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

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

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи візуалізації системного та реального часу ПК”

Виконав здобувач вищої освіти
IV курсу, групи КІ-19
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Подвалков Є.М.
«____» _____ 2023 р.

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

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

Галузь знань ._12 _____*“*Інформаційні технології*”*_____

Спеціальність *123 “*Комп’ютерна інженерія*”*

Освітньо-професійна (освітньо-наукова) програма _*“*Комп’ютерна інженерія*”*_

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

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

Подвалкову Єгору Михайловичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи візуалізації системного та реального часу ПК*

2. Керівник роботи *Коваленко Анна Степанівна, канд. техн. наук, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 7-02 від 5.01.2023 року

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

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи візуалізації системного та реального часу ПК*

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

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

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

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

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

КАЛЕНДАРНИЙ ПЛАН

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

Дата видачі завдання
« 17 » січня 2023 р.

Підпис керівника
_____ Коваленко А.С.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2023 р.

Підпис здобувача
_____ Подвалков Є.М.
(прізвище та ініціали)

АНОТАЦІЯ

Подвалков Є.М. Програмне забезпечення системи візуалізації системного та реального часу ПК. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи візуалізації системного та реального часу ПК.

Метою розробки є програмне забезпечення системи візуалізації системного та реального часу ПК.

Результат роботи – програмна реалізація системи візуалізації системного та реального часу ПК.

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

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

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

Програму розроблено в середовищі Builder C++.

Ключові слова: комп'ютерна інженерія, системний та реальний час ПК

ABSTRACT

Podvalkov E.M. PC system and real-time visualization system software. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this final qualification work for the first (bachelor) level of higher education, software is developed, which is intended for the PC system and real-time visualization system.

The goal of development is PC system and real-time visualization system software.

The result of the work is the software implementation of the PC system and real-time visualization system.

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

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

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

The program was developed in the Builder C++ environment.

Keywords: computer engineering, system and real-time PC

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ЕОМ	–	Електронно-обчислювальна машина
ПК	–	Персональний комп'ютер
ОС	–	Операційна система
AS	–	Адресний строб
BIOS	–	Базова система введення/виведення
CE	–	Вихід дешифратора адреси
CKOUT	–	Синхросигнал
CMOS	–	Complementary-symmetry/metal-oxide semiconductor
DS	–	Строб даних
INT	–	Переривання
IRQ	–	Запит на переривання
NMI	–	Немаскуєме переривання
PS	–	Сигнал стану живлення
Rijndael	–	Алгоритм шифрування
RTC	–	Годинник реального часу
R/W	–	Читання/запис
SQW	–	Вихідні прямокутні імпульси – меандр
VCL	–	Інтегрована бібліотека візуальних компонентів

ВСТУП

Актуальність теми. Розвиток засобів обчислювальної техніки відбувається в багатьох напрямках, що розширюють сферу застосування ЕОМ і підвищують ефективність їхнього використання. Але для того, щоб обслуговувати цю техніку, необхідно підготовити спеціалістів високого рівня, які б були спроможні це зробити.

У процесі підготовки потрібно вивчати, не тільки ЕОМ як систему, але й різні функціональні та структурні частини ЕОМ, на фізичному та прикладному рівні.

Обов'язковим елементом системної плати комп'ютера є RTC ("Real-Time Clock" – годинник реального часу). Цей годинник має автономне джерело живлення й продовжує функціонувати, навіть якщо комп'ютер виключений. При включенні комп'ютера BIOS зчитує показання годинника реального часу й надалі здійснює відлік часу самостійно. Операційна система при завантаженні одержує поточний час із BIOS і надалі також веде самостійний відлік часу. Існує безліч додатків, які самостійно ведуть відлік часу або ведуть свій календар. При цьому при первісному завантаженні додаток може одержувати поточний час від операційної системи, з BIOS або безпосередньо з RTC.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи візуалізації системного та реального часу ПК.

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

– Огляд існуючих систем візуалізації системного та реального часу ПК.

– Дослідження системи візуалізації системного та реального часу ПК.

– Програмна реалізація системи візуалізації системного та реального часу ПК.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі візуалізації системного та реального часу ПК.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи візуалізації системного та реального часу ПК, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Годинник реального часу (RTC – Real Time Clock) використовується в комп'ютері для зберігання й постійного відліку поточного системного часу й дати (рисунок 1.1). Для забезпечення безперебійної роботи вони живляться від батарейки. Годинники мають власну пам'ять (CMOS) невеликого обсягу (64 байта), у якій вони використовують тільки 15 байтів, комірки що залишилися використовуються програмою BIOS Setup для зберігання конфігурації устаткування комп'ютера. Цілісність цієї інформації перевіряється за допомогою контрольної суми.

Доступ до комірок пам'яті CMOS здійснюється через порти введення-виведення 070h (індекс комірки) і 071h (читання/запис вмісту комірки).

Рисунок 1.1 – Мікросхема RTC фірми EPSON

Підсистема годинника реального часу в перших комп'ютерах виконувалася на мікросхемі контролера MC146818 фірми Motorola.

Для вхідного тактового сигналу контролера застосовується спеціальний "годинніковий" кварцовий генератор із частотою 32 768 кГц, що дозволяє за допомогою розподілу частоти одержати імпульси із частотою 1 Гц. Контролер рахує секунди, хвилини, години, дні тижні, місяці й роки. Причому працює він навіть при відключенні живлення комп'ютера, підживлюючись від батареї або

акумулятора. Це дозволяє зберігати інформацію про поточний час постійно.

Крім лічильника поточного часу, контролер має у своєму складі будильник. Будильник може формувати переривання (IRQ8) із програмно заданою періодичністю. Стан всіх лічильників (секунд, хвилин, годин і т.д.) програмно доступний як для читання, так і для запису, що дозволяє встановлювати потрібний час і стежити за ним.

Первинною функцією RTC є точний рахунок днів та часу, що досягається за допомогою чотирьох лічильників:

– лічильник на 60 секунд;
– лічильник на 60 хвилин;
– лічильник на 24 години;
– лічильник на 32768 днів.

RTC інкрементує лічильник секунд один раз у секунду; інші три лічильники інкрементуються у відповідні моменти часу. Лічильник днів інкрементується щодня опівночі (0 годин, 0 хвилин, 0 секунд).

Можлива періодична генерація переривання щосекунди, щохвилини, щогодини або щодня. Керування кожним із цих переривань може здійснюватися незалежно.

RTC забезпечує дві функції будильника, програмуємі в регістрі будильника RTC (RTC_ALARM).

Перша функція – будильник за часом дня (певній годині, хвилині й секунді). Коли переривання будильника дозволене, RTC генерує переривання щодня в заданий час.

Друга функція будильника дозволяє крім часу дня також задавати конкретний день. Коли переривання будильника по дню дозволено, RTC генерує переривання в заданий день і час.

Переривання будильника й переривання будильника по дню можуть дозволятися й заборонятися незалежно.

RTC забезпечує функцію секундоміра, що представляє собою таймер

зворотного рахунку. Кількість секунд, що підраховуються, програмується в регістрі лічильника секундоміра RTC (RTC_SWCNT). Коли переривання секундоміра дозволене, RTC генерує переривання після закінчення заданої кількості секунд.

1.2 Область застосування

Розроблене програмне забезпечення можна використовувати для навчання студентів роботі і програмуванню системного та реального часу ПК. Даний емулятор можна використовувати для виконання лабораторних робіт по програмуванню.

Якість навчання, одержаного студентом, визначається не тільки рівнем теоретичної підготовки, але й умінням використовувати отримані знання на практиці. Для підвищення рівня практичної підготовки наукових і технічних фахівців в останні роки все більше застосування знаходять інформаційні технології, зокрема, при організації лабораторних робіт.

Лабораторні стенди, що припускають фізичне моделювання технологічного процесу навіть у спрощеному вигляді, означають серйозні фінансові витрати на їхнє створення й підтримку в працездатному стані. При такому підході необхідно розробити й реалізувати не тільки модель технологічного процесу, але також оснастити її датчиками, виконавчими органами й забезпечити сполучення з комп'ютером через модулі вводу-виводу. Вартість створення такого стенда часто дуже висока. Стенди громіздкі, займають багато місця, а помилки, що допускаються студентами при роботі з ними, найчастіше призводять до виходу стенда з ладу. У цих умовах найбільш перспективний спосіб організації практичних занять для студентів повинен бути заснований на використанні програмних імітаторів – віртуальних лабораторних стендів. У порівнянні з фізичними моделями, емулятори мають ряд очевидних переваг, що дають істотне скорочення матеріальних і часових витрат на

створення, тиражування й супровід лабораторних стендів.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи візуалізації системного та реального часу ПК, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Годинники реального часу – це мікросхеми з вбудованими засобами реєстрації часу, що дозволяють відраховувати секунди, хвилини, години, дні тижня, числа місяця, роки й навіть сторіччя. Деякі мікросхеми дозволяють відраховувати соті частки секунди, враховувати переходи на літній і зимовий час, мають убудовану енергонезалежну пам'ять, супервізор живлення. Повністю енергонезалежні годинники реального часу випускаються в комплекті з вбудованою літієвою батареєю живлення, що повністю виключає порушення коректного функціонування таких мікросхем.

Годинники реального часу підрозділяються на три типи:

- годинники реального часу з дуже низьким споживанням (Low power Real-time clocks);
- годинники реального часу промислового стандарту (Industry standard Real-time clocks);
- годинники реального часу високого ступеня інтеграції (High integration Real-time clocks).

Годинники реального часу з дуже низьким споживанням

Годинники реального часу з дуже низьким споживанням і допустимим напруженням живлення в робочому режимі до 1,3 В (відлік часу триває аж до мінімальної напруги 1,0 В) представлені серією M41T6x. Мікросхеми споживають усього 350 на в режимі очікування при напрузі живлення 3 В. Низьке споживання особливо важливо в режимі очікування, тому що в цьому випадку живлення на мікросхеми годинника реального часу подається від резервної батареї.

Мікросхеми цієї серії обновляють поточну інформацію про рік, місяць, день, даті, годинниках, хвилинах, секундах, десятих і сотих частках секунди й навіть про сторіччя. Керування й обмін інформацією з мікроконтролером відбувається по стандартному інтерфейсу I^2C з тактовою частотою 400 кГц. Автоматичні переходи на зимовий і літній час спрощують використання годинника реального часу серії M41T6x. Мініатюрний корпус QFN16 з розмірами всього 3x3 мм дозволяє вбудовувати ці мікросхеми в компактні прилади.

Годинники реального часу промислового стандарту

Серед мікросхем цього типу годинники реального часу M41T82, M41T83 і M41T93 мають аналогове й цифрове калібрування для підстроювання частоти кварцового резонатора 32,768 кГц. Аналогове калібрування здійснюється регулюванням еквівалентної навантажувальної ємності CLOAD, що проілюстровано на рисунку 2.1.

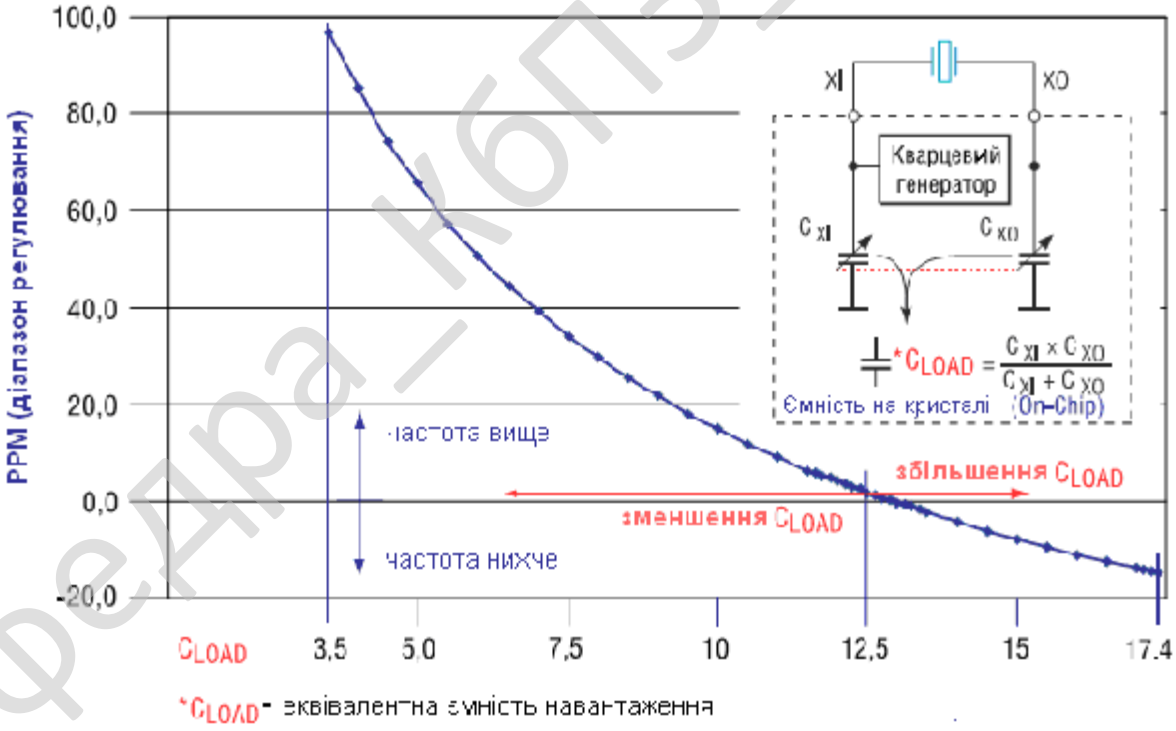

Рисунок 2.1 – Аналогове калібрування M41T82, M41T83 і M41T93, що здійснюється регулюванням сумарної навантажувальної ємності C_{load} кварцового резонатора

Завантажуючи через послідовний інтерфейс у відповідні регістри мікросхем певні значення (точні дані наведені в документації для конкретних годинників реального часу), можна домогтися мінімального відхилення від необхідної частоти 32,768 кГц. Крім того, у цих мікросхемах є можливість і цифрове калібрування або підстроювання частоти низькочастотного кварцового резонатора, що задає, 32,768 кГц по високочастотному кварцовому резонаторі мікроконтролера. Справа в тому, що стабільність частоти високочастотних кварцових резонаторів, частота яких виміряється одиницями й десятками Мгц, істотно вище, ніж аналогічний параметр низькочастотних часових кварців із частотою 32,768 кГц. Це наочно показано на рисунку 2.2, узятому з посібника із застосування AN2678 компанії STMicroelectronics.

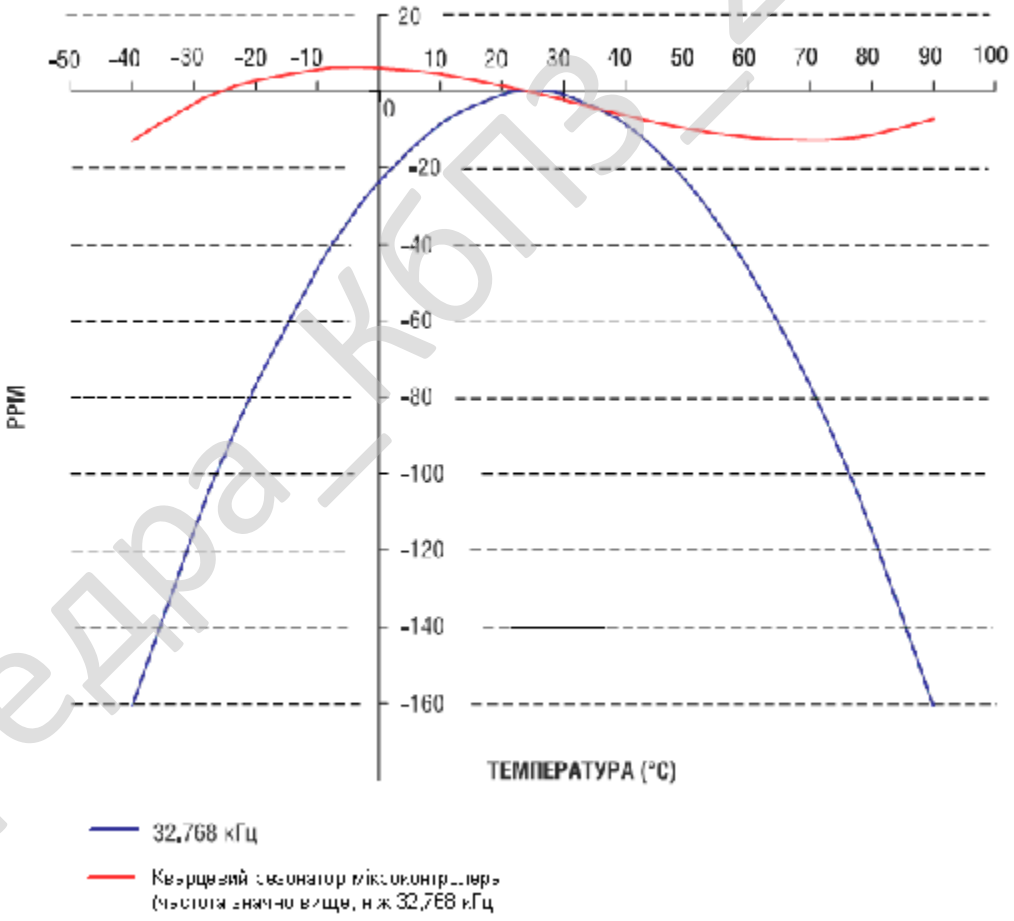

Рисунок 2.2 – Стабільність частоти низькочастотних і високочастотних кварцових резонаторів індустріального діапазону робочих температур від -40 до 85°C

З рисунку 2.2 добре видно, що діапазон зміни частоти кварцового резонатора 32,768 кГц у кілька разів більше в порівнянні з діапазоном високочастотного кварцового резонатора, що задає тактову частоту мікроконтролера. Ці властивості резонаторів дозволяють реалізувати додатково до аналогового підстроювання частоти цифрове калібрування резонатора 32,768 кГц по частоті генератора, що задає, мікроконтролера. Два калібрування, засновані на різних принципах, істотно підвищують стабільність частоти задаючого генератора годинника реального часу. Попереднє заводське калібрування мікросхем M41T82, M41T83 і M41T93 дозволяють забезпечити стабільність частоти ±5 ppm (ppm – одна мільйонна частина), що дозволяє одержати точність відліку часу з помилкою не більше 12 секунд на місяць при кімнатній температурі. Цифрове калібрування істотно підвищує точність ходу годин не тільки при кімнатній температурі, але й у всьому індустріальному діапазоні робочих температур. Мікросхеми M41T82, M41T83 і M41T93 мають убудовані схеми формування сигналів, що інформують мікроконтролер про неприпустимо низьку напругу живлення основного джерела. У цей момент відбувається автоматичне перемикання на живлення від резервної батареї.

Мікросхеми годинника реального часу мають убудовані додаткові функції. Мікросхема містить інтегровану схему керування й вибору живлення (від основного джерела або резервної батареї), формувач сигналу зупинки кварцового генератора, два формувачі сигналів оповіщення (ALARM1 і ALARM2), таймер Watchdog, вихід тестової частоти, 8 біт пам'яті з однократним програмуванням (OTP), 7 байт пам'яті SRAM. Мікросхеми випускаються в мініатюрних корпусах QFN16, SO18 або SO8.

Мікросхеми годинника реального часу з високим ступенем інтеграції

Цей тип годинників реального часу має підвищений об'єм вбудованої енергонезалежної пам'яті NVRAM, а деякі мікросхеми містять у комплекті вбудовану літієву батарею живлення (корпус SOH28), що забезпечує стабільний безперебійний відлік часу протягом декількох років.

Заслуженою популярністю серед розроблювачів користуються мікросхеми годинників реального часу M41T56C64 з напругою живлення 5 В ±10%, що мають найбільший об'єм вбудованої пам'яті (56 байт NVRAM + 8 кБ EEPROM). Убудована пам'ять EEPROM – це добре відома розроблювачам M24C64. Низьке споживання інтегрованої пам'яті цих мікросхем дозволяє зберігати інформацію при відключеному живленні більше 40 років. Убудований кварцовий резонатор, індустріальний діапазон робочих температур, убудована схема для автоматичного переходу на резервне живлення в сполученні з низькою ціною забезпечують високий успіх цих мікросхем серед широкої розмаїтості годинників реального часу. Безсумнівно, розроблювачів зацікавить нова мікросхема M41T00SC64 з аналогічними функціональними можливостями, але із широким діапазоном напруг живлення від 2,7 до 5,5 В. Це особливо актуально, тому що гнітюче число сучасних мікроконтролерів працюють при напрузі живлення 3,3 В и нижче.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Оскільки потрібно розробити просту та легку у користуванні програму, яка б виконувалась під операційною системою Windows, то для її реалізації я обрав Builder C++. Існує велике число бібліотек написаних під Builder C++, тому це одна з важливих причин вибору мови програмування. Середовище Builder C++ досить просте в користуванні, його вихідний код значно менше по об’єму в порівнянні з Delphi чи деякими іншими програмами такого типу. Досить легко організувати взаємодію між модулями програм, об’єктно-орієнтований підхід дає можливість значно скоротити код програми, а отже і час його виконання.

На заміну старого розробленого набору елементів управління у Builder C++ інтегрована бібліотека візуальних компонентів VCL, представлених на палітрі компонентів. Після переносу на форму методом перетягування (drag-and-drop) компоненти відразу становляться діючими об’єктами вашої програми. Окрім

типізованих інтерфейсних елементів Windows (кнопки, смуги прокручування, редагуємі текстові області, прості та комбіновані списки, та інше) у бібліотеку включені елементи підтримки діалогових вікон, обслуговування баз даних та багато іншого. Можливо не тільки модифікувати поведінку існуючих компонентів, але і будувати нові.

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

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

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

Builder C++ підтримує зв'язок з різними базами даних 3-х видів: dBASE і Paradox: Sybase, Oracle, InterBase і Informix; Excel, Access, FoxPro і Btrieve. Механізм BDE (Borland Database Engine) додає обслуговуванню зв'язків з базами даних дивовижну простоту і прозорість. Провідник Database Explorer дозволяє зображати зв'язки і об'єкти баз даних графічно. Використовуючи компоненти баз даних, я побудував електронний записник згідно таблиці dBASE за півгодини

роботи на комп'ютері. Спадкоємство готових форм і їх "підгонка" під специфічні вимоги помітно скорочують тимчасові витрати на вирішення подібних завдань.

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

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

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи візуалізації системного та реального часу ПК.

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

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

б) вибрати та обґрунтувати методику побудови системи 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

Опис будови та принципу роботи годинника реального часу RTC

Розглянемо принцип роботи годинника реального часу. На рисунку 3.1 показана схема взаємодії CMOS-BIOS-ОС.

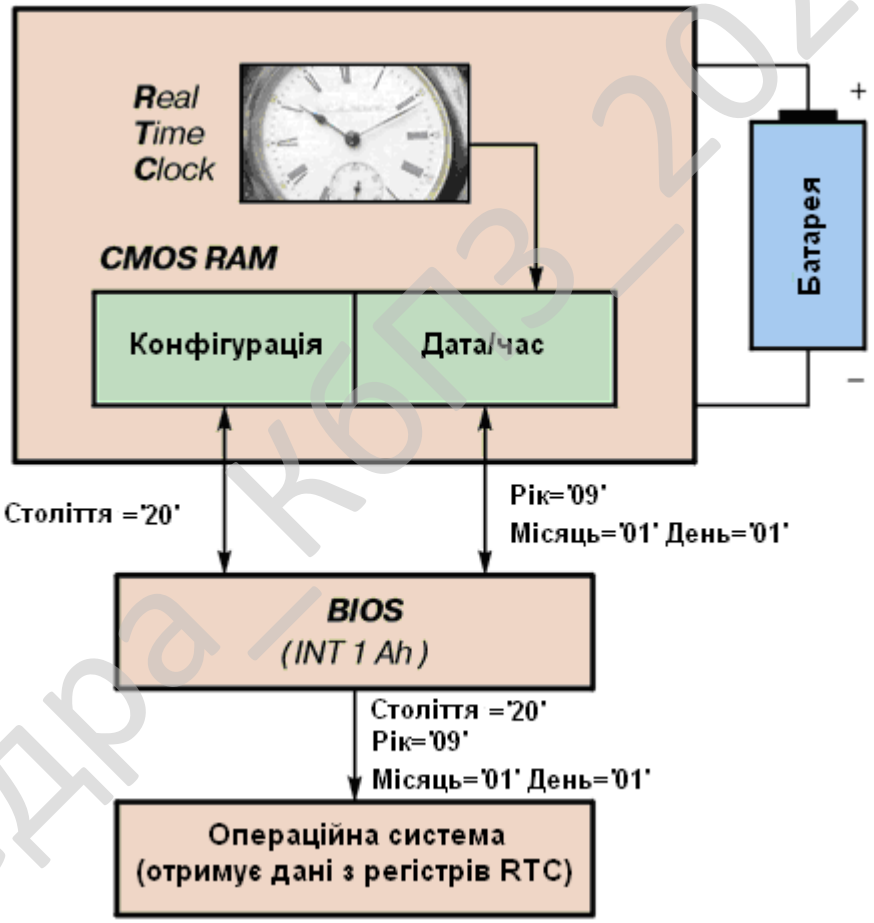

Рисунок 3.1 – Схема взаємодії CMOS – BIOS – Операційна система

Годинник живиться від акумулятора, тому його показання не пропадають при вимиканні комп'ютера. Доступ до годинників реального часу можливий або

через комырки CMOS-пам'яті, або через спеціальні функції BIOS (що краще з погляду незалежності роботи програми від особливостей апаратури).

RTC являється двійковим таймером, що зберігає інформацію про поточний час дня й формує субсекундні переривання системи. RTC складається з каскадно включених 32-розрядного й 8-бітного лічильників імпульсів, які містять кількість секунд (загальною сумою близько 136 років) і субсекунд (з дозволом 1/256 секунди) відповідно. 8– бітний субсекундний лічильник збільшується із частотою 256 Гц, сформованою з 32.768 кГц синхроімпульсів. Окремий субсекундний лічильник аварійної сигналізації з перезавантаженням може використовуватися для формування запитів на переривання з періодом до 1/256 секунди. 32-розрядний лічильник секунд збільшується при кожному переповненні 8-бітного субсекундного лічильника. 32-розрядний лічильник може використовуватися в якості програмувального одноденного аварійного таймера подій. Для початкового запису даних у регістр RTC повинен бути зупинений, але, будучи один раз запущеними, RTC працює увесь час, поки вони активні й не потрібно їхньої зупинки для читання регістра лічильника. Також RTC підтримують цифрове налаштування ходу, що дозволяє використовувати їх у тих додатках, у яких потрібна висока точність.

Функція цифрового налаштування ходу RTC

Точність ходу некоректуємого годинника реального часу залежить від кварцового резонатора (і температурного дрейфу його характеристик). Щоб забезпечити вимоги деяких систем, годинники реального часу містять модуль цифрового налаштування ходу (рисунок 3.2).

Модуль цифрового налаштування ходу може додавати або виключати 256 Гц синхроімпульси. П'ять бітів налаштування (TRM4:0) використовуються для визначення величини цифрового налаштування ходу, а знаковий біт (TSGN) визначає, чи належні імпульси бути додані (TSGN = 0) або вилучені (TSGN = 1) із синхросигнала. Кожні 16 секунд п'ять бітів налаштування додаються до попереднього значення. Щоразу, коли формується імпульс додавання фази

відбувається підміна 256 Гц імпульсів на 512 Гц імпульси, додаючи 128 додаткових зовнішніх 32 кГц імпульсів. Якщо обрано негативне підстроювання, то відбувається маскування 256 Гц імпульсу, що дозволяє відняти 128 32 кГц імпульсів. Шляхом установки бітів FT і SQE можна вивести на зовнішній вивід 1 Гц або 512 Гц прямокутні імпульси. На рисунку 3.3 наведені епюри сигналів, що пояснюють принцип роботи модуля цифрового налаштування ходу.

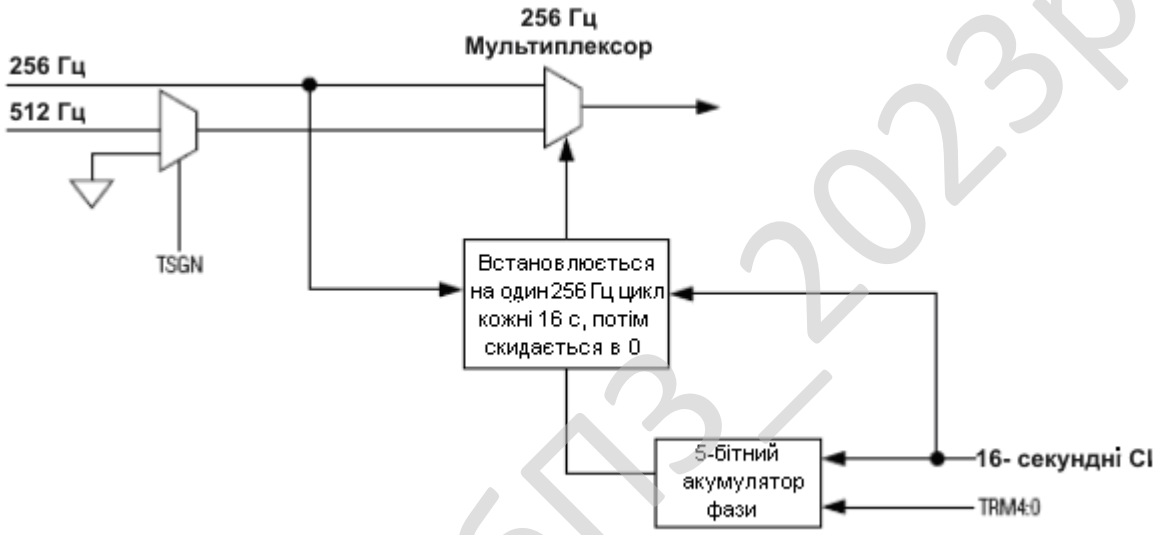

Рисунок 3.2 – Модуль цифрового налаштування ходу RTC

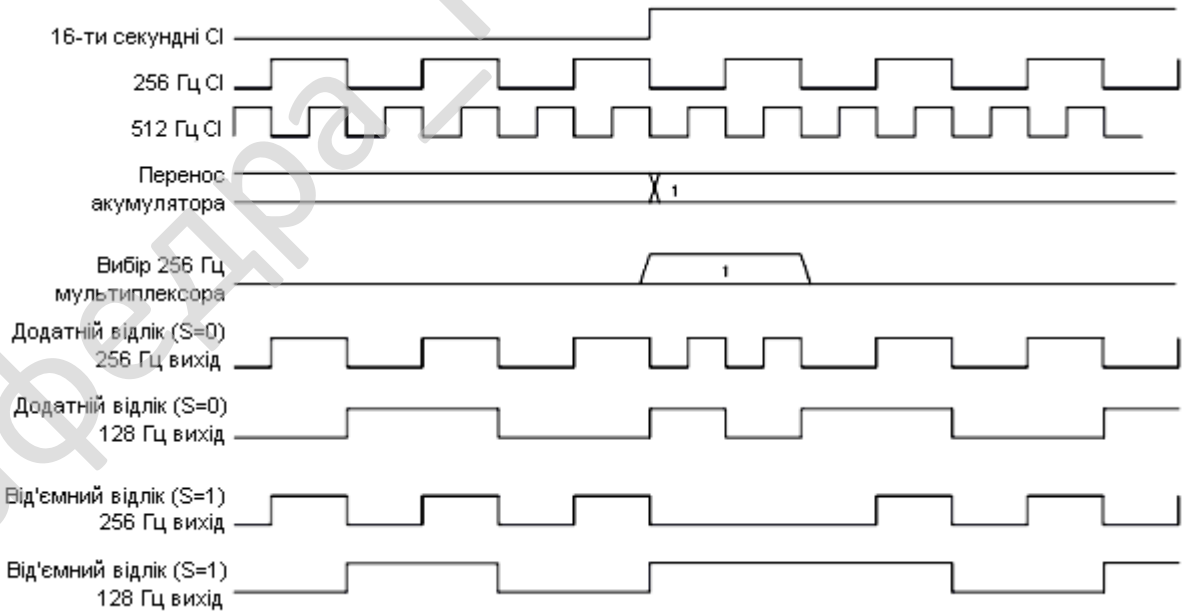

Рисунок 3.3 – Епюри сигналів, що пояснюють роботу модуля цифрового налаштування ходу RTC

Мінімальне налаштування (00001h) приведе до формування імпульсу корекції кожні 32 х 16 секунд = 512 секунд, що приведе до додавання/виключення одного імпульсу (=128 циклів 32.768 кГц) кожні 512 секунд. Це здійснить корекцію величиною 128 / (32 768 Гц х 512 сек.) = ±7.63 ppm. Максимальне значення налаштування забезпечується установкою всіх бітів в 1 (TRM4:0 = 11111b). Це приведе до додавання/виключення 31 імпульсу кожний 512-секундний інтервал, що еквівалентно корекції (31 х 128) / (32 768 Гц х 512 сек.) = ±236.5 ppm. Такий діапазон налаштування ходу дозволяє скорегувати температурний відхід характеристик більшості 32 кГц кварцових резонаторів при роботі в індустріальному температурному діапазоні.

Регістри годинника реального часу

Регістри годинника реального часу містяться у CMOS-пам'яті в діапазоні адрес 00h-0Dh. Їх адреси та призначення показані рисунку 3.4.

Адреса	Призначення
00h	лічильник секунд
01h	регістр секунд будильника
02h	лічильник хвилин
03h	регістр хвилин будильника
04h	лічильник годин
05h	регістр годин будильника
06h	лічильник днів тижня (1 – неділя)
07h	лічильник днів місяця
08h	лічильник місяців
09h	лічильник років (останні 2 цифри року)
0Ah	Регистр статуса RTC A
0Bh	Регистр статуса RTC B
0Ch	Регистр статуса RTC C
0Dh	Регистр статуса RTC D
0Eh…3Fh	Інші регістри CMOS-пам'яті

Рисунок 3.4 – Регістри годинника реального часу

Всі дані RTC зберігаються у двійково-десятковому форматі (BCD). Байти будильника 1h, 3h, 5h служать для створення часу виробітку сигналу переривання від RTC. Кожний байт може містити конкретне значення часу (секунди й хвилини в межах 0 – 59, а години в межах 0 – 23) й "неважливе" значення – код у межах C0h-FFh, тобто два старших розряди містять 1.

При конкретному завданні часу переривання виробляється раз у добу (2 рази при 12-ти годинах). Якщо байт годин містить "неважливий" код, переривання щогодини; якщо байт годин і хвилин "неважливі", переривання щохвилини; і при всіх " неважливих" байтах – щосекунди.

Розглянемо регістри стану годинника реального часу A, B, C, D.

Регістр А стану RTC. Формат регістра показаний на рисунку 3.5.

7	6	5	4	3	2	1	0
UIP	DV2	DV1	DV0	RS3	RS2	RS1	RS0

Рисунок 3.5 – Формат регістра А стану RTC

RS0-RS3 – швидкість відліку (дорівнює 0110).

DV0-DV2 – 22-розрядний дільник (дорівнює 010).

UIP – прапор відновлення (0 – можна читати дані RTC, 1 – іде коректування).

Регістр В стану RTC. Формат регістра показаний на рисунку 3.6.

7	6	5	4	3	2	1	0
SET	PIE	A1E	UIE	SQWE	DM	24/12	DSE

Рисунок 3.6 – Формат регістра В стану RTC

DSE – дозвіл літнього часу; 1 -дозволено, 0 – заборонено (дорівнює 0).

24/12 – 12– або 24-годинний час (1 – 24-годинний, 0 – 12-годинний).

DM – формат даних: 0 – BCD, 1 – двійковий (дорівнює 0).

SQWE – дозволити прямокутний імпульс (дорівнює 0).

UIE – дозволити переривання по кінці відновлення – щосекунди (дорівнює 0).

A1E – дозволити сигнальне переривання (дорівнює 0).

PIE – дозволити періодичні переривання із частотою RS3-RS0 регістри A;

SET – заборона коректування; 1 – немає корекції, 0 – є корекція (дорівнює 0).

Регістр C стану RTC – біти стану переривань, доступні тільки для читання. При читанні з регістра всі розряди скидаються. Формат регістра показаний на рисунку 3.7.

7	6	5	4	3	2	1	0
IRQF	PF	AF	UF	0	0	0	0

Рисунок 3.7 – Формат регістра C стану RTC

Біти 0-3 – зарезервовані. Біти 4-6 – причини запиту.

PF – періодичне переривання.

AF – сигнальне переривання (поточний час збігся із часом будильника).

UF – переривання по кінці коректування.

IRQF – запит на переривання.

Регістр D стану RTC. Біт 7=1, якщо CMOS одержує живлення; 0=немає живлення від автономного джерела.

Програмування годинника реального часу

Годинник реального часу виробляє апаратне переривання IRQ8, якому відповідає переривання з номером 70h. Це переривання може вироблятися по

трьох причинах:

– Переривання по закінченню зміни даних. Виробляється при встановленому в 1 біті 4 регістру стану B після кожного відновлення регістрів годин.

– Переривання будильника виробляється при збігу регістрів годин і регістрів будильника й при встановленому в 1 біті 5 регістра станів B.

– Періодичне переривання виробляється з інтервалом приблизно 1 мілісекунда при встановленому в 1 біті 6 регістра станів B.

При спрацьовуванні будильника BIOS виробляє переривання INT 4Ah. Програма може підготувати власний оброблювач для цього переривання.

Для роботи з годинником реального часу можна безпосередньо звертатися до комірок CMOS-пам'яті, використовуючи порти 70h і 71h. Однак найкраще користатися функціями 2 – 7 переривання 1Ah, описаними нижче.

Прочитати показання годинника реального часу

На вході: AH = 02h.

На виході: CH = години в BCD-форматі (наприклад, 13h означає 13 годин);

CL = хвилини в BCD-форматі;

DH = секунди в BCD-форматі;

CF = CY = 1, якщо годинник реального часу не встановлений.

Установити годинник реального часу

На вході: AH = 03h;

CH = години в BCD-формат;

CL = хвилини в BCD-форматі;

DH = секунди в BCD-форматі;

DL = 1, якщо необхідно використовувати літній час.

На виході: не використовуються.

Прочитати дату з годинника реального часу

На вході: AH = 04h.

На виході: CH = сторіччя в BCD-форматі ;

CL = рік в BCD-форматі (наприклад, CX=1991h – 1991 рік);

DH = місяць в BCD-форматі;

DL = число в BCD-форматі;

CF = CY = 1, якщо годинник реального часу не встановлений.

Установити дату в годиннику реального часу

На вході: AH = 05h;

CH = сторіччя в BCD-форматі ;

CL = рік в BCD-форматі;

DH = місяць в BCD-форматі;

DL = число в BCD-форматі;

На виході: не використовуються.

Установити будильник

На вході: AH = 06h;

CH = години в BCD-форматі;

CL = хвилини в BCD-форматі;

DH = секунди в BCD-форматі.

На виході: CF = CY = 1, якщо годинник реального часу не встановлені.

Ця функція дозволяє встановити будильник на заданий час. Коли будильник "задзвенить", буде викликане переривання INT 4Ah (це переривання викликають модулі BIOS після приходу апаратного переривання від годин реального часу IRQ8, тобто переривання з номером 70h). Програма, що використовує функцію будильника, повинна підготувати оброблювач переривання INT 4Ah, що завершує свою роботу виконанням команди IRET.

Програма може встановити тільки один будильник.

Скидання будильника

На вході: AH = 07h.

На виході: не використовуються.

Ця функція дозволяє скинути будильник, наприклад, для того щоб установити його заново на інший час.

Використання годинника реального часу

Можна використовувати годинник реального часу для рішення двох завдань. По-перше, годинники дозволяють визначити поточну дату й час із точністю до секунди. По-друге, будильник можна використовувати для виконання яких-небудь дій у заданий час або періодично.

Тому що встановлений час спрацьовування будильника зберігається в CMOS -пам'яті, що харчується від акумулятора, будильник не буде скинутий при випадковому вимиканні комп'ютера.

Для роботи з годинниками реального часу можна використовувати наступну функцію:

```
/**
*.Name          timer
*.Title         Робота з годинником реального часу
*.Descr  Ця функція призначена для обслуговування
*        системного годинника реального часу через
*        переривання INT 1Ah.
*.Proto         int timer(char fn, SYSTIMER *tm)
*.Params        char     fn – виконувана функція:
*  RTC_GET_TIME     – прочитати показання годинника;
*  RTC_SET_TIME      – установити годинник;
*  RTC_GET_DATE      – прочитати дату;
*  RTC_SET_DATE      – установити дату;
*  RTC_SET_ALARM     – установити будильник;
*  RTC_CLEAR_ALARM   – скинути будильник.
*  Всі ці константи описані у файлі sysp.h
*  SYSTIMER tm – структура, що містить дані
*                для установки годинника або
*                показання годин:
*               typedef struct _SYSTIMER_ {
*                  char hour;     // години
*                  char min;      // хвилини
*                  char sec;      // секунди
*                  unsigned year; // рік
*                  char month;    // місяць
*                  char day;      // число
```

```
*                  char daylight_savings; // прапор
*                          // використання
*                         // літнього часу
*                         // (для включення режиму
*                         // повинен дорівнювати 1)
*              } SYSTIMER;
*.Return       0   - успішне виконання функції;
*              -1  - годинник реального часу відсутній
*                    у комп'ютері;
*.Sample       setalarm.c
**/

#include <stdio.h>
#include <dos.h>
#include "sysp.h"

union REGS reg;

int timer(char fn, SYSTIMER *tm) {

        reg.h.ah = fn;
        switch (fn) {
          case RTC_SET_TIME:
                 reg.h.ch = tm->hour;
                 reg.h.cl = tm->min;
                 reg.h.dh = tm->sec;
                 reg.h.dl = tm->daylight_savings;
            break;
            case RTC_SET_DATE:
                 reg.x.cx = tm->year;
                 reg.h.dh = tm->month;
                 reg.h.dl = tm->day;
            break;
            case RTC_SET_ALARM:
                 reg.h.ch = tm->hour;
                 reg.h.cl = tm->min;
                 reg.h.dh = tm->sec;
            break;
        }
        int86(0x1a,&reg,&reg);
        if(reg.x.cflag == 1) return(-1);
        switch (fn) {
```

```
                case RTC_GET_TIME:
                    tm->hour = reg.h.ch;
                    tm->min = reg.h.cl;
                    tm->sec = reg.h.dh;
                break;
                case RTC_GET_DATE:
                     tm->year = reg.x.cx;
                     tm->month = reg.h.dh;
                     tm->day = reg.h.dl;
                 break;
            }
            return(0);
}
```

Ця функція виконує всі види обслуговування годинника реального часу, які підтримуються BIOS.

Для ілюстрації основних прийомів роботи з годинником ми підготували програму, що виводить на екран поточну дату й час. Потім установлюється будильник. Він повинен спрацювати через одну хвилину й подати звуковий сигнал.

Перед установкою будильника програма підключає свій оброблювач переривання 4Ah. Це переривання викликається при спрацьовуванні будильника. Перед завершенням роботи програма скидає будильник і відновлює вектор переривання 4Ah.

```
#include <stdio.h>
#include <stdlib.h>
#include <dos.h>
#include "sysp.h"

// Виключаємо перевірку стека й покажчиків
#pragma check_stack( off )
#pragma check_pointer( off )
// Макрос для видачі звукового сигналу
#define BEEP() _asm { \
        _asm mov bx,0 \
        _asm mov ax, 0E07h \
        _asm int 10h \
}
void main(void);
```

```
// Опис програми-оброблювача переривання
// будильника
void _interrupt _far alarm(void);
// Змінна для зберігання старого
// вектора будильника
void (_interrupt _far *old_4a)(void);
void main(void) {
char *month_to_text[] = {
        "січень",
        "лютий",
        "березень",
        "квітень",
        "травень",
        "червень",
        "липень",
        "серпень",
        "вересень",
        "жовтень",
        "листопад",
        "грудень"
};
SYSTIMER tmr;
// Визначаємо поточну дату й час
        timer(RTC_GET_DATE, &tmr);
        timer(RTC_GET_TIME, &tmr);
// Виводимо дату й час на екран
        printf("\nзараз %d рік, %s, %d число."
                "\n",
             bcd2bin(&(tmr.year)),
             month_to_text[bcd1bin(&(tmr.month)) - 1],
                bcd1bin(&(tmr.day)));
        printf("\nчас - %02.2d:%02.2d:%02.2d""\n",
              bcd1bin(&(tmr.hour)),
              bcd1bin(&(tmr.min)),
              bcd1bin(&(tmr.sec)));
// Для установки будильника збільшуємо
// лічильник хвилин на одиницю. Для спрощення
// програми ми не перевіряємо лічильник на
// переповнення, тому якщо поточне
// значення лічильника хвилин дорівнює 59,
// будильник не спрацює.
        bin1bcd(bcd1bin(&(tmr.min)) + 1, &(tmr.min));
```

```
        // Виводимо на екран час, коли спрацює
        // будильник.
                printf("\nчас              спрацьовування              будильника-",
 %02.2d:%02.2d:%02.2d""\n",
                        bcd1bin(&(tmr.hour)),
                        bcd1bin(&(tmr.min)),
                        bcd1bin(&(tmr.sec)));
        // Підключаємо свій оброблювач переривання
        // будильника, старе значення вектора
        // 0x4a зберігаємо
                old_4a = _dos_getvect(0x4a);
                _dos_setvect(0x4a, alarm);
        // Установлюємо будильник
                timer(RTC_SET_ALARM, &tmr);
                printf("\nбудильник  установлений.  Для  скасування  і  завершення
програми натисніть\nбудь-яку клавішу...");
        getch();
        // Скидаємо будильник і відновлюємо
        // вектор переривання будильника
                timer(RTC_CLEAR_ALARM, &tmr);
                _dos_setvect(0x4a, old_4a);
                exit(0);
        }
        // Перетворення однобайтового числа з формату BCD
        //у двійковий формат.
        int bcd1bin(char *bcd) {
                return( ((*bcd) & 0x0f) +
                        10 * (((*bcd) & 0xf0) >> 4) );
        }
        // Перетворення двобаштового числа з формату BCD
        //у двійковий формат.
        int bcd2bin(char *bcd) {
                return( bcd1bin(bcd) +
                                100 * bcd1bin(bcd + 1) );
        }
        // Перетворення однобайтового
        // числа із двійкового формату
        // формат BCD.
        int bin1bcd(int bin, char *bcd) {
                int i;
                i = bin / 10;
                *bcd = (i << 4) + (bin - (i * 10));
```

```
}
// Програма одержує керування
// при спрацьовуванні будильника.
// Її призначення – видати звуковий сигнал.
void _interrupt _far alarm(void) {
        BEEP();
        BEEP();
        BEEP();
        BEEP();
        BEEP();
        BEEP();
        BEEP();
}
```

3.2 Розробка структурної схеми

До складу IBM PC входять годинник реального часу Real Time Clock (RTC) і 64 байта CMOS-пам'ять, що живляться від автономного джерела живлення. При вмиканні ПК вміст CMOS аналізується POST, що витягає з неї конфігурацію системи й поточну дату й час. Годинник реального часу RTC і CMOS-пам'ять спочатку виконувалися на базі мікросхеми MC146818 фірми Motorola, а в сучасних комп'ютерах реалізуються чипсетом. Спрощена структурна схема RTC представлена на рисунку 3.8.

Призначення сигналів наступне:

– CE – дозвіл кристала (вихід дешифратора адреси);

– DS – строб даних;

– AS – адресний строб;

– R/W – читання/запис;

– SQW – вихідні прямокутні імпульси – меандр (в IBM PC не використовуються);

– IRQ – запит на переривання від RTC;

– PS – сигнал стану живлення (використовується для контролю вірогідності даних);

– CKOUT – синхросигнал, що може використовуватися як вхідний синхросигнал інших пристроїв.

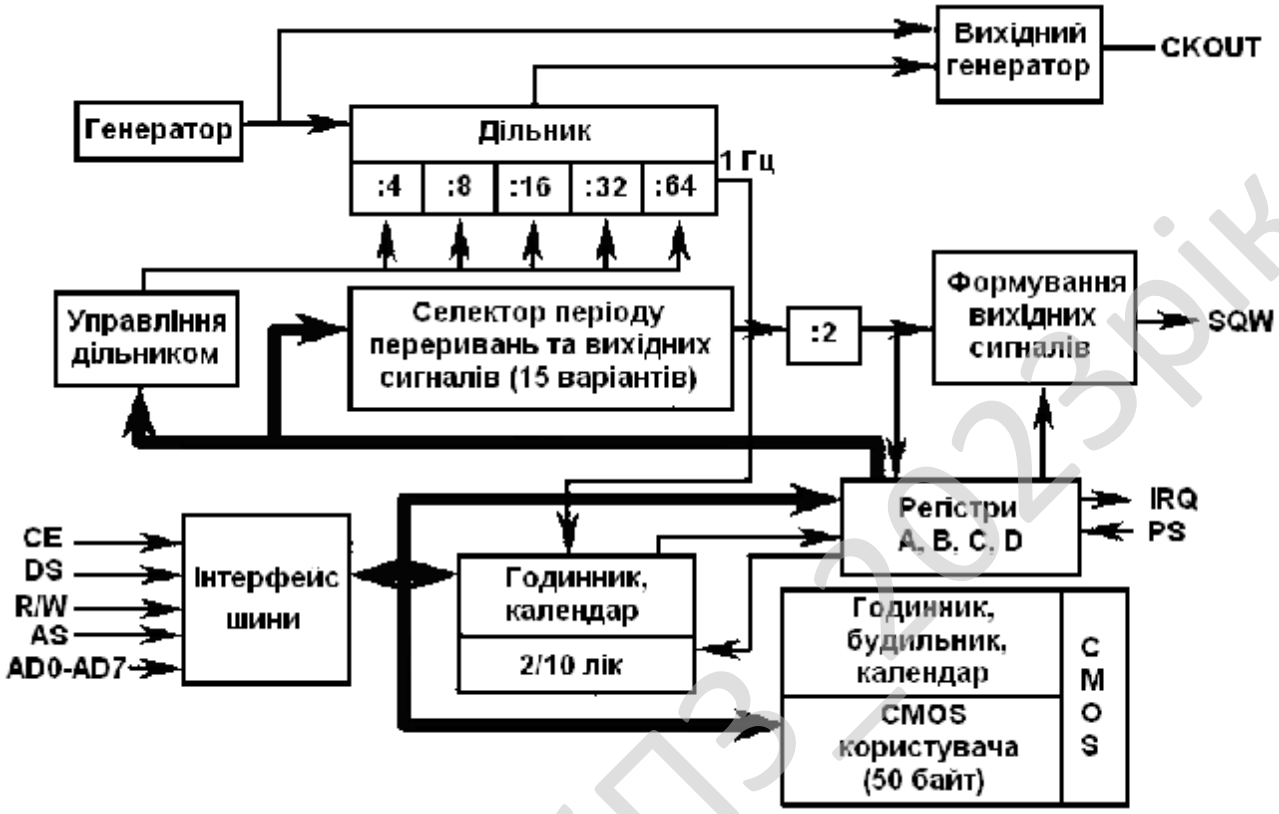

Рисунок 3.8 – Структурна схема RTC і CMOS пам'яті

Основна частина RTC – задаючий генератор (внутрішній або зовнішній) із частотою 4.194304 Мгц, 1.048576 Мгц і дільник частоти, у якого вихід останнього каскаду (1 Гц) управляє годинником.

Для доступу до даних CMOS використовуються порти 70h і 71h, причому адреса регістра подається в 70h порт, а дані читаються/пишуться через 71h порт.

Адреси CMOS з 10h по 20h захищені контрольною сумою, що зберігається по адресах 2Eh-2Fh. Тому зміни вмісту цих адрес необхідно супроводжувати перерахуванням і зміною контрольної суми.

Порт 70h застосовується не тільки для завдання адреси CMOS, але й для дозволу або заборони NMI (немаскуємого переривання). Якщо біт 7 дорівнює 0, то NMI дозволяється, якщо 1 – забороняється.

3.3 Розробка функціональної схеми

На рисунку 3.9 показана функціональна схема розробленої системи.

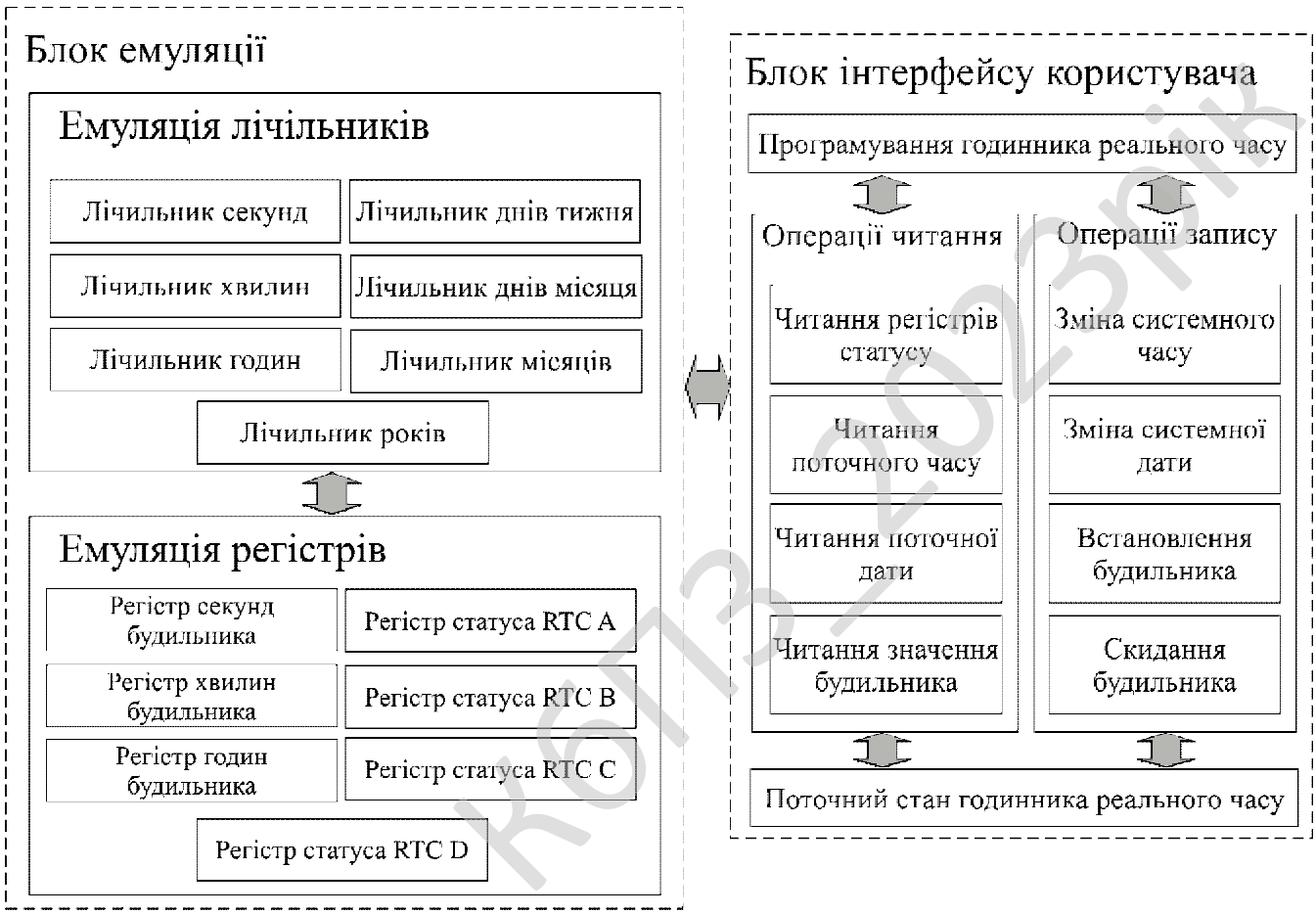

Рисунок 3.9 – Функціональна схема системи

З рисунку видно, що розроблена система складається з блоку емуляції та блоку інтерейсу користувача.

Блок емуляції в свою чергу складається з емуляції лічильників та емуляції регістрів RTC.

Емуляція лічильників складається з таких елементів:

– Лічильник секунд.

– Лічильник хвилин.

– Лічильник годин.

– Лічильник днів тижня.
– Лічильник днів місяця.
– Лічильник місяців.
– Лічильник років.

Емуляція регістрів складається з наступних складових:

– Регістр секунд будильника.
– Регістр хвилин будильника.
– Регістр годин будильника.
– Регістр статуса RTC A.
– Регістр статуса RTC B.
– Регістр статуса RTC C.
– Регістр статуса RTC D.

Інтерфейс користувача дозволяє переглядати поточний стан RTC та здійснювати його системне програмування.

Програмування годинника реального часу полягає у здійсненні операцій читання та запису регістрів та лічильників.

Операції читання включають у себе: читання регістрів статусу, читання поточного часу, читання поточної дати та читання значення будильника.

Операції запису включають у себе: зміну системного часу, зміну системної дати, встановлення будильника та скидання будильника.

3.4 Розробка діаграми процесів

Розглянемо діаграму процесів системи, зображену на рисунку 3.10. Після запуску системи запускається процес ініціалізації лічільників та регістрів. Потім відбувається запуск відліку імпульсів, що формують системні дату та час. Далі на екран виводиться поточний стан RTC. Після чого користувач може замустити операції читання чи запису.

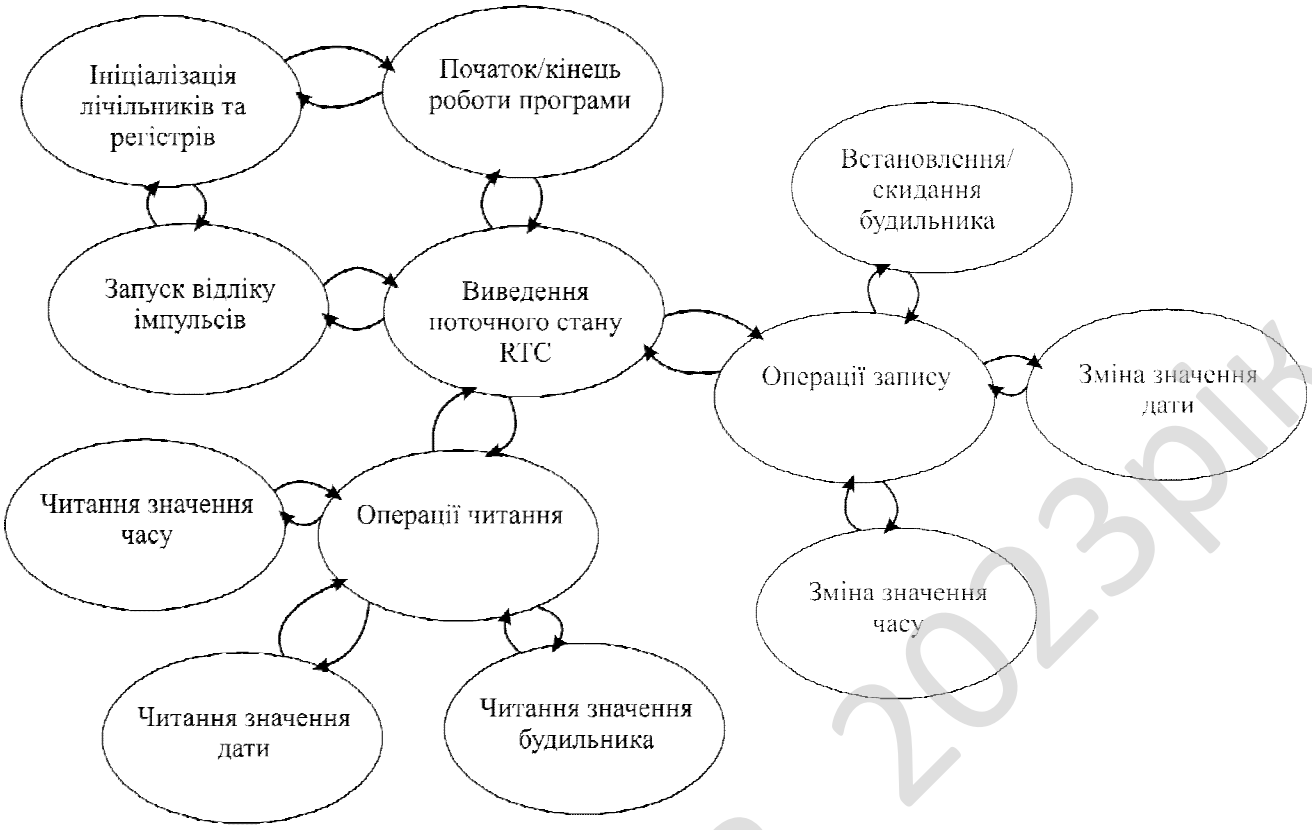

Рисунок 3.10 – Діаграма процесів системи

Операції читання складаються з читання значення часу, дати та будильника.

Операції запису складаються зі зміни значення часу і дати та встановлення/скидання будильника.

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Розглянемо алгоритм роботи основної програми. Блок-схема її роботи зображена на рисунку 4.1.

Як видно з рисунку, спочатку на екран виводиться головне вікно програми, потім відбувається ініціалізація регістрів та лічільників RTC та вивід поточного стану RTC. Початкові значення змінних:

```
int s=0; //лічільник секунд
int m=0; //лічільник хвилин
int h=0; //лічільник годин
int dn=1; //лічільник днів тижня
int dm=1; //лічільник днів місяця
int n=1; //лічільник місяців
int r=2009; //лічільник років
bool _a0=false; //біт 0 регістру A
bool _a1=true; //біт 1 регістру A
bool _a2=true; //біт 2 регістру A
bool _a3=false; //біт 3 регістру A
bool _a4=false; //біт 4 регістру A
bool _a5=true; //біт 5 регістру A
bool _a6=false; //біт 6 регістру A
bool _a7=false; //біт 7 регістру A
bool _b0=false; //біт 0 регістру A
bool _b1=true; //біт 1 регістру B
bool _b2=true; //біт 2 регістру B
bool _b3=false; //біт 3 регістру B
bool _b4=false; //біт 4 регістру B
bool _b5=false; //біт 5 регістру B
bool _b6=false; //біт 6 регістру B
bool _b7=false; //біт 7 регістру B
bool _c4=false; //біт 4 регістру C
bool _c5=false; //біт 5 регістру C
```

```
bool _c6=false; //біт 6 регістру C
bool _c7=false; //біт 7 регістру C
bool _d7=false; //біт 7 регістру D
```

Підпрограма ініціалізації виглядає наступним чином:

```
void __fastcall TForm1::FormCreate(TObject *Sender)
{
r1->Text=s;
r2->Text=m;
r3->Text=h;
r4->Text=dn;
r5->Text=dm;
r6->Text=n;
r7->Text=r;
r8->Text=s;
r9->Text=m;
r10->Text=h;
if(_b0==true) r11->Text="Дозволено";  else r11->Text="Заборонено";
if(_b1==true) r12->Text="24 години";  else r12->Text="12 години";
if(_b2==true) r13->Text="Двійковий";  else r13->Text="BCD";
a0->Checked=_a0;
a1->Checked=_a1;
a2->Checked=_a2;
a3->Checked=_a3;
a4->Checked=_a4;
a5->Checked=_a5;
a6->Checked=_a6;
a7->Checked=_a7;
b0->Checked=_b0;
b1->Checked=_b1;
b2->Checked=_b2;
b3->Checked=_b3;
b4->Checked=_b4;
b5->Checked=_b5;
b6->Checked=_b6;
b7->Checked=_b7;
c4->Checked=_c4;
c5->Checked=_c5;
c6->Checked=_c6;
c7->Checked=_c7;
d7->Checked=_d7;
}
```

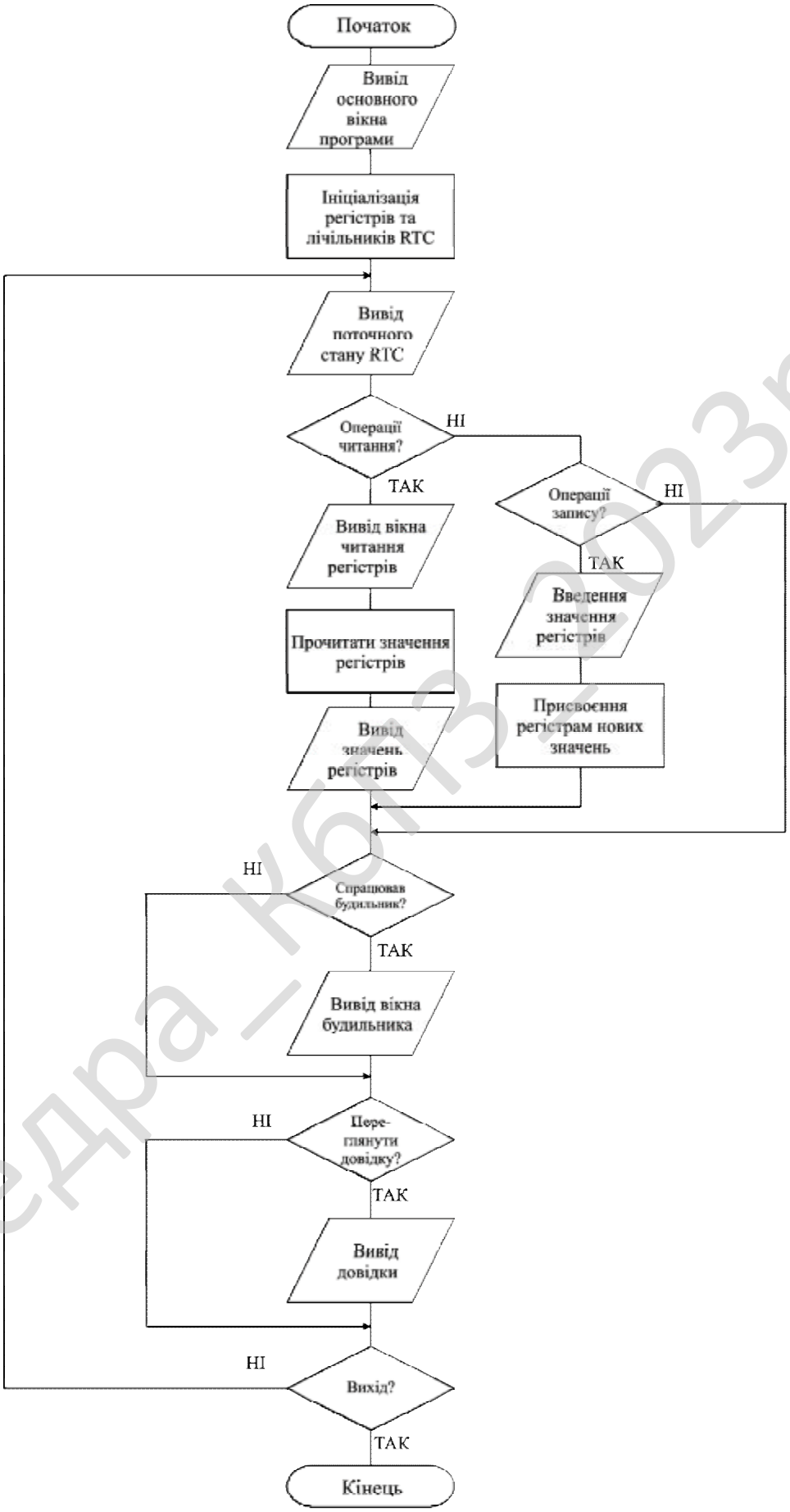

Рисунок 4.1 – Блок-схема роботи основної програми

Якщо користувач обирає операцію читання, то відкривається діалогове вікно читання регістрів та виводяться їх поточні значення. Якщо ж обрана операція запису, по з'являється діалогове вікно запису з полями, в які користувач ввобить нові значення регістрів.

Якщо користувачем було встановлено будильник і його час співпав з поточним, то на екран виводиться вікно будильника зі звуковим супроводом.

Також користувач може переглянути довідку про формат регістрів та лічильники RTC.

На рисунку 4.2 наведена блок-схема роботи підпрограми відліку імпульсів та обчислення поточної дати і часу.

Як видно з блок-схеми, поки користувач не вийде з програми, кожну секунду відбувається генерація імпульсу. При кожному імпульсі відбувається інкрементація лічильника секунд. Наступні лічильники інкрементуються після спрацьовування перевірок на настання відповідного часу. Так лічильник хвилин інкрементується коли настає 60 секунд, після чого лічильник секунд обнуляється. По такому ж принципу інкрементуються й інші лічильники.

В даній підпрограмі також здійснюється перевірка на високосний рік та на перехід на літній чи зимовий час.

У Юліанському календарі (за старим стилем) високосний рік - це той, котрий ділиться без залишку на чотири.

В Григоріанському календарі (за новим стилем) для усунення невідповідності календарного й сонячного (астрономічного) року 100-й рік не вважається високосним, але кожний 400-й - вважається.

Таким чином, алгоритм визначення високосного року виглядає в такий спосіб:

```
if((r%4==0)&(r%100!=0)|(r%400==0)) //перевірка на високосний рік
{
if((dm==29)&(n==2)){dm=1; n++; }
}
else
{
```

```
if((dm==28)&(n==2)){dm=1; n++; }
}
```

Перевірка на літній/зимовий час:

```
if((dm==29)&(n==3)) _b0=true; //перехід на літній час
if((dm==26)&(n==7)) _b0=true; //перехід на зимовий час
```

В цілому підпрограма відліку імпульсів та обчислення поточної дати і часу виглядає наступним чином:

```
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
r1->Text=s; // вивід поточного часу
r2->Text=m;
r3->Text=h;
if(dn==2) r4->Text="Понеділок"; //вивід поточного дня тижня
if(dn==3) r4->Text="Вівторок";
if(dn==4) r4->Text="Середа";
if(dn==5) r4->Text="Четвер";
if(dn==6) r4->Text="П""ятниця";
if(dn==7) r4->Text="Субота";
if(dn==1) r4->Text="Неділя";
r5->Text=dm; //вивід поточного дня місяця
if(n==1) r6->Text="Січень"; //вивід поточного місяця
if(n==2) r6->Text="Лютий";
if(n==3) r6->Text="Березень";
if(n==11) r6->Text="Квітень";
if(n==12) r6->Text="Травень";
if(n==4) r6->Text="Червень";
if(n==5) r6->Text="Липень";
if(n==6) r6->Text="Серпень";
if(n==7) r6->Text="Вересень";
if(n==8) r6->Text="Жовтень";
if(n==9) r6->Text="Листопад";
if(n==10) r6->Text="Грудень";
r7->Text=r; //вивід поточного року
if(s==60){m++; s=0;} else s++; //обчислення секунд
if(m==10){m=0; h++;} //обчислення хвилин
if(h==24){h=0; dn++; dm++;}//обчислення годин
if(dn==7) dn=0; //обчислення днів тижня
if((dm==31)&(n==1)){dm=1; n++; } //обчислення днів місяця
if((r%4==0)&(r%100!=0)|(r%400==0)) //перевірка на високосний рік
{
```

```
if((dm==29)&(n==2)){dm=1; n++; }
}
else
{
if((dm==28)&(n==2)){dm=1; n++; }
}
if((dm==28)&(n==2)){dm=1; n++; }
if((dm==31)&(n==3)){dm=1; n++; }
if((dm==30)&(n==4)){dm=1; n++; }
if((dm==31)&(n==5)){dm=1; n++; }
if((dm==30)&(n==6)){dm=1; n++; }
if((dm==31)&(n==7)){dm=1; n++; }
if((dm==31)&(n==8)){dm=1; n++; }
if((dm==30)&(n==9)){dm=1; n++; }
if((dm==31)&(n==10)){dm=1; n++; }
if((dm==30)&(n==11)){dm=1; n++; }
if((dm==31)&(n==12)){dm=1; n++; }
if(n==12){n=0; r++;}
if((dm==29)&(n==3)) _b0=true; //перехід на літній час
if((dm==26)&(n==7)) _b0=true; //перехід на зимовий час
}
```

На рисунку 4.3 зображена блок-схема роботи підпрограми обробки сигнального переривання. Під сигнальним перериванням мається на увазі будильник. Для того, щоб дозволити це переривання треба записати у 5 біт регістра стану B одиницю. Потім заповнити регістри будильника, які містять секунду, хвилину та годину, коли він повинен спрацювати.

Як тільки поточний час збігається зі значеннями регістрів будильника, на екран виводиться вікно будильника та програється мелодія для звернення уваги користувача.

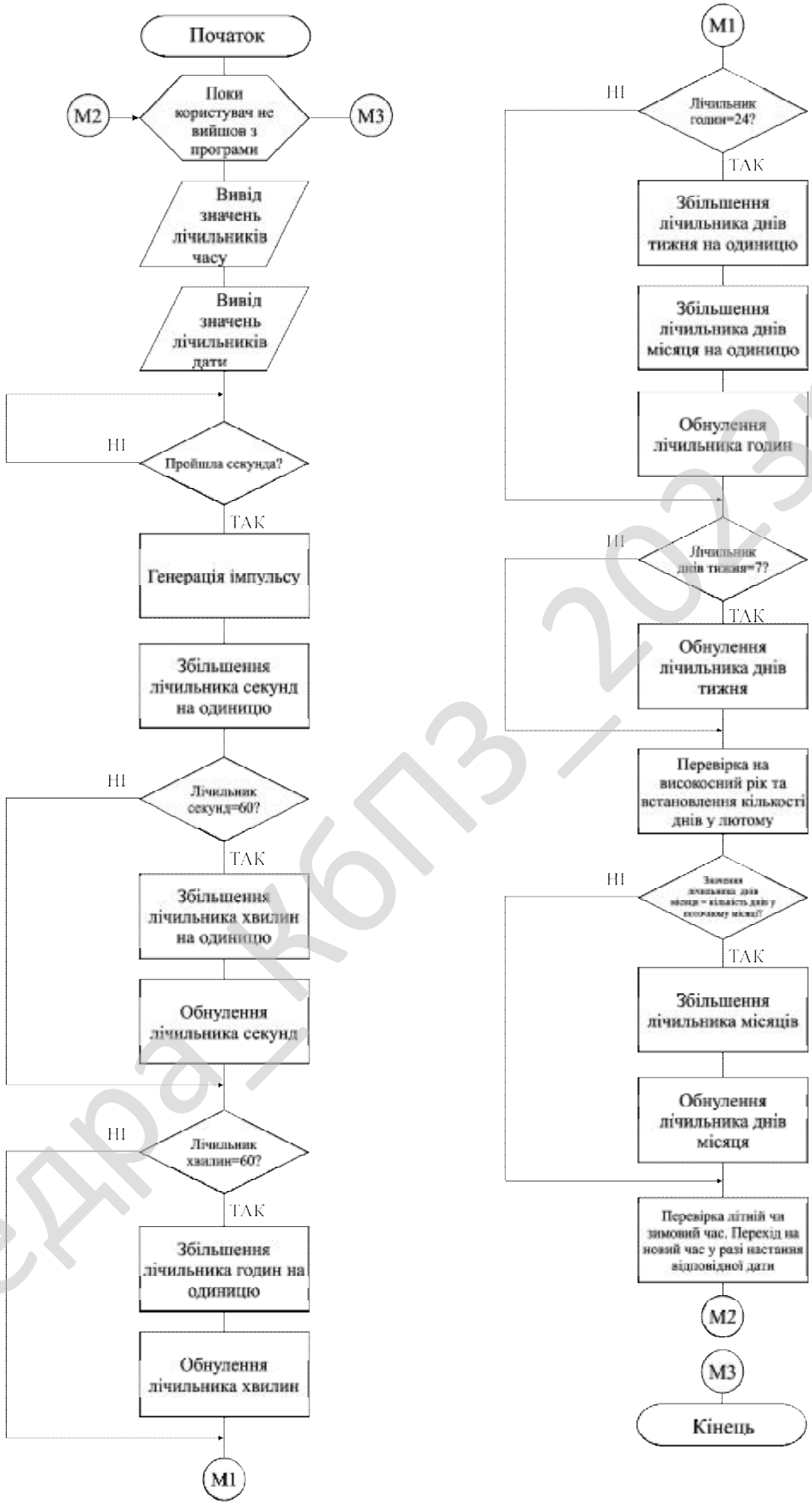

Рисунок 4.2 – Блок-схема роботи підпрограми відліку імпульсів та обчислення поточної дати і часу

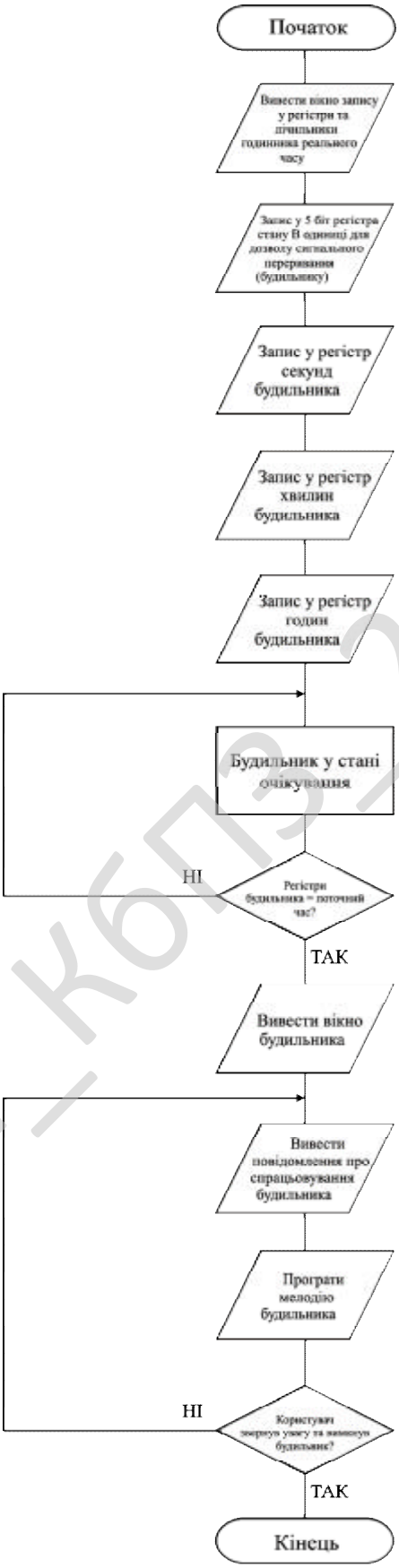

Рисунок 4.3 – Блок-схема роботи підпрограми обробки сигнального переривання

Підпрограма перевірки спрацьовування будильника:

```
void __fastcall TForm1::Timer2Timer(TObject *Sender)
{
if((s==sb)&(m==mb)&(h==hb)) Form4->Show();
}
```

Підпрограма програвання мелодії будильника:

```
void __fastcall TForm4::FormShow(TObject *Sender)
{
//програвання звукового файлу при спрацьовуванні будильника
MediaPlayer1->FileName="tada.wav";
MediaPlayer1->Open();
MediaPlayer1->Play();
}
```

4.2 Захист розробленого програмного забезпечення

Дані в програмі захищаються за допомогою використання алгоритму CAST-128 (або CAST5) у криптографії, це блоковий алгоритм симетричного шифрування на основі мережі Фейстеля, який використовується в цілому ряді продуктів криптографічного захисту, зокрема деяких версіях PGP і GPG і крім того схвалений для використання Канадським урядом.

Основні відомості

Алгоритм був створений в 1996 році Карлайлом Адамсом (Carlisle Adams) і Стаффордом Таваресом (Stafford Tavares) використовуючи метод побудови шифрів CAST, який використовується також і іншим їхнім алгоритмом CAST-256 (алгоритм-кандидат AES).

CAST-128 складається з 12 або 16 раундів мережі Фейстеля з розміром блоку 64 біта й довжиною ключа від 40 до 128 біт (але тільки з інкрементацією по 8 біт). 16 раундів використовуються коли розміри ключа перевищують 80 біт. В алгоритмі використовуються 8x16 S- блоки, засновані на бент-функції, операції XOR і модулярній арифметиці (модулярне додавання й вирахування). Є три різні типи функцій раундів, але вони схожі за структурою й різняться тільки у виборі виконуваної операції (додавання, вирахування або XOR) у різних місцях.

Хоча CAST-128 захищений патентом Entrust, його можна використовувати в усьому світі для комерційних або некомерційних цілей безкоштовно.

Опис

CAST – це популярний 64-бітовий шифр, що допускає розміри ключа аж до 128 біт

Алгоритм CAST використовує 64-бітовий блок і 64-бітовий ключ. CAST стійкий до диференціального й лінійного криптоаналізу. Сила алгоритму CAST укладена в його S-блоках. В CAST немає фіксованих S-блоків і для кожного додатка вони конструюються заново. Створений для конкретної реалізації CAST S-блок уже більше ніколи не міняється. Інакше кажучи, S-блоки залежать від реалізації, а не від ключа. Northern Telecom використовує CAST у своєму пакеті програм Entrust для комп'ютерів Macintosh, PC і робочих станцій UNIX. Обрані ними S-блоки не опубліковані, що втім не дивно.

CAST-128 належить компанії Entrust Technologies, але є безкоштовним як для комерційного, так і для некомерційного використання. CAST-256 – безкоштовне доступне розширення CAST-128, яке ухвалює розмір ключа до 256 біт і має розмір блоку 128 біт. CAST-256 був одним з первісних кандидатів на AES.

Опис алгоритму

CAST-128 заснований на мережі Фейстеля. Повний алгоритм шифрування викладений у наступних чотирьох кроках:

ВХІД: текст m_1 ... m_{64}, ключ $K = k_1 ... k_{128}$.

ВИХІД: зашифрований текст c_1 ... C_{64}.

1. (розгорнення ключа) становить 16 пару підключів $\{Km_i, Kr_i\}$ отриманих з K (див. розділи Пари раундових ключів і Неідентичні раунди).

2. (L_0, R_0) <- $(m_1 .. m_{64})$. (Розділяє текст на ліву й праву 32-бітні половини $L_0 = m_1 ... m_{32}$ і $R_0 = m_{33} ... m_{64}$).

3. (16 раундів) for i from 1 to 16, обчислити L_i і R_i у такий спосіб: $L_i = R_{i-1}$; $R_i = L_{i-1}$ ^ $F(R_{i-1},Km_i,Kr_i)$, де F визначена в розділі «Пари раундових ключів» (F має тип 1, тип 2, тип 3 або, залежно від i).

4. c_1 ... c_{64} <- (R_{16}, L_{16}). (Міняємо остаточні блоки місцями L_{16}, R_{16} і поєднуємо, щоб сформувати зашифрований текст.)

Розшифруванні збігається з алгоритмом шифрування, наведеним вище, крім того, що раунди (i, отже, пари підключів), використовуються у зворотному порядку, щоб обчислити (L_0, R_0) з (R_{16}, L_{16}).

Пари раундових ключів

CAST-128 використовує пару підключів за раунд: 32-бітні величини Km використовується в якості "маскування" ключа й Kr використовують як "перестановки" ключа, з яких використовуються тільки початкові 5-біт.

Неідентичні раунди

Три різні типів функції використовуються в CAST-128. Типи виглядає в такий спосіб (де "D" є вхідними даними у функцію F і "Ia"-"Id" є найбільш значимий байт – найменш значимий байт I, відповідно). Зверніть увагу, що "+" і "-" додавання й вирахування по модулю 2 ** 32, "" є побітове XOR і "<<<" є циклічним зрушенням уліво.

Раунди	$I = ((Km_i + R_{i-1}) <<< Kr_i)$
1,4,7,10,13,16	$F = ((S1[I_a]\ \ S2[I_b]) – (S3[I_c]))+ S4[I_d]$
Раунди	$I = ((Km_i \wedge R_{i-1}) <<< Kr_i)$
2,5,8,11,14	$F = ((S1[I_a] – S2[I_b]) + (S3[I_c]))\ \ S4[I_d]$
Раунди	$I = ((Km_i – R_{i-1}) <<< Kr_i)$
3,6,9,12,15	$F = ((S1[I_a] + S2[I_b])\ \ (S3[I_c]))- S4[I_d]$

Поля заміни

CAST-128 використовує вісім полів заміни: поля S1, S2, S3 і S4 раундові функції полів заміни, S5, S6, S7 і S8 є ключами розгорнення полів заміни. Незважаючи на те, що 8 полів заміни вимагають у цілому 8 Кбайт для зберігання, зверніть увагу на те, що тільки 4 Кбайта потрібні під час фактичного шифрування

/ дешифрування, тому що генерація підключа звичайно робиться до будь-якого введення даних.

Ключі розгорнення

Представимо 128-розрядний ключ у вигляді x0x1x2x3x4x5x6x7x8x9xaxbxcxdxexf, де x0 старший байт, і xf молодший байт.

Представимо z0..zf проміжними (тимчасовими) байтами. Si[] представляє поле заміни і і " ^ " представляє додавання по Xor'у.

Поля заміни формуються із ключа x0x1x2x3x4x5x6x7x8x9xaxbxcxdxexf у такий спосіб.

z0z1z2z3 = x0x1x2x3 ^ S5[xD] ^ S6[xF] ^ S7[xC] ^ S8[xE] ^ S7[x8]

z4z5z6z7 = x8x9xAxB ^ S5[z0] ^ S6[z2] ^ S7[z1] ^ S8[z3] ^ S8[xA]

z8z9zAzB = xCxDxExF ^ S5[z7] ^ S6[z6] ^ S7[z5] ^ S8[z4] ^ S5[x9]

zCzDzEzF = x4x5x6x7 ^ S5[zA] ^ S6[z9] ^ S7[zB] ^ S8[z8] ^ S6[xB]

K1 = S5[z8] ^ S6[z9] ^ S7[z7] ^ S8[z6] ^ S5[z2]

K2 = S5[zA] ^ S6[zB] ^ S7[z5] ^ S8[z4] ^ S6[z6]

K3 = S5[zC] ^ S6[zD] ^ S7[z3] ^ S8[z2] ^ S7[z9]

K4 = S5[zE] ^ S6[zF] ^ S7[z1] ^ S8[z0] ^ S8[zC]

x0x1x2x3 = z8z9zAzB ^ S5[z5] ^ S6[z7] ^ S7[z4] ^ S8[z6] ^ S7[z0]

x4x5x6x7 = z0z1z2z3 ^ S5[x0] ^ S6[x2] ^ S7[x1] ^ S8[x3] ^ S8[z2]

x8x9xAxB = z4z5z6z7 ^ S5[x7] ^ S6[x6] ^ S7[x5] ^ S8[x4] ^ S5[z1]

xCxDxExF = zCzDzEzF ^ S5[xA] ^ S6[x9] ^ S7[xB] ^ S8[x8] ^ S6[z3]

K5 = S5[x3] ^ S6[x2] ^ S7[xC] ^ S8[xD] ^ S5[x8]

K6 = S5[x1] ^ S6[x0] ^ S7[xE] ^ S8[xF] ^ S6[xD]

K7 = S5[x7] ^ S6[x6] ^ S7[x8] ^ S8[x9] ^ S7[x3]

K8 = S5[x5] ^ S6[x4] ^ S7[xA] ^ S8[xB] ^ S8[x7]

z0z1z2z3 = x0x1x2x3 ^ S5[xD] ^ S6[xF] ^ S7[xC] ^ S8[xE] ^ S7[x8]

z4z5z6z7 = x8x9xAxB ^ S5[z0] ^ S6[z2] ^ S7[z1] ^ S8[z3] ^ S8[xA]

z8z9zAzB = xCxDxExF ^ S5[z7] ^ S6[z6] ^ S7[z5] ^ S8[z4] ^ S5[x9]

zCzDzEzF = x4x5x6x7 ^ S5[zA] ^ S6[z9] ^ S7[zB] ^ S8[z8] ^ S6[xB]

K9 = S5[z3] ^ S6[z2] ^ S7[zC] ^ S8[zD] ^ S5[z9]

K10 = S5[z1] ^ S6[z0] ^ S7[zE] ^ S8[zF] ^ S6[zC]

K11 = S5[z7] ^ S6[z6] ^ S7[z8] ^ S8[z9] ^ S7[z2]

K12 = S5[z5] ^ S6[z4] ^ S7[zA] ^ S8[zB] ^ S8[z6]

x0x1x2x3 = z8z9zAzB ^ S5[z5] ^ S6[z7] ^ S7[z4] ^ S8[z6] ^ S7[z0]

x4x5x6x7 = z0z1z2z3 ^ S5[x0] ^ S6[x2] ^ S7[x1] ^ S8[x3] ^ S8[z2]

x8x9xAxB = z4z5z6z7 ^ S5[x7] ^ S6[x6] ^ S7[x5] ^ S8[x4] ^ S5[z1]

xCxDxExF = zCzDzEzF ^ S5[xA] ^ S6[x9] ^ S7[xB] ^ S8[x8] ^ S6[z3]

K13 = S5[x8] ^ S6[x9] ^ S7[x7] ^ S8[x6] ^ S5[x3]

K14 = S5[xA] ^ S6[xB] ^ S7[x5] ^ S8[x4] ^ S6[x7]

K15 = S5[xC] ^ S6[xD] ^ S7[x3] ^ S8[x2] ^ S7[x8]

K16 = S5[xE] ^ S6[xF] ^ S7[x1] ^ S8[x0] ^ S8[xD]

половина, що залишається, ідентична тому, що дане вище, продовження від останнього створило x0..xf, щоб генерувати ключі K17 – K32.

z0z1z2z3 = x0x1x2x3 ^ S5[xD] ^ S6[xF] ^ S7[xC] ^ S8[xE] ^ S7[x8]

z4z5z6z7 = x8x9xAxB ^ S5[z0] ^ S6[z2] ^ S7[z1] ^ S8[z3] ^ S8[xA]

z8z9zAzB = xCxDxExF ^ S5[z7] ^ S6[z6] ^ S7[z5] ^ S8[z4] ^ S5[x9]

zCzDzEzF = x4x5x6x7 ^ S5[zA] ^ S6[z9] ^ S7[zB] ^ S8[z8] ^ S6[xB]

K17 = S5[z8] ^ S6[z9] ^ S7[z7] ^ S8[z6] ^ S5[z2]

K18 = S5[zA] ^ S6[zB] ^ S7[z5] ^ S8[z4] ^ S6[z6]

K19 = S5[zC] ^ S6[zD] ^ S7[z3] ^ S8[z2] ^ S7[z9]

K20 = S5[zE] ^ S6[zF] ^ S7[z1] ^ S8[z0] ^ S8[zC]

x0x1x2x3 = z8z9zAzB ^ S5[z5] ^ S6[z7] ^ S7[z4] ^ S8[z6] ^ S7[z0]

x4x5x6x7 = z0z1z2z3 ^ S5[x0] ^ S6[x2] ^ S7[x1] ^ S8[x3] ^ S8[z2]

x8x9xAxB = z4z5z6z7 ^ S5[x7] ^ S6[x6] ^ S7[x5] ^ S8[x4] ^ S5[z1]

xCxDxExF = zCzDzEzF ^ S5[xA] ^ S6[x9] ^ S7[xB] ^ S8[x8] ^ S6[z3]

K21 = S5[x3] ^ S6[x2] ^ S7[xC] ^ S8[xD] ^ S5[x8]

K22 = S5[x1] ^ S6[x0] ^ S7[xE] ^ S8[xF] ^ S6[xD]

K23 = S5[x7] ^ S6[x6] ^ S7[x8] ^ S8[x9] ^ S7[x3]

K24 = S5[x5] ^ S6[x4] ^ S7[xA] ^ S8[xB] ^ S8[x7]

z0z1z2z3 = x0x1x2x3 ^ S5[xD] ^ S6[xF] ^ S7[xC] ^ S8[xE] ^ S7[x8]

z4z5z6z7 = x8x9xAxB ^ S5[z0] ^ S6[z2] ^ S7[z1] ^ S8[z3] ^ S8[xA]

z8z9zAzB = xCxDxExF ^ S5[z7] ^ S6[z6] ^ S7[z5] ^ S8[z4] ^ S5[x9]

zCzDzEzF = x4x5x6x7 ^ S5[zA] ^ S6[z9] ^ S7[zB] ^ S8[z8] ^ S6[xB]

K25 = S5[z3] ^ S6[z2] ^ S7[zC] ^ S8[zD] ^ S5[z9]

K26 = S5[z1] ^ S6[z0] ^ S7[zE] ^ S8[zF] ^ S6[zC]

K27 = S5[z7] ^ S6[z6] ^ S7[z8] ^ S8[z9] ^ S7[z2]

K28 = S5[z5] ^ S6[z4] ^ S7[zA] ^ S8[zB] ^ S8[z6]

x0x1x2x3 = z8z9zAzB ^ S5[z5] ^ S6[z7] ^ S7[z4] ^ S8[z6] ^ S7[z0]

x4x5x6x7 = z0z1z2z3 ^ S5[x0] ^ S6[x2] ^ S7[x1] ^ S8[x3] ^ S8[z2]

x8x9xAxB = z4z5z6z7 ^ S5[x7] ^ S6[x6] ^ S7[x5] ^ S8[x4] ^ S5[z1]

xCxDxExF = zCzDzEzF ^ S5[xA] ^ S6[x9] ^ S7[xB] ^ S8[x8] ^ S6[z3]

K29 = S5[x8] ^ S6[x9] ^ S7[x7] ^ S8[x6] ^ S5[x3]

K30 = S5[xA] ^ S6[xB] ^ S7[x5] ^ S8[x4] ^ S6[x7]

K31 = S5[xC] ^ S6[xD] ^ S7[x3] ^ S8[x2] ^ S7[x8]

K32 = S5[xE] ^ S6[xF] ^ S7[x1] ^ S8[x0] ^ S8[xD]

Маскування й перестановка підключів

Km_1, ..., Km_{16} 32-розрядні підключи маскування (один на раунд).

Kr_1, …, Kr_{16} 32-розрядні перестановки підключів (один на раунд); тільки молодші 5 бітів використовуються в кожному раунді.

for (i=1; i<=16; i++) { Km_i = Ki; Kr_i = K16+i; }

Змінний розмір ключа

CAST-128 Алгоритм шифрування був розроблений, щоб розмір ключа міг варіюватися від 40 до 128 біт, в 8-бітному кроці (тобто припустимі розміри ключа рівняються 40, 48, 56, 64..., 112, 120, і 128 бітам). Для змінної роботи розміру ключа специфікація наступні:

1) Для розмірів ключа до й включаючи 80 бітів (тобто, 40, 48, 56, 64, 72, і 80 бітів) алгоритм точно такої ж, але використовує 12 раундів замість 16;

2) Для розмірів ключа більше, чим 80 бітів, алгоритм використовує повні 16 раундів;

3) Для розмірів ключа менше, чим 128 бітів ключ доповнений нульовими байтами (у самих правих, або молодших, позиціях) до 128 биток (тому що розклад ключа CAST 128 ухвалює вхідний ключ 128 бітів).

Розшифрування

Розшифрування для CAST-128 відносно проста. Розшифрування працює в тому ж алгоритмічному напрямку, що й шифрування, починаючи із зашифрованого тексту як вхідних даних. При цьому підключ використовуються у зворотному напрямку.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Програма має простий і інтуїтивно зрозумілий інтерфейс. Головне вікно програми зображене на рисунку 5.1. Воно містить значення регістрів та лічильників годинника реального часу. А також містить наступні кнопки:

– "Читання" - відкриває діалогове вікно читання бітів регістрів стану RTC (рисунок 5.2).

– "Запис" - відкриває діалогове вікно запису у регістри та лічильники RTC (рисунок 5.3).

– "Довідка" - відкриває діалогове вікно довідки (рисунок 5.4).

– "Про програму" - відкриває діалогове вікно про програму (рисунок 5.5).

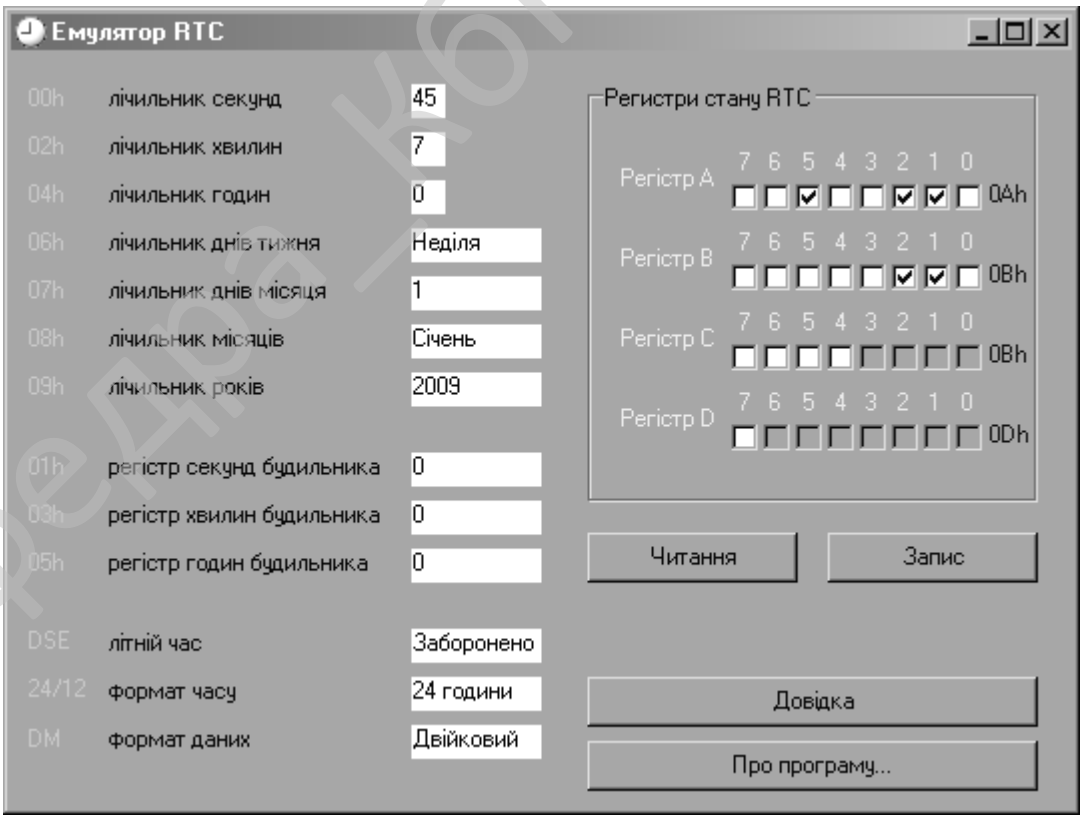

Рисунок 5.1 – Головне вікно програми

Значення регістрів та лічильників, що містять дату та час показані у десятковому форматі, крім лічильника дня тижня і місяця – вони виводяться на екран відповідними назвами. Значення регістрів стану – у двійковому форматі.

Крім регістрів на екран виводяться такі параметри годинника реального часу:

- Дозвіл літнього часу.
- Формат часу (24 або 12).
- Формат даних (двійковий або двійково десятковий).

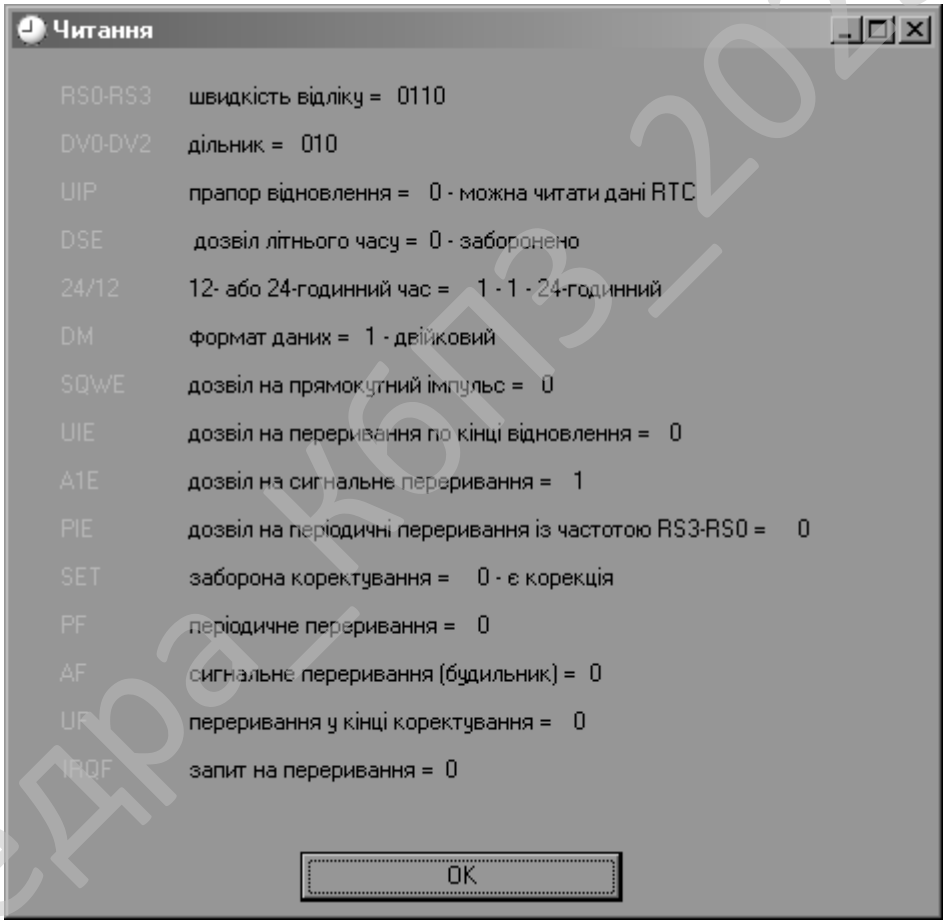

Рисунок 5.2 – Діалогове вікно читанная бітів регістрів стану RTC

Діалогове вікно читанная бітів регістрів стану RTC, що зображене на рисунку 5.2, містить назви і значення бітів та розшифрування їх значень.

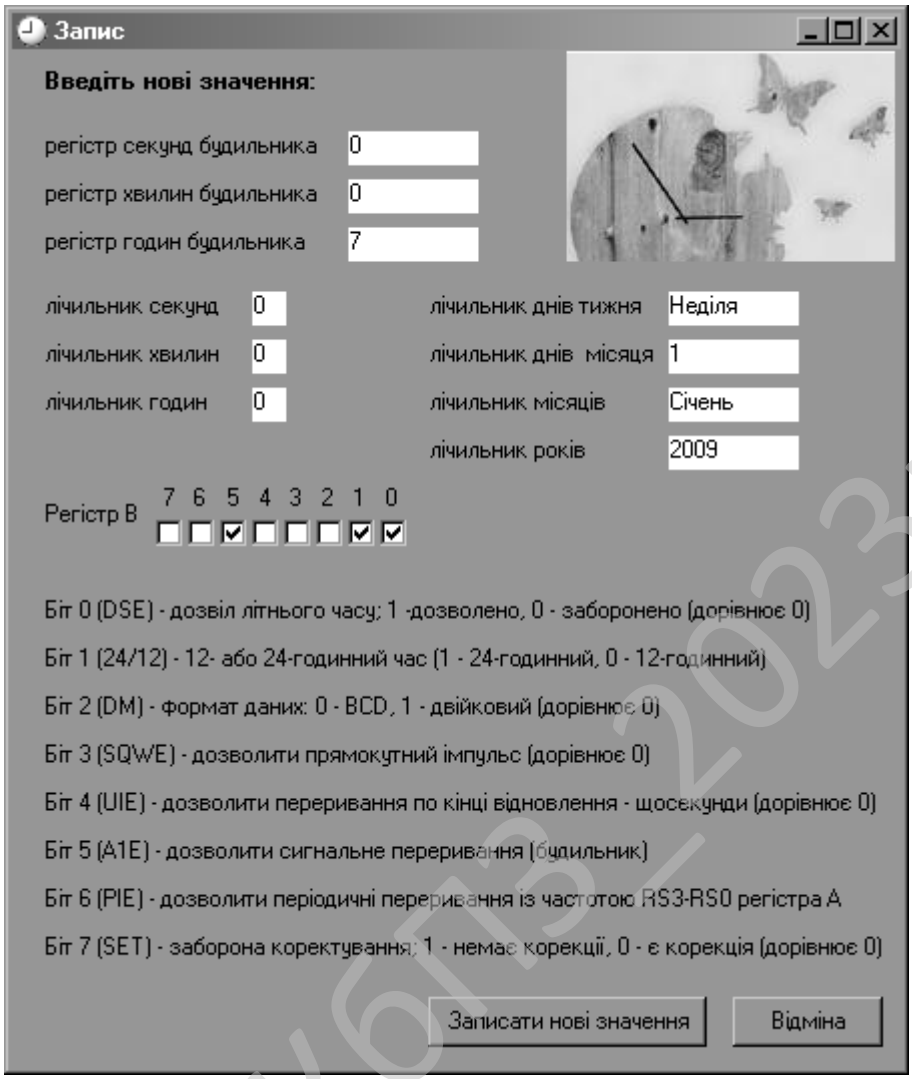

Рисунок 5.3 – Діалогове вікно запису у регістри та лічильники RTC

Діалогове вікно запису у регістри та лічильники RTC, що зображене на рисунку 5.3, містить поля для завдання нових значень лічильнкам часу і дати та регістрам будильника, а також регістру стану B, який можна змінювати програмно. Також у вікні, для зручності, наведені значення бітів регістру стану B.

Після того як введені нові значення, слід натиснути кнопку "Записати нові значення" для підтвердження вводу.

Якщо встановлені значення для регістрів будильника (рисунок 5.4) та при цьому вони співпали з поточним часом, то на екран виводиться вікно будильника з повідомленням про його спрацьовування, а також програється мелодія для звернення уваги користувача.

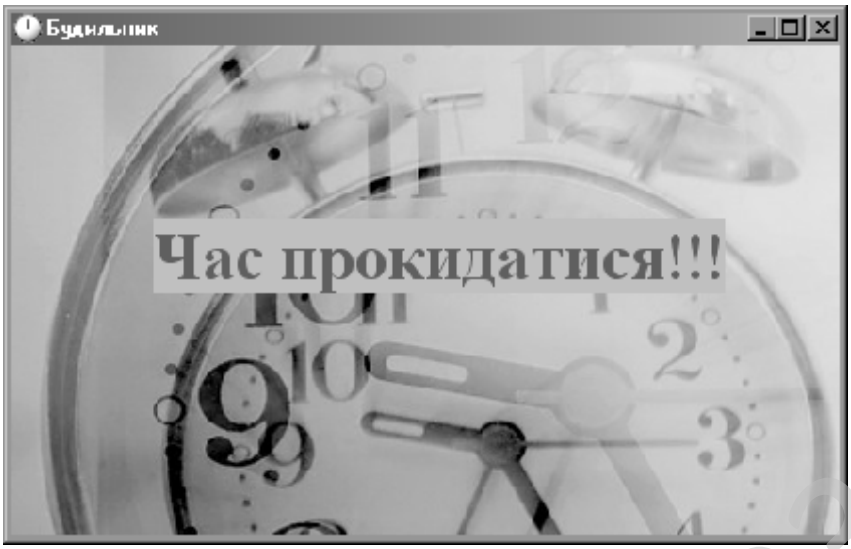

Рисунок 5.4 – Діалогове вікно будильника

В програмі є довідка (рисунок 5.5), що містить наступні розділи:

- Лічільники та регістри.
- Регистр A.
- Регистр B.
- Регистр C.
- Регистр D.

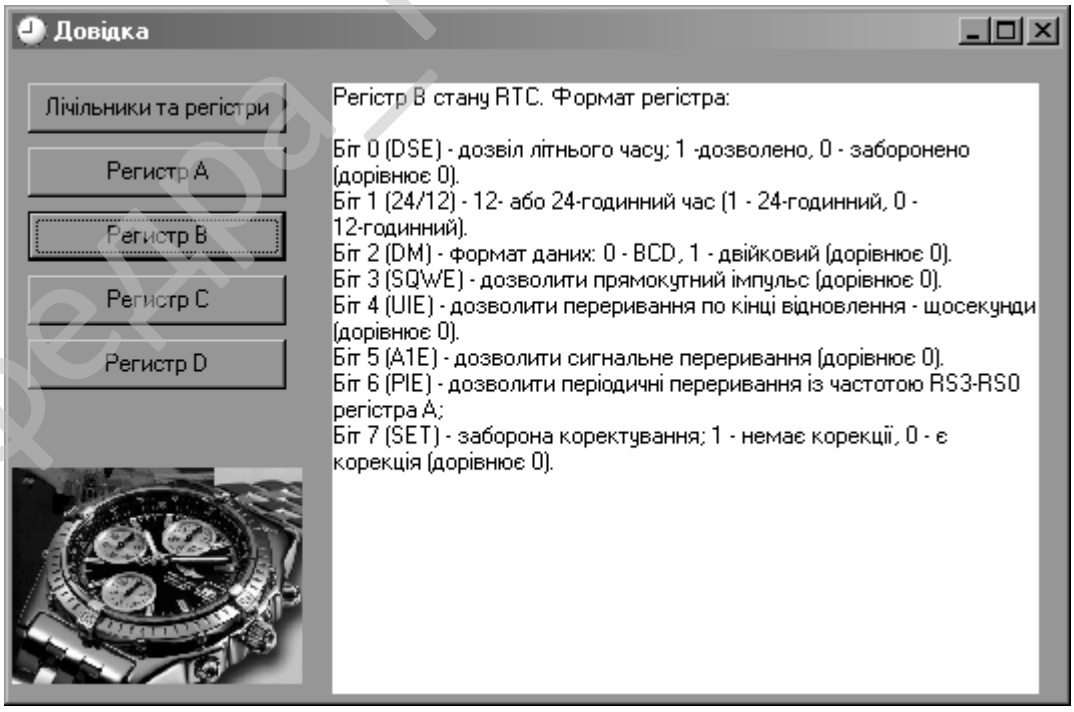

Рисунок 5.5 – Діалогове вікно довідки про регістри та лічильники RTC

Також користувач може переглянути коротку довідку про програму та її автора у діалоговому вікні "Про програму…" (рисунок 5.6).

Рисунок 5.6 – Діалогове вікно доводки про програму

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи візуалізації системного та реального часу ПК.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем візуалізації системного та реального часу ПК.

– Досліджена система візуалізації системного та реального часу ПК.

– На основі отриманих результатів досліджень створена програмна реалізація системи візуалізації системного та реального часу ПК.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання візуалізації системного та реального часу ПК.

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

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

Програма реалізована на мові високого рівня Builder C++. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи візуалізації системного та реального часу ПК. 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 CAST-128.

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

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Клюев В.В. Неразрушающий контроль и диагностика. Справочник, 2-е изд., перераб. и доп. / В.В. Клюев – М: Машиностроение, 2003. – 656 с.

2. Клюня В.Л. Основы экономической теории / В.Л. Клюня, Н.В. Черченко. – Минск: Минск, 2006. – 238 с.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

16. Коваленко А.С. Метод визначення оптимального комплексу робіт з відновлення працездатності інтегрованої системи технічної діагностики в умовах ресурсних обмежень / А.С. Коваленко // Системи обробки інформації. – Х.: ХУПС, 2016. – Вип. 3(140). – С. 04-72.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

50. Онищук А.Г. Радиоприемные устройстваю: Учебн. пособ. – 2-е изд., испр. / А.Г. Онищук, И.И. Забеньков, А.М. Амелин. – Минск: Новое знание, 2007. – 240 с.

51. Осипов В. Базы данных и Delphi. Теория и практика / В. Осипов. – БХВ-Петербург, 2011. – 752 с.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи візуалізації системного та реального часу ПК.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 7-02 від 5.01.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи візуалізації системного та реального часу ПК.

4 Джерела розробки

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

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

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

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи візуалізації системного та реального часу ПК;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Builder C++.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Перелік документів, що розробляються

– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Пояснювальна записка – 63 аркуші.

8 Етапи розробки

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

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

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

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

11.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 1.06.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Коваленко А.С.

Програмне забезпечення системи візуалізації системного та реального часу ПК

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 22

Літера: РП

Кропивницький – 2023 року

Основна програма

Файл ProjectRTC.cpp основної програми

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------------------------
USEFORM("main.cpp", Form1);
USEFORM("about_pr.cpp", Form3);
USEFORM("help.cpp", Form2);
USEFORM("alarm_clock.cpp", Form4);
USEFORM("write.cpp", Form5);
USEFORM("read.cpp", Form6);
//---------------------------------------------------------------------------
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
        try
        {
                Application->Initialize();
                Application->CreateForm(__classid(TForm1), &Form1);
                Application->CreateForm(__classid(TForm3), &Form3);
                Application->CreateForm(__classid(TForm2), &Form2);
                Application->CreateForm(__classid(TForm4), &Form4);
                Application->CreateForm(__classid(TForm5), &Form5);
                Application->CreateForm(__classid(TForm6), &Form6);
                Application->Run();
        }
        catch (Exception &exception)
        {
                Application->ShowException(&exception);
        }
        catch (...)
        {
                try
                {
                        throw Exception("");
                }
                catch (Exception &exception)
                {
                        Application->ShowException(&exception);
                }
        }
        return 0;
}
```

Файл main.cpp основної програми

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "main.h"
#include "about_pr.h"
#include "help.h"
#include "write.h"
#include "read.h"
#include "alarm_clock.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;



//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{

}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
Form3->Show(); //відкриття вікна "Про програму..."
}
//---------------------------------------------------------------------------

//підпрограма, що здійснює відлік імпульсів та обчислює поточну дату і час
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
r1->Text=s; // вивід поточного часу
r2->Text=m;
r3->Text=h;
if(dn==2) r4->Text="Понеділок"; //вивід поточного дня тижня
if(dn==3) r4->Text="Вівторок";
if(dn==4) r4->Text="Середа";
if(dn==5) r4->Text="Четвер";
if(dn==6) r4->Text="П""ятниця";
if(dn==7) r4->Text="Субота";
if(dn==1) r4->Text="Неділя";

r5->Text=dm; //вивід поточного дня місяця

if(n==1) r6->Text="Січень"; //вивід поточного місяця
if(n==2) r6->Text="Лютий";
if(n==3) r6->Text="Березень";
if(n==11) r6->Text="Квітень";
if(n==12) r6->Text="Травень";
if(n==4) r6->Text="Червень";
if(n==5) r6->Text="Липень";
if(n==6) r6->Text="Серпень";
if(n==7) r6->Text="Вересень";
if(n==8) r6->Text="Жовтень";
if(n==9) r6->Text="Листопад";
if(n==10) r6->Text="Грудень";


r7->Text=r; //вивід поточного року

if(s==60){m++; s=0;} else s++; //обчислення секунд
if(m==10){m=0; h++;} //обчислення хвилин
```

```
if(h==24){h=0; dn++; dm++;}//обчислення годин
if(dn==7) dn=0; //обчислення днів тижня

if((dm==31)&(n==1)){dm=1; n++; } //обчислення днів місяця

if((r%4==0)&(r%100!=0)|(r%400==0)) //перевірка на високосний рік
{
if((dm==29)&(n==2)){dm=1; n++; }
}
else
{
if((dm==28)&(n==2)){dm=1; n++; }
}

if((dm==28)&(n==2)){dm=1; n++; }
if((dm==31)&(n==3)){dm=1; n++; }
if((dm==30)&(n==4)){dm=1; n++; }
if((dm==31)&(n==5)){dm=1; n++; }
if((dm==30)&(n==6)){dm=1; n++; }
if((dm==31)&(n==7)){dm=1; n++; }
if((dm==31)&(n==8)){dm=1; n++; }
if((dm==30)&(n==9)){dm=1; n++; }
if((dm==31)&(n==10)){dm=1; n++; }
if((dm==30)&(n==11)){dm=1; n++; }
if((dm==31)&(n==12)){dm=1; n++; }
if(n==12){n=0; r++;}

if((dm==29)&(n==3)) _b0=true; //перехід на літній час
if((dm==26)&(n==7)) _b0=true; //перехід на зимовий час

}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button4Click(TObject *Sender)
{
Form2->Show(); //відкриття вікна довідки
}
//---------------------------------------------------------------------------

//підпрограма ініціалізації основних змінних
void __fastcall TForm1::FormCreate(TObject *Sender)
{

r1->Text=s;
r2->Text=m;
r3->Text=h;
r4->Text=dn;
r5->Text=dm;
r6->Text=n;
r7->Text=r;
r8->Text=sb;
r9->Text=mb;
r10->Text=hb;

if(_b0==true) r11->Text="Дозволено";  else r11->Text="Заборонено";
if(_b1==true) r12->Text="24 години";  else r12->Text="12 години";
if(_b2==true) r13->Text="Двійковий";  else r13->Text="BCD";

a0->Checked=_a0;
a1->Checked=_a1;
a2->Checked=_a2;
a3->Checked=_a3;
a4->Checked=_a4;
a5->Checked=_a5;
a6->Checked=_a6;
a7->Checked=_a7;
b0->Checked=_b0;
b1->Checked=_b1;
b2->Checked=_b2;
```

```
b3->Checked=_b3;
b4->Checked=_b4;
b5->Checked=_b5;
b6->Checked=_b6;
b7->Checked=_b7;
c4->Checked=_c4;
c5->Checked=_c5;
c6->Checked=_c6;
c7->Checked=_c7;
d7->Checked=_d7;

}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button3Click(TObject *Sender)
{
Form6->Show(); //вивід вікна читання значень регістрів
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button2Click(TObject *Sender)
{
Form5->Show();  //вивід вікна запису у регістри
}
//---------------------------------------------------------------------------

//підпрограма перевірки спрацьовування будильника
void __fastcall TForm1::Timer2Timer(TObject *Sender)
{
if((s==sb)&(m==mb)&(h==hb)) Form4->Show();
}
//---------------------------------------------------------------------------
```

Файл main.h – бібліотека для файлу main.cpp

```
//---------------------------------------------------------------------------

#ifndef mainH
#define mainH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published:      // IDE-компоненти управління
        TButton *Button1;
        TGroupBox *GroupBox1;
        TLabel *Label1;
        TLabel *Label2;
        TLabel *Label3;
        TLabel *Label4;
        TLabel *Label6;
        TLabel *Label7;
        TLabel *Label8;
        TLabel *Label9;
        TLabel *Label10;
        TLabel *Label11;
        TLabel *Label12;
        TLabel *Label13;
        TLabel *Label14;
        TLabel *Label15;
        TLabel *Label16;
        TLabel *Label17;
        TLabel *Label18;
        TLabel *Label19;
        TLabel *Label20;
        TLabel *Label21;
        TLabel *Label22;
        TLabel *Label23;
        TLabel *Label24;
        TLabel *Label25;
        TLabel *Label26;
        TLabel *Label27;
        TLabel *Label28;
        TLabel *Label29;
        TLabel *Label30;
        TLabel *Label31;
        TLabel *Label32;
        TLabel *Label33;
        TLabel *Label34;
        TLabel *Label35;
        TLabel *Label36;
        TLabel *Label37;
        TCheckBox *a7;
        TCheckBox *a6;
        TCheckBox *a5;
        TCheckBox *a4;
        TCheckBox *a3;
        TCheckBox *a2;
        TCheckBox *a1;
        TCheckBox *a0;
        TCheckBox *b7;
        TCheckBox *b6;
        TCheckBox *b5;
        TCheckBox *b4;
        TCheckBox *b3;
```

```
TCheckBox *b2;
TCheckBox *b1;
TCheckBox *b0;
TCheckBox *c7;
TCheckBox *c6;
TCheckBox *c5;
TCheckBox *c4;
TCheckBox *c3;
TCheckBox *c2;
TCheckBox *c1;
TCheckBox *c0;
TCheckBox *d7;
TCheckBox *d6;
TCheckBox *d5;
TCheckBox *d4;
TCheckBox *d3;
TCheckBox *d2;
TCheckBox *d1;
TCheckBox *d0;
TLabel *Label5;
TLabel *Label38;
TLabel *Label39;
TLabel *Label40;
TLabel *Label41;
TLabel *Label42;
TLabel *Label43;
TLabel *Label44;
TLabel *Label45;
TLabel *Label46;
TEdit *r1;
TEdit *r2;
TEdit *r3;
TEdit *r4;
TEdit *r6;
TEdit *r7;
TTimer *Timer1;
TButton *Button2;
TButton *Button3;
TButton *Button4;
TEdit *r5;
TEdit *r8;
TEdit *r9;
TEdit *r10;
TLabel *Label47;
TEdit *r11;
TLabel *Label48;
TEdit *r12;
TLabel *Label49;
TEdit *r13;
TTimer *Timer2;
TLabel *Label50;
TLabel *Label51;
TLabel *Label52;
TLabel *Label53;
TLabel *Label54;
TLabel *Label55;
TLabel *Label56;
TLabel *Label57;
TLabel *Label58;
TLabel *Label59;
TLabel *Label62;
TLabel *Label60;
TLabel *Label61;
TLabel *Label63;
TLabel *Label64;
TLabel *Label65;
TLabel *Label66;
void __fastcall Button1Click(TObject *Sender);
void __fastcall Timer1Timer(TObject *Sender);
```

```
        void __fastcall Button4Click(TObject *Sender);
        void __fastcall FormCreate(TObject *Sender);
        void __fastcall Button3Click(TObject *Sender);
        void __fastcall Button2Click(TObject *Sender);
        void __fastcall Timer2Timer(TObject *Sender);

private:    // Визначається користувачем

public:           // Визначається користувачем

        __fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
int s=0;
int m=0;
int h=0;
int dn=1;
int dm=1;
int n=1;
int r=2009;
bool _a0=false;
bool _a1=true;
bool _a2=true;
bool _a3=false;
bool _a4=false;
bool _a5=true;
bool _a6=false;
bool _a7=false;
bool _b0=false;
bool _b1=true;
bool _b2=true;
bool _b3=false;
bool _b4=false;
bool _b5=false;
bool _b6=false;
bool _b7=false;
bool _c4=false;
bool _c5=false;
bool _c6=false;
bool _c7=false;
bool _d7=false;
int sb=-1;
int mb=-1;
int hb=-1;
//---------------------------------------------------------------------------
#endif
```

Файл Read.cpp - читанная регістрів RTC

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "read.h"
#include "main.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm6 *Form6;
//---------------------------------------------------------------------------
__fastcall TForm6::TForm6(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm6::Button1Click(TObject *Sender)
{
Form6->Close();
}
//---------------------------------------------------------------------------

void __fastcall TForm6::FormCreate(TObject *Sender)
{
//вивід значень бітів регістрів стану
int a0;
int a1;
int a2;
int a3;
int a4;
int a5;
int a6;
if(Form1->a0->Checked==true) a0=1; else  a0=0;
if(Form1->a1->Checked==true) a1=1; else  a1=0;
if(Form1->a2->Checked==true) a2=1; else  a2=0;
if(Form1->a3->Checked==true) a3=1; else  a3=0;
if(Form1->a4->Checked==true) a4=1; else  a4=0;
if(Form1->a5->Checked==true) a5=1; else  a5=0;
if(Form1->a6->Checked==true) a6=1; else  a6=0;
L1->Caption=IntToStr(a3)+IntToStr(a2)+IntToStr(a1)+IntToStr(a0);
L2->Caption=IntToStr(a6)+IntToStr(a5)+IntToStr(a4);
if(Form1->a7->Checked==true) L3->Caption="1 - іде коректування";
else L3->Caption="0 - можна читати дані RTC";

if(Form1->b0->Checked==true) L4->Caption="1 -дозволено";
else L4->Caption="0 - заборонено";
if(Form1->b1->Checked==true) L5->Caption="1 - 1 - 24-годинний";
else L5->Caption="0 - 0 - 12-годинний";
if(Form1->b2->Checked==true) L6->Caption="1 - двійковий";
else L6->Caption="0 - BCD";
if(Form1->b3->Checked==true) L7->Caption="1";
else L7->Caption="0";
if(Form1->b4->Checked==true) L8->Caption="1";
else L8->Caption="0";
if(Form1->a5->Checked==true) L9->Caption="1";
else L9->Caption="0";
if(Form1->b6->Checked==true) L10->Caption="1";
else L10->Caption="0";
if(Form1->b7->Checked==true) L11->Caption="1 - немає корекції";
else L11->Caption="0 - є корекція";
if(Form1->c4->Checked==true) L12->Caption="1";
else L12->Caption="0";
if(Form1->c5->Checked==true) L13->Caption="1";
else L13->Caption="0";
```

```
if(Form1->c6->Checked==true) L14->Caption="1";
else L14->Caption="0";
if(Form1->c7->Checked==true) L15->Caption="1";
else L15->Caption="0";
}
//---------------------------------------------------------------------------
```

Файл Read.h – бібліотека для файлу Read.cpp

```
//---------------------------------------------------------------------------

#ifndef readH
#define readH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
//---------------------------------------------------------------------------
class TForm6 : public TForm
{
__published:      // IDE-компоненти керування
        TLabel *Label1;
        TLabel *Label2;
        TLabel *Label3;
        TLabel *Label4;
        TLabel *Label5;
        TLabel *Label6;
        TLabel *Label7;
        TLabel *Label8;
        TLabel *Label9;
        TLabel *Label10;
        TLabel *Label11;
        TLabel *Label12;
        TLabel *Label13;
        TLabel *Label14;
        TLabel *Label15;
        TLabel *L1;
        TLabel *L2;
        TLabel *L3;
        TLabel *L4;
        TLabel *L5;
        TLabel *L6;
        TLabel *L7;
        TLabel *L8;
        TLabel *L9;
        TLabel *L10;
        TLabel *L11;
        TLabel *L12;
        TLabel *L13;
        TLabel *L14;
        TLabel *L15;
        TButton *Button1;
        TLabel *Label17;
        TLabel *Label18;
        TLabel *Label19;
        TLabel *Label20;
        TLabel *Label21;
        TLabel *Label22;
        TLabel *Label23;
        TLabel *Label24;
        TLabel *Label25;
        TLabel *Label26;
        TLabel *Label27;
        TLabel *Label28;
        TLabel *Label29;
        TLabel *Label30;
        TLabel *Label31;
        void __fastcall Button1Click(TObject *Sender);
        void __fastcall FormCreate(TObject *Sender);
private:    // Визначається користувачем
public:           // Визначається користувачем
        __fastcall TForm6(TComponent* Owner);
};
```

```
//---------------------------------------------------------------------------
extern PACKAGE TForm6 *Form6;
//---------------------------------------------------------------------------
#endif
```

Файл Write.cpp - запис у регістри та лічильники RTC

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "write.h"
#include "main.h"

//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm5 *Form5;
//---------------------------------------------------------------------------
__fastcall TForm5::TForm5(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm5::FormCreate(TObject *Sender)
{
//вивід на екран поточних значень регістрів та лічільників
rr1->Text=Form1->r1->Text;
rr2->Text=Form1->r2->Text;
rr3->Text=Form1->r3->Text;
rr4->Text=Form1->r4->Text;
rr5->Text=Form1->r5->Text;
rr6->Text=Form1->r6->Text;
rr7->Text=Form1->r7->Text;
rr8->Text=Form1->r8->Text;
rr9->Text=Form1->r9->Text;
rr10->Text=Form1->r10->Text;
}
//---------------------------------------------------------------------------

//введення нових значень регістрів та лічільників
void __fastcall TForm5::Button1Click(TObject *Sender)
{
s=StrToInt(rr1->Text);
m=StrToInt(rr2->Text);
h=StrToInt(rr3->Text);
dn=StrToInt(rr4->Text);
dm=StrToInt(rr5->Text);
n=StrToInt(rr6->Text);
r=StrToInt(rr7->Text);
sb=StrToInt(rr9->Text);
mb=StrToInt(rr9->Text);
hb=StrToInt(rr10->Text);

_b0=bb0->Checked;
_b1=bb1->Checked;
_b2=bb2->Checked;
_b3=bb3->Checked;
_b4=bb4->Checked;
_b5=bb5->Checked;
_b6=bb6->Checked;
_b7=bb7->Checked;

Form5->Close();

}
//---------------------------------------------------------------------------
```

Файл Read.h – бібліотека для файлу Read.cpp

```
//---------------------------------------------------------------------------

#ifndef readH
#define readH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
//---------------------------------------------------------------------------
class TForm6 : public TForm
{
__published:      // IDE-компоненти керування
        TLabel *Label1;
        TLabel *Label2;
        TLabel *Label3;
        TLabel *Label4;
        TLabel *Label5;
        TLabel *Label6;
        TLabel *Label7;
        TLabel *Label8;
        TLabel *Label9;
        TLabel *Label10;
        TLabel *Label11;
        TLabel *Label12;
        TLabel *Label13;
        TLabel *Label14;
        TLabel *Label15;
        TLabel *L1;
        TLabel *L2;
        TLabel *L3;
        TLabel *L4;
        TLabel *L5;
        TLabel *L6;
        TLabel *L7;
        TLabel *L8;
        TLabel *L9;
        TLabel *L10;
        TLabel *L11;
        TLabel *L12;
        TLabel *L13;
        TLabel *L14;
        TLabel *L15;
        TButton *Button1;
        TLabel *Label17;
        TLabel *Label18;
        TLabel *Label19;
        TLabel *Label20;
        TLabel *Label21;
        TLabel *Label22;
        TLabel *Label23;
        TLabel *Label24;
        TLabel *Label25;
        TLabel *Label26;
        TLabel *Label27;
        TLabel *Label28;
        TLabel *Label29;
        TLabel *Label30;
        TLabel *Label31;
        void __fastcall Button1Click(TObject *Sender);
        void __fastcall FormCreate(TObject *Sender);
private:    // Визначається користувачем
public:           // Визначається користувачем
        __fastcall TForm6(TComponent* Owner);
};
```

```
//---------------------------------------------------------------------------
extern PACKAGE TForm6 *Form6;
//---------------------------------------------------------------------------
#endif
```

Файл alarm_clock.cpp - вікно будильника

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "alarm_clock.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm4 *Form4;
//---------------------------------------------------------------------------
__fastcall TForm4::TForm4(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TForm4::FormShow(TObject *Sender)
{

//програвання звукового файлу при спрацьовуванні будильника
MediaPlayer1->FileName="tada.wav";
MediaPlayer1->Open();
MediaPlayer1->Play();
}
//---------------------------------------------------------------------------
```

Файл alarm_clock.h – бібліотека для файлу alarm_clock.cpp

```
//---------------------------------------------------------------------------

#ifndef alarm_clockH
#define alarm_clockH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
#include <Graphics.hpp>
#include <MPlayer.hpp>
//---------------------------------------------------------------------------
class TForm4 : public TForm
{
__published:      // IDE-компоненти керування
        TImage *Image1;
        TLabel *Label1;
        TMediaPlayer *MediaPlayer1;
        void __fastcall FormShow(TObject *Sender);
private:    // Визначається користувачем
public:           // Визначається користувачем
        __fastcall TForm4(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm4 *Form4;
//---------------------------------------------------------------------------
#endif
```

Файл help.cpp - довідка

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "help.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm2 *Form2;
//---------------------------------------------------------------------------
__fastcall TForm2::TForm2(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
// довідка про регістри та лічильники RTC
void __fastcall TForm2::Button1Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("00h лічильник секунд");
Memo1->Lines->Add("01h регістр секунд будильника");
Memo1->Lines->Add("02h лічильник хвилин");
Memo1->Lines->Add("03h регістр хвилин будильника");
Memo1->Lines->Add("04h лічильник годин");
Memo1->Lines->Add("05h регістр годин будильника");
Memo1->Lines->Add("06h лічильник днів тижня (1 - неділя)");
Memo1->Lines->Add("07h лічильник днів місяця");
Memo1->Lines->Add("08h лічильник місяців");
Memo1->Lines->Add("09h лічильник років (останні 2 цифри року)");
Memo1->Lines->Add("0Ah Регистр статуса RTC A");
Memo1->Lines->Add("0Bh Регистр статуса RTC B");
Memo1->Lines->Add("0Ch Регистр статуса RTC C");
Memo1->Lines->Add("0Dh Регистр статуса RTC D");
}
//---------------------------------------------------------------------------
// довідка про регістр А стану RTC
void __fastcall TForm2::Button2Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Регістр А стану RTC. Формат регістра: ");
Memo1->Lines->Add("");
Memo1->Lines->Add("Біти 0-3 (RS0-RS3) - швидкість відліку (дорівнює 0110). ");
Memo1->Lines->Add("Біти 4-6 (DV0-DV2) - 22-розрядний дільник (дорівнює 010). ");
Memo1->Lines->Add("Біт 7 (UIP) - прапор відновлення (0 - можна читати дані RTC,
1 - іде коректування). ");
}
//---------------------------------------------------------------------------

// довідка про регістр В стану RTC
void __fastcall TForm2::Button3Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Регістр В стану RTC. Формат регістра: ");
Memo1->Lines->Add("");
Memo1->Lines->Add("Біт 0 (DSE) - дозвіл літнього часу; 1 -дозволено, 0 -
заборонено (дорівнює 0). ");
Memo1->Lines->Add("Біт 1 (24/12) - 12- або 24-годинний час (1 - 24-годинний, 0 -
12-годинний). ");
Memo1->Lines->Add("Біт 2 (DM) - формат даних: 0 - BCD, 1 - двійковий (дорівнює
0). ");
Memo1->Lines->Add("Біт 3 (SQWE) - дозволити прямокутний імпульс (дорівнює 0).
");
Memo1->Lines->Add("Біт 4 (UIE) - дозволити переривання по кінці відновлення -
щосекунди (дорівнює 0). ");
```

```
Memo1->Lines->Add("Біт 5 (A1E) - дозволити сигнальне переривання (дорівнює 0).
");
Memo1->Lines->Add("Біт 6 (PIE) - дозволити періодичні переривання із частотою
RS3-RS0 регістра A; ");
Memo1->Lines->Add("Біт 7 (SET) - заборона коректування; 1 - немає корекції, 0 -
є корекція (дорівнює 0). ");
}
//---------------------------------------------------------------------------
// довідка про регістр C стану RTC
void __fastcall TForm2::Button4Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Регістр C стану RTC - біти стану переривань, доступні тільки
для читання. При читанні з регістра всі розряди скидаються. Формат регістра: ");
Memo1->Lines->Add("");
Memo1->Lines->Add("Біти 0-3 - зарезервовані. ");
Memo1->Lines->Add("Біти 4-6 - причини запиту. ");
Memo1->Lines->Add("Біт 4 (PF) - періодичне переривання. ");
Memo1->Lines->Add("Біт 5 (AF) - сигнальне переривання (поточний час збігłося із
часом будильника). ");
Memo1->Lines->Add("Біт 6 (UF) - переривання по кінці коректування. ");
Memo1->Lines->Add("Біт 7 (IRQF) - запит на переривання. ");
}
//---------------------------------------------------------------------------
// довідка про регістр D стану RTC
void __fastcall TForm2::Button5Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Регістр D стану RTC. Біт 7=1, якщо CMOS одержує живлення;
0=немає живлення від автономного джерела.");
}
//---------------------------------------------------------------------------
```

Файл help.h – бібліотека для файлу help.cpp

```
//---------------------------------------------------------------------------

#ifndef helpH
#define helpH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
#include <Graphics.hpp>
//---------------------------------------------------------------------------
class TForm2 : public TForm
{
__published:      // IDE-компоненти керування
        TMemo *Memo1;
        TButton *Button1;
        TButton *Button2;
        TButton *Button3;
        TButton *Button4;
        TButton *Button5;
        TImage *Image1;
        void __fastcall Button1Click(TObject *Sender);
        void __fastcall Button2Click(TObject *Sender);
        void __fastcall Button3Click(TObject *Sender);
        void __fastcall Button4Click(TObject *Sender);
        void __fastcall Button5Click(TObject *Sender);
private:    // Визначається користувачем
public:           // Визначається користувачем
        __fastcall TForm2(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm2 *Form2;
//---------------------------------------------------------------------------
#endif
```

Файл about_pr.cpp – вікно "про програму..."

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "about_pr.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm3 *Form3;
//---------------------------------------------------------------------------
__fastcall TForm3::TForm3(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm3::Button1Click(TObject *Sender)
{
Form3->Close();
}
//---------------------------------------------------------------------------
```

Файл about_pr.h – бібліотека для файлу about_pr.cpp

```
//---------------------------------------------------------------------------

#ifndef about_prH
#define about_prH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
#include <Graphics.hpp>
//---------------------------------------------------------------------------
class TForm3 : public TForm
{
__published:      // IDE-компоненти керування
        TImage *Image1;
        TLabel *Label1;
        TLabel *Label2;
        TLabel *Label3;
        TLabel *Label4;
        TLabel *Label5;
        TLabel *Label6;
        TLabel *Label7;
        TButton *Button1;
        TLabel *Label8;
        void __fastcall Button1Click(TObject *Sender);
private:    // Визначається користувачем
public:           // Визначається користувачем
        __fastcall TForm3(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm3 *Form3;
//---------------------------------------------------------------------------
#endif
```