	56,16

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

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

$$Ф_{др}^{с} = \frac{56 \cdot 1}{1,2} = 47 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

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

$$Ч_{ел} = 47/(24 \cdot 8) = 0{,}25 \text{ ставки.}$$

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

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

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

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 2012 R2, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN, PPPoE, Frame Relay, Wi-Fi	2	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

Посада	Вид роботи	Час	К-ть штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	2	0,5
	Підтримка постійних клієнтів	1	
	Оформлення договорів, ведення тендерів	0,5	
	Контроль взаєморозрахунків з постачальниками	0,5	
Всього		4	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	2	0,5
	Створення графічних і стилістичних елементів сайту	1	
	Оформлення банерів і промо-сторінок	0,5	
	Розміщення графіки і контенту на Інтернет сторінках	0,5	
Всього		4	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,25
	Верстка друкованих видань	0,5	
	Додрукова підготовка макетів	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	0,25	21000	5250
Продакт-менеджер	0,5	17136	8568
Інженер-програміст	6,8	19000	129200
Інженер-електронщик	0,25	16000	4000
Інженер-системотехнік	0,25	16000	4000
Адміністратор мережі	0,5	19000	9500
Системний програміст	0,25	16000	4000
Дизайнер WEB	0,5	16000	8000
Інженер-верстальник	0,25	18000	4500
Бухгалтер-економіст	0,5	15000	7500
Всього за період розробки	$R_{сп}$ = 10,05	-	$Ф_{роб}$ = 184518

Розрахуємо середньоденну зарплату одного виконавця:

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

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

$$З_{сд} = \frac{184518}{10,05 \cdot 24} = 765 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

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

$$Б_{уд} = R_{сп}^{1} S_{у} Ц_{пл}, \quad (7.9)$$

де: $R_{сп}^{1}$ – кількість робочих місць виконавців, шт. Приймаємо 13 робочих місць;

$S_{у}$ – питома площа на одне робоче місце, $м^2$;

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

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

$$Б_{уд} = 13 \cdot 8 \cdot 20000 = 2080000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 208000 грн.

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

$$I_{нв} = R_{сп}^{1} \cdot Ц_{м}, \quad (7.10)$$

де: $Ц_{м}$ – ціна меблів для одного робочого місця, грн.

$$I_{нв} = 13 \cdot 3500 = 45500 \text{ грн.}$$

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

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались за пропозицією інтернет ресурсу hotline за 24.10.23 – джерело https://hotline.ua

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		11457
Системний блок		7509
Процесор	Intel Core i3-8145U (2 (4) ядра по 2.1 - 3. GHz), 4 MB Smart Cache	-
Системна плата	Intel SoC platform, NUC8i3PN, 3x USB 3.1 2x USB 2.0, 1x USB Type-C, 2x Audio, 1 LAN (RJ-45), 1x HDMI, 1x DisplayPort	-
Відеокарта	Інтегрована Intel UHD Graphics 610 (до 179 МВ з ОЗП)	-
Жорсткий диск	SSD 240 Gb	-
Оперативна пам’ять	8 GB DDR4	-
DVD-привод	-	-
Корпус	Lenovo ThinkCentre M630e Tiny USFF, 65W	-
Кулер	–	–
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-E int. 3.5", 1*USB2.0+AUDIO+1394, multi: A Type Cards, black	-
інше	Клавіатура, мишка	Подару-нок
Монітор	22" TFT, ASUS VW223D (5ms, 300/3000: 1 170/160, D-SUB, Wide)	2600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Сканер	Epson Perfection V37 Photo	2970
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965
Пристрій безперебійног живлення	UPS APC BACK-UPS ES 525VA 230V RUSSIA (BE525-RS)	1348

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

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп’ютери	13	11457	14894,1	163835,1
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	1	2970	297	3267
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	185653,6

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	2080000	-	-
2. Передавальні пристрої	208000	-	-
Всього по групі	2288000	5	114400
Група 4			
3. Обчислювальна техніка	185654	-	-
Всього по групі	185654	50	92827
Група 5, 6			
4. Вимірювальні пристрої	3999	25	1000
5. Транспортні засоби	97500	20	19500
6. Господарський інвентар	45500	25	11375
Всього по групі	146999	-	31875
7. Нематеріальні активи	32000	10	3200
Разом	$К_р$ = 2652653		$А_р$ = 242302

Примітка: вартість автомобіля взята по даним з прайс-листа автосалону автотрейдінг, вкладки автобазар, джерело http: //www.auto-trading.com.ua/sale/lot20772.html, складає 97500 грн.

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

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

де: $N_е$ – кількість екземплярів програм, шт.

$$З_о = 765 \cdot 143/320 = 342 \text{ грн.}$$

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

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

де: $Н_q$ – норматив додаткової зарплати, %.

$$З_д = 342 \cdot 10 \cdot 0{,}01 = 34 \text{ грн.}$$

Відрахування на соціальні потреби за нормативом $Н_с = 22\%$ від суми основної та додаткової зарплати:

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

де: $Н_с$ – відрахування на соціальні потреби, %.

$$С_{оц} = 0{,}01 \cdot 22(342+34) = 83 \text{ грн.}$$

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

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

де: $Н_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 342 \cdot 15 \cdot 0{,}01 = 51 \text{ грн.}$$

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

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

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

Згідно прийнятих норм на підприємстві $n_{вит}$ приймаємо 0,2 пачек паперу на період розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=200 грн., визначаємо вартість паперу за період розробки:

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

$$З_{М1} = 200 \cdot 0{,}2 = 40 \text{ грн.}$$

Згідно прийнятих норм по комплектації до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків. Їх кількість дорівнює кількості коробочних версій запропонованого продукту (приймаємо 100):

$$З_{М2} = \sum Ц_д\ , \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR box – 23 грн./шт., DVD-R box – 32,15 грн./шт.

$$З_{М2} = 32{,}15 \cdot 100 = 3215 \text{ грн.}$$

Згідно виданих викладачем норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_з., \quad (7.18)$$

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

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

$$З_М = (40 + 3215 + 1702)/320 = 15 \text{ грн.}$$

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

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

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

$$О_п = 342 \cdot 15 \cdot 0{,}01 = 51 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 320 прим.):

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

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

$$A_м = 242302 \cdot 1/(320 \cdot 12) = 63 \text{ грн.}$$

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

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

$$C_п = 342+34+83+51+15+51+63 = 639 \text{ грн.}$$

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

Для даного програмного забезпечення рівень рентабельності складає 50%.

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

де: $P_п$ – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 50 \cdot 639 = 320 \text{ грн.}$$

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

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн
1	2	3
1. Основна зарплата виконавців	$З_о$	342
2. Додаткова зарплата виконавців	$З_д$	34
3. Відрахування на соціальні потреби	$С_{оц}$	83
4. Загальногосподарські витрати	$Г_{осп}$	51
5. Витрати на матеріали	$З_М$	15
6. Освоєння нових операційних систем, мов програмування	$О_п$	51

Продовження таблиці 7.9

1	2	3
7. Амортизація основних фондів	$А_м$	63
8. Повна собівартість програмного забезпечення	$С_п$	639
9. Плановий прибуток	$П_р$	320
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	959
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	191,8
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	1150,8

7.6 Визначення об'єму капітальних вкладень у споживача програмної продукції

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

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

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	1151
Всього капітальних витрат	–	1151

7.7 Визначення експлуатаційних витрат

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

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на технічне обслуговування	$З_р$	25498	19800
2. Витрати на електроенергію	$З_{ел}$	0	0
3. Витрати на амортизацію	$З_{ам}$	0	576
Всього витрат за рік	$І$	25498	20376

Витрати на профілактичні роботи:

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

де: $Т_р$ – кількість годин обслуговування кожного комп'ютера за рік, *год.*;

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

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 190 годин на рік до 150 годин на рік, тому витрати на технічне обслуговування зменшилися з:

$$З_{р\,баз} = 190 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 25498 \text{ грн},$$

до:

$$З_{р\,нов} = 150 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 19800 \text{ грн}.$$

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

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

$$З_{ел\ баз} = З_{ел\ нов}$$

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

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	50	–	1151	–	575,5
Всього відрахувань	-	–	1151	–	575,5

7.8 Визначення економічної ефективності програмної продукції

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

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою:

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

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

$$Ев = (959 - 639) \cdot 320 - (0,05 \cdot 2288000 + 0,5 \cdot 185654 +$$
$$+0,25 \cdot 49499 + 0,2 \cdot 97500 + 0,1 \cdot 32000) \cdot 1/12 = 82208\ грн.$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

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

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

$$T_в = \frac{364653}{(959-639)\cdot 320\cdot 12/1} = 0{,}3 \ року .$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	320
2. Повна собівартість розробленої програми	Грн.	639
3. Ціна розробленої програми	Грн.	959
4. Плановий прибуток від реалізації розробленої програми	Грн.	320
5. Рентабельність програмної продукції	%	50
6. Об’єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	2652653
7. Загальний прибуток від реалізації програмної продукції	Грн.	102400
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	82208
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років.	0,3
10.Об’єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	1151
11.Величина економічного ефекту у користувача програмної продукції	Грн.	4547
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,22

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

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

де: $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно;

$K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються.

$$E_{сп} = (25498\text{-}20376)\text{-}0{,}5 \cdot 1151 = 4547 \text{ грн.}$$

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

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

$$T_{сп} = \frac{1151}{(25498\text{-}20376)} = 0{,}22 \text{ року}.$$

7.9 Висновки

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

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Загальна комп'ютеризація суспільства призвела до того, що використання комп'ютерів стало повсюдним у всіх сферах економіки та народного господарства. Застосування персональних комп'ютерів і ЕОМ дозволило значно підвищити продуктивність праці, змінити характер і зміст праці [4]. Комп'ютеризація, поряд з незаперечними перевагами, тягне за собою і багато проблем. Для того, щоб активне застосування комп'ютерних технологій не стало додатковим чинником погіршення здоров'я, вкрай необхідно щоб робоче місце відповідало гігієнічним вимогам. Темою дипломного проекту є розробка та дослідження та реалізація програмного продукту, тому актуально буде розгляд умов праці програміста.

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

В кожній ІТ компанії є трудові відносини з працівниками. Згідно закону України "Про охорону праці" [3] кожна компанія впроваджує заходи з охорони праці. Реалізується трудові відносини з вживанням необхідних засобів з охорони праці та розробки відповідних документів:

– Інструкцій з охорони праці по кожній професії і загальні.

– Положення про охорону праці.

– Накази з охорони праці.

– Журнали реєстрації та інструктажу.

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

За недотриманням вимог, керівники ІТ компаній можуть бути притягнуті до відповідальності, яка виглядає у виді накладання штрафу. Якщо в результаті порушення умов охорони праці є постраждалі працівники то керівні особи ІТ компаній притягуються до кримінальної відповідальності.

Законом України “Про охорону праці” [3] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» [5], яким затверджено нормативно-правовий акт з охорони праці НПАОП 0.00-7.15-18, «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98.

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

Ці шкідливі фактори можуть привести до професійних захворювань.

Розглянемо шкідливі чинники роботи програмістів керуючись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98 [5], та «Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» НПАОП 0.00-7.15-18.

Умови праці програміста включають наступні фактори:

– параметри повітряного середовища в приміщенні;

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

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

8.2 Аналіз умов праці на робочому місці програміста

Робота програміста пов'язана з постійною роботою на ЕОМ, яка відбувається у кімнаті розмірами 5м×7,2м×2,8м. Одна з її більших стін має шість двостулкових вікон, розмірами 2,2м×1,8м, які виходять на північний схід. Вікна розташовані рівномірно по всій довжині стіни. Підлога в кімнаті покрита леноліумом, всі стіни пофарбовані світло оранжевого кольору до висоти 2,8м, а далі підвісна стеля. Уздовж стін розташовані комп'ютерні столи. На них розташовуються 2 персональні комп'ютери й інша оргтехніка (сканер принтери, телефони й ксерокс). Столи мають пластикове покриття. Габарити їхньої робочої поверхні 1250 мм×850 мм. Висота столів 750 мм. Висота стільців від рівня підлоги становить 430 мм.

Згідно НПАОП 0.00 – 1.28 – 10 «Правила охорони праці під час електронно-обчислювальних машин» площа повинна задовольняти умові – не менш 6 $м^2$ на одне робоче місце. Кратність повітрообміну в приміщенні вузла також регламентується ДСанПіН 3.3.2.007 – 98, вона повинна становити 20 $м^3$/годину на одне місце. Виконання даних вимог забезпечить підтримку в приміщенні вузла оптимального значення вологості й складу повітря.

Відповідно ДБН В.2.5 – 28 – 2006 роботу програміста можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм)

V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення вузла можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при сполученому висвітленні), повинен становити 0,5%, освітленість при штучному висвітленні повинна становити 300 лк.

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

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

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

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

Таблиця 8.1 – Допустимі спектри рівнів звукового тиску

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

Таблиця 8.2 – Відмінності реальних параметрів робочого місця від параметрів відповідні вимоги нормативного акту

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

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

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

У приміщеннях, де встановлені комп'ютери, повинні дотримуватися певні параметри мікроклімату. У санітарних нормах ДСН 3.3.6.042 – 99 встановлені величини параметрів мікроклімату, що створюють комфортні умови. Ці норми встановлюються в залежності від пори року, характеру трудового процесу і характеру виробничого приміщення (див. табл. 8.3).

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

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

8.3 Розробка заходів з умов поліпшення охорони праці

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

Різними фірмами в сукупності розроблено понад 11 схем регулювань параметрів робочого крісла, які забезпечують: плавне переміщення сидіння по

висоті за допомогою газової пружини; плавна зміна нахилу спинки і сидіння; регулювання пружинного противотиску спинки крісла на спину оператора; перестановку спинки по висоті; зміна глибини сидіння шляхом зміни вигину краю сидіння; синхронне повторення рухів оператора сидінням і спинкою в правильному кутовому співвідношенні; синхронне повторення спинкою крісла рухів верхньої частини тулуба того, що сидить; амортизацію сидіння.

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

8.4 Розрахункова частина

Проведемо розрахунок штучного освітлення за методом коефіцієнта використання світлового потоку для приміщення ширина якого складає 5 м, довжина – 7,2 м, висота – 2,8 м.

У зазначеному приміщенні працює 5 людей.

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

$$F=ESKZ/n,$$

де: F – світловий потік, що розраховується, Лм;

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

S – площа освітлюваного приміщення (у нашому випадку S=5 х 7,2 = 36 $м^2$);

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

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

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

Обчислимо індекс приміщення за формулою:

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

де: *S* – площа приміщення, *S* = 36 *м*^{2}*;*

h – розрахункова висота підвісу, *h = 3 м (співпадає з в*исотою стелі, т.я. лампи освітлення закріплюються на стелі)*;*

A – ширина приміщення, *A = 5 м;*

B – довжина приміщення, *B =* 7,2 *м.*

Підставимо всі значення у формулу та визначемо індекса приміщення: i=1,4.

Знаючи індекс приміщення, за знаходимо *n = 0,29 (з табличних даних коефіцієнтів використання світлового потоку (n) світильників з відповідним типом лампам).* Підставимо всі значення у формулу, визначемо світловий потік: *F=77478 Лм.*

Для розрахунку дудемо використовувати *світлодіодні панелі LED панель 42Вт 6000К SUNLED 000000127*, світловий потік яких $F_л$ *= 3990 Лм.*

Число ламп визначається по формулі:

$$N=F/F_л$$

де: F – світловий потік,

$F_л$ – світловий потік однієї лампи.

Підставимо всі значення у формулу та визначемо індекса приміщення: $N=$

$77478/ 3990=19,4$ *шт.*

Приймаємо необхідну кількість *світлодіодних світильників* 20 шт.

8.5 Висновки до розділу

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

З цих міркувань було здійснено аналіз умов праці, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Виконано розрахунок штучного освітлення, як одного з ключових факторів впливу на працездатність та здоров'я програміста. Розроблено заходи з умов поліпшення охорони праці.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи моніторингу стану SSD диску на основі технології Life Left.

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

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів моніторингу стану SSD диску на основі технології Life Left.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем моніторингу стану SSD диску на основі технології Life Left.

– Досліджена система моніторингу стану SSD диску на основі технології Life Left.

– На основі отриманих результатів досліджень створена програмна реалізація системи моніторингу стану SSD диску на основі технології Life Left.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання моніторингу стану SSD диску на основі технології Life Left.

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

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

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

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

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм TEA.

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

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 4547 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,22 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ергашев С.Н.у. Дослідження та програмна реалізація системи моніторингу стану SSD диску на основі технології Life Left // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2023.

2. Adam Freeman. Pro Go The Complete Guide to Programming Reliable and Efficient Software Using Golang. Apress Media. 2022. 1078 p.

3. Fernando Doglio. Skills of a Successful Software Engineer. Manning. 2022. 182 c.

4. M. Holmes He. Creating Apps with React Native. Apress Media. 2022. 445 p.

5. Maurício Aniche. Effective Software Testing. Manning Publications. 2021. 372 p

6. Priscila Heller. Automating Workflows with GitHub Actions. Packt Publishing. 2021. 216 p.

7. JJ Geewax. API Design Patterns. Manning Publications Co. 2021. 481 p.

8. Prateek Prasad. App Design Apprentice. Razeware LLC. 2020. 272 p.

9. Dawn Griffiths, David Griffiths. Head First Android Development. O'Reilly Media, Inc. 2021. 1414 p.

10. Nathan Metzler. Kotlin Programming for Beginners. Independently published. 2021. 158 p.

11. Aaron Torres. Go Programming Cookbook Second Edition. Packt Publishing Ltd. 2019. 427 p.

12. Мелешко Є.В., Якименко М.С., Поліщук Л.І. Алгоритми та структури даних: Навчальний посібник для студентів технічних спеціальностей денної та заочної форми навчання. – Кропивницький: Видавець – Лисенко В.Ф., 2019. – 156 с.

13. Knuth D. The Art of Computer Programming, Vol. 1: Fundamental Algorithms, 3rd Edition 3rd Edition. – Addison-Wesley Professional, 2019. – 672 p.

14. Knuth D. The Art of Computer Programming: Vol. 3: Sorting and Searching 2nd Edition, Kindle Edition. – Addison-Wesley Professional, 2019. – 800 p.

15. Knuth D. Art of Computer Programming, Vol. 2: Seminumerical Algorithms 3rd Edition, Kindle Edition. – Addison-Wesley Professional, 2019. – 672 p.

16. Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. Introduction to Algorithms, 3rd Edition (The MIT Press) 3rd Edition – The MIT Press, 2019. – 1292 p.

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

18. Smirnov, O., Karapetyan, A., Fedorov, E., «Creating Neural Network and Single Solution Human-Based Metaheuristic Methods of Solving the Traveling Salesman Problem». CEUR Workshop Proceedings, Volume 3312, 2022, pp. 47-58.

19. Smirnov O., Kuznetsov A., Kryvinska N., Kiian A., Kuznetsova K. «Full Non-Binary Constant-Weight Codes». SN Computer Science, Vol 2, 337, 2021. https://doi.org/10.1007/s42979-021-00739-w.

20. Smirnov O., Kovalenko O., Kovalenko A., Kavun S. «Quantitative Risk Assessment Method Development in the Context of the SDLC-model». 2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T), 2021, pp. 203-208, doi: 10.1109/PICST54195.2021.9772143

21. 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.

22. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А., Коваленко А.С. «Дослідження нормативних документів та галузевих стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС,

важливих для безпеки». Системи управління, навігації та зв'язку, 2023, вип. 2(72), С. 170-178.

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

24. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А. «Дослідження нормативної документації та стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». VI міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології", м. Кропивницький. 20-21 квітня 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 35-36.

25. Смірнов, О.А., Усік П.С., Полігенько О.О., Одарченко Р.С., Терещенко Л.Ю. «Інформаційна технологія та програмне забезпечення для підвищення ефективності планування підсистеми базових станцій стільникового зв'язку». Проблеми телекомунікацій. № 1(26). С. 83-96. 2020.

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

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

28. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». Центральноукраїнський науковий вісник. Технічні науки. № 2(33). с. 161-172, 2019.

29. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019:

Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

30. 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.

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

32. O. Smirnov, O. Kovalenko, A. Kovalenko, S. Smirnov, V. Vialkova. The mathematical model of the testing technology for DOM XSS vulnerabilities. Scientific & practical cyber security journal (SPCSJ) Vol 2 Issue 1, 22-28 pp. [Электронный Журнал]. Georgia. Tbilisi: SCSA – 2018.

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

34. Смірнов О.А., Коваленко О.В., Коваленко А.С., Смірнов С.А. Розробка методу передтестової компіляції й розподілу доступу. Збірник наукових праць III міжнародної науково-практичної конференції “Інформаційна безпека та комп’ютерні технології”, м. Кропивницький. 19-20 квітня 2018р. – Кропивницький: ЦНТУ. – 2018. – С. 214-215

35. Smirnov Oleksii, Kovalenko Oleksandr, Kovalenko Anna, Smirnov Serhii. Method of testing the DOM XSS vulnerability. International Conference «Information technologies, systems and networks ITSN-2017». Chisinau, Republic of Moldova. 17 – 18 October 2017. – Chisinau: Academy of Sciences of Moldova, Military Academy of Armed Forces “Alexandru cel Bun”. 2017. P7.

36. Смірнов О.А., Смірнов С.А., Коваленко О.В., Коваленко А.С. Технологія тестування DOM XSS уразливості. Науково-практичний журнал кібер безпеки (SPCSJ) № 1. [Електронний журнал]. Грузія. Тбілісі: SCSA - 2017.

37. Смірнов О.А., Лисенко І.А. Інформаційна технологія проектування тестових наборів з урахуванням вимог до програмного забезпечення. Системи управління, навігації та зв'язку. – Випуск 4 (44). - Полтава: ПолтНТУ. - 2017. - С. 112-115.

38. Смірнов О.А., Смірнов С.А., Рябой Д.К., Рябая О.В. Модель вузла комутації з відносними пріоритетами, резервуванням ресурсів і обліком реальної надійності обслуговуючих приладів .Збірник тез всеукраїнської науково-практичної інтернет-конференції «Автоматика та комп'ютерно-інтегровані технології у промисловості, телекомунікаціях, енергетиці та транспорті». м. Кропивницький. 16-17 листопада 2017 р. – Кропивницький: ЦНТУ. – 2017. – С. 198-199.

39. Смірнов О.А., Коваленко О.В. Використання псевдобулевих методів бівалентного програмування для управління ризиками розробки програмного забезпечення. Системи управління, навігації та зв'язку. – Випуск 1 (37). - Полтава: ПолтНТУ. - 2016. - С. 98-103.

40. Смірнов О.А., Лисенко І.А. Формалізація процесу проектування тестових наборів. Збірник наукових праць Харківського університету Повітряних Сил. Випуск 3 (48). - Харків: ХУПС. - 2016. - С.96-100.

41. Смірнов О.А., Лисенко І.А. Удосконалення методу перевірки коректності таблиць рішень для подання тестових наборів. Збірник наукових праць "Системи обробки інформації". - Випуск 8 (145). - Х.: ХУПС - 2016. - С. 77-80.

42. Смірнов О.А., Лисенко І.А. Розробка впорядкованих каскадних таблиць рішень із використанням матриць слідування. Збірник наукових праць "Системи обробки інформації". - Випуск 6 (143). - Х.: ХУПС - 2016. - С. 216-220.

43. Смірнов О.А., Коваленко О.В., Якименко Н.М., Доренський О.П. Метод кількісної оцінки ризиків розроблення програмного забезпечення. Збірник наукових праць Харківського університету Повітряних Сил. Випуск 2 (47). - Харків: ХУПС. - 2016. - С. 128-133.

44. Смірнов О.А., Коваленко О.В., Якименко Н.М., Доренський О.П. Метод якісного аналізу ризиків розроблення програмного забезпечення. Наука і техніка Збройних Сил України. – Випуск 2(23). - Харків: ХУПС. - 2016. - С. 150-158.

45. Смірнов О.А., Коваленко О.В., Якименко Н.М., Доренський О.П. Проблеми аналізу та оцінки ризиків інформаційної діяльності. Збірник наукових праць "Системи обробки інформації". - Випуск 3 (140). - Х.: ХУПС - 2016. - С. 40-42.

46. Смірнов О.А., Коваленко А.С., Коваленко О.В., Доренський О.П. Удосконалення методу технічного обслуговування об’єктів інтегрованої інформаційної системи. Системи озброєння і військова техніка. – Випуск 2(46) – Х.: ХУПС – 2016. – С. 103-107.

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

48. Смірнов О.А., Євсеєв С.П., Король О.Г., Коваленко О.В., Коваленко А.С., Смірнов С.А. Архітектура мікропроцесорів та компонентів ЕОМ. Навчальний посібник – Кіровоград: Вид. Лисенко В.Ф., 2015. – 550 с.

49. Смірнов О.А., Коваленко О.В., Мелешко Є.В., Константинова Л.В., Кожанова А.С. Інженерія програмного забезпечення. Навчальний посібник для студентів вищих навчальних закладів напрямів підготовки 8.050102 «Комп’ютерна інженерія». За ред. О.А. Смірнова Гриф “Навчальний посібник” надано у відповідності з листом Міністерства освіти і науки України від 18.03.2013 року № 1/11-5584. – Кіровоград: КНТУ 2013. – 409с.

50. Смірнов О.А., Осадчій С.І., Мелешко Є.В., Іванов С.Г., Павленко М.А., Усачов О.М. Основи технічної експлуатації АСУ. Навчальний посібник для студентів вищих навчальних закладів напрямів підготовки 8.050102 «Комп'ютерна інженерія». За ред. О.А. Смірнова, Гриф "Навчальний посібник" надано у відповідності з листом Міністерства освіти і науки України від 27.02.2013 року № 1/11-4487 . – Кіровоград: КНТУ 2013. – 322с.

51. Смірнов О.А., Коваленко О.В., Кожанова А.С., Левошко О.Л., Константинова Л.В. Основи системного програмування. Навчальний посібник для студентів вищих навчальних закладів напрямів підготовки 8.050102 «Комп'ютерна інженерія». За ред. Коваленка О.В., Гриф "Навчальний посібник" надано у відповідності з листом Міністерства освіти і науки України від 26.02.2013 року № 1/11-4368. – Кіровоград: КНТУ 2013. – 257с.

52. Смірнов О.А., Мелешко Є.В., Семенов С.Г. Методи та засоби обробки сигналів і даних в інформаційних системах. Навчальний посібник для студентів вищих навчальних закладів напрямів підготовки 8.050102 «Комп'ютерна інженерія». За ред. О.А. Смірнова Гриф "Навчальний посібник" надано у відповідності з листом Міністерства освіти і науки, молоді та спорту України від 17.04.2012 року № 1/11-5249. – Кіровоград: КНТУ 2012. – 250 с.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи моніторингу стану SSD диску на основі технології Life Left.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 34-13 від 04.08.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи моніторингу стану SSD диску на основі технології Life Left.

4 Джерела розробки

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

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

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

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

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи моніторингу стану SSD диску на основі технології Life Left;

– 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 Visual C++.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Економічні вимоги

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

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2023 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинно бути проведено розрахунок штучного освітлення за методом коефіцієнта використання світлового потоку.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 100 аркушів.

10 Етапи розробки

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

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

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

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

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

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

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

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

_____ Коваленко О.В.

Дослідження та програмна реалізація системи моніторингу стану SSD диску на основі технології Life Left

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 72

Літера: РП

Кропивницький – 2023 року

Файл DiskInfo.cpp – основна програма

```
#include "stdafx.h"
#include "DiskInfo.h"
#include "DiskInfoDlg.h"
#include "GraphDlg.h"

#include "GetFileVersion.h"
#include "GetOsInfo.h"
#include "IsCurrentUserLocalAdministrator.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif

// CDiskInfoApp

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


// CDiskInfoApp конструктор

CDiskInfoApp::CDiskInfoApp()
{
      // ПРИМІТКА: Додаємо код конструктору,
      // Встановлюємо усі значиму ініціалізацію в  InitInstance
}


// Опис CDiskInfoApp об’єкту

CDiskInfoApp theApp;
CString m_Ini;

//-----------------------------------------------------------------------------
// Опис прототипів
//-----------------------------------------------------------------------------
static BOOL IsFileExistEx(const TCHAR* path, const TCHAR* fileName);
static BOOL RunAsRestart();
// CDiskInfoApp ініціалізація

BOOL CDiskInfoApp::InitInstance()
{
      BOOL flagEarthlight = FALSE;
      DWORD defaultDisk = 0;
      HANDLE hMutex = NULL;

      INITCOMMONCONTROLSEX InitCtrls;
      InitCtrls.dwSize = sizeof(InitCtrls);
      InitCtrls.dwICC = ICC_WIN95_CLASSES;
      InitCommonControlsEx(&InitCtrls);
      CWinApp::InitInstance();

      ZeroMemory(&m_OsVer, sizeof(OSVERSIONINFO));
      m_OsVer.dwOSVersionInfoSize = sizeof(OSVERSIONINFO);
      GetVersionEx(&m_OsVer);

      // Якщо версія підтримується.
      if(GetFileVersion(_T("Shdocvw.dll")) < 471)
      {
            AfxMessageBox(_T("CrystalDiskInfo має потребу у  IE 6.0 або
вище."));
      }

      // ініціалізація m_Ini
```

```
	TCHAR *ptrEnd;
	TCHAR ini[MAX_PATH];
	::GetModuleFileName(NULL, ini, MAX_PATH);
	if((ptrEnd = _tcsrchr(ini, '.')) != NULL)
	{
		*ptrEnd = '\0';
		_tcscat_s(ini, MAX_PATH, _T(".ini"));
	}
	m_Ini = ini;

	int argc;
	LPWSTR *argv = CommandLineToArgvW(GetCommandLineW(), &argc);

	if(argc > 1)
	{
		CString cstr;
		cstr = argv[1];
		if(cstr.Compare(_T("/Earthlight")) == 0)
		{
			flagEarthlight = TRUE;
			if(argc > 2)
			{
				defaultDisk = _tstoi(argv[2]);
			}
		}
		if(cstr.Compare(_T("/Startup")) == 0)
		{
			int time = 0;
			time = GetPrivateProfileInt(_T("Setting"),
_T("StartupWaitTime"), 30, m_Ini);
			if(time >= 0)
			{
				Sleep(time * 1000);
			}
			TCHAR str[MAX_PATH];
			::GetModuleFileName(NULL, str, MAX_PATH);
			ShellExecute(NULL, NULL, str, NULL, NULL, SW_SHOWNORMAL);
			return FALSE;
		}
	}

	// Розшифрування отриманих даних
	//flagEarthlight = TRUE;

	if(! flagEarthlight)
	{
		hMutex = ::CreateMutex(NULL, FALSE, PRODUCT_NAME PRODUCT_VERSION);
		if(GetLastError() == ERROR_ALREADY_EXISTS)
		{
			return FALSE;
		}
	}

	CString DefaultTheme;
	CString DefaultLanguage;
	TCHAR tmp[MAX_PATH];

	GetModuleFileName(NULL, tmp, MAX_PATH);
	if((ptrEnd = _tcsrchr(tmp, '\\')) != NULL)
	{
		*ptrEnd = '\0';
	}

	m_ExeDir.Format(_T("%s\\"), tmp);
	m_ThemeDir.Format(_T("%s\\%s"), tmp, THEME_DIR);
	m_LangDir.Format(_T("%s\\%s"), tmp, LANGUAGE_DIR);
	m_LifeLeftDir.Format(_T("%s\\%s"), tmp, LifeLeft_DIR);

	m_ThemeIndex = MENU_THEME_INDEX;
```

```
	m_LangIndex = MENU_LANG_INDEX;

	DefaultTheme.Format(_T("%s\\%s"), tmp, DEFAULT_THEME);
	DefaultLanguage.Format(_T("%s\\%s"), tmp, DEFAULT_LANGUAGE);

/*
	if(IsClassicSystem())
	{
		m_MainDlgPath.Format(_T("%s\\") DIALOG_DIR CLASSIC_DIALOG, tmp);
	}
*/
	if(! IsFileExist(m_MainDlgPath))
	{
		m_MainDlgPath.Format(_T("%s\\") DIALOG_DIR MAIN_DIALOG, tmp);
	}

	m_AboutDlgPath.Format(_T("%s\\") DIALOG_DIR ABOUT_DIALOG, tmp);
	m_SettingDlgPath.Format(_T("%s\\") DIALOG_DIR SETTING_DIALOG, tmp);
	if(GetIeVersion() == 800)
	{
		m_GraphDlgPath.Format(_T("%s\\") DIALOG_DIR GRAPH_DIALOG_IE8, tmp);
	}
	else
	{
		m_GraphDlgPath.Format(_T("%s\\") DIALOG_DIR GRAPH_DIALOG, tmp);
	}
	m_OptionDlgPath.Format(_T("%s\\") DIALOG_DIR OPTION_DIALOG, tmp);

	if(! IsFileExistEx(m_MainDlgPath, MAIN_DIALOG))
	{
		return FALSE;
	}
	if(! IsFileExistEx(m_AboutDlgPath, ABOUT_DIALOG))
	{
		return FALSE;
	}
	if(! IsFileExistEx(m_SettingDlgPath, SETTING_DIALOG))
	{
		return FALSE;
	}

// для сімейства Windows NT
#ifdef _UNICODE
	if(! IsCurrentUserLocalAdministrator())
	{
		if(m_OsVer.dwMajorVersion < 6)
		{
			AfxMessageBox(_T("CrystalDiskInfo is required Administrator
Privileges."));
		}
		RunAsRestart();
		return FALSE;
	}
#endif

	if(flagEarthlight)
	{
		CGraphDlg dlg(NULL, defaultDisk);
		m_pMainWnd = &dlg;
		INT_PTR nResponse = dlg.DoModal();
	}
	else
	{
		CDiskInfoDlg dlg;
		m_pMainWnd = &dlg;
		INT_PTR nResponse = dlg.DoModal();
		::ReleaseMutex(hMutex);
		::CloseHandle(hMutex);
	}
```

```
	return FALSE;
}

BOOL IsFileExistEx(const TCHAR* path, const TCHAR* fileName)
{
	if(! IsFileExist(path))
	{
		CString cstr;
		cstr.Format(_T("Не знайдено\"%s\"."), fileName);
		AfxMessageBox(cstr);
		return FALSE;
	}
	return TRUE;
}

BOOL RunAsRestart()
{
	int count;
#ifdef _UNICODE
	TCHAR** cmd = ::CommandLineToArgvW(::GetCommandLine(), &count);
#else
	TCHAR** cmd = ::__argv;
	count = ::__argc;
#endif

	if(count < 2 || _tcscmp(cmd[1], _T("runas")) != 0)
	{
		TCHAR path[MAX_PATH];
		::GetModuleFileName(NULL, path, MAX_PATH);
		if(::ShellExecute(NULL, _T("runas"), path, _T("runas"), NULL,
SW_SHOWNORMAL)
			> (HINSTANCE)32)
		{
			return TRUE;
		}
	}
	return FALSE;
}
```

Файл DiskInfo.h – бібліотека для файлу DiskInfo.cpp

```
#pragma once

#ifndef __AFXWIN_H__
	#error "include 'stdafx.h' перед включенням цього файлу для PCH"
#endif

#include "resource.h"		// головні символи

#define THEME_DIR                       _T("CdiResource\\theme\\")
#define LANGUAGE_DIR                    _T("CdiResource\\language\\")
#define DIALOG_DIR                          _T("CdiResource\\dialog\\")
#define LifeLeft_DIR                        _T("LifeLeft\\")
#define LifeLeft_INI                        _T("LifeLeft.ini")
#define EXCHANGE_INI                    _T("Exchange.ini")

#define MENU_THEME_INDEX                3
#define MENU_LANG_INDEX                 6
#define MENU_DRIVE_INDEX                4

#define MAIN_DIALOG                         _T("Main.html")
//#define CLASSIC_DIALOG                    _T("Classic.html")
#define ABOUT_DIALOG                    _T("About.html")
#define SETTING_DIALOG                  _T("Setting.html")
#define GRAPH_DIALOG                    _T("Graph.html")
#define GRAPH_DIALOG_IE8                _T("GraphIe8.html")
#define OPTION_DIALOG                   _T("Option.html")

#define DEFAULT_THEME                   THEME_DIR _T("default\\Main.css")
#define DEFAULT_LANGUAGE                LANGUAGE_DIR _T("English.lang")


class CDiskInfoApp : public CWinApp
{
public:
	CDiskInfoApp();

	OSVERSIONINFO m_OsVer;
	BOOL m_IsNT;

	CString m_MainDlgPath;
	CString m_AboutDlgPath;
	CString m_SettingDlgPath;
	CString m_GraphDlgPath;
	CString m_OptionDlgPath;
	CString m_LifeLeftDir;
	CString m_ExeDir;
	CString m_Ini;

	CString m_ThemeDir;
	CString m_LangDir;
	DWORD m_ThemeIndex;
	DWORD m_LangIndex;

// Аннулюється
	public:
	virtual BOOL InitInstance();

// Реалізація

	DECLARE_MESSAGE_MAP()
};

extern CDiskInfoApp theApp;
```

Файл AtaLifeLeft.cpp – тестування жорсткого диску з використанням технології LifeLeft

```
#include "stdafx.h"
#include "AtaLifeLeft.h"
#include <wbemcli.h>

#include "DebugPrint.h"
#include "DnpService.h"

#pragma comment(lib, "wbemuuid.lib")
#define SAFE_RELEASE(p) { if(p) { (p)->Release(); (p)=NULL; } }

static const TCHAR *commandTypeString[] =
{
	_T("pd"),
	_T("sm"),
	_T("sa"),
	_T("sp"),
	_T("io"),
	_T("lo"),
	_T("jm"),
	_T("cy")
};
// Визначення ATA
CAtaLifeLeft::CAtaLifeLeft()
{
	BOOL bosVersionInfoEx;
	ZeroMemory(&m_Os, sizeof(OSVERSIONINFOEX));
	m_Os.dwOSVersionInfoSize = sizeof(OSVERSIONINFOEX);
	if(!(bosVersionInfoEx = GetVersionEx((OSVERSIONINFO *)&m_Os)))
	{
		m_Os.dwOSVersionInfoSize = sizeof(OSVERSIONINFO);
		GetVersionEx((OSVERSIONINFO *)&m_Os);
	}

	m_FlagAtaPassThrough = FALSE;
	if(m_Os.dwMajorVersion >= 6 || (m_Os.dwMajorVersion == 5 &&
m_Os.dwMinorVersion == 2))
	{
		m_FlagAtaPassThrough = TRUE;
	}
	else if(m_Os.dwMajorVersion == 5 && m_Os.dwMinorVersion == 1)
	{
		CString cstr;
		cstr = m_Os.szCSDVersion;
		cstr.Replace(_T("Service Pack "), _T(""));
		if(_tstoi(cstr) >= 2)
		{
			m_FlagAtaPassThrough = TRUE;
		}
	}
}

CAtaLifeLeft::~CAtaLifeLeft()
{
}

/* ЗАГАЛЬНІ ФУНКЦІЇ*/
DWORD CAtaLifeLeft::UpdateLifeLeftInfo(DWORD i)
{
	if(vars.GetCount() == 0)
	{
		return LifeLeft_STATUS_NO_CHANGE;
	}

	static LifeLeft_ATTRIBUTE attribute[MAX_DISK][MAX_ATTRIBUTE] = {0};
```

```
// Читання таблиці атрибутів
      if(vars[i].IsLifeLeftEnabled)
      {
            switch(vars[i].CommandType)
            {
            case CMD_TYPE_PHYSICAL_DRIVE:
                  GetLifeLeftAttributePd(vars[i].PhysicalDriveId, &(vars[i]));
                  vars[i].DiskStatus = CheckDiskStatus(vars[i].Attribute,
vars[i].Threshold, vars[i].AttributeCount, vars[i].VendorId);
                  break;

            case CMD_TYPE_SCSI_MINIPORT:
                  GetLifeLeftAttributeScsi(vars[i].ScsiPort,
vars[i].ScsiTargetId, &(vars[i]));
                  vars[i].DiskStatus = CheckDiskStatus(vars[i].Attribute,
vars[i].Threshold, vars[i].AttributeCount, vars[i].VendorId);
                  break;
            case CMD_TYPE_SAT:
            case CMD_TYPE_SUNPLUS:
            case CMD_TYPE_IO_DATA:
            case CMD_TYPE_LOGITEC:
            case CMD_TYPE_JMICRON:
            case CMD_TYPE_CYPRESS:
                  GetLifeLeftAttributeSat(vars[i].PhysicalDriveId, &(vars[i]));
                  vars[i].DiskStatus = CheckDiskStatus(vars[i].Attribute,
vars[i].Threshold, vars[i].AttributeCount, vars[i].VendorId);
                  break;
            default:
                  return LifeLeft_STATUS_NO_CHANGE;
            }

            return CheckLifeLeftAttributeUpdate(i, attribute[i],
vars[i].Attribute);
      }

      return LifeLeft_STATUS_NO_CHANGE;
}

/* ЗАГАЛЬНІ ФУНКЦІЇ*/
BOOL CAtaLifeLeft::UpdateIdInfo(DWORD i)
{
      BOOL flag = FALSE;
      switch(vars[i].CommandType)
      {
      case CMD_TYPE_PHYSICAL_DRIVE:
            flag =  DoIdentifyDevicePd(vars[i].PhysicalDriveId,
&(vars[i].IdentifyDevice));
            break;
      case CMD_TYPE_SCSI_MINIPORT:
            flag =  DoIdentifyDeviceScsi(vars[i].ScsiPort, vars[i].ScsiTargetId,
&(vars[i].IdentifyDevice));
            break;
      case CMD_TYPE_SAT:
      case CMD_TYPE_SUNPLUS:
      case CMD_TYPE_IO_DATA:
      case CMD_TYPE_LOGITEC:
      case CMD_TYPE_JMICRON:
      case CMD_TYPE_CYPRESS:
            flag = DoIdentifyDeviceSat(vars[i].PhysicalDriveId,
&(vars[i].IdentifyDevice), vars[i].CommandType);
            break;
      default:
            return FALSE;
            break;
      }

      if(vars[i].Major >= 3 && vars[i].IdentifyDevice.CommandSetSupported2 & (1
<< 3))
      {
```

```
            vars[i].IsApmSupported = TRUE;
            if(vars[i].IdentifyDevice.CommandSetEnabled2 & (1 << 3))
            {
                  vars[i].IsApmEnabled = TRUE;
            }
            else
            {
                  vars[i].IsApmEnabled = FALSE;
            }
      }
      if(vars[i].Major >= 5 && vars[i].IdentifyDevice.CommandSetSupported2 & (1
<< 9))
      {
            vars[i].IsAamSupported = TRUE;
            if(vars[i].IdentifyDevice.CommandSetEnabled2 & (1 << 9))
            {
                  vars[i].IsAamEnabled = TRUE;
            }
            else
            {
                  vars[i].IsAamEnabled = FALSE;
            }
      }

      return flag;
}

/* ЗАГАЛЬНІ ФУНКЦІЇ*/
BYTE CAtaLifeLeft::GetAamValue(DWORD i)
{
      return LOBYTE(vars[i].IdentifyDevice.AcoustricManagement);
}

/* ЗАГАЛЬНІ ФУНКЦІЇ*/
BYTE CAtaLifeLeft::GetApmValue(DWORD i)
{
      return LOBYTE(vars[i].IdentifyDevice.CurrentPowerManagement);
}

/* ЗАГАЛЬНІ ФУНКЦІЇ*/
BYTE CAtaLifeLeft::GetRecommendAamValue(DWORD i)
{
      return HIBYTE(vars[i].IdentifyDevice.AcoustricManagement);
}

/* ЗАГАЛЬНІ ФУНКЦІЇ*/
BYTE CAtaLifeLeft::GetRecommendApmValue(DWORD i)
{
      return HIBYTE(vars[i].IdentifyDevice.CurrentPowerManagement);
}

/* ЗАГАЛЬНІ ФУНКЦІЇ*/
BOOL CAtaLifeLeft::EnableAam(DWORD i, BYTE param)
{
      return SendAtaCommand(i, 0xEF, 0x42, param);
}

/* ЗАГАЛЬНІ ФУНКЦІЇ*/
BOOL CAtaLifeLeft::DisableAam(DWORD i)
{
      return SendAtaCommand(i, 0xEF, 0xC2, 0);
}

/* ЗАГАЛЬНІ ФУНКЦІЇ*/
BOOL CAtaLifeLeft::EnableApm(DWORD i, BYTE param)
{
      return SendAtaCommand(i, 0xEF, 0x05, param);
}
```

```
/* ЗАГАЛЬНІ ФУНКЦІЇ*/
BOOL CAtaLifeLeft::DisableApm(DWORD i)
{
	return SendAtaCommand(i, 0xEF, 0x85, 0);
}

BOOL CAtaLifeLeft::SendAtaCommand(DWORD i, BYTE main, BYTE sub, BYTE param)
{
	switch(vars[i].CommandType)
	{
	case CMD_TYPE_PHYSICAL_DRIVE:
		return SendAtaCommandPd(vars[i].PhysicalDriveId, main, sub, param,
NULL, 0);
		break;
	case CMD_TYPE_SCSI_MINIPORT:
		return SendAtaCommandScsi(vars[i].ScsiPort, vars[i].ScsiTargetId,
main, sub, param);
		break;
	case CMD_TYPE_SAT:
	case CMD_TYPE_SUNPLUS:
	case CMD_TYPE_IO_DATA:
	case CMD_TYPE_LOGITEC:
	case CMD_TYPE_JMICRON:
	case CMD_TYPE_CYPRESS:
		return SendAtaCommandSat(vars[i].PhysicalDriveId, main, sub, param,
vars[i].CommandType);
		break;
	default:
		return FALSE;
		break;
	}
	return FALSE;
}

DWORD CAtaLifeLeft::CheckLifeLeftAttributeUpdate(DWORD index,
LifeLeft_ATTRIBUTE* pre, LifeLeft_ATTRIBUTE* cur)
{

	if(memcmp(pre, cur, sizeof(LifeLeft_ATTRIBUTE) * MAX_ATTRIBUTE) == 0)
	{
		return LifeLeft_STATUS_NO_CHANGE;
	}
	else
	{
		for(int i = 0; i < MAX_ATTRIBUTE; i++)
		{
			switch(cur[i].Id)
			{
			case 0x09: // Кількість відпрацьованих годин у включеному
стані
				{
					DWORD preRawValue = MAKELONG(
						MAKEWORD(pre[i].RawValue[0],
pre[i].RawValue[1]),
						MAKEWORD(pre[i].RawValue[2],
pre[i].RawValue[3])
						);
					DWORD curRawValue = MAKELONG(
						MAKEWORD(cur[i].RawValue[0],
cur[i].RawValue[1]),
						MAKEWORD(cur[i].RawValue[2],
cur[i].RawValue[3])
						);

					if(GetPowerOnHours(preRawValue,
vars[index].DetectedTimeUnitType)
					!= GetPowerOnHours(curRawValue,
vars[index].DetectedTimeUnitType))
					{
```

```
                                    memcpy(pre, cur, sizeof(LifeLeft_ATTRIBUTE)
* MAX_ATTRIBUTE);
                                    return LifeLeft_STATUS_MAJOR_CHANGE;
                              }
                              if(GetPowerOnHours(preRawValue,
vars[index].MeasuredTimeUnitType)
                              != GetPowerOnHours(curRawValue,
vars[index].MeasuredTimeUnitType))
                              {
                                    memcpy(pre, cur, sizeof(LifeLeft_ATTRIBUTE)
* MAX_ATTRIBUTE);
                                    return LifeLeft_STATUS_MAJOR_CHANGE;
                              }
                        }
                        break;
                  case 0x0C: // Кількість зафіксованих повторів
включення/вимикання живлення накопичувача
                        {
                              DWORD preRawValue = MAKELONG(
                                    MAKEWORD(pre[i].RawValue[0],
pre[i].RawValue[1]),
                                    MAKEWORD(pre[i].RawValue[2],
pre[i].RawValue[3])
                                    );
                              DWORD curRawValue = MAKELONG(
                                    MAKEWORD(cur[i].RawValue[0],
cur[i].RawValue[1]),
                                    MAKEWORD(cur[i].RawValue[2],
cur[i].RawValue[3])
                                    );
                              if(preRawValue != curRawValue)
                              {
                                    memcpy(pre, cur, sizeof(LifeLeft_ATTRIBUTE)
* MAX_ATTRIBUTE);
                                    return LifeLeft_STATUS_MAJOR_CHANGE;
                              }
                        }
                        break;
                  case 0xC2: // Температура
                        if(pre[i].RawValue[0] != cur[i].RawValue[0]
                        || pre[i].CurrentValue != cur[i].CurrentValue)
                        {
                              memcpy(pre, cur, sizeof(LifeLeft_ATTRIBUTE) *
MAX_ATTRIBUTE);
                              return LifeLeft_STATUS_MAJOR_CHANGE;
                        }
                        break;
                  default:
                        break;
                  }
            }
            return LifeLeft_STATUS_MINOR_CHANGE;
      }
}

/* ЗАГАЛЬНІ ФУНКЦІЇ*/
BOOL CAtaLifeLeft::MeasuredTimeUnit()
{
      DWORD getTickCount = GetTickCount();
      if(getTickCount > MeasuredGetTickCount + 155000 || MeasuredGetTickCount +
125000 > getTickCount)
      {
            return FALSE;
      }

      for(int i = 0; i < vars.GetCount(); i++)
      {
            if(vars[i].PowerOnRawValue < 0)
            {
```

```
                continue;
            }
            UpdateLifeLeftInfo(i);

            DWORD test = vars[i].PowerOnRawValue - vars[i].PowerOnStartRawValue;

            if(vars[i].Model.Find(_T("SAMSUNG")) == 0)
            {
                if(test >= 2)
                {
                    vars[i].MeasuredTimeUnitType = POWER_ON_HALF_MINUTES;
                }
                else
                {
                    vars[i].MeasuredTimeUnitType = POWER_ON_HOURS;
                }
            }
            else if(vars[i].Model.Find(_T("FUJITSU")) == 0)
            {
                if(test >= 2)
                {
                    vars[i].MeasuredTimeUnitType = POWER_ON_SECONDS;
                }
                else
                {
                    vars[i].MeasuredTimeUnitType = POWER_ON_HOURS;
                }
            }
            else if(vars[i].Model.Find(_T("MAXTOR")) == 0)
            {
                if(test >= 2)
                {
                    vars[i].MeasuredTimeUnitType = POWER_ON_MINUTES;
                    vars[i].IsMaxtorMinute = TRUE;
                }
                else
                {
                    vars[i].MeasuredTimeUnitType = POWER_ON_HOURS;
                    vars[i].IsMaxtorMinute = FALSE;
                }
            }
            else
            {
                if(test >= 2)
                {
                    vars[i].MeasuredTimeUnitType = POWER_ON_MINUTES;
                }
                else
                {
                    vars[i].MeasuredTimeUnitType = POWER_ON_HOURS;
                }
            }
        }

        return TRUE;
}

/* ЗАГАЛЬНІ ФУНКЦІЇ*/
BOOL CAtaLifeLeft::Init(BOOL useWmi, BOOL advancedDiskSearch, PBOOL
flagChangeDisk)
{
        IsEnabledWmi = FALSE;
        CArray<DISK_POSITION, DISK_POSITION> previous;

        if(flagChangeDisk != NULL)
        {
            *flagChangeDisk = FALSE;
            for(int i = 0; i < vars.GetCount(); i++)
            {
```

```
                DISK_POSITION dp;
                dp.PhysicalDriveId = vars.GetAt(i).PhysicalDriveId;
                dp.ScsiTargetId = vars.GetAt(i).ScsiTargetId;
                dp.ScsiPort = vars.GetAt(i).ScsiPort;

                previous.Add(dp);
          }
     }

     // Ініціалізація
     vars.RemoveAll();
     m_ControllerMap = _T("");

     if(useWmi)
     {
          HRESULT hRes = S_OK;
          ULONG   uReturned = 1;

          IWbemLocator*                pIWbemLocator = NULL;
          IWbemServices*               pIWbemServices = NULL;
          IEnumWbemClassObject*   pEnumCOMDevs = NULL;
          IEnumWbemClassObject*   pEnumCOMDevs2 = NULL;
          IWbemClassObject*       pCOMDev = NULL;

          DebugPrint(_T("CAtaLifeLeft::Init WMI on - Start"));

          bool initWmi = true;
          CDnpService cService;

          for(int i = 0; i < 3; i++)
          {
                if(! cService.IsServiceRunning(_T("Winmgmt")))
                {
                     DebugPrint(_T("Waiting... Winmgmt"));
                     initWmi = cService.EasyStart(_T("Winmgmt"));
                     continue;
                }
                else
                {
                     break;
                }
          }

          if(initWmi)
          {
                try
                {
                     DebugPrint(_T("CoInitialize()"));
                     CoInitialize(NULL);
                     DebugPrint(_T("CoInitializeSecurity()"));
                     CoInitializeSecurity(NULL, -1, NULL, NULL,
RPC_C_AUTHN_LEVEL_DEFAULT,
                          RPC_C_IMP_LEVEL_IMPERSONATE, NULL, EOAC_NONE,
NULL);
                     DebugPrint(_T("CoCreateInstance()"));
                     if(FAILED(CoCreateInstance(CLSID_WbemLocator, NULL,
CLSCTX_INPROC_SERVER,
                          IID_IWbemLocator, (LPVOID *)&pIWbemLocator)))
                     {
                          CoUninitialize();
                          DebugPrint(_T("NG:WMI Init 1"));
                     }
                     else
                     {
                          long securityFlag = 0;
                          if( m_Os.dwMajorVersion >= 6 // Vista або пізніша
версія
                          || (m_Os.dwMajorVersion == 5 &&
m_Os.dwMinorVersion >= 1) // XP або пізніша версія
```

```
                                )
                                {
                                      securityFlag =
WBEM_FLAG_CONNECT_USE_MAX_WAIT;
                                }

                                DebugPrint(_T("ConnectServer()"));
                                if(FAILED(pIWbemLocator-
>ConnectServer(SysAllocString(L"\\\\.\\root\\cimv2"),
                                      NULL, NULL, 0L,
                                      securityFlag,
                                      NULL, NULL, &pIWbemServices)))
                                {
                                      CoUninitialize();
                                      DebugPrint(_T("NG:WMI Init 2"));
                                }
                                else
                                {
                                      DebugPrint(_T("CoSetProxyBlanket()"));
                                      hRes = CoSetProxyBlanket(pIWbemServices,
RPC_C_AUTHN_WINNT, RPC_C_AUTHZ_NONE,
                                            NULL, RPC_C_AUTHN_LEVEL_CALL,
RPC_C_IMP_LEVEL_IMPERSONATE, NULL, EOAC_NONE);
                                      if(FAILED(hRes))
                                      {
                                            CoUninitialize();
                                            CString cstr;
                                            cstr.Format(_T("NG:WMI Init - %08X"),
hRes);
                                            DebugPrint(cstr);
                                      }
                                      else
                                      {
                                            IsEnabledWmi = TRUE;
                                            DebugPrint(_T("OK:WMI Init"));
                                      }
                                }
                                SAFE_RELEASE(pIWbemLocator);
                          }
                    }
                    catch(...)
                    {
                          DebugPrint(_T("EX:WMI Init"));
                    }
              }
              else
              {
                    DebugPrint(_T("NG:WMI Init 3"));
              }

              if(IsEnabledWmi)
              {
                    CStringArray csa;
                    CString temp, cstr, cstr1, cstr2;
                    try
                    {// Win32_IDE Контролер
                          pIWbemServices->ExecQuery(SysAllocString(L"WQL"),
                                SysAllocString(L"select Name, DeviceID from
Win32_IDEController"), WBEM_FLAG_FORWARD_ONLY | WBEM_FLAG_RETURN_IMMEDIATELY,
NULL, &pEnumCOMDevs);
                          while(pEnumCOMDevs && SUCCEEDED(pEnumCOMDevs-
>Next(10000, 1, &pCOMDev, &uReturned)) && uReturned == 1)
                          {
                                VARIANT  pVal;
                                VariantClear(&pVal);
                                CString name1, deviceId, channel;
                                if(pCOMDev->Get(L"DeviceID", 0L, &pVal, NULL,
NULL) == WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                                {
```

```
                                    deviceId = pVal.bstrVal;
                                    if(deviceId.Find(_T("PCIIDE\\IDECHANNEL"))
== 0)
                                    {
                                          channel = deviceId.Right(1);
                                    }
                                    deviceId.Replace(_T("\\"), _T("\\\\"));
                                    VariantClear(&pVal);
                              }
                              if(pCOMDev->Get(L"Name", 0L, &pVal, NULL, NULL) ==
WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                              {
                                    name1 = pVal.bstrVal;
                                    if(! channel.IsEmpty())
                                    {
                                          name1 += _T(" (") + channel + _T(")");
                                    }
                                    m_IdeController.Add(name1);

                                    VariantClear(&pVal);
                              }
                              SAFE_RELEASE(pCOMDev);

                              if(cstr.Find(name1) == -1 || cstr.Find(name1 +
_T(" [ATA]")) >= 0)
                              {
                                    csa.Add(cstr);
                                    cstr = _T("%%%") + name1 + _T(" [ATA]");
                                    cstr += _T("\r\n");
                              }

                              CString mapping;
                              mapping.Format(_T("ASSOCIATORS OF
{Win32_IDEController.DeviceID=\"%s\"} WHERE AssocClass =
Win32_IDEControllerDevice"), deviceId);
                              pIWbemServices->ExecQuery(SysAllocString(L"WQL"),
SysAllocString(mapping), WBEM_FLAG_FORWARD_ONLY | WBEM_FLAG_RETURN_IMMEDIATELY,
NULL, &pEnumCOMDevs2);
                              while(pEnumCOMDevs2 && SUCCEEDED(pEnumCOMDevs2-
>Next(10000, 1, &pCOMDev, &uReturned)) && uReturned == 1)
                              {
                                    VARIANT pVal;
                                    VariantClear(&pVal);
                                    CString name2, deviceId, channel;
                                    if(pCOMDev->Get(L"DeviceID", 0L, &pVal,
NULL, NULL) == WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                                    {
                                          deviceId = pVal.bstrVal;

      if(deviceId.Find(_T("PCIIDE\\IDECHANNEL")) == 0)
                                          {
                                                channel = deviceId.Right(1);
                                          }
                                          VariantClear(&pVal);
                                    }
                                    if(pCOMDev->Get(L"Name", 0L, &pVal, NULL,
NULL) == WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                                    {
                                          name2 = pVal.bstrVal;
                                          if(! channel.IsEmpty())
                                          {
                                                name2 += _T(" (") + channel +
_T(")");
                                          }
                                          VariantClear(&pVal);
                                    }
                                    SAFE_RELEASE(pCOMDev);
```

```
                                    if(cstr.Find(_T("   - ") + name1) >= 0 ||
cstr.Find(_T("   + ") + name1) >= 0)
                                    {
                                          cstr1 = _T("- ") + name1;
                                          cstr2 = _T("+ ") + name1;
                                          cstr.Replace(cstr1, cstr2);

                                          cstr1 = name1 + _T("\r\n     - ") +
name2;
                                          cstr.Replace(name1, cstr1);
                                    }
                                    else
                                    {
                                          cstr += _T("   - ") + name2 +
_T("\r\n");
                                    }
                                    cstr.Replace(_T("%%%"), _T(" + "));
                              }
                              cstr.Replace(_T("%%%"), _T(" - "));
                              SAFE_RELEASE(pEnumCOMDevs2);
                        }
                        csa.Add(cstr);
                        SAFE_RELEASE(pEnumCOMDevs);
                        DebugPrint(_T("OK:Win32_IDEController"));
                  }
                  catch(...)
                  {
                        DebugPrint(_T("EX:Win32_IDEController"));
                  }

                  try
                  {
                        cstr = _T("");
                        pIWbemServices->ExecQuery(SysAllocString(L"WQL"),
                              SysAllocString(L"select Name, DeviceID from
Win32_SCSIController"), WBEM_FLAG_FORWARD_ONLY | WBEM_FLAG_RETURN_IMMEDIATELY,
NULL, &pEnumCOMDevs);
                        while(pEnumCOMDevs && SUCCEEDED(pEnumCOMDevs-
>Next(10000, 1, &pCOMDev, &uReturned)) && uReturned == 1)
                        {
                              VARIANT  pVal;
                              VariantClear(&pVal);
                              CString name1, deviceId;
                              if(pCOMDev->Get(L"DeviceID", 0L, &pVal, NULL,
NULL) == WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                              {
                                    deviceId = pVal.bstrVal;
                                    deviceId.Replace(_T("\\"), _T("\\\\"));
                                    VariantClear(&pVal);
                              }
                              if(pCOMDev->Get(L"Name", 0L, &pVal, NULL, NULL) ==
WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                              {
                                    name1 = pVal.bstrVal;
                                    m_ScsiController.Add(name1);
                                    VariantClear(&pVal);
                              }
                              SAFE_RELEASE(pCOMDev);
                              if(cstr.Find(name1) == -1 || cstr.Find(name1 +
_T(" [SCSI]")) >= 0)
                              {
                                    csa.Add(cstr);
                                    cstr = _T("%%%") + name1 + _T(" [SCSI]");

                                    cstr += _T("\r\n");
                              }

                              CString mapping;
```

```
                              mapping.Format(_T("ASSOCIATORS OF
{Win32_SCSIController.DeviceID=\"%s\"} WHERE AssocClass =
Win32_SCSIControllerDevice"), deviceId);
                              pIWbemServices->ExecQuery(SysAllocString(L"WQL"),
SysAllocString(mapping), WBEM_FLAG_FORWARD_ONLY | WBEM_FLAG_RETURN_IMMEDIATELY,
NULL, &pEnumCOMDevs2);
                              while(pEnumCOMDevs2 && SUCCEEDED(pEnumCOMDevs2-
>Next(10000, 1, &pCOMDev, &uReturned)) && uReturned == 1)
                              {
                                    VARIANT pVal;
                                    VariantClear(&pVal);
                                    CString name2, deviceId;
                                    if(pCOMDev->Get(L"DeviceID", 0L, &pVal,
NULL, NULL) == WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                                    {
                                          deviceId = pVal.bstrVal;
                                          VariantClear(&pVal);
                                    }
                                    if(pCOMDev->Get(L"Name", 0L, &pVal, NULL,
NULL) == WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                                    {
                                          name2 = pVal.bstrVal;
                                          VariantClear(&pVal);
                                    }
                                    SAFE_RELEASE(pCOMDev);

                                    if(cstr.Find(_T("   - ") + name1) >= 0 ||
cstr.Find(_T("   + ") + name1) >= 0)
                                    {
                                          cstr1 = _T("- ") + name1;
                                          cstr2 = _T("+ ") + name1;
                                          cstr.Replace(cstr1, cstr2);

                                          cstr1 = name1 + _T("\r\n     - ") +
name2;
                                          cstr.Replace(name1, cstr1);
                                    }
                                    else
                                    {
                                          cstr += _T("   - ") + name2 +
_T("\r\n");
                                    }
                                    cstr.Replace(_T("%%%"), _T(" + "));
                              }
                              cstr.Replace(_T("%%%"), _T(" - "));
                              SAFE_RELEASE(pEnumCOMDevs2);
                        }
                        csa.Add(cstr);
                        SAFE_RELEASE(pEnumCOMDevs);
                        DebugPrint(_T("OK:Win32_SCSIController"));
                  }
                  catch(...)
                  {
                        DebugPrint(_T("EX:Win32_SCSIController"));
                  }

                  for(int i = 0; i < csa.GetCount(); i++)
                  {
                        m_ControllerMap += csa.GetAt(i);
                  }

                  try
                  {// Win32_USB Контролер
                        pIWbemServices->ExecQuery(SysAllocString(L"WQL"),
                              SysAllocString(L"select Name, DeviceID from
Win32_USBController"), WBEM_FLAG_FORWARD_ONLY | WBEM_FLAG_RETURN_IMMEDIATELY,
NULL, &pEnumCOMDevs);
                        while(pEnumCOMDevs && SUCCEEDED(pEnumCOMDevs-
>Next(10000, 1, &pCOMDev, &uReturned)) && uReturned == 1)
```

```
                        {
                              VARIANT  pVal;
                              VariantClear(&pVal);
                              CString deviceId, channel;
                              if(pCOMDev->Get(L"DeviceID", 0L, &pVal, NULL,
NULL) == WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                              {
                                    deviceId = pVal.bstrVal;
                                    deviceId.Replace(_T("\\"), _T("\\\\"));
                                    VariantClear(&pVal);
                              }
                              SAFE_RELEASE(pCOMDev);

                              CString mapping, enclosure;
                              mapping.Format(_T("ASSOCIATORS OF
{Win32_USBController.DeviceID=\"%s\"} WHERE AssocClass =
Win32_USBControllerDevice"), deviceId);
                              pIWbemServices->ExecQuery(SysAllocString(L"WQL"),
SysAllocString(mapping), WBEM_FLAG_FORWARD_ONLY | WBEM_FLAG_RETURN_IMMEDIATELY,
NULL, &pEnumCOMDevs2);
                              while(pEnumCOMDevs2 && SUCCEEDED(pEnumCOMDevs2-
>Next(10000, 1, &pCOMDev, &uReturned)) && uReturned == 1)
                              {
                                    VARIANT pVal;
                                    VariantClear(&pVal);
                                    if(pCOMDev->Get(L"DeviceID", 0L, &pVal,
NULL, NULL) == WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                                    {
                                          cstr = pVal.bstrVal;
                                          VariantClear(&pVal);
                                          if(cstr.Find(_T("USBSTOR")) >= 0)
                                          {
                                                EXTERNAL_DISK_INFO edi = {0};
                                                int curPos= 0;
                                                CString resToken;
                                                resToken =
deviceId.Tokenize(_T("\\&"), curPos);
                                                while(resToken != _T(""))
                                                {

      if(resToken.Replace(_T("VID_"), _T("")) > 0)
                                                      {
                                                            edi.VendorId =
_tcstol(resToken, NULL, 16);
                                                      }
                                                      else
if(resToken.Replace(_T("PID_"), _T("")) > 0)
                                                      {
                                                            edi.ProductId =
_tcstol(resToken, NULL, 16);
                                                      }
                                                      resToken =
deviceId.Tokenize(_T("\\&"), curPos);
                                                };

                                                if(pCOMDev->Get(L"Name", 0L,
&pVal, NULL, NULL) == WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                                                {
                                                      edi.Enclosure =
pVal.bstrVal;
                                                      VariantClear(&pVal);
                                                }
                                                externals.Add(edi);
                                          }
                                          deviceId = cstr;
                                    }
                                    SAFE_RELEASE(pCOMDev);
                              }
                              SAFE_RELEASE(pEnumCOMDevs2);
```

```
                        }
                        SAFE_RELEASE(pEnumCOMDevs);
                        DebugPrint(_T("OK:Win32_USBController"));
                  }
                  catch(...)
                  {
                        DebugPrint(_T("EX:Win32_USBController"));
                  }

                  try
                  {// Win32_1394 Контролер
                        pIWbemServices->ExecQuery(SysAllocString(L"WQL"),
                              SysAllocString(L"select Name, DeviceID from
Win32_1394Controller"), WBEM_FLAG_FORWARD_ONLY | WBEM_FLAG_RETURN_IMMEDIATELY,
NULL, &pEnumCOMDevs);
                        while(pEnumCOMDevs && SUCCEEDED(pEnumCOMDevs-
>Next(10000, 1, &pCOMDev, &uReturned)) && uReturned == 1)
                        {
                              VARIANT  pVal;
                              VariantClear(&pVal);
                              CString deviceId, channel;
                              if(pCOMDev->Get(L"DeviceID", 0L, &pVal, NULL,
NULL) == WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                              {
                                    deviceId = pVal.bstrVal;
                                    deviceId.Replace(_T("\\"), _T("\\\\"));
                                    VariantClear(&pVal);
                              }
                              SAFE_RELEASE(pCOMDev);

                              CString mapping, enclosure;
                              mapping.Format(_T("ASSOCIATORS OF
{Win32_1394Controller.DeviceID=\"%s\"} WHERE AssocClass =
Win32_1394ControllerDevice"), deviceId);
                              pIWbemServices->ExecQuery(SysAllocString(L"WQL"),
SysAllocString(mapping), WBEM_FLAG_FORWARD_ONLY | WBEM_FLAG_RETURN_IMMEDIATELY,
NULL, &pEnumCOMDevs2);
                              while(pEnumCOMDevs2 && SUCCEEDED(pEnumCOMDevs2-
>Next(10000, 1, &pCOMDev, &uReturned)) && uReturned == 1)
                              {
                                    VARIANT pVal;
                                    VariantClear(&pVal);
                                    if(pCOMDev->Get(L"DeviceID", 0L, &pVal,
NULL, NULL) == WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                                    {
                                          deviceId = pVal.bstrVal;
                                          VariantClear(&pVal);
                                    }
                                    SAFE_RELEASE(pCOMDev);
                              }
                              SAFE_RELEASE(pEnumCOMDevs2);
                        }
                        SAFE_RELEASE(pEnumCOMDevs);
                        DebugPrint(_T("OK:Win32_1394Controller"));
                  }
                  catch(...)
                  {
                        DebugPrint(_T("EX:Win32_1394Controller"));
                  }

                  /* РОЗШИФРУВАННЯ ЗНАЧЕНЬ
                  for(int i = 0; i < externals.GetCount(); i++)
                  {
                        CString cstr;
                        cstr.Format(_T("Enclosure=%s, VID=%04X, PID=%04X"),
                              externals.GetAt(i).Enclosure,
                              externals.GetAt(i).VendorId,
                              externals.GetAt(i).ProductId);
```

```
                        AfxMessageBox(cstr);
                  }
                  */

                  try
                  {
                        pIWbemServices->ExecQuery(SysAllocString(L"WQL"),
                              SysAllocString(L"SELECT * FROM Win32_DiskDrive"),
WBEM_FLAG_FORWARD_ONLY | WBEM_FLAG_RETURN_IMMEDIATELY, NULL, &pEnumCOMDevs);
                        DebugPrint(_T("DO:SELECT * FROM Win32_DiskDrive"));
                        while(pEnumCOMDevs && SUCCEEDED(pEnumCOMDevs-
>Next(10000, 1, &pCOMDev, &uReturned)) && uReturned == 1)
                        {
                              CString mapping1, mapping2;
                              CString model, deviceId, diskSize;
                              INT physicalDriveId = -1, scsiPort = -1,
scsiTargetId = -1;

                              VARIANT pVal;
                              VariantClear(&pVal);
                              if(pCOMDev->Get(L"Size", 0L, &pVal, NULL, NULL) ==
WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                              {
                                    diskSize = pVal.bstrVal;
                                    VariantClear(&pVal);
                              }
                              if(pCOMDev->Get(L"DeviceID", 0L, &pVal, NULL,
NULL) == WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                              {
                                    deviceId = pVal.bstrVal;
                                    deviceId.Replace(_T("\\"), _T("\\\\"));
                                    if(_ttoi(deviceId.Right(2)) >= 10)
                                    {
                                          physicalDriveId =
_ttoi(deviceId.Right(2));
                                    }
                                    else
                                    {
                                          physicalDriveId =
_ttoi(deviceId.Right(1));
                                    }
                                    VariantClear(&pVal);
                              }
                              if(pCOMDev->Get(L"Model", 0L, &pVal, NULL, NULL)
== WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                              {
                                    model = pVal.bstrVal;
                                    VariantClear(&pVal);
                              }
                              if(pCOMDev->Get(L"SCSIPort", 0L, &pVal, NULL,
NULL) == WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                              {
                                    scsiPort = pVal.intVal;
                                    VariantClear(&pVal);
                              }
                              if(pCOMDev->Get(L"SCSITargetId", 0L, &pVal, NULL,
NULL) == WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                              {
                                    scsiTargetId = pVal.intVal;
                                    VariantClear(&pVal);
                              }
                              SAFE_RELEASE(pCOMDev);

                              try
                              {
                                    // GetDiskInfo - інформація про диск
                                    CString cstr;
                                    cstr.Format(_T("DO:GetDiskInfo pd=%d, sp=%d,
st=%d"), physicalDriveId, scsiPort, scsiTargetId);
```

```
                                    DebugPrint(cstr);

                                    INTERFACE_TYPE interfaceType =
INTERFACE_TYPE_UNKNOWN;
                                    VENDOR_ID vendorId = VENDOR_UNKNOWN;

                                    if(model.Find(_T(" USB Device")) >= 0)
                                    {
                                          interfaceType = INTERFACE_TYPE_USB;
                                    }
                                    else if(model.Find(_T(" IEEE 1394 SBP2
Device")) >= 0)
                                    {
                                          interfaceType =
INTERFACE_TYPE_IEEE1394;
                                    }

                                    for(int i = 0; i < externals.GetCount();
i++)
                                    {

      if(model.Find(externals.GetAt(i).Enclosure) == 0)
                                          {
                                                vendorId =
(VENDOR_ID)externals.GetAt(i).VendorId;
                                          }
                                    }

                                    if(GetDiskInfo(physicalDriveId, scsiPort,
scsiTargetId, interfaceType, vendorId))
                                    {
                                          int index = (int)vars.GetCount() - 1;
                                          if(! diskSize.IsEmpty())
                                          {
                                                DWORD totalDiskSize =
(DWORD)(_ttoi64(diskSize) / 1000 / 1000 - 50);
                                                if((totalDiskSize <
vars[index].TotalDiskSize - 100
                                                || vars[index].TotalDiskSize +
100 < totalDiskSize)
                                                &&
vars[index].DiskSizeLba48
                                                )
                                                {
                                                      vars[index].TotalDiskSize =
totalDiskSize;
                                                }
                                          }

                                          BOOL flagSkipModelCheck = FALSE;

                                          vars[index].ModelWmi = model;
                                          // Модель
                                          model.Replace(_T(" SCSI Disk Device"),
_T(""));
                                          model.Replace(_T(" ATA Device"),
_T(""));

                                          if(model.Replace(_T(" USB Device"),
_T("")) > 0)
                                          {
                                                flagSkipModelCheck = TRUE;
                                                cstr.Format(_T("USB (%s)"),
vars[index].Interface);
                                                vars[index].Interface = cstr;
                                                vars[index].InterfaceType =
INTERFACE_TYPE_USB;
```

```
                                                for(int i = 0; i <
externals.GetCount(); i++)
                                                {

      if(externals.GetAt(i).Enclosure.Find(vars[index].ModelWmi) == 0)
                                                      {
                                                            vars[index].Enclosure
= externals.GetAt(i).Enclosure;
                                                            vars[index].VendorId
= externals.GetAt(i).VendorId;
                                                            vars[index].ProductId
= externals.GetAt(i).ProductId;
                                                      }
                                                }
                                          }
                                          else if(model.Replace(_T(" IEEE 1394
SBP2 Device"), _T("")) > 0)
                                          {
                                                flagSkipModelCheck = TRUE;
                                                cstr.Format(_T("IEEE 1394 (%s)"),
vars[index].Interface);
                                                vars[index].Interface = cstr;
                                                vars[index].InterfaceType =
INTERFACE_TYPE_IEEE1394;
                                                for(int i = 0; i <
externals.GetCount(); i++)
                                                {

      if(externals.GetAt(i).Enclosure.Find(vars[index].ModelWmi) == 0)
                                                      {
                                                            vars[index].Enclosure
= externals.GetAt(i).Enclosure;
                                                            vars[index].VendorId
= externals.GetAt(i).VendorId;
                                                            vars[index].ProductId
= externals.GetAt(i).ProductId;
                                                      }
                                                }
                                          }

                                          CString cmp, cmp1, cmp2, cmp3;
                                          cmp = model;
                                          cmp.Replace(_T(" "), _T(""));
                                          cmp1 = cmp.Left(16);

                                          cmp = vars[index].Model;
                                          cmp.Replace(_T(" "), _T(""));
                                          cmp2 = cmp.Left(16);

                                          cmp = vars[index].ModelReverse;
                                          cmp.Replace(_T(" "), _T(""));
                                          cmp3 = cmp.Left(16);

                                          if(vars[index].Model.IsEmpty())
                                          {
                                                vars.RemoveAt(index);
                                          }
                                          else if(flagSkipModelCheck)
                                          {
                                                // Не використовується
                                          }
                                          else if(model.IsEmpty() ||
cmp1.Compare(cmp2) == 0)
                                          {
                                                // Не використовується
                                          }
                                          else if(cmp1.Compare(cmp3) == 0)
                                          {
```

```
                                                vars[index].SerialNumber =
vars[index].SerialNumberReverse;
                                                vars[index].FirmwareRev =
vars[index].FirmwareRevReverse;
                                                vars[index].Model =
vars[index].ModelReverse;
                                                vars[index].ModelSerial =
vars[index].Model + vars[index].SerialNumber;

      vars[index].ModelSerial.Replace(_T("/"), _T(""));
                                          }
                                          else
                                          {
                                                vars.RemoveAt(index);
                                          }

                                          // РОЗШИФРУВАННЯ ЗНАЧЕНЬ
                                          // vars[index].VendorId =
VENDOR_MTRON;


                                          DebugPrint(_T("OK:Check Model Name"));
                                    }
                              }
                              catch(...)
                              {
                                    DebugPrint(_T("EX:GetDiskInfo pd=%d, sp=%d,
st=%d"));
                              }
                        }
                        SAFE_RELEASE(pEnumCOMDevs);
                        DebugPrint(_T("OK:SELECT * FROM Win32_DiskDrive"));
                  }
                  catch(...)
                  {
                        DebugPrint(_T("EX:SELECT * FROM Win32_DiskDrive"));
                  }

                  // Список дисків
                  try
                  {
                        DWORD driveLetterMap[256] = {0};

                        pIWbemServices->ExecQuery(SysAllocString(L"WQL"),
                              SysAllocString(L"SELECT * FROM
Win32_DiskPartition"), WBEM_FLAG_FORWARD_ONLY | WBEM_FLAG_RETURN_IMMEDIATELY,
NULL, &pEnumCOMDevs);
                        while(pEnumCOMDevs && SUCCEEDED(pEnumCOMDevs-
>Next(10000, 1, &pCOMDev, &uReturned)) && uReturned == 1)
                        {
                              DWORD physicalDriveId = 0;
                              CString partition, drive, mapping, cstr;
                              VARIANT pVal;
                              VariantClear(&pVal);
                              if(pCOMDev->Get(L"DeviceID", 0L, &pVal, NULL,
NULL) == WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                              {
                                    partition = pVal.bstrVal;
                                    VariantClear(&pVal);
                              }
                              cstr = partition;
                              cstr.Replace(_T("Disk #"), _T(""));
                              physicalDriveId = _ttoi(cstr);
                              SAFE_RELEASE(pCOMDev);

                              pIWbemServices->ExecQuery(SysAllocString(L"WQL"),
                                    SysAllocString(L"SELECT * FROM
Win32_LogicalDisk Where DriveType = 3"), WBEM_FLAG_FORWARD_ONLY |
WBEM_FLAG_RETURN_IMMEDIATELY, NULL, &pEnumCOMDevs2);
```

```
                              while(pEnumCOMDevs2 && SUCCEEDED(pEnumCOMDevs2-
>Next(10000, 1, &pCOMDev, &uReturned)) && uReturned == 1)
                              {
                                    VariantClear(&pVal);
                                    if(pCOMDev->Get(L"DeviceID", 0L, &pVal,
NULL, NULL) == WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                                    {
                                          drive = pVal.bstrVal;
                                          VariantClear(&pVal);
                                    }
                                    SAFE_RELEASE(pCOMDev);

                                    IWbemContext *pCtx = 0;
                                    IWbemCallResult *pResult = 0;

      mapping.Format(_T("Win32_LogicalDiskToPartition.Antecedent=\"Win32_DiskPar
tition.DeviceID=\\\"%s\\\"\",Dependent=\"Win32_LogicalDisk.DeviceID=\\\"%s\\\"\"
"),
                                          partition, drive);

                                    BSTR bstr;
                                    bstr = mapping.AllocSysString();
                                    pIWbemServices->GetObject(bstr, 0, pCtx,
&pCOMDev, &pResult);
                                    SysFreeString(bstr);
                                    if(pCOMDev)
                                    {
                                          driveLetterMap[physicalDriveId] |= 1
<< (drive.GetAt(0) - 'A');
                                    }
                                    SAFE_RELEASE(pCOMDev);
                              }
                              SAFE_RELEASE(pEnumCOMDevs2);
                        }
                        SAFE_RELEASE(pEnumCOMDevs);

                        for(int i = 0; i < vars.GetCount(); i++)
                        {
                              CString driveLetter = _T("");
                              for(int j = 0; j < 26; j++)
                              {
                                    if(driveLetterMap[vars[i].PhysicalDriveId] &
(1 << j))
                                    {
                                          CString cstr;
                                          cstr.Format(_T("%C"), j + 'A');
                                          driveLetter += cstr + _T(": ");
                                          vars[i].DriveLetterMap += (1 << j);
                                    }
                              }
                              vars[i].DriveMap.Append(driveLetter);
                        }
                        DebugPrint(_T("OK:Список дисків"));

                  }
                  catch(...)
                  {
                        DebugPrint(_T("EX:Список дисків"));
                  }

                  SAFE_RELEASE(pCOMDev);
                  SAFE_RELEASE(pEnumCOMDevs);
                  SAFE_RELEASE(pEnumCOMDevs2);
                  SAFE_RELEASE(pIWbemServices);
                  CoUninitialize();

                  DebugPrint(_T("OK:CoUninitialize()"));
            }
      }
```

```
	else
	{
		DebugPrint(_T("CAtaLifeLeft::Init WMI off - Start"));
	}

	// \\\\.\\PhysicalDrive%d
	for(int i = 0; i < MAX_SEARCH_PHYSICAL_DRIVE; i++)
	{
		BOOL  bRet;
		HANDLE	hIoCtrl;
		DWORD dwReturned;
		DISK_GEOMETRY dg;

		hIoCtrl = GetIoCtrlHandle(i);
		if(hIoCtrl == INVALID_HANDLE_VALUE)
		{
			continue;
		}
		bRet = ::DeviceIoControl(hIoCtrl, IOCTL_DISK_GET_DRIVE_GEOMETRY,
			NULL, 0, &dg, sizeof(DISK_GEOMETRY),
			&dwReturned, NULL);
		::CloseHandle(hIoCtrl);
		if(bRet == FALSE || dwReturned != sizeof(DISK_GEOMETRY) ||
dg.MediaType != FixedMedia)
		{
			continue;
		}
// Визначення виробника
		if(GetDiskInfo(i, -1, -1, INTERFACE_TYPE_UNKNOWN, VENDOR_UNKNOWN))
		{
			int index = (int)vars.GetCount() - 1;
			CString cmp;
			cmp = vars[index].Model;
			if(cmp.Find(_T("DW C")) == 0 // WDC
			|| cmp.Find(_T("iHat")) == 0 // Hitachi
			|| cmp.Find(_T("ASSM")) == 0 // SAMSUNG
			|| cmp.Find(_T("aMtx")) == 0 // Maxtor
			|| cmp.Find(_T("OTHS")) == 0 // TOSHIBA
			|| cmp.Find(_T("UFIJ")) == 0 // FUJITSU
			)
			{
				vars[index].SerialNumber =
vars[index].SerialNumberReverse;
				vars[index].FirmwareRev =
vars[index].FirmwareRevReverse;
				vars[index].Model = vars[index].ModelReverse;
				vars[index].ModelSerial = vars[index].Model +
vars[index].SerialNumber;
				vars[index].ModelSerial.Replace(_T("/"), _T(""));
			}
		}
	}

	// сортування
	ATA_LifeLeft_INFO* p = vars.GetData();
	qsort(p, vars.GetCount(), sizeof(ATA_LifeLeft_INFO), Compare);

	DebugPrint(_T("OK:GetDiskInfo - PhysicalDrive"));

	// Визначення типу знайдених дисків
	if(advancedDiskSearch)
	{
		// \\\\.\\Scsi%d:
		for(int i = 0; i < MAX_SEARCH_SCSI_PORT; i++)
		{
			for(int j = 0; j < MAX_SEARCH_SCSI_TARGET_ID; j++)
			{
				if(GetDiskInfo(-1, i, j, INTERFACE_TYPE_UNKNOWN,
VENDOR_UNKNOWN))
```

```
                        {
                              int index = (int)vars.GetCount() - 1;
                              CString cmp;
                              cmp = vars[index].Model;
                              if(cmp.Find(_T("DW C")) == 0 // WDC
                              || cmp.Find(_T("iHat")) == 0 // Hitachi
                              || cmp.Find(_T("ASSM")) == 0 // SAMSUNG
                              || cmp.Find(_T("aMtx")) == 0 // Maxtor
                              || cmp.Find(_T("OTHS")) == 0 // TOSHIBA
                              || cmp.Find(_T("UFIJ")) == 0 // FUJITSU
                              )
                              {
                                    vars[index].SerialNumber =
vars[index].SerialNumberReverse;
                                    vars[index].FirmwareRev =
vars[index].FirmwareRevReverse;
                                    vars[index].Model =
vars[index].ModelReverse;
                                    vars[index].ModelSerial = vars[index].Model
+ vars[index].SerialNumber;
                                    vars[index].ModelSerial.Replace(_T("/"),
_T(""));
                              }
                        }
                  }
            }
            DebugPrint(_T("OK:GetDiskInfo - Scsi"));
      }

      MeasuredGetTickCount = GetTickCount();
      DebugPrint(_T("CAtaLifeLeft::Init - Complete"));

      if(flagChangeDisk != NULL)
      {
            if(vars.GetCount() != previous.GetCount())
            {
                  *flagChangeDisk = TRUE;
            }
            else
            {
                  for(int i = 0; i < vars.GetCount(); i++)
                  {
                        if(vars.GetAt(i).PhysicalDriveId !=
previous.GetAt(i).PhysicalDriveId
                        || vars.GetAt(i).ScsiTargetId !=
previous.GetAt(i).ScsiTargetId
                        || vars.GetAt(i).ScsiPort != previous.GetAt(i).ScsiPort
                        )
                        {
                              *flagChangeDisk = TRUE;
                              break;
                        }
                  }
            }
      }

      return IsEnabledWmi;
}

int CAtaLifeLeft::Compare(const void *p1, const void *p2)
{
      return ((ATA_LifeLeft_INFO*)p1)->PhysicalDriveId -
((ATA_LifeLeft_INFO*)p2)->PhysicalDriveId;
}

BOOL CAtaLifeLeft::AddDisk(INT physicalDriveId, INT scsiPort, INT scsiTargetId,
COMMAND_TYPE commandType, IDENTIFY_DEVICE* identify)
{
      ATA_LifeLeft_INFO asi;
```

```
memcpy(&(asi.IdentifyDevice), identify, sizeof(IDENTIFY_DEVICE));
asi.PhysicalDriveId = physicalDriveId;
asi.ScsiPort =  scsiPort;
asi.ScsiTargetId = scsiTargetId;
asi.CommandType = commandType;
asi.CommandTypeString = commandTypeString[commandType];

for(int i = 0; i < MAX_ATTRIBUTE; i++)
{
      ::ZeroMemory(&(asi.Attribute[i]), sizeof(LifeLeft_ATTRIBUTE));
      ::ZeroMemory(&(asi.Threshold[i]), sizeof(LifeLeft_THRESHOLD));
}

asi.IsLifeLeftEnabled = FALSE;
asi.IsWord88 = FALSE;
asi.IsWord64_76 = FALSE;
asi.IsCheckSumError = FALSE;

asi.IsLifeLeftSupported = FALSE;
asi.IsLba48Supported = FALSE;
asi.IsNcqSupported = FALSE;
asi.IsAamSupported = FALSE;
asi.IsApmSupported = FALSE;
asi.IsAamEnabled = FALSE;
asi.IsApmEnabled = FALSE;
asi.IsNvCacheSupported = FALSE;
asi.IsMaxtorMinute = FALSE;

asi.TotalDiskSize = 0;
asi.Cylinder = 0;
asi.Head = 0;
asi.Sector = 0;
asi.Sector28 = 0;
asi.Sector48 = 0;
asi.DiskSizeChs = 0;
asi.DiskSizeLba28 = 0;
asi.DiskSizeLba48 = 0;
asi.BufferSize = 0;
asi.NvCacheSize = 0;
asi.TransferModeType = 0;
asi.DetectedTimeUnitType = 0;
asi.MeasuredTimeUnitType = 0;
asi.AttributeCount = 0;
asi.DetectedPowerOnHours = -1;
asi.MeasuredPowerOnHours = -1;
asi.PowerOnRawValue = -1;
asi.PowerOnStartRawValue = -1;
asi.PowerOnCount = 0;
asi.Temperature = 0;
asi.Speed = 0.0;
asi.Life = -1;

asi.Major = 0;
asi.Minor = 0;

asi.DiskStatus = 0;
asi.DriveLetterMap = 0;

asi.AlarmTemperature = 0;

asi.VendorId = VENDOR_UNKNOWN;
asi.InterfaceType = INTERFACE_TYPE_UNKNOWN;

asi.VendorId = 0;
asi.ProductId = 0;

asi.SerialNumber = _T("");
asi.FirmwareRev = _T("");
```

```
	asi.Model = _T("");
	asi.ModelReverse = _T("");
	asi.ModelWmi = _T("");
	asi.ModelSerial = _T("");
	asi.DriveMap = _T("");
	asi.MaxTransferMode = _T("");
	asi.CurrentTransferMode = _T("");
	asi.MajorVersion = _T("");
	asi.MinorVersion = _T("");
	asi.Interface = _T("");
	asi.Enclosure = _T("");

	CHAR buf[64];

	// Перевірка помилок
	BYTE sum = 0;
	BYTE checkSum[IDENTIFY_BUFFER_SIZE];
	memcpy(checkSum, (void *)identify, IDENTIFY_BUFFER_SIZE);
	for(int j = 0; j < IDENTIFY_BUFFER_SIZE; j++)
	{
		sum += checkSum[j];
	}
	if(sum != 0)
	{
		asi.IsCheckSumError = TRUE;
	}

	// Обооротні дії
	strncpy_s(buf, 64, identify->SerialNumber, sizeof(identify-
>SerialNumber));
	asi.SerialNumberReverse = buf;
	asi.SerialNumberReverse.TrimLeft();
	asi.SerialNumberReverse.TrimRight();
	strncpy_s(buf, 64, identify->FirmwareRev, sizeof(identify->FirmwareRev));
	asi.FirmwareRevReverse = buf;
	asi.FirmwareRevReverse.TrimLeft();
	asi.FirmwareRevReverse.TrimRight();
	strncpy_s(buf, 64, identify->Model, sizeof(identify->Model));
	asi.ModelReverse = buf;
	asi.ModelReverse.TrimLeft();
	asi.ModelReverse.TrimRight();

	ChangeByteOrder(identify->SerialNumber, sizeof(identify->SerialNumber));
	ChangeByteOrder(identify->FirmwareRev, sizeof(identify->FirmwareRev));
	ChangeByteOrder(identify->Model, sizeof(identify->Model));

	if(CheckAsciiStringError(identify->SerialNumber, sizeof(identify-
>SerialNumber))
	|| CheckAsciiStringError(identify->FirmwareRev, sizeof(identify-
>FirmwareRev))
	|| CheckAsciiStringError(identify->Model, sizeof(identify->Model)))
	{
		return FALSE;
	}



	// Запис даних у структуру
	strncpy_s(buf, 64, identify->SerialNumber, sizeof(identify-
>SerialNumber));
	asi.SerialNumber = buf;
	asi.SerialNumber.TrimLeft();
	asi.SerialNumber.TrimRight();
	strncpy_s(buf, 64, identify->FirmwareRev, sizeof(identify->FirmwareRev));
	asi.FirmwareRev = buf;
	asi.FirmwareRev.TrimLeft();
	asi.FirmwareRev.TrimRight();
	strncpy_s(buf, 64, identify->Model, sizeof(identify->Model));
	asi.Model = buf;
```

```
	asi.Model.TrimLeft();
	asi.Model.TrimRight();

	if(asi.Model.IsEmpty() || asi.FirmwareRev.IsEmpty())
	{
		return FALSE;
	}

// РОЗШИФРУВАННЯ ЗНАЧЕНЬ
//	asi.Model = _T(" MTRON ") + asi.Model;

	asi.ModelSerial = asi.Model + asi.SerialNumber;
	asi.ModelSerial.Replace(_T("/"), _T(""));

	asi.Major = GetAtaMajorVersion(identify->MajorVersion, asi.MajorVersion);
	asi.TransferModeType = GetTransferMode(identify->MultiWordDma, identify-
>SerialAtaCapabilities,
					identify->UltraDmaMode,
asi.CurrentTransferMode, asi.MaxTransferMode,
					asi.Interface, &asi.InterfaceType);
	asi.DetectedTimeUnitType = GetTimeUnitType(asi.Model, asi.FirmwareRev,
asi.Major, asi.TransferModeType);

	// Feature
	if(asi.Major >= 3 && asi.IdentifyDevice.CommandSetSupported1 & (1 << 0))
	{
		asi.IsLifeLeftSupported = TRUE;
	}
	if(asi.Major >= 3 && asi.IdentifyDevice.CommandSetSupported2 & (1 << 3))
	{
		asi.IsApmSupported = TRUE;
		if(asi.IdentifyDevice.CommandSetEnabled2 & (1 << 3))
		{
			asi.IsApmEnabled = TRUE;
		}
	}
	if(asi.Major >= 5 && asi.IdentifyDevice.CommandSetSupported2 & (1 << 9))
	{
		asi.IsAamSupported = TRUE;
		if(asi.IdentifyDevice.CommandSetEnabled2 & (1 << 9))
		{
			asi.IsAamEnabled = TRUE;
		}
	}

	if(asi.Major >= 5 && asi.IdentifyDevice.CommandSetSupported2 & (1 << 10))
	{
		asi.IsLba48Supported = TRUE;
	}

	if(asi.Major >= 6 && asi.IdentifyDevice.SerialAtaCapabilities & (1 << 8))
	{
		asi.IsNcqSupported = TRUE;
	}
	if(asi.Major >= 7 && asi.IdentifyDevice.NvCacheCapabilities & (1 << 0))
	{
		asi.IsNvCacheSupported = TRUE;
	}

	CString model = asi.Model;
	model.MakeUpper();
	if(model.Find(_T("MAXTOR")) == 0 && asi.DetectedTimeUnitType ==
POWER_ON_MINUTES)
	{
		asi.IsMaxtorMinute = TRUE;
	}

	// Розмір диску та розмір буферу
	asi.Cylinder = identify->LogicalCylinders;
```

```
	asi.Head = identify->LlogicalHeads;
	asi.Sector = identify->LogicalSectors;
	asi.Sector28 = identify->TotalAddressableSectors;
	asi.Sector48 = identify->MaxUserLba;
	asi.DiskSizeChs   = (DWORD)(((ULONGLONG)identify->LogicalCylinders *
identify->LlogicalHeads * identify->LogicalSectors * 512) / 1000 / 1000  - 50);
	asi.DiskSizeLba28 = (DWORD)(((ULONGLONG)identify->TotalAddressableSectors
* 512) / 1000 / 1000 - 50);
	if(asi.IsLba48Supported)
	{
		asi.DiskSizeLba48 = (DWORD)(((ULONGLONG)identify->MaxUserLba * 512)
/ 1000 / 1000 - 50);
	}
	asi.BufferSize = identify->BufferSize * 512;
	if(asi.IsNvCacheSupported)
	{
		asi.NvCacheSize = identify->NvCacheSizeLogicalBlocks * 512;
	}

	if(asi.DiskSizeChs == 0)
	{
		asi.TotalDiskSize = 0;
	}
	else if(asi.DiskSizeLba48 > asi.DiskSizeLba28)
	{
		asi.TotalDiskSize = asi.DiskSizeLba48;
	}
	else if(asi.DiskSizeLba28 > asi.DiskSizeChs)
	{
		asi.TotalDiskSize = asi.DiskSizeLba28;
	}
	else
	{
		asi.TotalDiskSize = asi.DiskSizeChs;
	}

	// Перевірка помилок для контролеру External ATA
	if(asi.IsLba48Supported && (identify->TotalAddressableSectors < 268435455
&& asi.DiskSizeLba28 != asi.DiskSizeLba48))
	{
		asi.DiskSizeLba48 = 0;
	}

	// SSD Життя
	if(asi.Model.Find(_T("MTRON")) == 0)
	{
		asi.VendorId = VENDOR_MTRON;
	}

	// перевірка підтримки S.M.A.R.T.
	switch(asi.CommandType)
	{
	case CMD_TYPE_PHYSICAL_DRIVE:
// РОЗШИФРУВАННЯ ЗНАЧЕНЬ
//		SendAtaCommandPd(physicalDriveId, 0xEF, 0x42, 0xFE);
//		SendAtaCommandPd(physicalDriveId, 0xEF, 0x05, 0xFE);

		if(GetLifeLeftAttributePd(physicalDriveId, &asi))
		{
			GetLifeLeftThresholdPd(physicalDriveId, &asi);
			asi.DiskStatus = CheckDiskStatus(asi.Attribute, asi.Threshold,
asi.AttributeCount, asi.VendorId);
			asi.IsLifeLeftEnabled = TRUE;
		}
		else if(ControlLifeLeftStatusPd(physicalDriveId, ENABLE_LifeLeft))
		{
			if(GetLifeLeftAttributePd(physicalDriveId, &asi))
			{
				GetLifeLeftThresholdPd(physicalDriveId, &asi);
```

```
                        asi.DiskStatus = CheckDiskStatus(asi.Attribute,
asi.Threshold, asi.AttributeCount, asi.VendorId);
                        asi.IsLifeLeftEnabled = TRUE;
                  }
            }
            break;
      case CMD_TYPE_SCSI_MINIPORT:
// РОЗШИФРУВАННЯ ЗНАЧЕНЬ
//          SendAtaCommandScsi(scsiPort, scsiTargetId, 0xEC, 0x00, 0x00); // ID
//          SendAtaCommandScsi(scsiPort, scsiTargetId, 0xEF, 0x05, 0x80); // APM
//          SendAtaCommandScsi(scsiPort, scsiTargetId, 0xEF, 0x42, 0x80); // AAM

            if(GetLifeLeftAttributeScsi(scsiPort, scsiTargetId, &asi))
            {
                  GetLifeLeftThresholdScsi(scsiPort, scsiTargetId, &asi);
                  asi.DiskStatus = CheckDiskStatus(asi.Attribute, asi.Threshold,
asi.AttributeCount, asi.VendorId);
                  asi.IsLifeLeftEnabled = TRUE;
            }
            else if(ControlLifeLeftStatusScsi(scsiPort, scsiTargetId,
ENABLE_LifeLeft))
            {
                  if(GetLifeLeftAttributeScsi(scsiPort, scsiTargetId, &asi))
                  {
                        GetLifeLeftThresholdScsi(scsiPort, scsiTargetId, &asi);
                        asi.DiskStatus = CheckDiskStatus(asi.Attribute,
asi.Threshold, asi.AttributeCount, asi.VendorId);
                        asi.IsLifeLeftEnabled = TRUE;
                  }
            }
            break;
      case CMD_TYPE_SAT:
      case CMD_TYPE_SUNPLUS:
      case CMD_TYPE_IO_DATA:
      case CMD_TYPE_LOGITEC:
      case CMD_TYPE_JMICRON:
      case CMD_TYPE_CYPRESS:
// РОЗШИФРУВАННЯ ЗНАЧЕНЬ
//          SendAtaCommandSat(physicalDriveId, 0xEF, 0x05, 0xC0,
asi.CommandType);
//          SendAtaCommandSat(physicalDriveId, 0xEF, 0x42, 0xC0,
asi.CommandType);

            if(GetLifeLeftAttributeSat(physicalDriveId, &asi))
            {
                  GetLifeLeftThresholdSat(physicalDriveId, &asi);
                  asi.DiskStatus = CheckDiskStatus(asi.Attribute, asi.Threshold,
asi.AttributeCount, asi.VendorId);
                  asi.IsLifeLeftEnabled = TRUE;
            }
            else if(ControlLifeLeftStatusSat(physicalDriveId, ENABLE_LifeLeft,
asi.CommandType))
            {
                  if(GetLifeLeftAttributeSat(physicalDriveId, &asi))
                  {
                        GetLifeLeftThresholdSat(physicalDriveId, &asi);
                        asi.DiskStatus = CheckDiskStatus(asi.Attribute,
asi.Threshold, asi.AttributeCount, asi.VendorId);
                        asi.IsLifeLeftEnabled = TRUE;
                  }
            }
            break;
      default:
            return FALSE;
            break;
      }

      for(int i = 0; i < vars.GetCount(); i++)
      {
```

```
            if(asi.Model.Compare(vars[i].Model) == 0 &&
asi.SerialNumber.Compare(vars[i].SerialNumber) == 0)
            {
                  return FALSE;
            }
      }

      asi.PowerOnStartRawValue = asi.PowerOnRawValue;

      vars.Add(asi);

      return TRUE;
}

BOOL CAtaLifeLeft::GetDiskInfo(INT physicalDriveId, INT scsiPort, INT
scsiTargetId, INTERFACE_TYPE interfaceType, VENDOR_ID vendorId)
{
      if(vars.GetCount() > MAX_DISK)
      {
            return FALSE;
      }
      // Перевірка перекриття
      for(int i = 0; i < vars.GetCount(); i++)
      {
            if(physicalDriveId >= 0 && vars[i].PhysicalDriveId ==
physicalDriveId)
            {
                  return FALSE;
            }
            else if(scsiPort >= 0 && scsiTargetId >= 0
                  &&    vars[i].ScsiPort == scsiPort && vars[i].ScsiTargetId ==
scsiTargetId)
            {
                  return FALSE;
            }
      }

      IDENTIFY_DEVICE identify = {0};

      if(interfaceType == INTERFACE_TYPE_UNKNOWN || interfaceType ==
INTERFACE_TYPE_PATA || interfaceType == INTERFACE_TYPE_SATA)
      {
            if(physicalDriveId >= 0 && DoIdentifyDevicePd(physicalDriveId,
&identify))
            {
                  return AddDisk(physicalDriveId, scsiPort, scsiTargetId,
CMD_TYPE_PHYSICAL_DRIVE, &identify);
            }
            else if(scsiPort >= 0 && scsiTargetId >= 0 &&
DoIdentifyDeviceScsi(scsiPort, scsiTargetId, &identify))
            {
                  return AddDisk(physicalDriveId, scsiPort, scsiTargetId,
CMD_TYPE_SCSI_MINIPORT, &identify);
            }
      }
      else if(physicalDriveId >= 0)
      {
            /** РОЗШИФРУВАННЯ ЗНАЧЕНЬ
            if(TRUE)
            {
                  DoIdentifyDeviceSat(physicalDriveId, &identify,
CMD_TYPE_DEBUG);
            }
            else
            */
            if(interfaceType == INTERFACE_TYPE_USB && vendorId ==
USB_VENDOR_IO_DATA)
            {
```

```
                if(DoIdentifyDeviceSat(physicalDriveId, &identify,
CMD_TYPE_IO_DATA))
                {
                        return AddDisk(physicalDriveId, scsiPort, scsiTargetId,
CMD_TYPE_IO_DATA, &identify);
                }
            }
            if(interfaceType == INTERFACE_TYPE_USB && vendorId ==
USB_VENDOR_LOGITEC)
            {
                if(DoIdentifyDeviceSat(physicalDriveId, &identify,
CMD_TYPE_LOGITEC))
                {
                        return AddDisk(physicalDriveId, scsiPort, scsiTargetId,
CMD_TYPE_LOGITEC, &identify);
                }
            }
            if(interfaceType == INTERFACE_TYPE_USB && vendorId ==
USB_VENDOR_SUNPLUS)
            {
                if(DoIdentifyDeviceSat(physicalDriveId, &identify,
CMD_TYPE_SUNPLUS))
                {
                        return AddDisk(physicalDriveId, scsiPort, scsiTargetId,
CMD_TYPE_SUNPLUS, &identify);
                }
            }
            else if(interfaceType == INTERFACE_TYPE_USB && vendorId ==
USB_VENDOR_CYPRESS)
            {
                if(DoIdentifyDeviceSat(physicalDriveId, &identify,
CMD_TYPE_CYPRESS))
                {
                        return AddDisk(physicalDriveId, scsiPort, scsiTargetId,
CMD_TYPE_CYPRESS, &identify);
                }
            }
            else if(interfaceType == INTERFACE_TYPE_USB &&
                (vendorId == USB_VENDOR_INITIO || vendorId ==
USB_VENDOR_OXFORD)
                )
            {
                if(DoIdentifyDeviceSat(physicalDriveId, &identify,
CMD_TYPE_SAT))
                {
                        return AddDisk(physicalDriveId, scsiPort, scsiTargetId,
CMD_TYPE_SAT, &identify);
                }
            }
            else
            {
                if(DoIdentifyDeviceSat(physicalDriveId, &identify,
CMD_TYPE_SAT))
                {
                        return AddDisk(physicalDriveId, scsiPort, scsiTargetId,
CMD_TYPE_SAT, &identify);
                }
                else if(DoIdentifyDeviceSat(physicalDriveId, &identify,
CMD_TYPE_JMICRON))
                {
                        return AddDisk(physicalDriveId, scsiPort, scsiTargetId,
CMD_TYPE_JMICRON, &identify);
                }
                else if(DoIdentifyDeviceSat(physicalDriveId, &identify,
CMD_TYPE_SUNPLUS))
                {
                        return AddDisk(physicalDriveId, scsiPort, scsiTargetId,
CMD_TYPE_SUNPLUS, &identify);
                }
```

```
                  else if(DoIdentifyDeviceSat(physicalDriveId, &identify,
CMD_TYPE_CYPRESS))
                  {
                        return AddDisk(physicalDriveId, scsiPort, scsiTargetId,
CMD_TYPE_CYPRESS, &identify);
                  }
                  else if(DoIdentifyDeviceSat(physicalDriveId, &identify,
CMD_TYPE_LOGITEC))
                  {
                        return AddDisk(physicalDriveId, scsiPort, scsiTargetId,
CMD_TYPE_LOGITEC, &identify);
                  }
            }
      }

      return FALSE;
}

/*-------------------------------------------------------------------------*/
// \\\\.\\Фізичний диск X
/*-------------------------------------------------------------------------*/
HANDLE CAtaLifeLeft::GetIoCtrlHandle(BYTE index)
{
      CString     strDevice;
      strDevice.Format(_T("\\\\.\\PhysicalDrive%d"), index);

      return ::CreateFile(strDevice, GENERIC_READ | GENERIC_WRITE,
            FILE_SHARE_READ | FILE_SHARE_WRITE,
            NULL, OPEN_EXISTING, 0, NULL);
}

BOOL CAtaLifeLeft::DoIdentifyDevicePd(INT physicalDriveId, IDENTIFY_DEVICE*
data)
{
      BOOL  bRet;
      HANDLE      hIoCtrl;
      DWORD dwReturned;

      IDENTIFY_DEVICE_OUTDATA sendCmdOutParam;
      SENDCMDINPARAMS   sendCmd;

      if(data == NULL)
      {
            return      FALSE;
      }

      if(! SendAtaCommandPd(physicalDriveId, 0xEC, 0x00, 0x00, (PBYTE)data,
sizeof(IDENTIFY_DEVICE)))
      {
            ::ZeroMemory(data, sizeof(IDENTIFY_DEVICE));
            hIoCtrl = GetIoCtrlHandle(physicalDriveId);
            if(hIoCtrl == INVALID_HANDLE_VALUE)
            {
                  return      FALSE;
            }
            ::ZeroMemory(&sendCmdOutParam, sizeof(IDENTIFY_DEVICE_OUTDATA));
            ::ZeroMemory(&sendCmd, sizeof(SENDCMDINPARAMS));

            sendCmd.irDriveRegs.bCommandReg                 = ID_CMD;
            sendCmd.irDriveRegs.bSectorCountReg       = 1;
            sendCmd.irDriveRegs.bSectorNumberReg      = 1;
            sendCmd.cBufferSize                                   =
IDENTIFY_BUFFER_SIZE;

            bRet = ::DeviceIoControl(hIoCtrl, DFP_RECEIVE_DRIVE_DATA,
                  &sendCmd, sizeof(SENDCMDINPARAMS),
                  &sendCmdOutParam, sizeof(IDENTIFY_DEVICE_OUTDATA),
                  &dwReturned, NULL);
```

```
            ::CloseHandle(hIoCtrl);

            if(bRet == FALSE || dwReturned != sizeof(IDENTIFY_DEVICE_OUTDATA))
            {
                  return      FALSE;
            }

            memcpy_s(data, sizeof(IDENTIFY_DEVICE),
sendCmdOutParam.SendCmdOutParam.bBuffer, sizeof(IDENTIFY_DEVICE));
      }

      return      TRUE;
}

BOOL CAtaLifeLeft::GetLifeLeftAttributePd(INT PhysicalDriveId,
ATA_LifeLeft_INFO* asi)
{
      BOOL  bRet;
      HANDLE      hIoCtrl;
      DWORD dwReturned;

      LifeLeft_READ_DATA_OUTDATA    sendCmdOutParam;
      SENDCMDINPARAMS   sendCmd;

      hIoCtrl = GetIoCtrlHandle(PhysicalDriveId);
      if(hIoCtrl == INVALID_HANDLE_VALUE)
      {
            return      FALSE;
      }

      ::ZeroMemory(&sendCmdOutParam, sizeof(LifeLeft_READ_DATA_OUTDATA));
      ::ZeroMemory(&sendCmd, sizeof(SENDCMDINPARAMS));

      sendCmd.irDriveRegs.bFeaturesReg    = READ_ATTRIBUTES;
      sendCmd.irDriveRegs.bSectorCountReg = 1;
      sendCmd.irDriveRegs.bSectorNumberReg= 1;
      sendCmd.irDriveRegs.bCylLowReg           = LifeLeft_CYL_LOW;
      sendCmd.irDriveRegs.bCylHighReg          = LifeLeft_CYL_HI;
      sendCmd.irDriveRegs.bCommandReg          = LifeLeft_CMD;
      sendCmd.cBufferSize                            =
READ_ATTRIBUTE_BUFFER_SIZE;

      bRet = ::DeviceIoControl(hIoCtrl, DFP_RECEIVE_DRIVE_DATA,
            &sendCmd, sizeof(SENDCMDINPARAMS),
            &sendCmdOutParam, sizeof(LifeLeft_READ_DATA_OUTDATA),
            &dwReturned, NULL);

      ::CloseHandle(hIoCtrl);

      if(bRet == FALSE || dwReturned != sizeof(LifeLeft_READ_DATA_OUTDATA))
      {
            return      FALSE;
      }

      CString str;
      asi->AttributeCount = 0;
      int j = 0;
      for(int i = 0; i < MAX_ATTRIBUTE; i++)
      {
            DWORD rawValue = 0;
            memcpy(     &(asi->Attribute[j]),
                        &(sendCmdOutParam.Data[i * sizeof(LifeLeft_ATTRIBUTE) +
1]), sizeof(LifeLeft_ATTRIBUTE));

            if(asi->Attribute[j].Id != 0)
            {
                  switch(asi->Attribute[j].Id)
                  {
```

```
                  case 0x09: // Кількість відпрацьованих годин у включеному
стані
                        rawValue = MAKELONG(
                              MAKEWORD(asi->Attribute[j].RawValue[0], asi-
>Attribute[j].RawValue[1]),
                              MAKEWORD(asi->Attribute[j].RawValue[2], asi-
>Attribute[j].RawValue[3])
                              );
                        asi->PowerOnRawValue = rawValue;
                        asi->DetectedPowerOnHours = GetPowerOnHours(rawValue,
asi->DetectedTimeUnitType);
                        asi->MeasuredPowerOnHours = GetPowerOnHours(rawValue,
asi->MeasuredTimeUnitType);
                        break;
                  case 0x0C: // Кількість зафіксованих повторів
включення/вимикання живлення накопичувача
                        rawValue = MAKELONG(
                              MAKEWORD(asi->Attribute[j].RawValue[0], asi-
>Attribute[j].RawValue[1]),
                              MAKEWORD(asi->Attribute[j].RawValue[2], asi-
>Attribute[j].RawValue[3])
                              );
                        asi->PowerOnCount = rawValue;
                        break;
                  case 0xC2: // Температура
                        if(asi->Attribute[j].RawValue[0] > 0)
                        {
                              asi->Temperature = asi->Attribute[j].RawValue[0];
                        }
                        else
                        {
                              asi->Temperature = asi->Attribute[j].CurrentValue;
                        }
                  case 0xBB: // Визначення фірми-розробника
                        if(asi->VendorId == VENDOR_MTRON)
                        {
                              asi->Life = asi->Attribute[j].CurrentValue;
                        }
                        break;
                  default:
                        break;
                  }
                  j++;
            }
      }
      asi->AttributeCount = j;

      if(asi->AttributeCount > 0)
      {
            return TRUE;
      }
      else
      {
            return FALSE;
      }
}

BOOL CAtaLifeLeft::GetLifeLeftThresholdPd(INT physicalDriveId,
ATA_LifeLeft_INFO* asi)
{
      BOOL  bRet;
      HANDLE      hIoCtrl;
      DWORD dwReturned;

      LifeLeft_READ_DATA_OUTDATA    sendCmdOutParam;
      SENDCMDINPARAMS   sendCmd;

      hIoCtrl = GetIoCtrlHandle(physicalDriveId);
      if(hIoCtrl == INVALID_HANDLE_VALUE)
```

```
	{
		return	FALSE;
	}

	::ZeroMemory(&sendCmdOutParam, sizeof(LifeLeft_READ_DATA_OUTDATA));
	::ZeroMemory(&sendCmd, sizeof(SENDCMDINPARAMS));

	sendCmd.irDriveRegs.bFeaturesReg    = READ_THRESHOLDS;
	sendCmd.irDriveRegs.bCylLowReg           = LifeLeft_CYL_LOW;
	sendCmd.irDriveRegs.bCylHighReg          = LifeLeft_CYL_HI;
	sendCmd.irDriveRegs.bCommandReg          = LifeLeft_CMD;
	sendCmd.cBufferSize                           =
READ_THRESHOLD_BUFFER_SIZE;

	bRet = ::DeviceIoControl(hIoCtrl, DFP_RECEIVE_DRIVE_DATA,
		&sendCmd, sizeof(SENDCMDINPARAMS),
		&sendCmdOutParam, sizeof(LifeLeft_READ_DATA_OUTDATA),
		&dwReturned, NULL);

	::CloseHandle(hIoCtrl);

	if(bRet == FALSE || dwReturned != sizeof(LifeLeft_READ_DATA_OUTDATA))
	{
		return	FALSE;
	}

	CString str;
	int j = 0;
	for(int i = 0; i < MAX_ATTRIBUTE; i++)
	{
		memcpy(	&(asi->Threshold[i]),
			&(sendCmdOutParam.Data[i * sizeof(LifeLeft_THRESHOLD) +
1]), sizeof(LifeLeft_THRESHOLD));

		if(asi->Threshold[j].Id != 0)
		{
			j++;
		}
	}

	return	TRUE;
}

BOOL CAtaLifeLeft::ControlLifeLeftStatusPd(INT physicalDriveId, BYTE command)
{
	BOOL  bRet;
	HANDLE	hIoCtrl;
	DWORD dwReturned;

	SENDCMDINPARAMS		sendCmd;
	SENDCMDOUTPARAMS  sendCmdOutParam;

	hIoCtrl = GetIoCtrlHandle(physicalDriveId);
	if(hIoCtrl == INVALID_HANDLE_VALUE)
	{
		return	FALSE;
	}

	::ZeroMemory(&sendCmd, sizeof(SENDCMDINPARAMS));
	::ZeroMemory(&sendCmdOutParam, sizeof(SENDCMDOUTPARAMS));

	sendCmd.irDriveRegs.bFeaturesReg    = command;
	sendCmd.irDriveRegs.bSectorCountReg = 1;
	sendCmd.irDriveRegs.bSectorNumberReg= 1;
	sendCmd.irDriveRegs.bCylLowReg           = LifeLeft_CYL_LOW;
	sendCmd.irDriveRegs.bCylHighReg          = LifeLeft_CYL_HI;
	sendCmd.irDriveRegs.bCommandReg          = LifeLeft_CMD;
	sendCmd.cBufferSize                           = 0;
```

```
	bRet = ::DeviceIoControl(hIoCtrl, DFP_SEND_DRIVE_COMMAND,
		&sendCmd, sizeof(SENDCMDINPARAMS) - 1,
		&sendCmdOutParam, sizeof(SENDCMDOUTPARAMS) -1,
		&dwReturned, NULL);

	::CloseHandle(hIoCtrl);

	return	bRet;
}

BOOL CAtaLifeLeft::SendAtaCommandPd(INT physicalDriveId, BYTE main, BYTE sub,
BYTE param, PBYTE data, DWORD dataSize)
{
	BOOL  bRet;
	HANDLE	hIoCtrl;
	DWORD dwReturned;

	hIoCtrl = GetIoCtrlHandle(physicalDriveId);
	if(hIoCtrl == INVALID_HANDLE_VALUE)
	{
		return	FALSE;
	}

	if(m_FlagAtaPassThrough)
	{
		ATA_PASS_THROUGH_EX_WITH_BUFFERS ab;
		::ZeroMemory(&ab, sizeof(ab));
		ab.Apt.Length = sizeof(ATA_PASS_THROUGH_EX);
		ab.Apt.TimeOutValue = 10;
		DWORD size = offsetof(ATA_PASS_THROUGH_EX_WITH_BUFFERS, Buf);
		ab.Apt.DataBufferOffset = size;

		if(dataSize > 0)
		{
			if(dataSize > sizeof(ab.Buf))
			{
				return FALSE;
			}
			ab.Apt.AtaFlags = ATA_FLAGS_DATA_IN;
			ab.Apt.DataTransferLength = dataSize;
			size += dataSize;
		}

		ab.Apt.CurrentTaskFile.bFeaturesReg = sub;
		ab.Apt.CurrentTaskFile.bSectorCountReg = param;
		ab.Apt.CurrentTaskFile.bCommandReg = main;

		bRet = ::DeviceIoControl(hIoCtrl, IOCTL_ATA_PASS_THROUGH,
			&ab, size, &ab, size, &dwReturned, NULL);
		::CloseHandle(hIoCtrl);
		if(bRet && dataSize && data != NULL)
		{
			memcpy_s(data, dataSize, ab.Buf, dataSize);
		}
	}
	else if(m_Os.dwMajorVersion == 4)
	{
		return FALSE;
	}
	else
	{
		DWORD size = sizeof(CMD_IDE_PATH_THROUGH) - 1 + dataSize;
		CMD_IDE_PATH_THROUGH* buf =
(CMD_IDE_PATH_THROUGH*)VirtualAlloc(NULL, size, MEM_COMMIT, PAGE_READWRITE);

		buf->reg.bFeaturesReg		= sub;
		buf->reg.bSectorCountReg	= param;
		buf->reg.bSectorNumberReg	= 0;
		buf->reg.bCylLowReg			= 0;
```

```
            buf->reg.bCylHighReg          = 0;
            buf->reg.bDriveHeadReg        = 0;
            buf->reg.bCommandReg          = main;
            buf->reg.bReserved                = 0;
            buf->length                       = dataSize;

            bRet = ::DeviceIoControl(hIoCtrl, IOCTL_IDE_PASS_THROUGH,
                  buf, size, buf, size, &dwReturned, NULL);
            ::CloseHandle(hIoCtrl);
            if(bRet && dataSize && data != NULL)
            {
                  memcpy_s(data, dataSize, buf->buffer, dataSize);
            }
            VirtualFree(buf, 0, MEM_RELEASE);
      }

      return      bRet;
}

/*-------------------------------------------------------------------------*/
//  \\\\.\\ Диск SCSI X
/*-------------------------------------------------------------------------*/

BOOL CAtaLifeLeft::DoIdentifyDeviceScsi(INT scsiPort, INT scsiTargetId,
IDENTIFY_DEVICE* identify)
{
      int done = FALSE;
      int controller = 0;
      int current = 0;
      HANDLE hScsiDriveIOCTL = 0;
      CString driveName;
      driveName.Format(_T("\\\\.\\Scsi%d:"), scsiPort);
      hScsiDriveIOCTL = CreateFile(driveName, GENERIC_READ | GENERIC_WRITE,
                                          FILE_SHARE_READ | FILE_SHARE_WRITE,
NULL, OPEN_EXISTING, 0, NULL);
      if(hScsiDriveIOCTL != INVALID_HANDLE_VALUE)
      {
            BYTE buffer[sizeof(SRB_IO_CONTROL) + sizeof(SENDCMDOUTPARAMS) +
IDENTIFY_BUFFER_SIZE];
            SRB_IO_CONTROL *p = (SRB_IO_CONTROL *)buffer;
            SENDCMDINPARAMS *pin = (SENDCMDINPARAMS *)(buffer +
sizeof(SRB_IO_CONTROL));
            DWORD dummy;

            memset(buffer, 0, sizeof(buffer));
            p->HeaderLength = sizeof (SRB_IO_CONTROL);
            p->Timeout = 2;
            p->Length = sizeof(SENDCMDOUTPARAMS) + IDENTIFY_BUFFER_SIZE;
            p->ControlCode = IOCTL_SCSI_MINIPORT_IDENTIFY;
            memcpy((char *)p->Signature, "SCSIDISK", 8);
            pin->irDriveRegs.bCommandReg = ID_CMD;
            pin->bDriveNumber = scsiTargetId;

            if(DeviceIoControl(hScsiDriveIOCTL, IOCTL_SCSI_MINIPORT,
                                          buffer, sizeof(SRB_IO_CONTROL) +
sizeof(SENDCMDINPARAMS) - 1,
                                          buffer, sizeof(SRB_IO_CONTROL) +
sizeof(SENDCMDOUTPARAMS) + IDENTIFY_BUFFER_SIZE,
                                          &dummy, NULL))
            {
                  SENDCMDOUTPARAMS *pOut = (SENDCMDOUTPARAMS *)(buffer +
sizeof(SRB_IO_CONTROL));
                  if(*(pOut->bBuffer) > 0)
                  {
                        done = TRUE;
                        memcpy_s(identify, sizeof(IDENTIFY_DEVICE), pOut-
>bBuffer, sizeof(IDENTIFY_DEVICE));
                  }
            }
```

```
            CloseHandle(hScsiDriveIOCTL);
      }
      return done;
}

BOOL CAtaLifeLeft::GetLifeLeftAttributeScsi(INT scsiPort, INT scsiTargetId,
ATA_LifeLeft_INFO* asi)
{
      HANDLE hScsiDriveIOCTL = 0;
      CString driveName;
      driveName.Format(_T("\\\\.\\Scsi%d:"), scsiPort);
      hScsiDriveIOCTL = CreateFile(driveName, GENERIC_READ | GENERIC_WRITE,
                                    FILE_SHARE_READ | FILE_SHARE_WRITE,
NULL, OPEN_EXISTING, 0, NULL);
      if(hScsiDriveIOCTL != INVALID_HANDLE_VALUE)
      {
            BYTE buffer[sizeof(SRB_IO_CONTROL) + sizeof(SENDCMDOUTPARAMS) +
READ_ATTRIBUTE_BUFFER_SIZE];
            SRB_IO_CONTROL *p = (SRB_IO_CONTROL *)buffer;
            SENDCMDINPARAMS *pin = (SENDCMDINPARAMS *)(buffer +
sizeof(SRB_IO_CONTROL));
            DWORD dummy;
            memset(buffer, 0, sizeof(buffer));
            p->HeaderLength = sizeof(SRB_IO_CONTROL);
            p->Timeout = 2;
            p->Length = sizeof(SENDCMDOUTPARAMS) + READ_ATTRIBUTE_BUFFER_SIZE;
            p->ControlCode = IOCTL_SCSI_MINIPORT_READ_LifeLeft_ATTRIBS;
            memcpy((char *)p->Signature, "SCSIDISK", 8);
            pin->irDriveRegs.bFeaturesReg       = READ_ATTRIBUTES;
            pin->irDriveRegs.bSectorCountReg    = 1;
            pin->irDriveRegs.bSectorNumberReg   = 1;
            pin->irDriveRegs.bCylLowReg             = LifeLeft_CYL_LOW;
            pin->irDriveRegs.bCylHighReg        = LifeLeft_CYL_HI;
            pin->irDriveRegs.bCommandReg        = LifeLeft_CMD;
            pin->cBufferSize                        =
READ_ATTRIBUTE_BUFFER_SIZE;
            pin->bDriveNumber                       = scsiTargetId;

            if(DeviceIoControl(hScsiDriveIOCTL, IOCTL_SCSI_MINIPORT,
                                    buffer, sizeof(SRB_IO_CONTROL) +
sizeof(SENDCMDINPARAMS) - 1,
                                    buffer, sizeof(SRB_IO_CONTROL) +
sizeof(SENDCMDOUTPARAMS) + READ_ATTRIBUTE_BUFFER_SIZE,
                                    &dummy, NULL))
            {
                  SENDCMDOUTPARAMS *pOut = (SENDCMDOUTPARAMS *)(buffer +
sizeof(SRB_IO_CONTROL));
                  if(*(pOut->bBuffer) > 0)
                  {
                        CString str;
                        asi->AttributeCount = 0;
                        int j = 0;

                        for(int i = 0; i < MAX_ATTRIBUTE; i++)
                        {
                              DWORD rawValue = 0;

                              memcpy(     &(asi->Attribute[j]),
                                    &(pOut->bBuffer[i *
sizeof(LifeLeft_ATTRIBUTE) + 2]), sizeof(LifeLeft_ATTRIBUTE));

                              if(asi->Attribute[j].Id != 0)
                              {
                                    switch(asi->Attribute[j].Id)
                                    {
                                    case 0x09: // Кількість відпрацьованих годин
у включеному стані
                                          rawValue = MAKELONG(
```

```
                                                MAKEWORD(asi-
>Attribute[j].RawValue[0], asi->Attribute[j].RawValue[1]),
                                                MAKEWORD(asi-
>Attribute[j].RawValue[2], asi->Attribute[j].RawValue[3])
                                                );
                                          asi->PowerOnRawValue = rawValue;
                                          asi->DetectedPowerOnHours =
GetPowerOnHours(rawValue, asi->DetectedTimeUnitType);
                                          asi->MeasuredPowerOnHours =
GetPowerOnHours(rawValue, asi->MeasuredTimeUnitType);
                                          break;
                                    case 0x0C: // Кількість зафіксованих
повторів включення/вимикання живлення накопичувача
                                          rawValue = MAKELONG(
                                                MAKEWORD(asi-
>Attribute[j].RawValue[0], asi->Attribute[j].RawValue[1]),
                                                MAKEWORD(asi-
>Attribute[j].RawValue[2], asi->Attribute[j].RawValue[3])
                                                );
                                          asi->PowerOnCount = rawValue;
                                          break;
                                    case 0xC2: // Температура
                                          if(asi->Attribute[j].RawValue[0] > 0)
                                          {
                                                asi->Temperature = asi-
>Attribute[j].RawValue[0];
                                          }
                                          else
                                          {
                                                asi->Temperature = asi-
>Attribute[j].CurrentValue;
                                          }
                                          break;
                                    case 0xBB: // Визначення фірми-розробника
                                          if(asi->VendorId == VENDOR_MTRON)
                                          {
                                                asi->Life = asi-
>Attribute[j].CurrentValue;
                                          }
                                          break;
                                    default:
                                          break;
                                    }
                                    j++;
                              }
                        }
                        asi->AttributeCount = j;
                  }
            }
      }
      CloseHandle(hScsiDriveIOCTL);

      if(asi->AttributeCount > 0)
      {
            return      TRUE;
      }
      else
      {
            return FALSE;
      }
}

BOOL CAtaLifeLeft::GetLifeLeftThresholdScsi(INT scsiPort, INT scsiTargetId,
ATA_LifeLeft_INFO* asi)
{
      HANDLE hScsiDriveIOCTL = 0;
      CString driveName;
      driveName.Format(_T("\\\\.\\Scsi%d:"), scsiPort);
      hScsiDriveIOCTL = CreateFile(driveName, GENERIC_READ | GENERIC_WRITE,
```

```
                                          FILE_SHARE_READ | FILE_SHARE_WRITE,
NULL, OPEN_EXISTING, 0, NULL);
      if(hScsiDriveIOCTL != INVALID_HANDLE_VALUE)
      {
            BYTE buffer[sizeof(SRB_IO_CONTROL) + sizeof(SENDCMDOUTPARAMS) +
READ_THRESHOLD_BUFFER_SIZE];
            SRB_IO_CONTROL *p = (SRB_IO_CONTROL *)buffer;
            SENDCMDINPARAMS *pin = (SENDCMDINPARAMS *)(buffer +
sizeof(SRB_IO_CONTROL));
            DWORD dummy;
            memset(buffer, 0, sizeof(buffer));
            p->HeaderLength = sizeof(SRB_IO_CONTROL);
            p->Timeout = 2;
            p->Length = sizeof(SENDCMDOUTPARAMS) + READ_THRESHOLD_BUFFER_SIZE;
            p->ControlCode = IOCTL_SCSI_MINIPORT_READ_LifeLeft_THRESHOLDS;
            memcpy((char *)p->Signature, "SCSIDISK", 8);
            pin->irDriveRegs.bFeaturesReg       = READ_THRESHOLDS;
            pin->irDriveRegs.bSectorCountReg    = 1;
            pin->irDriveRegs.bSectorNumberReg   = 1;
            pin->irDriveRegs.bCylLowReg               = LifeLeft_CYL_LOW;
            pin->irDriveRegs.bCylHighReg        = LifeLeft_CYL_HI;
            pin->irDriveRegs.bCommandReg        = LifeLeft_CMD;
            pin->cBufferSize                          =
READ_THRESHOLD_BUFFER_SIZE;
            pin->bDriveNumber                         = scsiTargetId;

            if(DeviceIoControl(hScsiDriveIOCTL, IOCTL_SCSI_MINIPORT,
                                          buffer, sizeof(SRB_IO_CONTROL) +
sizeof(SENDCMDINPARAMS) - 1,
                                          buffer, sizeof(SRB_IO_CONTROL) +
sizeof(SENDCMDOUTPARAMS) + READ_THRESHOLD_BUFFER_SIZE,
                                          &dummy, NULL))
            {
                  SENDCMDOUTPARAMS *pOut = (SENDCMDOUTPARAMS *)(buffer +
sizeof(SRB_IO_CONTROL));
                  if(*(pOut->bBuffer) > 0)
                  {
                        int j = 0;
                        for(int i = 0; i < MAX_ATTRIBUTE; i++)
                        {
                              memcpy(     &(asi->Threshold[j]),
                                    &(pOut->bBuffer[i *
sizeof(LifeLeft_THRESHOLD) + 2]), sizeof(LifeLeft_THRESHOLD));
                              if(asi->Threshold[j].Id != 0)
                              {
                                    j++;
                              }
                        }
                  }
            }
      }
      CloseHandle (hScsiDriveIOCTL);

      if(asi->AttributeCount > 0)
      {
            return TRUE;
      }
      else
      {
            return FALSE;
      }
}

BOOL CAtaLifeLeft::ControlLifeLeftStatusScsi(INT scsiPort, INT scsiTargetId,
BYTE command)
{
      BOOL  bRet;
      HANDLE hScsiDriveIOCTL = 0;
      CString driveName;
```

```
	driveName.Format(_T("\\\\.\\Scsi%d:"), scsiPort);
	hScsiDriveIOCTL = CreateFile(driveName, GENERIC_READ | GENERIC_WRITE,
                                     FILE_SHARE_READ | FILE_SHARE_WRITE,
NULL, OPEN_EXISTING, 0, NULL);
	if(hScsiDriveIOCTL != INVALID_HANDLE_VALUE)
	{
		BYTE buffer[sizeof(SRB_IO_CONTROL) + sizeof(SENDCMDOUTPARAMS) +
SCSI_MINIPORT_BUFFER_SIZE];
		SRB_IO_CONTROL *p = (SRB_IO_CONTROL *)buffer;
		SENDCMDINPARAMS *pin = (SENDCMDINPARAMS *)(buffer +
sizeof(SRB_IO_CONTROL));
		DWORD dummy;
		memset(buffer, 0, sizeof(buffer));
		p->HeaderLength = sizeof(SRB_IO_CONTROL);
		p->Timeout = 2;
		p->Length = sizeof(SENDCMDOUTPARAMS) + SCSI_MINIPORT_BUFFER_SIZE;
		if(command == DISABLE_LifeLeft)
		{
			p->ControlCode = IOCTL_SCSI_MINIPORT_DISABLE_LifeLeft;
		}
		else
		{
			p->ControlCode = IOCTL_SCSI_MINIPORT_ENABLE_LifeLeft;
		}
		memcpy((char *)p->Signature, "SCSIDISK", 8);
		pin->irDriveRegs.bFeaturesReg     = command;
		pin->irDriveRegs.bSectorCountReg  = 1;
		pin->irDriveRegs.bSectorNumberReg = 1;
		pin->irDriveRegs.bCylLowReg             = LifeLeft_CYL_LOW;
		pin->irDriveRegs.bCylHighReg      = LifeLeft_CYL_HI;
		pin->irDriveRegs.bCommandReg      = LifeLeft_CMD;
		pin->cBufferSize                        =
SCSI_MINIPORT_BUFFER_SIZE;
		pin->bDriveNumber                       = scsiTargetId;

		bRet = DeviceIoControl(hScsiDriveIOCTL, IOCTL_SCSI_MINIPORT,
                                       buffer, sizeof(SRB_IO_CONTROL) +
sizeof(SENDCMDINPARAMS) - 1,
                                       buffer, sizeof(SRB_IO_CONTROL) +
sizeof(SENDCMDOUTPARAMS) + SCSI_MINIPORT_BUFFER_SIZE,
                                       &dummy, NULL);
	}
	CloseHandle(hScsiDriveIOCTL);

	return bRet;
}


BOOL CAtaLifeLeft::SendAtaCommandScsi(INT scsiPort, INT scsiTargetId, BYTE main,
BYTE sub, BYTE param)
{
/** Does not work...
	BOOL  bRet;
	HANDLE hScsiDriveIOCTL = 0;
	CString driveName;
	driveName.Format(_T("\\\\.\\Scsi%d:"), scsiPort);
	hScsiDriveIOCTL = CreateFile(driveName, GENERIC_READ | GENERIC_WRITE,
                                     FILE_SHARE_READ | FILE_SHARE_WRITE,
NULL, OPEN_EXISTING, 0, NULL);
	if(hScsiDriveIOCTL != INVALID_HANDLE_VALUE)
	{
		DWORD dummy;
		CMD_ATA_PASS_THROUGH_WITH_BUFFERS capt;
		::ZeroMemory(&capt, sizeof(CMD_ATA_PASS_THROUGH_WITH_BUFFERS));
		capt.apt.Length = sizeof(CMD_ATA_PASS_THROUGH);
		capt.apt.PathId = 0;
		capt.apt.TargetId = 0;
		capt.apt.Lun = 0;
		capt.apt.TimeOutValue = 10;
```

```
            DWORD size = offsetof(CMD_ATA_PASS_THROUGH_WITH_BUFFERS, DataBuf);
            capt.apt.DataBufferOffset = size;

            capt.apt.AtaFlags = 0x02;
            capt.apt.DataTransferLength = 512;
            size += 512;
            capt.DataBuf[0] = 0xCF;

            capt.apt.CurrentTaskFile.bFeaturesReg= sub;
            capt.apt.CurrentTaskFile.bSectorCountReg = param;
            capt.apt.CurrentTaskFile.bDriveHeadReg = 0xA0;
            capt.apt.CurrentTaskFile.bCommandReg = main;

            bRet = DeviceIoControl(hScsiDriveIOCTL, IOCTL_ATA_PASS_THROUGH,
                                                    &capt, size,
                                                    &capt, size,
                                                    &dummy, NULL);
      }
      CloseHandle(hScsiDriveIOCTL);

      return bRet;
*/
      return FALSE;
}

/*-------------------------------------------------------------------------*/
//  SCSI / ATA переклад (SAT)
/*-------------------------------------------------------------------------*/

BOOL CAtaLifeLeft::DoIdentifyDeviceSat(INT physicalDriveId, IDENTIFY_DEVICE*
data, COMMAND_TYPE type)
{
      BOOL  bRet;
      HANDLE      hIoCtrl;
      DWORD dwReturned;
      DWORD length;

      SCSI_PASS_THROUGH_WITH_BUFFERS sptwb;

      if(data == NULL)
      {
            return     FALSE;
      }

      ::ZeroMemory(data, sizeof(IDENTIFY_DEVICE));

      hIoCtrl = GetIoCtrlHandle(physicalDriveId);
      if(hIoCtrl == INVALID_HANDLE_VALUE)
      {
            return      FALSE;
      }

      ::ZeroMemory(&sptwb,sizeof(SCSI_PASS_THROUGH_WITH_BUFFERS));

    sptwb.Spt.Length = sizeof(SCSI_PASS_THROUGH);
    sptwb.Spt.PathId = 0;
    sptwb.Spt.TargetId = 0;
    sptwb.Spt.Lun = 0;
    sptwb.Spt.SenseInfoLength = 24;
    sptwb.Spt.DataIn = SCSI_IOCTL_DATA_IN;
    sptwb.Spt.DataTransferLength = IDENTIFY_BUFFER_SIZE;
    sptwb.Spt.TimeOutValue = 2;
    sptwb.Spt.DataBufferOffset = offsetof(SCSI_PASS_THROUGH_WITH_BUFFERS,
DataBuf);
    sptwb.Spt.SenseInfoOffset = offsetof(SCSI_PASS_THROUGH_WITH_BUFFERS,
SenseBuf);

      if(type == CMD_TYPE_SAT)
```

```
	{
		sptwb.Spt.CdbLength = 12;
		sptwb.Spt.Cdb[0] = 0xA1;//ATA проходів через(12) операцій коду(A1h)
		sptwb.Spt.Cdb[1] = (4 << 1) | 0; //MULTIPLE_COUNT=0,PROTOCOL=4(PIO
Data-In),Reserved
		sptwb.Spt.Cdb[2] = (1 << 3) | (1 << 2) |
2;//OFF_LINE=0,CK_COND=0,Reserved=0,T_DIR=1(ToDevice),BYTE_BLOCK=1,T_LENGTH=2
		sptwb.Spt.Cdb[3] = 0;//FEATURES (7:0)
		sptwb.Spt.Cdb[4] = 1;//SECTOR_COUNT (7:0)
		sptwb.Spt.Cdb[5] = 0;//LBA_LOW (7:0)
		sptwb.Spt.Cdb[6] = 0;//LBA_MID (7:0)
		sptwb.Spt.Cdb[7] = 0;//LBA_HIGH (7:0)
		sptwb.Spt.Cdb[9] = ID_CMD;//COMMAND
	}
	else if(type == CMD_TYPE_SUNPLUS)
	{
		sptwb.Spt.CdbLength = 12;
		sptwb.Spt.Cdb[0] = 0xF8;
		sptwb.Spt.Cdb[1] = 0x00;
		sptwb.Spt.Cdb[2] = 0x22;
		sptwb.Spt.Cdb[3] = 0x10;
		sptwb.Spt.Cdb[4] = 0x01;
		sptwb.Spt.Cdb[5] = 0x00;
		sptwb.Spt.Cdb[6] = 0x01;
		sptwb.Spt.Cdb[7] = 0x00;
		sptwb.Spt.Cdb[8] = 0x00;
		sptwb.Spt.Cdb[9] = 0x00;
		sptwb.Spt.Cdb[10] = 0xA0;
		sptwb.Spt.Cdb[11] = 0xEC; // ID_CMD
	}
	else if(type == CMD_TYPE_IO_DATA)
	{
		sptwb.Spt.CdbLength = 12;
		sptwb.Spt.Cdb[0] = 0xE3;
		sptwb.Spt.Cdb[1] = 0x00;
		sptwb.Spt.Cdb[2] = 0x00;
		sptwb.Spt.Cdb[3] = 0x01;
		sptwb.Spt.Cdb[4] = 0x01;
		sptwb.Spt.Cdb[5] = 0x00;
		sptwb.Spt.Cdb[6] = 0x00;
		sptwb.Spt.Cdb[7] = 0xA0;
		sptwb.Spt.Cdb[8] = 0xEC;  // ID_CMD
		sptwb.Spt.Cdb[9] = 0x00;
		sptwb.Spt.Cdb[10] = 0x00;
		sptwb.Spt.Cdb[11] = 0x00;
	}
	else if(type == CMD_TYPE_LOGITEC)
	{
		sptwb.Spt.CdbLength = 10;
		sptwb.Spt.Cdb[0] = 0xE0;
		sptwb.Spt.Cdb[1] = 0x00;
		sptwb.Spt.Cdb[2] = 0x00;
		sptwb.Spt.Cdb[3] = 0x00;
		sptwb.Spt.Cdb[4] = 0x00;
		sptwb.Spt.Cdb[5] = 0x00;
		sptwb.Spt.Cdb[6] = 0x00;
		sptwb.Spt.Cdb[7] = 0xA0;
		sptwb.Spt.Cdb[8] = 0xEC;  // ID_CMD
		sptwb.Spt.Cdb[9] = 0x4C;
	}
	else if(type == CMD_TYPE_JMICRON)
	{
		sptwb.Spt.CdbLength = 12;
		sptwb.Spt.Cdb[0] = 0xDF;
		sptwb.Spt.Cdb[1] = 0x10;
		sptwb.Spt.Cdb[2] = 0x00;
		sptwb.Spt.Cdb[3] = 0x02;
		sptwb.Spt.Cdb[4] = 0x00;
		sptwb.Spt.Cdb[5] = 0x00;
```

```
		sptwb.Spt.Cdb[6] = 0x01;
		sptwb.Spt.Cdb[7] = 0x00;
		sptwb.Spt.Cdb[8] = 0x00;
		sptwb.Spt.Cdb[9] = 0x00;
		sptwb.Spt.Cdb[10] = 0xA0;
		sptwb.Spt.Cdb[11] = 0xEC; // ID_CMD
	}
	else if(type == CMD_TYPE_CYPRESS)
	{
		sptwb.Spt.CdbLength = 16;
		sptwb.Spt.Cdb[0] = 0x24;
		sptwb.Spt.Cdb[1] = 0x24;
		sptwb.Spt.Cdb[2] = 0x00;
		sptwb.Spt.Cdb[3] = 0xBE;
		sptwb.Spt.Cdb[4] = 0x01;
		sptwb.Spt.Cdb[5] = 0x00;
		sptwb.Spt.Cdb[6] = 0x00;
		sptwb.Spt.Cdb[7] = 0x01;
		sptwb.Spt.Cdb[8] = 0x00;
		sptwb.Spt.Cdb[9] = 0x00;
		sptwb.Spt.Cdb[10] = 0x00;
		sptwb.Spt.Cdb[11] = 0xA0;
		sptwb.Spt.Cdb[12] = 0xEC; // ID_CMD
		sptwb.Spt.Cdb[13] = 0x00;
		sptwb.Spt.Cdb[14] = 0x00;
		sptwb.Spt.Cdb[15] = 0x00;
	}
/*
	else if(type == CMD_TYPE_DEBUG)
	{
		sptwb.Spt.CdbLength = 16;
		for(int i = 0xA0; i <= 0xFF; i++)
		{
			for(int j = 8; j < 16; j++)
			{
				::ZeroMemory(&sptwb.Spt.Cdb, 16);
				sptwb.Spt.Cdb[0] = i;
				sptwb.Spt.Cdb[j - 1] = 0xA0;
				sptwb.Spt.Cdb[j] = 0xEC;

				length = offsetof(SCSI_PASS_THROUGH_WITH_BUFFERS,
DataBuf) + sptwb.Spt.DataTransferLength;

				bRet = ::DeviceIoControl(hIoCtrl,
IOCTL_SCSI_PASS_THROUGH,
					&sptwb, sizeof(SCSI_PASS_THROUGH),
					&sptwb, length,   &dwReturned, NULL);


				if(bRet == FALSE || dwReturned != length)
				{
					continue;
				}

				CString cstr;
				cstr.Format(_T("i = %d, j = %d"), i, j);
				AfxMessageBox(cstr);

				::CloseHandle(hIoCtrl);
				memcpy_s(data, sizeof(IDENTIFY_DEVICE), sptwb.DataBuf,
sizeof(IDENTIFY_DEVICE));

				return TRUE;
			}
		}
	}
*/
	else
	{
```

```
            return FALSE;
      }

      length = offsetof(SCSI_PASS_THROUGH_WITH_BUFFERS, DataBuf) +
sptwb.Spt.DataTransferLength;

      bRet = ::DeviceIoControl(hIoCtrl, IOCTL_SCSI_PASS_THROUGH,
            &sptwb, sizeof(SCSI_PASS_THROUGH),
            &sptwb, length,   &dwReturned, NULL);

      ::CloseHandle(hIoCtrl);

      if(bRet == FALSE || dwReturned != length)
      {
            return      FALSE;
      }

      memcpy_s(data, sizeof(IDENTIFY_DEVICE), sptwb.DataBuf,
sizeof(IDENTIFY_DEVICE));

      return      TRUE;
}

BOOL CAtaLifeLeft::GetLifeLeftAttributeSat(INT PhysicalDriveId,
ATA_LifeLeft_INFO* asi)
{
      BOOL  bRet;
      HANDLE      hIoCtrl;
      DWORD dwReturned;
      DWORD length;

      SCSI_PASS_THROUGH_WITH_BUFFERS sptwb;

      hIoCtrl = GetIoCtrlHandle(PhysicalDriveId);
      if(hIoCtrl == INVALID_HANDLE_VALUE)
      {
            return      FALSE;
      }

      ::ZeroMemory(&sptwb,sizeof(SCSI_PASS_THROUGH_WITH_BUFFERS));

    sptwb.Spt.Length = sizeof(SCSI_PASS_THROUGH);
    sptwb.Spt.PathId = 0;
    sptwb.Spt.TargetId = 0;
    sptwb.Spt.Lun = 0;
    sptwb.Spt.SenseInfoLength = 24;
    sptwb.Spt.DataIn = SCSI_IOCTL_DATA_IN;
    sptwb.Spt.DataTransferLength = READ_ATTRIBUTE_BUFFER_SIZE;
    sptwb.Spt.TimeOutValue = 2;
    sptwb.Spt.DataBufferOffset = offsetof(SCSI_PASS_THROUGH_WITH_BUFFERS,
DataBuf);
    sptwb.Spt.SenseInfoOffset = offsetof(SCSI_PASS_THROUGH_WITH_BUFFERS,
SenseBuf);

      COMMAND_TYPE type = asi->CommandType;
      if(type == CMD_TYPE_SAT)
      {
            sptwb.Spt.CdbLength = 12;
            sptwb.Spt.Cdb[0] = 0xA1;//ATA прохід через (12) операцій коду (A1h)
            sptwb.Spt.Cdb[1] = (4 << 1) | 0; //MULTIPLE_COUNT=0,PROTOCOL=4(PIO
Data-In),Reserved
            sptwb.Spt.Cdb[2] = (1 << 3) | (1 << 2) |
2;//OFF_LINE=0,CK_COND=0,Reserved=0,T_DIR=1(ToDevice),BYTE_BLOCK=1,T_LENGTH=2
            sptwb.Spt.Cdb[3] = READ_ATTRIBUTES;//FEATURES (7:0)
            sptwb.Spt.Cdb[4] = 1;//SECTOR_COUNT (7:0)
            sptwb.Spt.Cdb[5] = 1;//LBA_LOW (7:0)
            sptwb.Spt.Cdb[6] = LifeLeft_CYL_LOW;//LBA_MID (7:0)
            sptwb.Spt.Cdb[7] = LifeLeft_CYL_HI;//LBA_HIGH (7:0)
            sptwb.Spt.Cdb[9] = LifeLeft_CMD;//COMMAND
```

```
}
else if(type == CMD_TYPE_SUNPLUS)
{
      sptwb.Spt.CdbLength = 12;
      sptwb.Spt.Cdb[0] = 0xF8;
      sptwb.Spt.Cdb[1] = 0x00;
      sptwb.Spt.Cdb[2] = 0x22;
      sptwb.Spt.Cdb[3] = 0x10;
      sptwb.Spt.Cdb[4] = 0x01;
      sptwb.Spt.Cdb[5] = 0xD0;  // READ_ATTRIBUTES
      sptwb.Spt.Cdb[6] = 0x01;
      sptwb.Spt.Cdb[7] = 0x00;
      sptwb.Spt.Cdb[8] = 0x4F;  // LifeLeft_CYL_LOW
      sptwb.Spt.Cdb[9] = 0xC2;  // LifeLeft_CYL_HIGH
      sptwb.Spt.Cdb[10] = 0xA0;
      sptwb.Spt.Cdb[11] = 0xB0; // LifeLeft_CMD
}
else if(type == CMD_TYPE_IO_DATA)
{
      sptwb.Spt.CdbLength = 12;
      sptwb.Spt.Cdb[0] = 0xE3;
      sptwb.Spt.Cdb[1] = 0x00; // ?
      sptwb.Spt.Cdb[2] = 0xD0; // READ_ATTRIBUTES
      sptwb.Spt.Cdb[3] = 0x00; // ?
      sptwb.Spt.Cdb[4] = 0x00; // ?
      sptwb.Spt.Cdb[5] = 0x4F; // LifeLeft_CYL_LOW
      sptwb.Spt.Cdb[6] = 0xC2; // LifeLeft_CYL_HIGH
      sptwb.Spt.Cdb[7] = 0xA0; //
      sptwb.Spt.Cdb[8] = 0xB0; // LifeLeft_CMD
      sptwb.Spt.Cdb[9] = 0x00;
      sptwb.Spt.Cdb[10] = 0x00;
      sptwb.Spt.Cdb[11] = 0x00;
}
else if(type == CMD_TYPE_LOGITEC)
{
      sptwb.Spt.CdbLength = 10;
      sptwb.Spt.Cdb[0] = 0xE0;
      sptwb.Spt.Cdb[1] = 0x00;
      sptwb.Spt.Cdb[2] = 0xD0; // READ_ATTRIBUTES
      sptwb.Spt.Cdb[3] = 0x00;
      sptwb.Spt.Cdb[4] = 0x00;
      sptwb.Spt.Cdb[5] = 0x4F; // LifeLeft_CYL_LOW
      sptwb.Spt.Cdb[6] = 0xC2; // LifeLeft_CYL_HIGH
      sptwb.Spt.Cdb[7] = 0xA0;
      sptwb.Spt.Cdb[8] = 0xB0; // LifeLeft_CMD
      sptwb.Spt.Cdb[9] = 0x4C;
}
else if(type == CMD_TYPE_JMICRON)
{
      sptwb.Spt.CdbLength = 12;
      sptwb.Spt.Cdb[0] = 0xDF;
      sptwb.Spt.Cdb[1] = 0x10;
      sptwb.Spt.Cdb[2] = 0x00;
      sptwb.Spt.Cdb[3] = 0x02;
      sptwb.Spt.Cdb[4] = 0x00;
      sptwb.Spt.Cdb[5] = 0xD0;  // READ_ATTRIBUTES
      sptwb.Spt.Cdb[6] = 0x01;
      sptwb.Spt.Cdb[7] = 0x01;
      sptwb.Spt.Cdb[8] = 0x4F;  // LifeLeft_CYL_LOW
      sptwb.Spt.Cdb[9] = 0xC2;  // LifeLeft_CYL_HIGH
      sptwb.Spt.Cdb[10] = 0xA0;
      sptwb.Spt.Cdb[11] = 0xB0; // LifeLeft_CMD
}
else if(type == CMD_TYPE_CYPRESS)
{
      sptwb.Spt.CdbLength = 16;
      sptwb.Spt.Cdb[0] = 0x24;
      sptwb.Spt.Cdb[1] = 0x24;
      sptwb.Spt.Cdb[2] = 0x00;
```

```
            sptwb.Spt.Cdb[3] = 0xBE;
            sptwb.Spt.Cdb[4] = 0x01;
            sptwb.Spt.Cdb[5] = 0x00;
            sptwb.Spt.Cdb[6] = 0xD0;  // READ_ATTRIBUTES
            sptwb.Spt.Cdb[7] = 0x00;
            sptwb.Spt.Cdb[8] = 0x00;
            sptwb.Spt.Cdb[9] = 0x4F;  // LifeLeft_CYL_LOW
            sptwb.Spt.Cdb[10] = 0xC2; // LifeLeft_CYL_HIGH
            sptwb.Spt.Cdb[11] = 0xA0;
            sptwb.Spt.Cdb[12] = 0xB0; // ID_CMD
            sptwb.Spt.Cdb[13] = 0x00;
            sptwb.Spt.Cdb[14] = 0x00;
            sptwb.Spt.Cdb[15] = 0x00;
      }
      else
      {
            return FALSE;
      }

      length = offsetof(SCSI_PASS_THROUGH_WITH_BUFFERS, DataBuf) +
sptwb.Spt.DataTransferLength;
      bRet = ::DeviceIoControl(hIoCtrl, IOCTL_SCSI_PASS_THROUGH,
            &sptwb, sizeof(SCSI_PASS_THROUGH),
            &sptwb, length,   &dwReturned, NULL);

      ::CloseHandle(hIoCtrl);

      if(bRet == FALSE || dwReturned != length)
      {
            return      FALSE;
      }

      CString str;
      asi->AttributeCount = 0;
      int j = 0;
      for(int i = 0; i < MAX_ATTRIBUTE; i++)
      {
            DWORD rawValue = 0;
            memcpy(     &(asi->Attribute[j]),
                        &(sptwb.DataBuf[i * sizeof(LifeLeft_ATTRIBUTE) + 2]),
sizeof(LifeLeft_ATTRIBUTE));

            if(asi->Attribute[j].Id != 0)
            {
                  switch(asi->Attribute[j].Id)
                  {
                  case 0x09: // Кількість відпрацьованих годин у включеному
стані
                        rawValue = MAKELONG(
                              MAKEWORD(asi->Attribute[j].RawValue[0], asi-
>Attribute[j].RawValue[1]),
                              MAKEWORD(asi->Attribute[j].RawValue[2], asi-
>Attribute[j].RawValue[3])
                              );
                        asi->PowerOnRawValue = rawValue;
                        asi->DetectedPowerOnHours = GetPowerOnHours(rawValue,
asi->DetectedTimeUnitType);
                        asi->MeasuredPowerOnHours = GetPowerOnHours(rawValue,
asi->MeasuredTimeUnitType);
                        break;
                  case 0x0C: // Кількість зафіксованих повторів
включення/вимикання живлення накопичувача
                        rawValue = MAKELONG(
                              MAKEWORD(asi->Attribute[j].RawValue[0], asi-
>Attribute[j].RawValue[1]),
                              MAKEWORD(asi->Attribute[j].RawValue[2], asi-
>Attribute[j].RawValue[3])
                              );
                        asi->PowerOnCount = rawValue;
```

```
                        break;
                  case 0xC2: // Температура
                        if(asi->Attribute[j].RawValue[0] > 0)
                        {
                              asi->Temperature = asi->Attribute[j].RawValue[0];
                        }
                        else
                        {
                              asi->Temperature = asi->Attribute[j].CurrentValue;
                        }
                        break;
                  case 0xBB: // Визначення фірми-розробника
                        if(asi->VendorId == VENDOR_MTRON)
                        {
                              asi->Life = asi->Attribute[j].CurrentValue;
                        // РОЗШИФРУВАННЯ ЗНАЧЕНЬ
                        //    asi->Life = 0;
                        //    asi->Attribute[j].CurrentValue = 0;
                        }
                        break;
                  default:
                        break;
                  }
                  j++;
            }
      }
      asi->AttributeCount = j;

      if(asi->AttributeCount > 0)
      {
            return TRUE;
      }
      else
      {
            return FALSE;
      }
}

BOOL CAtaLifeLeft::GetLifeLeftThresholdSat(INT physicalDriveId,
ATA_LifeLeft_INFO* asi)
{
      BOOL  bRet;
      HANDLE      hIoCtrl;
      DWORD dwReturned;
      DWORD length;

      SCSI_PASS_THROUGH_WITH_BUFFERS sptwb;

      hIoCtrl = GetIoCtrlHandle(physicalDriveId);
      if(hIoCtrl == INVALID_HANDLE_VALUE)
      {
            return      FALSE;
      }

      ::ZeroMemory(&sptwb,sizeof(SCSI_PASS_THROUGH_WITH_BUFFERS));

    sptwb.Spt.Length = sizeof(SCSI_PASS_THROUGH);
    sptwb.Spt.PathId = 0;
    sptwb.Spt.TargetId = 0;
    sptwb.Spt.Lun = 0;
    sptwb.Spt.SenseInfoLength = 24;
    sptwb.Spt.DataIn = SCSI_IOCTL_DATA_IN;
    sptwb.Spt.DataTransferLength = READ_THRESHOLD_BUFFER_SIZE;
    sptwb.Spt.TimeOutValue = 2;
    sptwb.Spt.DataBufferOffset = offsetof(SCSI_PASS_THROUGH_WITH_BUFFERS,
DataBuf);
    sptwb.Spt.SenseInfoOffset = offsetof(SCSI_PASS_THROUGH_WITH_BUFFERS,
SenseBuf);
```

```
	COMMAND_TYPE type = asi->CommandType;
	if(type == CMD_TYPE_SAT)
	{
		sptwb.Spt.CdbLength = 12;
		sptwb.Spt.Cdb[0] = 0xA1; ////ATA Проходів через (12) операцій коду
(A1h)
		sptwb.Spt.Cdb[1] = (4 << 1) | 0; //MULTIPLE_COUNT=0,PROTOCOL=4(PIO
Data-In),Reserved
		sptwb.Spt.Cdb[2] = (1 << 3) | (1 << 2) |
2;//OFF_LINE=0,CK_COND=0,Reserved=0,T_DIR=1(ToDevice),BYTE_BLOCK=1,T_LENGTH=2
		sptwb.Spt.Cdb[3] = READ_THRESHOLDS;//FEATURES (7:0)
		sptwb.Spt.Cdb[4] = 1;//SECTOR_COUNT (7:0)
		sptwb.Spt.Cdb[5] = 1;//LBA_LOW (7:0)
		sptwb.Spt.Cdb[6] = LifeLeft_CYL_LOW;//LBA_MID (7:0)
		sptwb.Spt.Cdb[7] = LifeLeft_CYL_HI;//LBA_HIGH (7:0)
		sptwb.Spt.Cdb[9] = LifeLeft_CMD;//COMMAND
	}
	else if(type == CMD_TYPE_SUNPLUS)
	{
		sptwb.Spt.CdbLength = 12;
		sptwb.Spt.Cdb[0] = 0xF8;
		sptwb.Spt.Cdb[1] = 0x00;
		sptwb.Spt.Cdb[2] = 0x22;
		sptwb.Spt.Cdb[3] = 0x10;
		sptwb.Spt.Cdb[4] = 0x01;
		sptwb.Spt.Cdb[5] = 0xD1; // READ_THRESHOLD
		sptwb.Spt.Cdb[6] = 0x01;
		sptwb.Spt.Cdb[7] = 0x01;
		sptwb.Spt.Cdb[8] = 0x4F; // LifeLeft_CYL_LOW
		sptwb.Spt.Cdb[9] = 0xC2; // LifeLeft_CYL_HIGH
		sptwb.Spt.Cdb[10] = 0xA0;
		sptwb.Spt.Cdb[11] = 0xB0;// LifeLeft_CMD
	}
	else if(type == CMD_TYPE_IO_DATA)
	{
		sptwb.Spt.CdbLength = 12;
		sptwb.Spt.Cdb[0] = 0xE3;
		sptwb.Spt.Cdb[1] = 0x00; // ?
		sptwb.Spt.Cdb[2] = 0xD1; // READ_THRESHOLD
		sptwb.Spt.Cdb[3] = 0x00; // ?
		sptwb.Spt.Cdb[4] = 0x00; // ?
		sptwb.Spt.Cdb[5] = 0x4F; // LifeLeft_CYL_LOW
		sptwb.Spt.Cdb[6] = 0xC2; // LifeLeft_CYL_HIGH
		sptwb.Spt.Cdb[7] = 0xA0; //
		sptwb.Spt.Cdb[8] = 0xB0; // LifeLeft_CMD
		sptwb.Spt.Cdb[9] = 0x00;
		sptwb.Spt.Cdb[10] = 0x00;
		sptwb.Spt.Cdb[11] = 0x00;
	}
	else if(type == CMD_TYPE_LOGITEC)
	{
		sptwb.Spt.CdbLength = 10;
		sptwb.Spt.Cdb[0] = 0xE0;
		sptwb.Spt.Cdb[1] = 0x00;
		sptwb.Spt.Cdb[2] = 0xD1; // READ_THRESHOLD
		sptwb.Spt.Cdb[3] = 0x00;
		sptwb.Spt.Cdb[4] = 0x00;
		sptwb.Spt.Cdb[5] = 0x4F; // LifeLeft_CYL_LOW
		sptwb.Spt.Cdb[6] = 0xC2; // LifeLeft_CYL_HIGH
		sptwb.Spt.Cdb[7] = 0xA0;
		sptwb.Spt.Cdb[8] = 0xB0; // LifeLeft_CMD
		sptwb.Spt.Cdb[9] = 0x4C;
	}
	else if(type == CMD_TYPE_JMICRON)
	{
		sptwb.Spt.CdbLength = 12;
		sptwb.Spt.Cdb[0] = 0xDF;
		sptwb.Spt.Cdb[1] = 0x10;
		sptwb.Spt.Cdb[2] = 0x00;
```

```
            sptwb.Spt.Cdb[3] = 0x02;
            sptwb.Spt.Cdb[4] = 0x00;
            sptwb.Spt.Cdb[5] = 0xD1;  // READ_THRESHOLD
            sptwb.Spt.Cdb[6] = 0x01;
            sptwb.Spt.Cdb[7] = 0x01;
            sptwb.Spt.Cdb[8] = 0x4F;  // LifeLeft_CYL_LOW
            sptwb.Spt.Cdb[9] = 0xC2;  // LifeLeft_CYL_HIGH
            sptwb.Spt.Cdb[10] = 0xA0;
            sptwb.Spt.Cdb[11] = 0xB0; // LifeLeft_CMD
      }
      else if(type == CMD_TYPE_CYPRESS)
      {
            sptwb.Spt.CdbLength = 16;
            sptwb.Spt.Cdb[0] = 0x24;
            sptwb.Spt.Cdb[1] = 0x24;
            sptwb.Spt.Cdb[2] = 0x00;
            sptwb.Spt.Cdb[3] = 0xBE;
            sptwb.Spt.Cdb[4] = 0x01;
            sptwb.Spt.Cdb[5] = 0x00;
            sptwb.Spt.Cdb[6] = 0xD1;  // READ_THRESHOLD
            sptwb.Spt.Cdb[7] = 0x00;
            sptwb.Spt.Cdb[8] = 0x00;
            sptwb.Spt.Cdb[9] = 0x4F;  // LifeLeft_CYL_LOW
            sptwb.Spt.Cdb[10] = 0xC2; // LifeLeft_CYL_HIGH
            sptwb.Spt.Cdb[11] = 0xA0;
            sptwb.Spt.Cdb[12] = 0xB0; // ID_CMD
            sptwb.Spt.Cdb[13] = 0x00;
            sptwb.Spt.Cdb[14] = 0x00;
            sptwb.Spt.Cdb[15] = 0x00;
      }
      else
      {
            return FALSE;
      }

      length = offsetof(SCSI_PASS_THROUGH_WITH_BUFFERS, DataBuf) +
sptwb.Spt.DataTransferLength;
      bRet = ::DeviceIoControl(hIoCtrl, IOCTL_SCSI_PASS_THROUGH,
            &sptwb, sizeof(SCSI_PASS_THROUGH),
            &sptwb, length,   &dwReturned, NULL);

      ::CloseHandle(hIoCtrl);

      if(bRet == FALSE || dwReturned != length)
      {
            return      FALSE;
      }

      CString str;
      int j = 0;
      for(int i = 0; i < MAX_ATTRIBUTE; i++)
      {
            memcpy(     &(asi->Threshold[i]),
                        &(sptwb.DataBuf[i * sizeof(LifeLeft_THRESHOLD) + 2]),
sizeof(LifeLeft_THRESHOLD));

            if(asi->Threshold[j].Id != 0)
            {
                  j++;
            }
      }

      return      TRUE;
}

BOOL CAtaLifeLeft::ControlLifeLeftStatusSat(INT physicalDriveId, BYTE command,
COMMAND_TYPE type)
{
      BOOL  bRet;
```

```
	HANDLE	hIoCtrl;
	DWORD dwReturned;

	SCSI_PASS_THROUGH_WITH_BUFFERS sptwb;

	hIoCtrl = GetIoCtrlHandle(physicalDriveId);
	if(hIoCtrl == INVALID_HANDLE_VALUE)
	{
		return	FALSE;
	}

	::ZeroMemory(&sptwb,sizeof(SCSI_PASS_THROUGH_WITH_BUFFERS));

    sptwb.Spt.Length = sizeof(SCSI_PASS_THROUGH);
    sptwb.Spt.PathId = 0;
    sptwb.Spt.TargetId = 0;
    sptwb.Spt.Lun = 0;
    sptwb.Spt.SenseInfoLength = 24;
    sptwb.Spt.DataIn = SCSI_IOCTL_DATA_IN;
    sptwb.Spt.DataTransferLength = 0;
    sptwb.Spt.TimeOutValue = 2;
    sptwb.Spt.DataBufferOffset = offsetof(SCSI_PASS_THROUGH_WITH_BUFFERS,
DataBuf);
    sptwb.Spt.SenseInfoOffset = offsetof(SCSI_PASS_THROUGH_WITH_BUFFERS,
SenseBuf);
	if(type == CMD_TYPE_SAT)
	{
		sptwb.Spt.CdbLength = 12;
		sptwb.Spt.Cdb[0] = 0xA1; //ATA Проходів через (12) операцій коду
(A1h)
		sptwb.Spt.Cdb[1] = (3 << 1) | 0; //MULTIPLE_COUNT=0,PROTOCOL=3(Non-
Data),Reserved
		sptwb.Spt.Cdb[2] = (1 << 3) | (1 << 2) |
2;//OFF_LINE=0,CK_COND=0,Reserved=0,T_DIR=1(ToDevice),BYTE_BLOCK=1,T_LENGTH=2
		sptwb.Spt.Cdb[3] = command;//FEATURES (7:0)
		sptwb.Spt.Cdb[4] = 0;//SECTOR_COUNT (7:0)
		sptwb.Spt.Cdb[5] = 1;//LBA_LOW (7:0)
		sptwb.Spt.Cdb[6] = LifeLeft_CYL_LOW;//LBA_MID (7:0)
		sptwb.Spt.Cdb[7] = LifeLeft_CYL_HI;//LBA_HIGH (7:0)
		sptwb.Spt.Cdb[9] = LifeLeft_CMD;//COMMAND
	}
	else if(type == CMD_TYPE_SUNPLUS)
	{
		sptwb.Spt.CdbLength = 12;
		sptwb.Spt.Cdb[0] = 0xF8;
		sptwb.Spt.Cdb[1] = 0x00;
		sptwb.Spt.Cdb[2] = 0x22;
		sptwb.Spt.Cdb[3] = 0x10;
		sptwb.Spt.Cdb[4] = 0x01;
		sptwb.Spt.Cdb[5] = command;
		sptwb.Spt.Cdb[6] = 0x01;
		sptwb.Spt.Cdb[7] = 0x00;
		sptwb.Spt.Cdb[8] = 0x4F;  // LifeLeft_CYL_LOW
		sptwb.Spt.Cdb[9] = 0xC2;  // LifeLeft_CYL_HIGH
		sptwb.Spt.Cdb[10] = 0xA0;
		sptwb.Spt.Cdb[11] = 0xB0; // LifeLeft_CMD
	}
	else if(type == CMD_TYPE_IO_DATA)
	{
		sptwb.Spt.CdbLength = 12;
		sptwb.Spt.Cdb[0] = 0xE3;
		sptwb.Spt.Cdb[1] = 0x00; // ?
		sptwb.Spt.Cdb[2] = command;
		sptwb.Spt.Cdb[3] = 0x00; // ?
		sptwb.Spt.Cdb[4] = 0x00; // ?
		sptwb.Spt.Cdb[5] = 0x4F; // LifeLeft_CYL_LOW
		sptwb.Spt.Cdb[6] = 0xC2; // LifeLeft_CYL_HIGH
		sptwb.Spt.Cdb[7] = 0xA0; //
		sptwb.Spt.Cdb[8] = 0xB0; // LifeLeft_CMD
```

```
            sptwb.Spt.Cdb[9] = 0x00;
            sptwb.Spt.Cdb[10] = 0x00;
            sptwb.Spt.Cdb[11] = 0x00;
      }
      else if(type == CMD_TYPE_LOGITEC)
      {
            sptwb.Spt.CdbLength = 10;
            sptwb.Spt.Cdb[0] = 0xE0;
            sptwb.Spt.Cdb[1] = 0x00;
            sptwb.Spt.Cdb[2] = command;
            sptwb.Spt.Cdb[3] = 0x00;
            sptwb.Spt.Cdb[4] = 0x00;
            sptwb.Spt.Cdb[5] = 0x4F; // LifeLeft_CYL_LOW
            sptwb.Spt.Cdb[6] = 0xC2; // LifeLeft_CYL_HIGH
            sptwb.Spt.Cdb[7] = 0xA0;
            sptwb.Spt.Cdb[8] = 0xB0; // LifeLeft_CMD
            sptwb.Spt.Cdb[9] = 0x4C;
      }
      else if(type == CMD_TYPE_JMICRON)
      {
            sptwb.Spt.CdbLength = 12;
            sptwb.Spt.Cdb[0] = 0xDF;
            sptwb.Spt.Cdb[1] = 0x10;
            sptwb.Spt.Cdb[2] = 0x00;
            sptwb.Spt.Cdb[3] = 0x02;
            sptwb.Spt.Cdb[4] = 0x00;
            sptwb.Spt.Cdb[5] = command;
            sptwb.Spt.Cdb[6] = 0x01;
            sptwb.Spt.Cdb[7] = 0x01;
            sptwb.Spt.Cdb[8] = 0x4F;  // LifeLeft_CYL_LOW
            sptwb.Spt.Cdb[9] = 0xC2;  // LifeLeft_CYL_HIGH
            sptwb.Spt.Cdb[10] = 0xA0;
            sptwb.Spt.Cdb[11] = 0xB0; // LifeLeft_CMD
      }
      else if(type == CMD_TYPE_CYPRESS)
      {
            sptwb.Spt.CdbLength = 16;
            sptwb.Spt.Cdb[0] = 0x24;
            sptwb.Spt.Cdb[1] = 0x24;
            sptwb.Spt.Cdb[2] = 0x00;
            sptwb.Spt.Cdb[3] = 0xBE;
            sptwb.Spt.Cdb[4] = 0x00;
            sptwb.Spt.Cdb[5] = 0x00;
            sptwb.Spt.Cdb[6] = command;
            sptwb.Spt.Cdb[7] = 0x00;
            sptwb.Spt.Cdb[8] = 0x00;
            sptwb.Spt.Cdb[9] = 0x4F;  // LifeLeft_CYL_LOW
            sptwb.Spt.Cdb[10] = 0xC2; // LifeLeft_CYL_HIGH
            sptwb.Spt.Cdb[11] = 0xA0;
            sptwb.Spt.Cdb[12] = 0xB0; // ID_CMD
            sptwb.Spt.Cdb[13] = 0x00;
            sptwb.Spt.Cdb[14] = 0x00;
            sptwb.Spt.Cdb[15] = 0x00;
      }
      else
      {
            return FALSE;
      }

      DWORD length = offsetof(SCSI_PASS_THROUGH_WITH_BUFFERS, DataBuf) +
sptwb.Spt.DataTransferLength;
      bRet = ::DeviceIoControl(hIoCtrl, IOCTL_SCSI_PASS_THROUGH,
            &sptwb, sizeof(SCSI_PASS_THROUGH),
            &sptwb, length,   &dwReturned, NULL);

      ::CloseHandle(hIoCtrl);

      return      bRet;
}
```

```
BOOL CAtaLifeLeft::SendAtaCommandSat(INT physicalDriveId, BYTE main, BYTE sub,
BYTE param, COMMAND_TYPE type)
{
      BOOL  bRet;
      HANDLE      hIoCtrl;
      DWORD dwReturned;

      SCSI_PASS_THROUGH_WITH_BUFFERS sptwb;

      hIoCtrl = GetIoCtrlHandle(physicalDriveId);
      if(hIoCtrl == INVALID_HANDLE_VALUE)
      {
            return      FALSE;
      }

      ::ZeroMemory(&sptwb,sizeof(SCSI_PASS_THROUGH_WITH_BUFFERS));

    sptwb.Spt.Length = sizeof(SCSI_PASS_THROUGH);
    sptwb.Spt.PathId = 0;
    sptwb.Spt.TargetId = 0;
    sptwb.Spt.Lun = 0;
    sptwb.Spt.SenseInfoLength = 24;
    sptwb.Spt.DataIn = SCSI_IOCTL_DATA_IN;
    sptwb.Spt.DataTransferLength = 0;
    sptwb.Spt.TimeOutValue = 2;
    sptwb.Spt.DataBufferOffset = offsetof(SCSI_PASS_THROUGH_WITH_BUFFERS,
DataBuf);
    sptwb.Spt.SenseInfoOffset = offsetof(SCSI_PASS_THROUGH_WITH_BUFFERS,
SenseBuf);
      if(type == CMD_TYPE_SAT)
      {
            sptwb.Spt.CdbLength = 12;
            sptwb.Spt.Cdb[0]  = 0xA1; //ATA Проходів через (12) операцій коду
(A1h)
            sptwb.Spt.Cdb[1]  = (3 << 1) | 0; //MULTIPLE_COUNT=0,PROTOCOL=3(Non-
Data),Reserved
            sptwb.Spt.Cdb[2]  = (1 << 3) | (1 << 2) |
2;//OFF_LINE=0,CK_COND=0,Reserved=0,T_DIR=1(ToDevice),BYTE_BLOCK=1,T_LENGTH=2
            sptwb.Spt.Cdb[3]  = sub;            //FEATURES (7:0)
            sptwb.Spt.Cdb[4]  = param;          //SECTOR_COUNT (7:0)
            sptwb.Spt.Cdb[5]  = 0x00;           //LBA_LOW (7:0)
            sptwb.Spt.Cdb[6]  = 0x00;           //LBA_MID (7:0)
            sptwb.Spt.Cdb[7]  = 0x00;           //LBA_HIGH (7:0)
            sptwb.Spt.Cdb[8]  = 0xA0;           //DEVICE_HEAD
            sptwb.Spt.Cdb[9]  = main;           //COMMAND
            sptwb.Spt.Cdb[10] = 0x00;
            sptwb.Spt.Cdb[11] = 0x00;
      }
      else if(type == CMD_TYPE_SUNPLUS)
      {
            sptwb.Spt.CdbLength = 12;
            sptwb.Spt.Cdb[0]  = 0xF8;
            sptwb.Spt.Cdb[1]  = 0x00;
            sptwb.Spt.Cdb[2]  = 0x22;
            sptwb.Spt.Cdb[3]  = 0x10;
            sptwb.Spt.Cdb[4]  = 0x01;
            sptwb.Spt.Cdb[5]  = sub;
            sptwb.Spt.Cdb[6]  = param;
            sptwb.Spt.Cdb[7]  = 0x00;
            sptwb.Spt.Cdb[8]  = 0x00;
            sptwb.Spt.Cdb[9]  = 0x00;
            sptwb.Spt.Cdb[10] = 0xA0;
            sptwb.Spt.Cdb[11] = main;
      }
      else if(type == CMD_TYPE_IO_DATA)
      {
            sptwb.Spt.CdbLength = 12;
            sptwb.Spt.Cdb[0]  = 0xE3;
```

```
            sptwb.Spt.Cdb[1]  = 0x00;
            sptwb.Spt.Cdb[2]  = sub;
            sptwb.Spt.Cdb[3]  = param;
            sptwb.Spt.Cdb[4]  = 0x00;
            sptwb.Spt.Cdb[5]  = 0x00;
            sptwb.Spt.Cdb[6]  = 0x00;
            sptwb.Spt.Cdb[7]  = 0xA0;
            sptwb.Spt.Cdb[8]  = main;
            sptwb.Spt.Cdb[9]  = 0x00;
            sptwb.Spt.Cdb[10] = 0x00;
            sptwb.Spt.Cdb[11] = 0x00;
      }
      else if(type == CMD_TYPE_LOGITEC)
      {
            sptwb.Spt.CdbLength = 10;
            sptwb.Spt.Cdb[0] = 0xE0;
            sptwb.Spt.Cdb[1] = 0x00;
            sptwb.Spt.Cdb[2] = sub;
            sptwb.Spt.Cdb[3] = param;
            sptwb.Spt.Cdb[4] = 0x00;
            sptwb.Spt.Cdb[5] = 0x00;
            sptwb.Spt.Cdb[6] = 0x00;
            sptwb.Spt.Cdb[7] = 0xA0;
            sptwb.Spt.Cdb[8] = main;
            sptwb.Spt.Cdb[9] = 0x4C;            // ?
      }
      else if(type == CMD_TYPE_JMICRON)
      {
            sptwb.Spt.CdbLength = 12;
            sptwb.Spt.Cdb[0]  = 0xDF;
            sptwb.Spt.Cdb[1]  = 0x10;
            sptwb.Spt.Cdb[2]  = 0x00;
            sptwb.Spt.Cdb[3]  = 0x02;
            sptwb.Spt.Cdb[4]  = 0x00;
            sptwb.Spt.Cdb[5]  = sub;
            sptwb.Spt.Cdb[6]  = param;
            sptwb.Spt.Cdb[7]  = 0x00;
            sptwb.Spt.Cdb[8]  = 0x00;
            sptwb.Spt.Cdb[9]  = 0x00;
            sptwb.Spt.Cdb[10] = 0xA0;
            sptwb.Spt.Cdb[11] = main;
      }
      else if(type == CMD_TYPE_CYPRESS)
      {
            sptwb.Spt.CdbLength = 16;
            sptwb.Spt.Cdb[0]  = 0x24;
            sptwb.Spt.Cdb[1]  = 0x24;
            sptwb.Spt.Cdb[2]  = 0x00;
            sptwb.Spt.Cdb[3]  = 0xBE;
            sptwb.Spt.Cdb[4]  = 0x00;
            sptwb.Spt.Cdb[5]  = 0x00;
            sptwb.Spt.Cdb[6]  = sub;
            sptwb.Spt.Cdb[7]  = param;
            sptwb.Spt.Cdb[8]  = 0x00;
            sptwb.Spt.Cdb[9]  = 0x00;
            sptwb.Spt.Cdb[10] = 0x00;
            sptwb.Spt.Cdb[11] = 0xA0;
            sptwb.Spt.Cdb[12] = main;
            sptwb.Spt.Cdb[13] = 0x00;
            sptwb.Spt.Cdb[14] = 0x00;
            sptwb.Spt.Cdb[15] = 0x00;
      }
      else
      {
            return FALSE;
      }

      DWORD length = offsetof(SCSI_PASS_THROUGH_WITH_BUFFERS, DataBuf) +
sptwb.Spt.DataTransferLength;
```

```
	bRet = ::DeviceIoControl(hIoCtrl, IOCTL_SCSI_PASS_THROUGH,
		&sptwb, sizeof(SCSI_PASS_THROUGH),
		&sptwb, length,   &dwReturned, NULL);

	::CloseHandle(hIoCtrl);

	return	bRet;
}

/*-------------------------------------------------------------------------*/
// Підтримка функцій
/*-------------------------------------------------------------------------*/

DWORD CAtaLifeLeft::CheckDiskStatus(LifeLeft_ATTRIBUTE* attribute,
LifeLeft_THRESHOLD* threshold, DWORD attributeCount, DWORD vendorId)
{
	int error = 0;
	int caution = 0;
	BOOL flagUnknown = TRUE;

	for(DWORD j = 0; j < attributeCount; j++)
	{
		if( attribute[j].Id != 0xBE // Температура воздуха
		&&	threshold[j].ThresholdValue != 0
		&&	attribute[j].CurrentValue <= threshold[j].ThresholdValue)
		{
			error++;
		}

		switch(attribute[j].Id)
		{
		case 0x05: // Кількість перепризначених секторів
//		case 0xC4: // Кількість операцій перепризначення (ремаппінгу).
		case 0xC5: // Поточна кількість нестабільних секторів
		case 0xC6: // Кількість нескоректованих помилок
			if(attribute[j].RawValue[0] == 0xFF
			&& attribute[j].RawValue[1] == 0xFF
			&& attribute[j].RawValue[2] == 0xFF
			&& attribute[j].RawValue[3] == 0xFF)
			{
				flagUnknown = FALSE;
			}
			else
			{
				caution += attribute[j].RawValue[0]
						+ attribute[j].RawValue[1];
				flagUnknown = FALSE;
			}
			break;
		case 0xBB: // Визначення фірми-розробника
			if(vendorId == VENDOR_MTRON)
			{
				if(attribute[j].CurrentValue == 0)
				{
					error = 1;
				}
				else if(attribute[j].CurrentValue < 10)
				{
					caution = 1;
				}
				else
				{
					flagUnknown = FALSE;
				}
			}
			break;
		default:
			break;
		}
```

```
	}

	if(error > 0)
	{
		return DISK_STATUS_BAD;
	}
	else if(flagUnknown)
	{
		return DISK_STATUS_UNKNOWN;
	}
	else if(caution > 0)
	{
		return DISK_STATUS_CAUTION;
	}
	else
	{
		return DISK_STATUS_GOOD;
	}
}

VOID CAtaLifeLeft::ChangeByteOrder(PCHAR str, DWORD length)
{
	CHAR  temp;
	for(DWORD i = 0; i < length; i += 2)
	{
		temp = str[i];
		str[i] = str[i+1];
		str[i+1] = temp;
	}
}

BOOL CAtaLifeLeft::CheckAsciiStringError(PCHAR str, DWORD length)
{
	BOOL flag = FALSE;
	for(DWORD i = 0; i < length; i++)
	{
		if((0x00 < str[i] && str[i] < 0x20) || str[i] >= 0x7f)
		{
			flag = TRUE;
			break;
		}
	}
	return flag;
}

DWORD CAtaLifeLeft::GetPowerOnHours(DWORD rawValue, DWORD timeUnitType)
{
	switch(timeUnitType)
	{
	case POWER_ON_UNKNOWN:
		return 0;
		break;
	case POWER_ON_HOURS:
		return rawValue;
		break;
	case POWER_ON_MINUTES:
		return rawValue / 60;
		break;
	case POWER_ON_HALF_MINUTES:
		return rawValue / 120;
		break;
	case POWER_ON_SECONDS:
		return rawValue / 60 / 60;
		break;
	default:
		return rawValue;
		break;
	}
}
```

```
DWORD CAtaLifeLeft::GetPowerOnHoursEx(DWORD i, DWORD timeUnitType)
{
	DWORD rawValue = vars[i].PowerOnRawValue;
	switch(timeUnitType)
	{
	case POWER_ON_UNKNOWN:
		return 0;
		break;
	case POWER_ON_HOURS:
		return rawValue;
		break;
	case POWER_ON_MINUTES:
		return rawValue / 60;
		break;
	case POWER_ON_HALF_MINUTES:
		return rawValue / 120;
		break;
	case POWER_ON_SECONDS:
		return rawValue / 60 / 60;
		break;
	default:
		return rawValue;
		break;
	}
}

DWORD CAtaLifeLeft::GetTransferMode(WORD w63, WORD w76, WORD w88, CString
&current, CString &max, CString &type, INTERFACE_TYPE* interfaceType)
{
	DWORD tm = TRANSFER_MODE_PIO;
	current = max = _T("");
	type = _T("Parallel ATA");
	*interfaceType = INTERFACE_TYPE_PATA;

	// Слово DMA або PIO
	if(w63 & 0x0700)
	{
		tm = TRANSFER_MODE_PIO_DMA;
		current = max = _T("PIO / DMA");
	}

	// Ultara DMA Максимальний режим передачі
		if(w88 & 0x0040){tm = TRANSFER_MODE_ULTRA_DMA_133; max = _T("Ultra
DMA/133");}
	else if(w88 & 0x0020){tm = TRANSFER_MODE_ULTRA_DMA_100; max = _T("Ultra
DMA/100");}
	else if(w88 & 0x0010){tm = TRANSFER_MODE_ULTRA_DMA_66;  max = _T("Ultra
DMA/66");}
	else if(w88 & 0x0008){tm = TRANSFER_MODE_ULTRA_DMA_44;  max = _T("Ultra
DMA/44");}
	else if(w88 & 0x0004){tm = TRANSFER_MODE_ULTRA_DMA_33;  max = _T("Ultra
DMA/33");}
	else if(w88 & 0x0002){tm = TRANSFER_MODE_ULTRA_DMA_25;  max = _T("Ultra
DMA/25");}
	else if(w88 & 0x0001){tm = TRANSFER_MODE_ULTRA_DMA_16;  max = _T("Ultra
DMA/16");}

	// Ultara DMA поточний режим передачі
		 if(w88 & 0x4000){current = _T("Ultra DMA/133");}
	else if(w88 & 0x2000){current = _T("Ultra DMA/100");}
	else if(w88 & 0x1000){current = _T("Ultra DMA/66");}
	else if(w88 & 0x0800){current = _T("Ultra DMA/44");}
	else if(w88 & 0x0400){current = _T("Ultra DMA/33");}
	else if(w88 & 0x0200){current = _T("Ultra DMA/25");}
	else if(w88 & 0x0100){current = _T("Ultra DMA/16");}

	// Serial ATA
	if(w76 != 0x0000 && w76 != 0xFFFF)
```

```
	{
		current = max = _T("SATA/150");
		type = _T("Serial ATA");
		*interfaceType = INTERFACE_TYPE_SATA;
	}

		 if(w76 & 0x0010){tm = TRANSFER_MODE_UNKNOWN;  current = max =
_T("Unknown");}
	else if(w76 & 0x0008){tm = TRANSFER_MODE_SATA_600; current = max =
_T("SATA/600");}
	else if(w76 & 0x0004){tm = TRANSFER_MODE_SATA_300; current = max =
_T("SATA/300");}
	else if(w76 & 0x0002){tm = TRANSFER_MODE_SATA_150; current = max =
_T("SATA/150");}

	return tm;
}

DWORD CAtaLifeLeft::GetTimeUnitType(CString model, CString firmware, DWORD
major, DWORD transferMode)
{
	model.MakeUpper();

	if(model.Find(_T("FUJITSU")) == 0)
	{
		if(major >= 8)
		{
			return POWER_ON_HOURS;
		}
		else
		{
			return POWER_ON_SECONDS;
		}
	}
	else if(model.Find(_T("HITACHI_DK")) == 0)
	{
		return POWER_ON_MINUTES;
	}
	else if(model.Find(_T("MAXTOR")) == 0)
	{
		if(transferMode >= TRANSFER_MODE_SATA_300
		|| model.Find(_T("MAXTOR 6H")) == 0	// Maxtor DiamondMax 11
сімейство
		|| model.Find(_T("MAXTOR 7H500")) == 0	// Maxtor MaXLine Pro 500
сімейство
		|| model.Find(_T("MAXTOR 6L0")) == 0	// Maxtor DiamondMax Plus
D740X сімейство
		|| model.Find(_T("MAXTOR 4K")) == 0	// Maxtor DiamondMax D540X-
4K сімейство
		)
		{
			return POWER_ON_HOURS;
		}
		else
		{
			return POWER_ON_MINUTES;
		}
	}
	else if(model.Find(_T("SAMSUNG")) == 0)
	{
		if(transferMode >= TRANSFER_MODE_SATA_300)
		{
			return POWER_ON_HOURS;
		}
		else if(-23 >= _tstoi(firmware.Right(3)) &&
_tstoi(firmware.Right(3)) >= -39)
		{
			return POWER_ON_HALF_MINUTES;
		}
```

```
		else if(model.Find(_T("SAMSUNG SV")) == 0
		||		model.Find(_T("SAMSUNG SP")) == 0
		||		model.Find(_T("SAMSUNG HM")) == 0
		)
		{
			return POWER_ON_HALF_MINUTES;
		}
		else
		{
			return POWER_ON_HOURS;
		}
	}
	else
	{
		return POWER_ON_HOURS;
	}
}

DWORD CAtaLifeLeft::GetAtaMajorVersion(WORD w80, CString &majorVersion)
{
	DWORD major = 0;

	if(w80 == 0x0000 || w80 == 0xFFFF)
	{
		return FALSE;
	}

	for(int i = 14; i > 0; i--)
	{
		if((w80 >> i) & 0x1)
		{
			major = i;
			break;
		}
	}

	if(major == 15)
	{
		majorVersion = _T("");
	}
	else if(major >= 8)
	{
		majorVersion.Format(_T("ATA%d-ACS"), major);
	}
	else if(major >= 4)
	{
		majorVersion.Format(_T("ATA/ATAPI-%d"), major);
	}
	else if(major == 0)
	{
		majorVersion = _T("");
	}
	else
	{
		majorVersion.Format(_T("ATA-%d"), major);
	}

	return major;
}


/*
DWORD CAtaLifeLeft::GetMaxtorPowerOnHours(DWORD currentValue, DWORD rawValue)
{
	if(200 < currentValue && currentValue <= 253)
	{
		return ((253 - currentValue) * 65536 + rawValue) / 60;
	}
	else if(100 < currentValue && currentValue <= 200)
```

```
        {
              return ((200 - currentValue) * 65536 + rawValue) / 60;
        }
        else if(currentValue <= 100)
        {
              return ((100 - currentValue) * 65536 + rawValue) / 60;
        }
        else
        {
              return rawValue / 60;
        }
}
*/
```

Файл AtaLifeLeft.h – бібліотека для файлу AtaLifeLeft.cpp

```
#pragma once

#include "winioctl.h"
#include "SPTIUtil.h"


class CAtaLifeLeft
{
public:
	static const int MAX_DISK = 32;// FIX
	static const int MAX_ATTRIBUTE = 30; // FIX
	static const int MAX_SEARCH_PHYSICAL_DRIVE = 32;
	static const int MAX_SEARCH_SCSI_PORT = 16;
	static const int MAX_SEARCH_SCSI_TARGET_ID = 8;

	static const int SCSI_MINIPORT_BUFFER_SIZE = 512;

public:
	CAtaLifeLeft();
	virtual ~CAtaLifeLeft();

	enum LifeLeft_STATUS
	{
		LifeLeft_STATUS_NO_CHANGE = 0,
		LifeLeft_STATUS_MINOR_CHANGE,
		LifeLeft_STATUS_MAJOR_CHANGE
	};

	enum TRANSFER_MODE
	{
		TRANSFER_MODE_UNKNOWN = 0,
		TRANSFER_MODE_PIO,
		TRANSFER_MODE_PIO_DMA,
		TRANSFER_MODE_ULTRA_DMA_16,
		TRANSFER_MODE_ULTRA_DMA_25,
		TRANSFER_MODE_ULTRA_DMA_33,
		TRANSFER_MODE_ULTRA_DMA_44,
		TRANSFER_MODE_ULTRA_DMA_66,
		TRANSFER_MODE_ULTRA_DMA_100,
		TRANSFER_MODE_ULTRA_DMA_133,
		TRANSFER_MODE_SATA_150,
		TRANSFER_MODE_SATA_300,
		TRANSFER_MODE_SATA_600
	};

	enum DISK_STATUS
	{
		DISK_STATUS_UNKNOWN = 0,
		DISK_STATUS_GOOD,
		DISK_STATUS_CAUTION,
		DISK_STATUS_BAD
	};

	enum POWER_ON_HOURS_UNIT
	{
		POWER_ON_UNKNOWN = 0,
		POWER_ON_HOURS,
		POWER_ON_MINUTES,
		POWER_ON_HALF_MINUTES,
		POWER_ON_SECONDS,
	};

	enum COMMAND_TYPE
	{
		CMD_TYPE_PHYSICAL_DRIVE = 0,
```

```
            CMD_TYPE_SCSI_MINIPORT,
            CMD_TYPE_SAT,                        // SAT = SCSI_ATA_TRANSLATION
            CMD_TYPE_SUNPLUS,
            CMD_TYPE_IO_DATA,
            CMD_TYPE_LOGITEC,
            CMD_TYPE_JMICRON,
            CMD_TYPE_CYPRESS,
            CMD_TYPE_PROLIFIC,
            CMD_TYPE_DEBUG
      };

      enum VENDOR_ID
      {
            VENDOR_UNKNOWN      = 0x0000,
            VENDOR_MTRON    = 0x0001,

            USB_VENDOR_BUFFALO  = 0x0411,
            USB_VENDOR_IO_DATA  = 0x04BB,
            USB_VENDOR_LOGITEC  = 0x0789,
            USB_VENDOR_INITIO   = 0x13FD,
            USB_VENDOR_SUNPLUS  = 0x04FC,
            USB_VENDOR_JMICRON  = 0x152D,
            USB_VENDOR_CYPRESS  = 0x04B4,
            USB_VENDOR_OXFORD   = 0x0928,
            USB_VENDOR_PROLIFIC = 0x067B,

      };

      enum INTERFACE_TYPE
      {
            INTERFACE_TYPE_UNKNOWN = 0,
            INTERFACE_TYPE_PATA,
            INTERFACE_TYPE_SATA,
            INTERFACE_TYPE_USB,
            INTERFACE_TYPE_IEEE1394
      };

protected:
      enum IO_CONTROL_CODE
      {
            DFP_SEND_DRIVE_COMMAND  = 0x0007C084,
            DFP_RECEIVE_DRIVE_DATA  = 0x0007C088,
            IOCTL_SCSI_MINIPORT     = 0x0004D008,
            IOCTL_IDE_PASS_THROUGH  = 0x0004D028, // 2000 або пізніша версія
            IOCTL_ATA_PASS_THROUGH  = 0x0004D02C, // XP SP2 and 2003 або пізніша
версія
      };

#pragma pack(push,1)

      typedef     struct _IDENTIFY_DEVICE_OUTDATA
      {
            SENDCMDOUTPARAMS  SendCmdOutParam;
            BYTE                    Data[IDENTIFY_BUFFER_SIZE - 1];
      } IDENTIFY_DEVICE_OUTDATA, *PIDENTIFY_DEVICE_OUTDATA;

      typedef     struct _LifeLeft_READ_DATA_OUTDATA
      {
            SENDCMDOUTPARAMS  SendCmdOutParam;
            BYTE                    Data[READ_ATTRIBUTE_BUFFER_SIZE - 1];
      } LifeLeft_READ_DATA_OUTDATA, *PLifeLeft_READ_DATA_OUTDATA;

      typedef struct _CMD_IDE_PATH_THROUGH
      {
            IDEREGS     reg;
            DWORD   length;
            BYTE    buffer[1];
      } CMD_IDE_PATH_THROUGH, *PCMD_IDE_PATH_THROUGH;
```

```
static const int ATA_FLAGS_DRDY_REQUIRED = 0x01;
static const int ATA_FLAGS_DATA_IN       = 0x02;
static const int ATA_FLAGS_DATA_OUT      = 0x04;
static const int ATA_FLAGS_48BIT_COMMAND = 0x08;

typedef struct _ATA_PASS_THROUGH_EX
{
      WORD    Length;
      WORD    AtaFlags;
      BYTE    PathId;
      BYTE    TargetId;
      BYTE    Lun;
      BYTE    ReservedAsUchar;
      DWORD   DataTransferLength;
      DWORD   TimeOutValue;
      DWORD   ReservedAsUlong;
      DWORD_PTR   DataBufferOffset;
      IDEREGS PreviousTaskFile;
      IDEREGS CurrentTaskFile;
} ATA_PASS_THROUGH_EX, *PCMD_ATA_PASS_THROUGH_EX;

typedef struct
{
      ATA_PASS_THROUGH_EX Apt;
      BYTE  Buf[512];
} ATA_PASS_THROUGH_EX_WITH_BUFFERS;

typedef     struct LifeLeft_ATTRIBUTE
{
      BYTE  Id;
      WORD  StatusFlags;
      BYTE  CurrentValue;
      BYTE  WorstValue;
      BYTE  RawValue[6];
      BYTE  Reserved;
};

typedef     struct LifeLeft_THRESHOLD
{
      BYTE  Id;
      BYTE  ThresholdValue;
      BYTE  Reserved[10];
};

typedef struct SRB_IO_CONTROL
{
   ULONG    HeaderLength;
   UCHAR    Signature[8];
   ULONG    Timeout;
   ULONG    ControlCode;
   ULONG    ReturnCode;
   ULONG    Length;
};

typedef struct SRB_IO_COMMAND
{
      SRB_IO_CONTROL    Cntrol;
      IDEREGS                 IdeRegs;
      BYTE              Data[512];
};

struct IDENTIFY_DEVICE
{
      WORD        GeneralConfiguration;                   //0
      WORD        LogicalCylinders;                       //1
Застарівший
      WORD        SpecificConfiguration;                  //2
      WORD        LlogicalHeads;                                  //3
Застарівший
```

```
            WORD        Retired1[2];                                //4-5
            WORD        LogicalSectors;                             //6
Застарівший
            DWORD       ReservedForCompactFlash;                    //7-8
            WORD        Retired2;                                   //9
            CHAR        SerialNumber[20];                           //10-19
            WORD        Retired3;                                   //20
            WORD        BufferSize;                                 //21
Застарівший
            WORD        Obsolute4;                                  //22
            CHAR        FirmwareRev[8];                             //23-26
            CHAR        Model[40];                                  //27-46
            WORD        MaxNumPerInterupt;                          //47
            WORD        Reserved1;                                  //48
            WORD        Capabilities1;                              //49
            WORD        Capabilities2;                              //50
            DWORD       Obsolute5;                                  //51-52
            WORD        Field88and7064;                             //53
            WORD        Obsolute6[5];                               //54-58
            WORD        MultSectorStuff;                            //59
            DWORD       TotalAddressableSectors;                    //60-61
            WORD        Obsolute7;                                  //62
            WORD        MultiWordDma;                               //63
            WORD        PioMode;                                    //64
            WORD        MinMultiwordDmaCycleTime;                   //65
            WORD        RecommendedMultiwordDmaCycleTime;   //66
            WORD        MinPioCycleTimewoFlowCtrl;                  //67
            WORD        MinPioCycleTimeWithFlowCtrl;        //68
            WORD        Reserved2[6];                               //69-74
            WORD        QueueDepth;                                 //75
            WORD        SerialAtaCapabilities;                      //76
            WORD        ReservedForFutureSerialAta;                 //77
            WORD        SerialAtaFeaturesSupported;                 //78
            WORD        SerialAtaFeaturesEnabled;                   //79
            WORD        MajorVersion;                               //80
            WORD        MinorVersion;                               //81
            WORD        CommandSetSupported1;                       //82
            WORD        CommandSetSupported2;                       //83
            WORD        CommandSetSupported3;                       //84
            WORD        CommandSetEnabled1;                         //85
            WORD        CommandSetEnabled2;                         //86
            WORD        CommandSetDefault;                          //87
            WORD        UltraDmaMode;                               //88
            WORD        TimeReqForSecurityErase;                    //89
            WORD        TimeReqForEnhancedSecure;                   //90
            WORD        CurrentPowerManagement;                     //91
            WORD        MasterPasswordRevision;                     //92
            WORD        HardwareResetResult;                        //93
            WORD        AcoustricManagement;                        //94
            WORD        StreamMinRequestSize;                       //95
            WORD        StreamingTimeDma;                           //96
            WORD        StreamingAccessLatency;                     //97
            DWORD       StreamingPerformance;                       //98-99
            ULONGLONG   MaxUserLba;                                 //100-103
            WORD        StremingTimePio;                            //104
            WORD        Reserved3;                                  //105
            WORD        SectorSize;                                 //106
            WORD        InterSeekDelay;                             //107
            WORD        IeeeOui;                                    //108
            WORD        UniqueId3;                                  //109
            WORD        UniqueId2;                                  //110
            WORD        UniqueId1;                                  //111
            WORD        Reserved4[4];                               //112-115
            WORD        Reserved5;                                  //116
            DWORD       WordsPerLogicalSector;                      //117-118
            WORD        Reserved6[8];                               //119-126
            WORD        RemovableMediaStatus;                       //127
            WORD        SecurityStatus;                             //128
            WORD        VendorSpecific[31];                         //129-159
```

```
            WORD        CfaPowerMode1;                              //160
            WORD        Reserved7[15];                              //161-175
            CHAR        CurrentMediaSerialNo[60];                   //176-205
            WORD        SctCommandTransport;                        //206  254
            WORD        ReservedForCeAta1[2];                       //207-208
            WORD        AlignmentOfLogicalBlocks;                   //209
            DWORD       WriteReadVerifySectorCountMode3;    //210-211
            DWORD       WriteReadVerifySectorCountMode2;    //212-213
            WORD        NvCacheCapabilities;                        //214
            DWORD       NvCacheSizeLogicalBlocks;                   //215-216
            WORD        NominalMediaRotationRate;                   //217
            WORD        Reserved8;                                  //218
            WORD        NvCacheOptions1;                            //219
            WORD        NvCacheOptions2;                            //220
            WORD        Reserved9;                                  //221
            WORD        TransportMajorVersionNumber;        //222
            WORD        TransportMinorVersionNumber;        //223
            WORD        ReservedForCeAta2[10];                      //224-233
            WORD        MinimumBlocksPerDownloadMicrocode;  //234
            WORD        MaximumBlocksPerDownloadMicrocode;  //235
            WORD        Reserved10[19];                             //236-254
            WORD        IntegrityWord;                              //255
      };
#pragma     pack(pop)

public:
      DWORD UpdateLifeLeftInfo(DWORD index);
      BOOL UpdateIdInfo(DWORD index);
      BYTE GetAamValue(DWORD index);
      BYTE GetApmValue(DWORD index);
      BOOL EnableAam(DWORD index, BYTE param);
      BOOL EnableApm(DWORD index, BYTE param);
      BOOL DisableAam(DWORD index);
      BOOL DisableApm(DWORD index);
      BYTE GetRecommendAamValue(DWORD index);
      BYTE GetRecommendApmValue(DWORD index);

      BOOL Init(BOOL useWmi, BOOL advancedDiskSearch, PBOOL flagChangeDisk);
      BOOL MeasuredTimeUnit();
      DWORD GetPowerOnHours(DWORD rawValue, DWORD timeUnitType);
      DWORD GetPowerOnHoursEx(DWORD index, DWORD timeUnitType);

      struct DISK_POSITION
      {
            INT                     PhysicalDriveId;
            INT                     ScsiPort;
            INT                     ScsiTargetId;
      };

      struct ATA_LifeLeft_INFO
      {
            IDENTIFY_DEVICE         IdentifyDevice;
            LifeLeft_ATTRIBUTE            Attribute[MAX_ATTRIBUTE];
            LifeLeft_THRESHOLD            Threshold[MAX_ATTRIBUTE];

            BOOL                    IsLifeLeftEnabled;
            BOOL                    IsCheckSumError;
            BOOL                    IsWord88;
            BOOL                    IsWord64_76;

            BOOL                    IsLifeLeftSupported;
            BOOL                    IsLba48Supported;
            BOOL                    IsAamSupported;
            BOOL                    IsApmSupported;
            BOOL                    IsAamEnabled;
            BOOL                    IsApmEnabled;
            BOOL                    IsNcqSupported;
            BOOL                    IsNvCacheSupported;
            BOOL                    IsMaxtorMinute;
```

```
		INT				PhysicalDriveId;
		INT				ScsiPort;
		INT				ScsiTargetId;
//		INT				AccessType;

		DWORD			TotalDiskSize;
		DWORD			Cylinder;
		DWORD			Head;
		DWORD			Sector;
		DWORD			Sector28;
		ULONGLONG		Sector48;
		DWORD			DiskSizeChs;
		DWORD			DiskSizeLba28;
		DWORD			DiskSizeLba48;
		DWORD			BufferSize;
		ULONGLONG		NvCacheSize;
		DWORD			TransferModeType;
		DWORD			DetectedTimeUnitType;
		DWORD			MeasuredTimeUnitType;
		DWORD			AttributeCount;
		INT				DetectedPowerOnHours;
		INT				MeasuredPowerOnHours;
		INT				PowerOnRawValue;
		INT				PowerOnStartRawValue;
		DWORD			PowerOnCount;
		DWORD			Temperature;
		double			Speed;

		INT				Life;

		DWORD			Major;
		DWORD			Minor;

		DWORD			DiskStatus;
		DWORD			DriveLetterMap;
		//
		DWORD			AlarmTemperature;
		BOOL			AlarmHealthStatus;

		INTERFACE_TYPE	InterfaceType;
		COMMAND_TYPE	CommandType;

		DWORD			VendorId;
		DWORD			ProductId;

		CString			SerialNumber;
		CString			SerialNumberReverse;
		CString			FirmwareRev;
		CString			FirmwareRevReverse;
		CString			Model;
		CString			ModelReverse;
		CString			ModelWmi;
		CString			ModelSerial;
		CString			DriveMap;
		CString			MaxTransferMode;
		CString			CurrentTransferMode;
		CString			MajorVersion;
		CString			MinorVersion;
		CString			Interface;
		CString			Enclosure;
		CString			CommandTypeString;
	};

	struct EXTERNAL_DISK_INFO
	{
		CString Enclosure;
		DWORD VendorId;
		DWORD ProductId;
```

```
	};

	CArray<ATA_LifeLeft_INFO, ATA_LifeLeft_INFO> vars;
	CArray<EXTERNAL_DISK_INFO, EXTERNAL_DISK_INFO> externals;

	CStringArray m_IdeController;
	CStringArray m_ScsiController;
	CStringArray m_UsbController;
	CString m_ControllerMap;

	BOOL IsEnabledWmi;
	DWORD MeasuredGetTickCount;

protected:
	OSVERSIONINFOEX m_Os;
	CString m_SerialNumberA_Z[26];
	BOOL m_FlagAtaPassThrough;

	BOOL GetDiskInfo(INT physicalDriveId, INT scsiPort, INT scsiTargetId,
INTERFACE_TYPE interfaceType, VENDOR_ID vendorId);
	BOOL AddDisk(INT PhysicalDriveId, INT ScsiPort, INT scsiTargetId,
COMMAND_TYPE commandType, IDENTIFY_DEVICE* identify);
	DWORD CheckLifeLeftAttributeUpdate(DWORD index, LifeLeft_ATTRIBUTE* pre,
LifeLeft_ATTRIBUTE* cur);

	VOID InitAtaInfo();
	VOID InitAtaInfoByWmi();
	VOID InitStruct();
	VOID ChangeByteOrder(PCHAR str, DWORD length);
	BOOL CheckAsciiStringError(PCHAR str, DWORD length);
	HANDLE GetIoCtrlHandle(BYTE index);
	BOOL SendAtaCommand(DWORD i, BYTE main, BYTE sub, BYTE param);

	BOOL DoIdentifyDevicePd(INT physicalDriveId, IDENTIFY_DEVICE* identify);
	BOOL GetLifeLeftAttributePd(INT physicalDriveId, ATA_LifeLeft_INFO* asi);
	BOOL GetLifeLeftThresholdPd(INT physicalDriveId, ATA_LifeLeft_INFO* asi);
	BOOL ControlLifeLeftStatusPd(INT physicalDriveId, BYTE command);
	BOOL SendAtaCommandPd(INT physicalDriveId, BYTE main, BYTE sub, BYTE
param, PBYTE data, DWORD dataSize);

	BOOL DoIdentifyDeviceScsi(INT scsiPort, INT scsiTargetId, IDENTIFY_DEVICE*
identify);
	BOOL GetLifeLeftAttributeScsi(INT scsiPort, INT scsiTargetId,
ATA_LifeLeft_INFO* asi);
	BOOL GetLifeLeftThresholdScsi(INT scsiPort, INT scsiTargetId,
ATA_LifeLeft_INFO* asi);
	BOOL ControlLifeLeftStatusScsi(INT scsiPort, INT scsiTargetId, BYTE
command);
	BOOL SendAtaCommandScsi(INT scsiPort, INT scsiTargetId, BYTE main, BYTE
sub, BYTE param);

	BOOL DoIdentifyDeviceSat(INT physicalDriveId, IDENTIFY_DEVICE* identify,
COMMAND_TYPE commandType);
	BOOL GetLifeLeftAttributeSat(INT physicalDriveId, ATA_LifeLeft_INFO* asi);
	BOOL GetLifeLeftThresholdSat(INT physicalDriveId, ATA_LifeLeft_INFO* asi);
	BOOL ControlLifeLeftStatusSat(INT physicalDriveId, BYTE command,
COMMAND_TYPE commandType);
	BOOL SendAtaCommandSat(INT physicalDriveId, BYTE main, BYTE sub, BYTE
param, COMMAND_TYPE commandType);

	DWORD CheckDiskStatus(LifeLeft_ATTRIBUTE* attribute, LifeLeft_THRESHOLD*
threshold, DWORD attributeCount, DWORD vendorId);

	DWORD GetTransferMode(WORD w63, WORD w76, WORD w88, CString
&currentTransferMode, CString &maxTransferMode, CString &Interface,
INTERFACE_TYPE *interfaceType);
	DWORD GetTimeUnitType(CString model, CString firmware, DWORD major, DWORD
transferMode);
	DWORD GetAtaMajorVersion(WORD w80, CString &majorVersion);
```

```
//     DWORD GetMaxtorPowerOnHours(DWORD currentValue, DWORD rawValue);

       static int Compare(const void *p1, const void *p2);
};
```

Файл AboutDlg.cpp – довідка

```
#include "stdafx.h"
#include "DiskInfo.h"
#include "AboutDlg.h"

IMPLEMENT_DYNCREATE(CAboutDlg, CDHtmlDialog)

CAboutDlg::CAboutDlg(CWnd* pParent /*=NULL*/)
                  : CDHtmlDialogEx(CAboutDlg::IDD, CAboutDlg::IDH, pParent)
{

}

CAboutDlg::~CAboutDlg()
{
}

BOOL CAboutDlg::OnInitDialog()
{
                  CDHtmlDialogEx::OnInitDialog();

                  InitDHtmlDialog(SIZE_X, SIZE_Y, ((CDiskInfoApp*)AfxGetApp())-
>m_AboutDlgPath);

                  return TRUE;
}

void CAboutDlg::OnDocumentComplete(LPDISPATCH pDisp, LPCTSTR szUrl)
{
                  CString cstr;
                  cstr = szUrl;
                  if(cstr.Find(_T("html")) != -1 || cstr.Find(_T("dlg")) != -1)
                  {
                        m_FlagShowWindow = TRUE;
                       m_Copyright = PRODUCT_COPYRIGHT;
                        UpdateData(FALSE);
                        ShowWindow(SW_SHOW);
                  }
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDHtmlDialogEx)
END_MESSAGE_MAP()

BEGIN_DHTML_EVENT_MAP(CAboutDlg)
                  DHTML_EVENT_ONCLICK(_T("CrystalDewWorld"), OnCrystalDewWorld)
END_DHTML_EVENT_MAP()

HRESULT CAboutDlg::OnCrystalDewWorld(IHTMLElement* /*pElement*/)
{
                  if(GetUserDefaultLCID() == 0x0411)// Japanese
                  {
                        ShellExecute(NULL, NULL, URL_CRYSTAL_DEW_WORLD_JA, NULL,
NULL, SW_SHOWNORMAL);
                  }
                  else // Other Language
                  {
                        ShellExecute(NULL, NULL, URL_CRYSTAL_DEW_WORLD_EN, NULL,
NULL, SW_SHOWNORMAL);
                  }

                  return S_FALSE;
}
```

Файл AboutDlg.h – бібліотека для файлу AboutDlg.cpp

```
#pragma once

class CAboutDlg : public CDHtmlDialogEx
{
	DECLARE_DYNCREATE(CAboutDlg)

	static const int SIZE_X = 240;
	static const int SIZE_Y = 200;

public:
	CAboutDlg(CWnd* pParent = NULL);
	virtual ~CAboutDlg();

	enum { IDD = IDD_ABOUT, IDH = IDR_HTML_DUMMY };

protected:
	virtual void DoDataExchange(CDataExchange* pDX);
	virtual BOOL OnInitDialog();
	virtual void OnDocumentComplete(LPDISPATCH pDisp, LPCTSTR szUrl);

	CString m_Version;
	CString m_Release;
	CString m_Copyright;

	HRESULT OnCrystalDewWorld(IHTMLElement *pElement);

	DECLARE_MESSAGE_MAP()
	DECLARE_DHTML_EVENT_MAP()
};
```