МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

До захисту допускається
Завідувач кафедри
_____Скарга-Бандурова І.С.
«____»_____ 2018 р.

ДИПЛОМНИЙ ПРОЕКТ (РОБОТА) БАКАЛАВРА

ПОЯСНЮВАЛЬНА ЗАПИСКА

НА ТЕМУ:

Інструментальні засоби налагоджування прикладних програм МСКУ-4

Освітньо-кваліфікаційний рівень "бакалавр"
Напрям 6.050102– "Комп'ютерна інженерія"

Керівник проекту:	_____ (підпис)	доц. Ларгін В. А. (ініціали, прізвище)
Консультант з охорони праці:	_____ (підпис)	ст.викл.Критська Я. О. (ініціали, прізвище)
Здобувач вищої освіти:	_____ (підпис)	Коверга М.О. (ініціали, прізвище)
Група:		КІ-14ад

Сєвєродонецьк 2018

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра Комп'ютерних наук та інженерії
Освітньо-кваліфікаційний рівень бакалавр
Напрям підготовки 6.050102 Комп'ютерна інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:

Завідувач кафедри _____
_____ І.С. Скарга-Бандурова
«_____» _____2018 р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) БАКАЛАВРА

Коверги Марка Олександровича
(прізвище, ім'я, по батькові)

1. Тема роботи Інструментальні засоби налагодження прикладних програм МСКУ-4

керівник проекту (роботи) Ларгін В. А., к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "14 " 05 2018р. №
2. Термін подання студентом роботи
3. Вихідні дані до роботи матеріали переддипломної практики

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Огляд об'єкту дослідження. Розробка алгоритму інструментальних засобів. Написання комплексу програм. Тестування роботи кожної програми. Охорона праці. Висновки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Електронні плакати

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	ст.викл. кафедри КНІ Критська Я.О.		

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Огляд літератури з теми ДП і постановка задачі	14.05.18-19.05.18	
2	Дослідження матеріалів	20.05.18-25.05.18	
3	Розробка програмної системи	26.05.18-02.06.18	
4	Тестування програмної системи	03.06.18-06.06.18	
5	Розробка розділу охорона праці	07.06.18-09.06.18	
6	Оформлення електронних плакатів	10.06.18-12.06.18	
7	Оформлення пояснювальної записки	13.06.18-15.06.18	

Здобувач вищої освіти _____ Коверга М.О.
(підпис) (прізвище та ініціали)

Керівник _____ Ларгін В. А.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Бакалаврська робота Коверги Марка Олександровича

На тему: «Інструментальні засоби налагоджування прикладних програм МСКУ-4»

Предметом дослідницької праці є розробка програмних засобів налагодження прикладних програм МСКУ-4.

У вступі вибирається об’єкт дослідження, наводиться його практична та теоретична важливість.

Перший розділ містить в собі загальні відомості про досліджуваний об’єкт, розкривається актуальність завдання, задається проблема, мета й задачі дослідження, виконується аналіз апаратних та програмних засобів також формуються вимоги до підсистеми налагодження.

В другій главі представлені й докладно розписані логічна структура інструментальних засобів налагодження.

В третьому розділі представлена алгоритмічна реалізація інструментальних засобів налагодження разом із програмними рішеннями.

В четвертому розділі представлена інструкція з користування розробленою підсистемою налагодження.

Висновки присвячені узагальненню данної роботи.

Перелік ключових скорочень: МСКУ, КС, ФБ, КП, КМп.

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ПТК - програмно – технічний комплекс

МСКУ - мікропроцесорні субкомплекси контролю й управління

КС - керуюча система

СДА - сервер діагностування та архівації

КМп - контролери мікропроцесорні

ІРП - інтерфейс радіальний

МЗв - модулі зв’язку

МЗО - модулі зв'язку з об'єктом

МКО - модулі контролю обладнання

ПКр - панелі кросові

ПСп - панелі сполучні

НР - нижній рівень

КП ОД - керуюча програма обробки даних

КНР - комплекс нижнього рівняі

ІЗН - інструментальні засоби налагодження

ФБ - функціональний блок

ПЗО пристрій зв’язку з об’єктом

ЗМІСТ

ВСТУП

Основними напрямками діяльності ПрАТ «СНВО «Імпульс» є розробка, виробництво, постачання, введення в експлуатацію та підтримка експлуатації систем контролю та управління (СКУ) для об’єктів атомної енергетики, залізнодорожнього транспорту та інших галузей промисловості [1] [2].

СКУ базуються на промислових контролерах МСКУ-4.

МСКУ-4 виконує наступні функції:

- введення та обробка даних від датчиків аналогових та дискретних сигналів;
- реалізація алгоритмів контролю та управління, різноманітних законів регулювання, захисту, блокування, пуску та зупину обладнання;
- формування та видача аналогових та дискретних сигналів, команд управління;
- взаємозв’язок з зовнішніми абонентами .

При розробці прикладних програм, що функціонують в МСКУ, необхідно надати розробнику сервісні засоби налагодження. Тому виникла необхідність розробки комплексу інструментальних програм, що функціонують в найбільш поширеній операційній системі Windows, що забезпечують налагодження прикладних програм, розроблених на мові технологічного програмування.

Метою даної дипломної роботи є розробка інструментальних засобів налагодження прикладних програм МСКУ-4.

1 ОГЛЯД ТЕХНІЧНИХ І ПРОГРАМНИХ ЗАСОБІВ МСКУ-4. ПОСТАНОВКА ЗАДАЧІ ПРОЕКТУВАННЯ

Даний розділ присвячується огляду технічних та програмних засобів МСКУ-4. В даному розділі розглянута структура МСКУ-4, розглянуті основні вимоги до підсистеми налагодження. На підставі аналізу сформована постановка задачі та сформовано технічне завдання.

1.1 Загальні відомості

1.1.1 Структура МСКУ-4

Склад конкретної МСКУ-4 визначається особливостями її застосування в системі автоматизації об'єкта.

МСКУ-4 представляє собою комплекс технічних засобів, до складу якого входять [3]:

– контролери мікропроцесорні (КМп) на основі інтерфейсу PCI-Express. Кожен КМп оснащений двома портами оптичного інтерфейсу Ethernet, двома портами Ethernet із мідними провідниками і 13 портами внутрішнього радіального інтерфейсу ІРП-4. Кожен КМп може мати 4, 8 або 12 додаткових портів оптичного Ethernet при установці одного, двох або трьох модулів зв'язку МЗО-41 відповідно;

– модулі зв'язку з об'єктом (МЗО), призначені для введення/виведення дискретних і аналогових сигналів, оснащені трьома портами ІРП-4;

– модулі зв'язку МЗО-42, призначені для зв'язку між оптичним Ethernet і 49 портами ІРП-4;

– модулі МЗО-43, призначені для зовнішніх зв'язків по інтерфейсах RS-422 і RS-485;

– модулі контролю обладнання МКО-4, призначені для контролю працездатності і стану обладнання, розміщеного в шафі МСКУ-4;

– панелі кросові (ПКр), призначені для підключення кабелів від об'єктів;

– панелі сполучні (ПСп), призначені для зв'язку ПКр з МЗО;

– каркаси монтажні (КМ), призначені для встановлення КМп і МЗО.

Центральна частина МСКУ-4 може бути нерезервованою (1 КМп) або резервованою (3 КМп). Передбачена можливість застосування нерезервованих, дубльованих і тройованих МЗО.

1.1.2 Загальні відомості про керуючу систему (КС) МСКУ

КС МСКУ [4] являє собою сукупність налаштувань конфігурації МСКУ й прикладних програм користувача, які організовують роботу МСКУ відповідно до визначених користувачем функцій.

Цільова КС МСКУ створюється під кожне конкретне застосування МСКУ розробником відповідної системи управління, до складу якої входить МСКУ.

Цільова КС МСКУ складається з завантажувального модуля, що включає:

- резидентное ядро КС МСКУ;
- налаштування та коди виконуваних прикладних програм.

Резидентне ядро КС МСКУ включає програмні модулі для виконання необхідних функцій усіх виконань МСКУ й записується в FLASH ПП КМп.

Завантажувальний (виконуваний) модуль КС МСКУ готується інструментальними засобами комплексу програм на підставі файлів вихідних текстів налаштувань [5] і вихідних текстів прикладних програм. Даний завантажувальний модуль записується до FLASH ПП КМп [6] [7].

У КС МСКУ обмін із компонентами ПТК може здійснюватися через мережеві контролери Ethernet.

Обробка даних у робочій станції можлива за допомогою прикладного процесу, організованого за принципом КС МСКУ. Прикладний процес являє собою програму обробки, що настроюється КП ОД, яка функціонує в робочій станції у середовищі ОС Linux і забезпечує обмін блоками даних із будь-якими абонентами мереж та іншими процесами (наприклад, ОБД СДА), виконання прикладних програм обробки даних, написаних користувачем на мові технологічного програмування.

КП ОД забезпечує виконання таких функцій:

− виконання прикладних програм, розроблених на мові технологічного програмування, для обробки даних, прийнятих від інших абонентів мережі (мереж) Ethernet нижнього рівня;

− підготовку і передачу даних іншим абонентам мережі(мереж) Ethernet НР.

− КП ОД є віртуальний нерезервований комплекс нижнього рівня (далі - КНР)[7].

− інструментальні засоби призначені для віддаленої налагодження прикладних програм в КМп, а КП ОД - в робочій станції, і надають такі можливості:

− встановлення й скасування контрольних точок зупину;

− запуск програми в покроковому або безперервному режимі;

− контроль і зміна значень вхідних/вихідних і локальних параметрів в оперативній пам'яті КМп (в тому числі, значення вхідних сигналів);

− автоматичне формування сценаріїв (як результату запам'ятовування всіх команд, заданих користувачем у діалоговому режимі), а також можливість їх коригування (для спрощеного повтору налагодження після коригувань програм);

− автоматичне формування протоколів налагодження, а також можливість порівняння протоколів налагодження, виконане в різний час (із формуванням протоколів відмінностей результатів налагодження);

− крос-налагодження прикладної програми безпосередньо в інструментальній ПЕОМ.

1.1.3 Структура керуючої програми

Загальна структура КП [8], а також основні групи модулів КП, показані на рис.1.1.

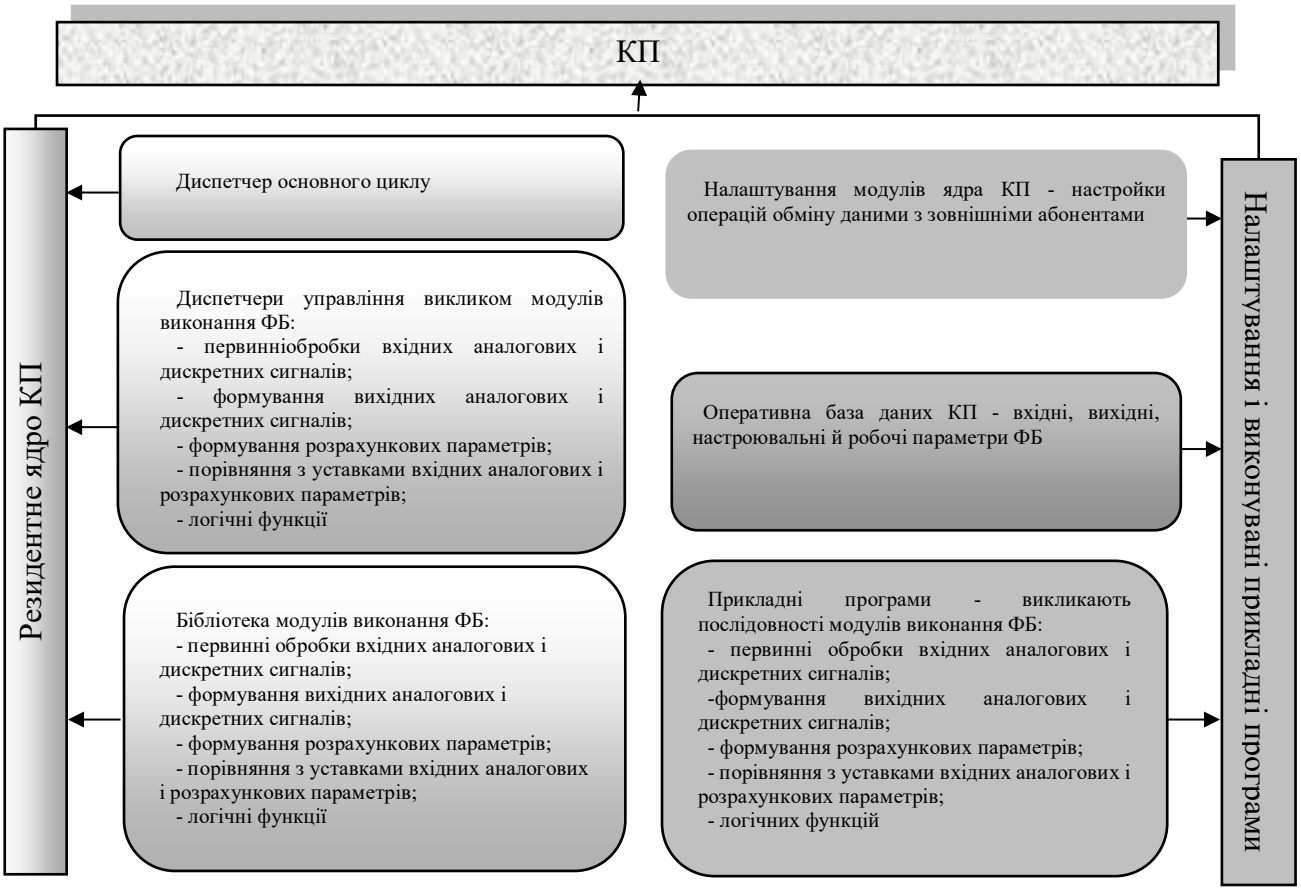


Рисунок 1.1 – Структура КП

1.1.4 Будова функціональних блоків

Виконуваний модуль ФБ представляє собою фрагмент коду, який запускається з диспетчера переходом по таблиці ФБ згідно з номером блоку - першим словом структури поточної послідовності. Старший байт слова - номер групи - відповідає функціональному призначенню диспетчера, молодший - вказує безпосередньо на місце адреси модуля блоку в таблиці.

Коду виконання передається адреса в списку викликів ФБ. Параметри списку викликів ФБ доступні через імена відповідної структури. ФБ повинен переінсталювати при виході вказівник списку викликів на позицію наступного ФБ.

Алгоритм ФБ наступний:

– зчитування зсувів вхідних і вихідних параметрів із структури послідовності, що викликає;

– зчитування вхідних параметрів із ОБД;

– формування результату;

– запис результату в ОБД щодо зміщення вихідного параметра;

– інсталяція вказівника в список викликів на наступний ФБ.

1.1.5 Загальні відомості про налагодження

Налагодження програми - етап розробки, в процесі якого відбуваються виявлення, локалізація й усунення очевидних помилок у програмі.

Використання зневаджувача дозволяє запустити прикладну програму в більш контрольованому оточенні й виконувати її у покроковому режимі по рядкам, виконувати програму в автоматичному режимі до заданої точки зупину, читати й змінювати значення змінних і т. п.

1.2 Технічне завдання на розробку

1.2.1 Актуальність розробки

При розробці прикладних програм, що функціонують в МСКУ-4, необхідно надавати розробнику сервісні засоби налагодження. Для вирішення цього завдання призначені інструментальні засоби налагодження. Існуючі засоби налагодження не відповідають вимогам, оскільки вони призначені для попереднього покоління МСКУ-3, а засоби налагодження інших виробників будуть несумісні. Отже, виходячи з вищесказаного є необхідність розробити інструментальні засоби налагодження для МСКУ-4, що функціонують у найбільш поширеній операційній системі Windows, що забезпечують налагодження прикладних програм, розроблених на мові технологічного програмування.

1.2.2 Мета дипломної роботи

Метою даної дипломної роботи є розробка інструментальних засобів налагодження прикладних програм МСКУ-4 в ПЕОМ.

1.3 Основні завдання даного дипломного проекту

1. дослідити об’єкт, розкрити актуальність завдання, сформувати мету й задачі дослідження, зробити аналіз апаратних та програмних засобів, використаних при вирішенні поставленної задачі та сформувати вимоги до підсистеми налагодження;
2. розглянути, дослідити та детально описати логічну структуру інструментальних засобів та допоміжних засобів підсистеми налагодження;
3. представити алгоритм та комплекс програм, що забезпечують налагодження прикладних програм МСКУ-4.

1.4 Вимоги до підсистеми налагодження – до програм і технічних засобів

1.4.1 Вимоги до складу технічних засобів

Інструментальні засоби налагодження повинні функціонувати на інструментальній IBM PC-сумісній ПЕОМ під управлінням операційної системи Windows XP/2000(Service Pack4) (далі - ОС Windows).

До складу інструментальної ПЕОМ повинні входити:

- процесор архітектури x86 з частотою 2 GHz або вище;
- оперативна пам’ять не менше 1 Gbyte;
- жорсткий диск з об’ємом не менше 10 Gbyte;
- кольоровий монітор стандарта WXGA з роздільною здатністю екрана не менше 1280 на 720 точок (при налаштуванні монітора в ОС Windows рекомендується використовувати встановлюваний за замовчування мілкий розмір шрифта дисплея (96

точок на дюйм) для правильного відображення надписів і повідомлень, виводимих у процесі роботи программ комплексу);

- пристрій введення з CD-ROM;
- клавіатура алфавітно-цифрова ;
- маніпулятор «миша»;
- апаратні засоби мережі Ethernet для обміну с КМп.

1.4.2 Вимоги до програмних засобів

До розроблюваних програмних засобів налагодження пред'являються наступні загальні вимоги:

– програми повинні бути розроблені на мові високого рівня C/C++;

– мати модульну структуру;

– повинні використовувати бібліотеки графічного інтерфейса, які дозволяють функціонувати або в операційній системі Windows, або в Linux.

1.4.2.1 Вимоги до складу програми

Сервісні засоби налагодження прикладних програм, завантажених в МСКУ/КМп, і КП ОД, завантажених у робочу станцію, повинні містити в собі:

– комплекс програмних засобів, що функціонують у підключеній до МСКУ/КМп інструментальній ПЕОМ (під управлінням ОС Windows) і забезпечують діалог із користувачем, який виконує процес налагодження й управління процесом виконання прикладних програм у МСКУ/КМп відповідно до вказівок користувача;

– модулі підтримки виконання прикладної програми в КМп у режимі налагодження, що забезпечують віддалене налагодження прикладної програми за командами зневаджувача.

– модулі підготовки виконуваного модуля УС для налагодження прикладних програм в МСКУ/КМп.

1.4.2.2 Вимоги до структури програми

Відповідно до правил використання додатків програмні засоби повинні мати формат виконуваного додатка з набором конфігураційних і допоміжних файлів із структурованими даними. Він повинен функціонувати в середовищі ОС Windows/Linux за умови завантаженої у МСКУ-4 виконавчої системи.

1.4.2.3 Вимоги до функцій програмних засобів налагодження

Розроблювані програмні засоби налагодження повинні забезпечувати виконання наступних функцій:

– завантаження КС із прикладної програми, яку потрібно налагодити, в оперативну пам'ять КМп(підготовка КС для режиму налагодження повинна бути виконана за допомогою інструментальних засобів підготовки);

– читання-запис глобальних змінних оперативної бази даних, пов'язаних із контекстом програми;

– читання-запис локальних змінних ОБД (внутрішніх змінних, використовуваних при виконанні функціональних блоків);

– читання-запис даних оперативної пам'яті МСКУ по фізичним адресам пам'яті й значень названих елементів КС МСКУ по ідентифікаторам елементів;

– передачі в МСКУ на вимогу оператора вихідних повідомлень операцій типу «запит-відповідь», прийом і виведення оператору вмісту відповідних повідомлень від МСКУ на дані операції;

– пуску програми в покроковому, тактовому або безперервному режимі;

– установа/скасування контрольних точок зупину;

– контролю й зміни значень вхідних дискретних і аналогових параметрів у режимі діалогу або в режимі сценарію.

1.4.2.4 Вимоги до режимів відладки

Зневаджувач повинен надавати користувачеві можливість налагодження:

– діалоговим способом із керуванням процесом налагодження за допомогою команд, що задаються через «меню-віконний» інтерфейс зневаджувача. Зневаджувач повинен автоматично формувати поточний сценарій процесу налагодження(текстовий файл), як результат запам'ятовування всіх заданих користувачем команд, і надавати можливість його збереження для подальшого використання. Сценарій у подальшому може використовуватися для повторення всіх необхідних дій (наприклад, після корегування програм) за допомогою команди запуску виконання сценарію й порівняння знову отриманого протоколу налагодження з раніше сформованим;

– за допомогою виконання сценарію(текстовий файл), в якому визначено необхідна послідовність команд для їх виконання зневаджувачем. Сценарій може бути сформований будь-яким текстовим редактором, або автоматично як результат запам'ятовування всіх команд, заданих користувачем в діалоговому режимі роботи з зневаджувачем.

1.5 Визначення функціональної структури розроблюваної підсистеми

Підсистема, що розробляється повинна бути комплексом програм функціонуючих під керівництвом диспетчера-монітора.

Монітор призначений для прийому команд оператора й передачі їх у відповідну секцію.

Диспетчер повинен забезпечувати виклик модулів, що виконують такі функції:

- пошук адреси змінної або адреси програми, що налагоджують, по map-файлу;
- виклик функцій читання/запису;
- виклик функцій перегляду по підсистемам;

– виконання команд в покроковому режимі (це посилання запиту в МСКУ-4 і очікування переходу програми в МСКУ-4 в наступну точку);

– установ контрольної точки.

Висновок до розділу 1

У данному розділі розкривається актуальність завдання дипломного проекту, сформовані вимоги до підсистеми налагодження та містяться загальні вимоги про досліджуваний об’єкт.

2 ОПИС ІНСТРУМЕНТАЛЬНИХ ПРОГРАМ НАЛАГОДЖЕННЯ. ЛОГІЧНА СТРУКТУРА.

2.1 Склад інструментальних програм налагодження

Інструментальні засоби налагодження [9], будуть мати структуру представлену на рис 2.2.

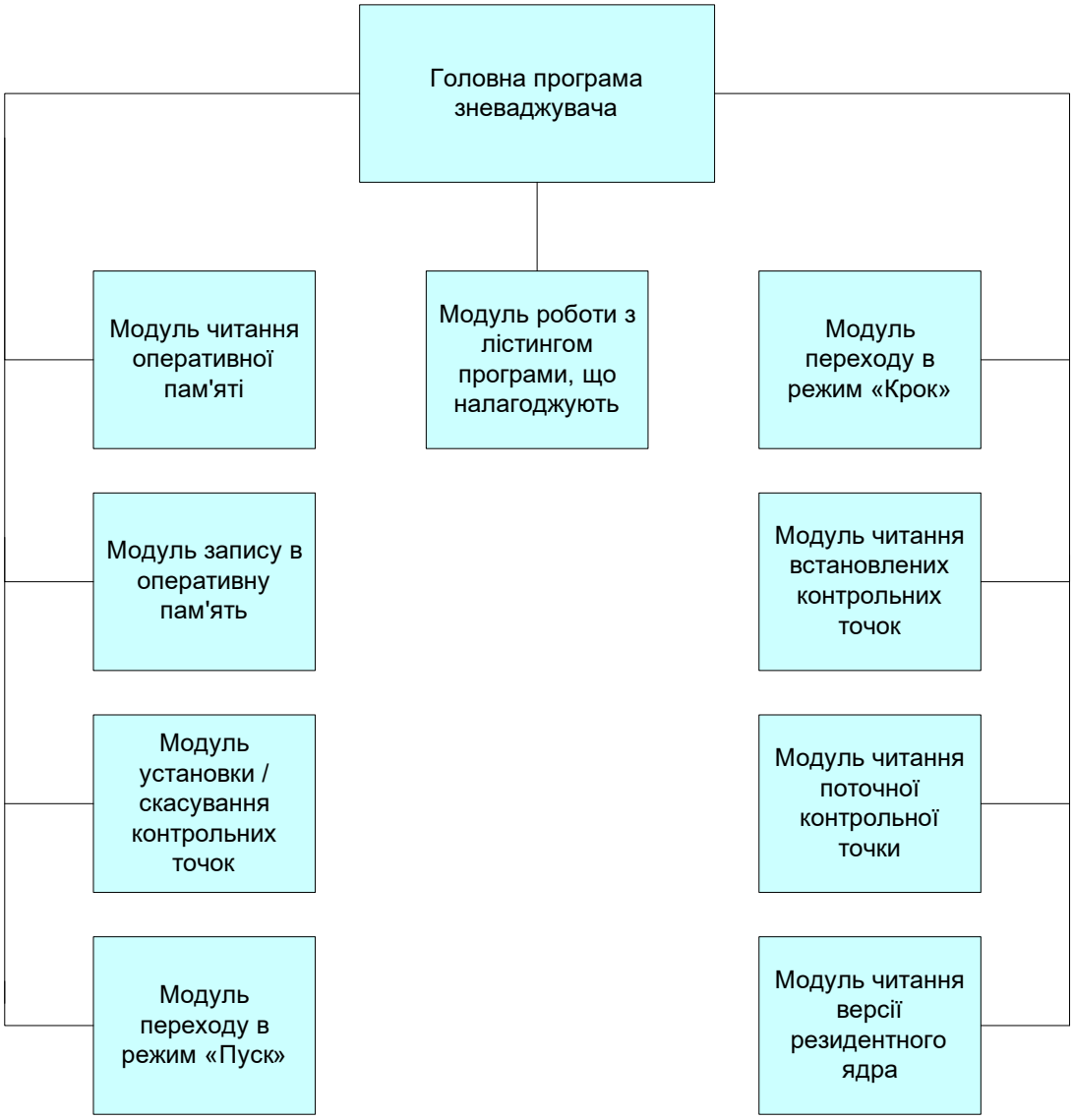


Рис 2.1 – Структура налагоджувача

Інструментальні засоби включають в себе наступні компоненти (більш детально у розділі 4):

- модуль читання оперативної пам’яті;
- модуль запису в оперативну пам’ять;

- модуль установки/скасування контрольних точок;
- модуль переходу в режим «пуск»;
- модуль переходу в режим «крок»;
- модуль читання контрольних точок;
- модуль читання поточної контрольної точки;
- модуль читання версії резидентного ядра.

2.2 Склад модулів резидентного ядра, що забезпечують виконання функцій налагодження

Склад модулів резидентного ядра [10] із зазначенням виконуваних ними функцій наведено в табл. 2.1.

Табл. 2.1 - Склад модулів резидентного ядра, що забезпечують виконання додаткових функцій

Ім'я файлу	Ім'я програми	Виконувані функції
fb_calc.asm, dispfbfo.asm, dispfblf.asm, dispfbpo.asm, dispfbtu.asm	?_DISP_FB_CA, ?_DISP_FB_FO, ?_DISP_FB_LF, ?_DISP_FB_PO, ?_DISP_FB_TU	Диспетчери викликів ФБ
dispalg.asm	???_Disp_alg	Функція диспетчеризації викликів ФБ
readd.asm	??_Ob_oper_read_m	Обробка повідомлення із завданням операцій читання / запису даних
	??_Ob_oper_ver	Обробка повідомлення із завданням операції читання дати складання і версії резидентного ядра
	??_Ob_oper_uokt	Обробка повідомлення із завданням операцій установки / скасування контрольних точок
	??_Ob_oper_go	Обробка повідомлення із завданням операції пуску прикладної програми, починаючи з контрольною точки
	??_Ob_oper_shag	Обробка повідомлення із завданням операції виконання кроку прикладної програми
	??_Ob_oper_spisok	Отримання списку контрольних точок
	??_Ob_oper_tek_adr	Отримання адреси поточної контрольної точки
	??_Ob_oper_takt	Режим «Такт»
	??_Ob_oper_scr	Роботи за сценарієм в режимі «Такт»

Продовження табл. 2.1

Ім'я файлу	Ім'я програми	Виконувані функції
readd.asm	??_Ob_oper_stakt	Отримання стану режиму «Такт»
	??_Ob_oper_iuso	Для встановлення / скасування ознаки імітації введення даних від модулів зв'язку з об'єктом
	??_Ob_oper_imsku	Встановлення/скасування ознаки імітації повідомлень із мережі Ethernet HP
	??_Ob_oper_hh	Встановлення/скасування/отримання стану холостого ходу

2.3 Структури даних, що використовуються засобами підтримки налагодження

2.3.1 Загальна структура повідомлення

Дистанційне налагодження прикладної програми виконується за допомогою обміну повідомленнями інструментальних засобів, що функціонують в ПЕОМ, і програм резидентного ядра, що функціонують в КМп і виконують вказівки, отримані з ПЕОМ.

Обмін інформацією між інструментальними засобами і програмами резидентного ядра виконується по інтерфейсу Ethernet з використанням протоколу мережі Ethernet НУ [11] [12] [13] [14].

Всі повідомлення, адресовані налагоджувальний засобів, супроводжуються ідентифікатором, рівним 1010_2.

Обробка завдання виконується в такий спосіб - аналізується вміст інформаційної частини повідомлення, отриманого з ПЕОМ, формується відповідь повідомлення, що містить результати виконання завдання.

Структура прийнятого для засобів налагодження з ПЕОМ повідомлення показана на рис. 2.2

	7	0
1	Код операції	
2	Код типу операції	
3	Службовий код (лічильник повідомлень)	
4 – *n*	Дані	

Рисунок 2.2- Структура прийнятого для засобів налагодження з ПЕОМ повідомлення

.

Структура переданого в ПЕОМ повідомлення-відповіді показана на рис. 2.3.

	7	0
1	Код операції	
2	Код завершення	
3	Код типу операції	
4	Службовий код	Службовий код
5		Службовий код
6 – *n*	Дані	

Рисунок 2.3- Структура переданого в ПЕОМ повідомлення-відповіді

Код операції визначає функцію, яку повинні виконати налагоджувальні засоби в КМп, і приймає значення, наведені в табл.2.2.

Коди завершення приймають значення:

0 – операція прийнята (формується будь-яким модулем);

2 – дана контрольна точка вже встановлена (модуль установки контрольної точки);

3 – немає вільних елементів в таблиці контрольних точок (модуль установки контрольної точки);

4 – дана контрольна точка не знайдена в таблиці контрольних точок (модуль скасування контрольної точки, модуль пуска с указанного адреса);

5 – контрольні точки не встановлені;

6 – помилка в завданні сценарію;

8 – помилка в параметрах запиту (неприпустимий запит, невірна довжина, формується будь-яким модулем).

Значення специфічних кодів завершення наведені при описі кожної операції.

Табл. 2.2 – Перелік операцій, які забезпечують виконання налагоджувальних функцій

Код операції	Код типу операції	Призначення
0x4	0x1	Читання оперативної пам'яті
0x4	0x3	Читання далекої пам'яті
0x4	0x4	Отримати версію і дату складання резидентного ядра
0x5	0x1	Запис у оперативну пам'ять
0x5	0x3	Запис у далеку пам'ять
0x6	0x1	Читання бітових полів з оперативної пам'яті
0x6	0x3	Читання бітових полів з далекої пам'яті
0x7	0x1	Запис бітових полів в оперативну пам'ять
0x7	0x3	Запис бітових полів в далеку пам'ять
0x10	0x11	Установка контрольної точки при налагодженні
0x10	0x12	Скасування контрольної точки при налагодженні
0x10	0x13	Пуск з адреси при налагодженні
0x10	0x14	Пуск в режимі «Крок» при налагодженні
0x10	0x15	Отримати список контрольних точок
0x10	0x16	Отримати адресу поточної контрольної точки
0x10	0x7	Встановити/зупинити/скасувати/не змінювати режим «Такт»
0x10	0x8	Встановити роботу за сценарієм режиму «Такт»
0x10	0x9	Отримати стан режиму «Такт»
0x10	0xA	Встановити/скасувати ознаку імітації введення даних від модулів МЗО
0x10	0xB	Встановити/скасувати ознаку імітації повідомлень з мережі Ethernet HP
0x10	0xD	Встановити/скасувати холостий хід або отримати стан холостого ходу

2.3.2 Структура таблиці контрольних точок

Для виконання прикладних програм в режимі контрольних точок або в покроковому режимі використовується внутрішня таблиця опису контрольних точок.

Програмні засоби цільового програмування забезпечують виконання наступних типів прикладних програм:

- первинної обробки вхідних аналогових і дискретних сигналів;
- логічних функцій;
- формування розрахункових аналогових сигналів;
- порівняння з уставками вхідних аналогових і розрахункових параметрів;
- формування вихідних аналогових і дискретних сигналів.

Структура таблиці опису контрольних точок показана на рис. 2.4.

Байт	Призначення
	Код списку ФБ
1-2	Код першого у списку ФБ
3-6	Повна адреса поточної контрольної точки (сегмент: зсув)
7-8	Код ФБ, замість якого встановлена поточна контрольна точка
9-12	Повна адреса контрольної точки 1 (сегмент: зсув)
13-14	Код ФБ, замість якого встановлена контрольна точка 1
…	…
n-n+4	Повна адреса контрольної точки k (сегмент: зсув)
n+5-n+6	Повна адреса контрольної точки k (сегмент: зсув)

Рисунок 2.4 - Структура таблиці опису контрольних точок

2.3.3 Структура таблиці зберігання даних в режимі «Такт» з заданим сценарієм

При виконанні прикладних програм в режимі «Такт» з заданим сценарієм для зберігання даних використовується внутрішня таблиця опису сценарію.

Структура внутрішньої таблиці опису сценарію показана на рис. 2.5.

Байт	Призначення
1	Кількість диспетчерів ФБ
2-5	Адреса описателя даних для диспетчера ФБ 1
6-7	Кількість циклів виконання диспетчера
8-9	Кількість виконаних циклів диспетчера дорівнює 0
	…
n-n+9	Адреса описателя даних для диспетчера ФБ з номером n
n-n+13	Кількість циклів виконання диспетчера
n-n+15	Кількість виконаних циклів диспетчера дорівнює 0

Рисунок 2.5- Структура внутрішньої таблиці опису сценарію

Внутрішня таблиця опису сценарію розміщується в оперативній пам'яті за адресою 0x600000.

Структура описувачу даних для диспетчера ФБ показана на рис. 2.6.

Байт	Призначення
1	Тип диспетчера ФБ
2-3	Кількість вхідних параметрів, для яких будуть встановлюватися значення за сценарієм (від 1 до 20000)
4-7	Адреса описувача даних для вхідного параметра 1
	…
m-m+4	Адреса описувача даних для вхідного параметра N

Рисунок 2.6- Структура внутрішньої таблиці опису сценарію

Тип диспетчера ФБ приймає значення, аналогічні описаним в 2.3.2.

Структура описувача даних для вхідного параметра показана на рис. 2.7.

Байт	Призначення
1-4	Повна адреса параметра в ОБД (сегмент: зсув)
5	Тип параметра
6	Кількість завдань
7	Номер поточного завдання дорівнює 0
8-23	Описувач завдання 1
	…
j-j+15	Описувач завдання N

Рисунок 2.7- Структура описувача даних для вхідного параметра

На рис. 2.7 позначення «Тип параметра» визначає розмірність і формат представлення даних в ОБД МСКУ / КМп. Приймає значення, відповідні форматам, описаним в таблиці 2.3:

1 – відповідає типу CHAR;

2 – відповідає типу UNSIGNED_CHAR;

3 – відповідає типу INT;

4 – відповідає типу UNSIGNED_INT;

5 – відповідає типу LONG;

6 – відповідає типу UNSIGNED_LONG;

7 – відповідає типу FLOAT;

8 – відповідає типу ST20;

9 – BOOL;

10 – BITS2.

Табл. 2.3 – Типи і формат представлення даних в ОБД МСКУ/КМп

Найменування типу значення	Позначення типу	Розмір значення (в байтах)	Діапазон значень
Ціле коротке знакове	CHAR	1	Від -128 до 127
Ціле коротке беззнакове	UNSIGNED_CHAR	1	Від 0 до 255
Ціле знакове	INT	2	Від -32768 до 32767
Ціле беззнакове	UNSIGNED_INT	2	Від 0 до 65535
Ціле довге знакове	LONG	4	Від -2^{31} до $2^{31}-1$
Ціле довге беззнакове	UNSIGNED_ LONG	4	Від 0 до $2^{32}-1$
Дійсне	FLOAT	4	Від $-3.4 \cdot 10^{38}$ до $+3.4 \cdot 10^{38}$
Тривалість часу (В одиницях по 20 ms)	ST20	2	Відповідає UNSIGNED_INT
Дискретне	BOOL	1	Від 0 до 1
Двухбітове	BITS2	1	Від 0 до 3

Структура описувача завдання показана на рис. 2.8.

Байт	Призначення
1	Тип завдання
2-3	Задана кількість тактів
4-5	Лічильник поточного такту дорівнює 0
6-9	Початкове значення параметра
10-13	Поточне значення параметра
14-15	Крок зміни

Рисунок 2.8- Структура описувача завдання

На рис. 2.8 позначення «Тип завдання» може приймати значення:

1 – значення вхідного параметра не змінюється протягом заданого кількості тактів;

2 – значення вхідного параметра протягом заданого кількості тактів збільшується на величину кроку;

3 – значення вхідного параметра протягом заданого кількості тактів зменшується на величину кроку;

4 – значення вхідного параметра протягом заданого кількості тактів змінюється за правилом: в парному такті значення вхідного параметра збільшується на величину кроку, а в непарному такті - зменшується на величину кроку.

2.3.4 Операція читання оперативної пам’яті

Структура інформаційної частини повідомлення для операції читання оперативної пам'яті показана на рис. 2.9.

	7 … 0
1	Код запиту (0x4)
2	0x1
3-6	Повна адреса оперативної пам'яті (сегмент, зміщення)
7-8	Довжина, в байтах (1440)

Рисунок 2.9- Структура інформаційної частини повідомлення для операції читання оперативної пам'яті

Структура інформаційної частини повідомлення-відповіді показана на рис. 2.10.

	7 … 0
1	Код запиту (0x4)
2	Код завершення
3	0x1
4-7	Повна адреса оперативної пам'яті (сегмент, зміщення)
8-9	Довжина, в байтах (1440)

Рисунок 2.10- Структура інформаційної частини повідомлення-відповіді

Код завершення може приймати значення, описані в 2.3.1.

В результаті виконання операції формується і видається по мережі Ethernet HP повідомлення, що містить набір даних, прочитаних за адресою, вказаною в повідомленні.

2.3.5 Операція читання далекої пам’яті

Структура інформаційної частини повідомлення для операції читання далекої пам'яті показана на рис. 2.11.

	7 … 0
1	Код запиту (0x4)
2	0x3
3-6	Абсолютна адреса далекої пам'яті
7-8	Довжина, в байтах (1440)

Рисунок 2.11- Структура інформаційної частини повідомлення для операції читання далекої пам'яті

Структура інформаційної частини повідомлення-відповіді показана на рис. 2.12.

	7 … 0
1	Код запиту (0x4)
2	Код завершення
3	0x3
4-7	Абсолютна адреса далекої пам'яті
8-9	Довжина, в байтах (1440)

Рисунок 2.12- Структура інформаційної частини повідомлення-відповіді

Код завершення може приймати значення, описані в 2.3.1.

В результаті виконання операції формується і видається по мережі Ethernet НУ повідомлення, що містить набір даних, прочитаних за адресою, вказаною в повідомленні.

2.3.6 Операція отримання версії і дати складання резидентного ядра

Структура інформаційної частини повідомлення для операції отримання версії і дати складання резидентного ядра показана на рис. 2.13.

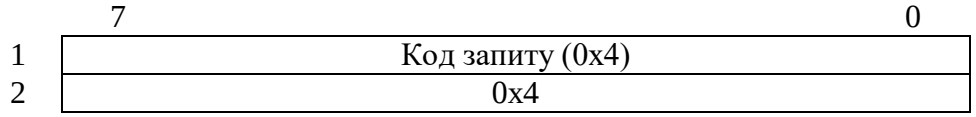


Рисунок 2.13- Структура інформаційної частини повідомлення для операції отримання версії і дати складання резидентного ядра

Структура інформаційної частини повідомлення-відповіді показана на рис. 2.14.

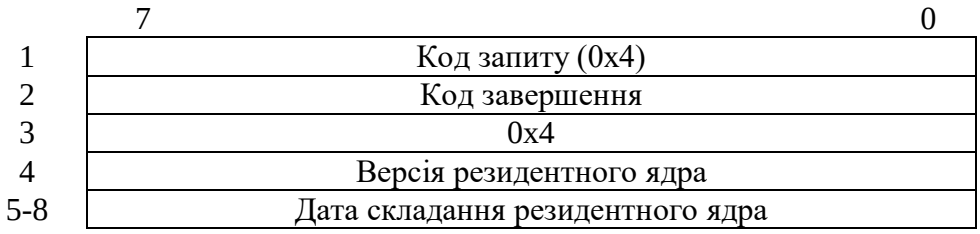


Рисунок 2.14- Структура інформаційної частини повідомлення-відповіді

Код завершення може приймати значення описані в 2.3.1.

При виконанні операції формується і видається по мережі Ethernet НР повідомлення, що містить версію резидентного ядра.

2.3.7 Операція запису в оперативну пам'ять

Структура інформаційної частини повідомлення для операції запису в оперативну пам'ять показана на малюнку 2.15.

	7	0
1	Код запиту (0x5)	
2	0x1	
3-6	Повна адреса оперативної пам'яті	
7-8	(сегмент, зміщення)	
9-n	Довжина, в байтах (1440)	

Рисунок 2.15- Структура інформаційної частини повідомлення для операції запису в оперативну пам'ять

Структура інформаційної частини повідомлення-відповіді показана на рис. 2.16.

	7	0
1	Код запиту (0x5)	
2	Код завершення	
3	0x1	
4-7	Повна адреса оперативної пам'яті	
8-9	(сегмент, зміщення)	

Рисунок 2.16- Структура інформаційної частини повідомлення-відповіді

Код завершення може приймати значення, описані в 2.3.1.

При виконанні операції за адресою, вказаною в прийнятому повідомленні, записуються дані з отриманого повідомлення, а потім формується і видається по мережі Ethernet HP повідомлення з кодом завершення операції.

2.3.8 Операція запису в далеку пам'ять

Структура інформаційної частини повідомлення для операції запису в далеку пам'ять показана на рис. 2.17.

	7	0
1	Код запиту (0x5)	
2	0x3	
3-6	Абсолютна адреса далекої пам'яті	
7-8	Довжина, в байтах (1440)	
9-n	Дані для запису	

Рисунок 2.17- Структура інформаційної частини повідомлення для операції запису в далеку пам'ять

Структура інформаційної частини повідомлення-відповіді показана на рис. 2.18

	7 … 0
1	Код запиту (0x5)
2	Код завершення
3	0x3
4-7	Абсолютна адреса далекої пам'яті
8-9	Довжина, в байтах (1440)

Рисунок 2.18- Структура інформаційної частини повідомлення-відповіді

Код завершення може приймати значення, описані в 2.3.1.

При виконанні операції за адресою, вказаною в прийнятому повідомленні, записуються дані з отриманого повідомлення, а потім формується і видається по мережі Ethernet HP повідомлення з кодом завершення операції.

2.3.9 Операція читання бітових полів оперативної пам'яті

Структура інформаційної частини повідомлення для операції читання оперативної пам'яті показана на рис. 2.19.

	7 … 0
1	Код запиту (0x6)
2	0x1
3	n - кількість бітових полів для читання
4-7	Повна адреса (сегмент, зміщення) оперативної пам'яті першого бітового поля
8	Зсув першого бітового поля (в бітах 0-31)
9	Розмір першого бітового поля (в бітах 1-32)
…	...
3+6·(*n*-1)	Повна адреса (сегмент, зміщення) оперативної пам'яті бітового поля з номером n
8+6·(*n*-1)	Зсув бітового поля з номером n (в бітах 0-31)
9+6·(*n*-1)	Розмір n бітового поля з номером n (в бітах 1-32)

Рисунок 2.19- Структура інформаційної частини повідомлення для операції читання оперативної пам'яті

Структура інформаційної частини повідомлення-відповіді показана на рис. 2.20.

	7 … 0
1	Код запиту (0x6)
2	Код завершення
3	0x1
4	n - кількість прочитаних бітових полів
5-8	Повна адреса (сегмент, зміщення) оперативної пам'яті першого бітового поля
9	Зсув першого бітового поля (в бітах 0-31)
10	Розмір першого бітового поля (в бітах 1-32)
11-14	Значення бітового поля
…	...
4+10·(*n*-1)	Повна адреса (сегмент, зміщення) оперативної пам'яті бітового поля з номером n
9+10·(*n*-1)	Зсув бітового поля з номером n (в бітах 0-31)
10+10·(*n*-1)	Розмір бітового поля з номером n (в бітах 1-32)
11+10·(*n*-1)	Значення бітового поля

Рисунок 2.20- Структура інформаційної частини повідомлення-відповіді

Код завершення може приймати значення, описані в 2.3.1.

В результаті виконання операції формується і видається по мережі Ethernet HP повідомлення, що містить набір даних, прочитаних за адресою, вказаною в повідомленні.

2.3.10 Операція читання бітових полів далекої пам'яті

Структура інформаційної частини повідомлення для операції читання оперативної пам'яті показана на рис. 2.21.

	7 … 0
1	Код запиту (0x6)
2	0x3
3	n - кількість бітових полів для читання
4-7	Абсолютний адреса оперативної пам'яті першого бітового поля
8	Зсув першого бітового поля (в бітах 0-31)
9	Розмір першого бітового поля (в бітах 1-32)
…	...
4+6·(*n*-1)	Абсолютний адреса оперативної пам'яті бітового поля з номером n
8+6·(*n*-1)	Зсув бітового поля з номером n (в бітах 0-31)
9+6·(*n*-1)	Розмір бітового поля з номером n (в бітах 1-32)

Рисунок 2.21- Структура інформаційної частини повідомлення для операції читання оперативної пам'яті

Структура інформаційної частини повідомлення-відповіді показана на рис. 2.22.

	7 … 0
1	Код запиту (0x6)
2	Код завершення
3	0x3
4	n - кількість прочитаних бітових полів
5-8	Абсолютна адреса оперативної пам'яті першого бітового поля
9	Зсув першого бітового поля (в бітах 0-31)
10	Розмір першого бітового поля (в бітах 1-32)
11-14	Значення бітового поля
…	...
5+10·(*n*-1)	Абсолютний адреса оперативної пам'яті бітового поля з номером n
9+10·(*n*-1)	Зсув бітового поля з номером n (в бітах 0-31)
10+10·(*n*-1)	Розмір бітового поля з номером n (в бітах 1-32)
11+10·(*n*-1)	Значення бітового поля

Рисунок 2.22- Структура інформаційної частини повідомлення-відповіді

Код завершення може приймати значення, описані в 2.3.1.

В результаті виконання операції формується і видається по мережі Ethernet HP повідомлення, що містить набір даних, прочитаних за адресою, вказаною в повідомленні.

2.3.11 Операція запису бітових полів в оперативну пам'ять

Структура інформаційної частини повідомлення для операції запису бітових полів в оперативну пам'ять показана на рис. 2.23.

	7 … 0
1	Код запиту (0x7)
2	0x1
3	n - кількість бітових полів для запису
4-7	Повна адреса (сегмент, зміщення) оперативної пам'яті першого бітового поля
8	Зсув першого бітового поля (в бітах 0-31)
9	Розмір першого бітового поля (в бітах 1-32)
10-13	Значення першого бітового поля
…	...
4+6·(*n*-1)	Повна адреса (сегмент, зміщення) оперативної пам'яті бітового поля з номером n
8+6·(*n*-1)	Зсув бітового поля з номером n (в бітах 0-31)
9+6·(*n*-1)	Розмір бітового поля з номером n (в бітах 1-32)
10+6·(*n*-1)	Значення бітового поля з номером n

Рисунок 2.23- Структура інформаційної частини повідомлення для операції запису бітових полів в оперативну пам'ять

Структура інформаційної частини повідомлення-відповіді показана на рис. 2.24.

	7 … 0
1	Код запиту (0x7)
2	код завершення
3	0x1
4	n - кількість записаних бітових полів
5-8	Повна адреса (сегмент, зміщення) оперативної пам'яті першого бітового поля
9	Зсув першого бітового поля (в бітах 0-31)
10	Розмір першого бітового поля (в бітах 1-32)
11-14	Значення бітового поля
…	...
5+10·(*n*-1)	Повна адреса (сегмент, зміщення) оперативної пам'яті бітового поля з номером n
9+10·(*n*-1)	Зсув бітового поля з номером n (в бітах 0-31)
10+10·(*n*-1)	Розмір бітового поля з номером n (в бітах 1-32)
11+10·(*n*-1)	Значення бітового поля

Рисунок 2.24- Структура інформаційної частини повідомлення-відповіді

Код завершення може приймати значення, описані в 2.3.1.

При виконанні операції за адресами, вказаними в прийнятому повідомленні, записуються дані з отриманого повідомлення, а потім формується і видається по мережі Ethernet HP повідомлення з кодом завершення операції.

2.3.12 Операція запису бітових полів в далеку пам'ять

Структура інформаційної частини повідомлення для операції запису бітових полів в далеку пам'ять показана на рис. 2.25.

	7 … 0
1	Код запиту (0x7)
2	0x3
3	n - кількість бітових полів для запису
4-7	Абсолютна адреса оперативної пам'яті першого бітового поля
8	Зсув першого бітового поля (в бітах 0-31)
9	Розмір першого бітового поля (в бітах 1-32)
10-13	Значення першого бітового поля
…	...
4+6·(*n*-1)	Абсолютна адреса оперативної пам'яті бітового поля з номером n
8+6·(*n*-1)	Зсув бітового поля з номером n (в бітах 0-31)
9+6·(*n*-1)	Розмір бітового поля з номером n (в бітах 1-32)
10+6·(*n*-1)	Значення бітового поля з номером n

Рисунок 2.25- Структура інформаційної частини повідомлення для операції запису бітових полів в далеку пам'ять

Структура інформаційної частини повідомлення-відповіді показана на рис. 2.26.

	7 … 0
1	Код запиту (0x7)
2	код завершення
3	0x3
4	n - кількість записаних бітових полів
5-8	Абсолютний адреса оперативної пам'яті першого бітового поля
9	Зсув першого бітового поля (в бітах 0-31)
10	Розмір першого бітового поля (в бітах 1-32)
11-14	Значення бітового поля
…	...
5+10·(*n*-1)	Абсолютний адреса оперативної пам'яті бітового поля з номером n
9+10·(*n*-1)	Зсув бітового поля з номером n (в бітах 0-31)
10+10·(*n*-1)	Розмір бітового поля з номером n (в бітах 1-32)
11+10·(*n*-1)	Значення бітового поля

Рисунок 2.26- Структура інформаційної частини повідомлення-відповіді

Код завершення може приймати значення, описані в 2.3.1.

При виконанні операції за адресами, вказаними в прийнятому повідомленні, записуються дані з отриманого повідомлення, а потім формується і видається по мережі Ethernet HP повідомлення з кодом завершення операції.

2.3.13 Операція установки контрольної точки

Структура інформаційної частини повідомлення для операції установки контрольної точки показана на рис. 2.27.

	7 … 0
1	Код запиту (0x10)
2	0x11
3	Код типу ФБ
4-7	Повна адреса контрольної точки

Рисунок 2.27- Структура інформаційної частини повідомлення для операції установки контрольної точки

Структура інформаційної частини повідомлення-відповіді показана на рис. 2.28.

	7 … 0
1	Код запиту (0x10)
2	Код завершення
3	0x11
4	Код типу ФБ
5-8	Повна адреса контрольної точки (зміщення, сегмент коду виклику ФБ)

Рисунок 2.28- Структура інформаційної частини повідомлення-відповіді

Код завершення може приймати значення, описані в 2.3.1.

При виконанні операції виділяється вільний елемент в таблиці контрольних точок і формується відповідно до структури, показаної на малюнку 9.

2.3.14 Операція скасування контрольної точки

Структура інформаційної частини повідомлення для операції скасування контрольної точки показана на рис. 2.29.

	7 … 0
1	Код запиту (0x10)
2	0x12
3	Код типу ФБ
4-7	Повна адреса контрольної точки (зміщення, сегмент коду виклику ФБ)

Рисунок 2.29- Структура інформаційної частини повідомлення для операції скасування контрольної точки

Структура інформаційної частини повідомлення-відповіді показана на рис. 2.30.

	7 … 0
1	Код запиту (0x10)
2	Код завершення
3	0x12
4	Код типу ФБ
5-8	Повна адреса контрольної точки (зміщення, сегмент коду виклику ФБ)

Рисунок 2.30- Структура інформаційної частини повідомлення-відповіді

Код завершення може приймати значення, описані в 2.3.1.

При виконанні операції з таблиці контрольних точок видаляється елемент з вказаною контрольної точкою, а на її адресу відновлюється вихідний код ФБ, збережений в елементі таблиці при установці цієї контрольної точки.

2.3.15 Операція пуску із зазначеної адреси

Структура інформаційної частини повідомлення для операції пуску із зазначеної адреси показана на рис. 2.31.

	7 … 0
1	Код запиту (0x10)
2	0x13
3	Код типу ФБ
4-7	Повна адреса пуску

Рисунок 2.31- Структура інформаційної частини повідомлення для операції пуску із зазначеної адреси

Структура інформаційної частини повідомлення-відповіді показана на рис. 2.32.

	7 … 0
1	Код запиту (0x10)
2	Код завершення
3	0x13
4	Код типу ФБ
5-8	Повна адреса пуску

Рисунок 2.32- Структура інформаційної частини повідомлення-відповіді

Код завершення може приймати значення, описані в 2.3.1.

При виконанні цієї операції встановлюється ознака режиму пуску і в робочій області запам'ятовується адресу пуску.

2.3.16 Операція пуску в режимі «Крок»

Структура інформаційної частини повідомлення для операції пуску в режимі «Крок» показана на рис. 2.33.

	7 … 0
1	Код запиту (0x10)
2	0x14
3	Код типу ФБ
4-7	Повна адреса (зсув, сегмент) поточного виклику ФБ)

Рисунок 2.33- Структура інформаційної частини повідомлення для операції пуску в режимі «Крок»

Структура інформаційної частини повідомлення-відповіді показана на рис. 2.34.

	7 … 0
1	Код запиту (0x10)
2	код завершення
3	0x14
4	Код типу ФБ
5-8	Повна адреса (зсув, сегмент) поточного виклику ФБ

Рисунок 2.34- Структура інформаційної частини повідомлення-відповіді

Код завершення може приймати значення, описані в 2.3.1.

При виконанні цієї операції встановлюється ознака режиму «Крок» і в робочій області запам'ятовується адреса поточного і наступного викликів ФБ.

2.3.17 Операція отримання списку контрольних точок

Структура інформаційної частини повідомлення для операції отримання списку контрольних точок показана на рис. 2.35.

	7	0
1	Код запиту (0x10)	
2	0x15	
3	Код типу ФБ	

Рисунок 2.35- Структура інформаційної частини повідомлення для операції отримання списку контрольних точок

Структура інформаційної частини повідомлення-відповіді показана на рис. 2.36.

	7	0
1	Код запиту (0x10)	
2	Код завершення	
3	0x15	
4	Код типу ФБ	
5	Кількість контрольних точок	
6-9	Повна адреса контрольної точки 1 (зміщення, сегмент)	
N	. . .	

Рисунок 2.36- Структура інформаційної частини повідомлення-відповіді

Код завершення може приймати значення, описані в 2.3.1.

При виконанні операції формується і видається по мережі Ethernet HP повідомлення, що містить дані з таблиці контрольних точок.

2.3.18 Операція отримання адреси поточної контрольної точки

Структура інформаційної частини повідомлення для операції отримання адреси поточної контрольної точки показана на рис. 2.37.

	7 … 0
1	Код запиту (0x10)
2	0x16
3	Код типу ФБ

Рисунок 2.37- Структура інформаційної частини повідомлення для операції отримання адреси поточної контрольної точки

Структура інформаційної частини повідомлення-відповіді показана на рис. 2.38.

	7 … 0
1	Код запиту (0x10)
2	код завершення
3	0x16
4	Код типу ФБ
5-8	Повна адреса поточної контрольної точки

Рисунок 2.38- Структура інформаційної частини повідомлення-відповіді

Код завершення може приймати значення, описані в 2.3.1.

При виконанні операції формується і видається по мережі Ethernet HP повідомлення, що містить адресу поточної контрольної точки.

2.3.19 Операції режиму «Такт»

Структура інформаційної частини повідомлення для виконання операцій режиму «Такт» показана на рис. 2.39.

	7 … 0
1	Код запиту (0x10)
2	0x7
3	Код типу ФБ 1
4	Операція режиму «Такт»
5-6	Кількість тактів
7	Код типу ФБ 2
8	Операція режиму «Такт»
9-10	Кількість тактів
11	Код типу ФБ 3
12	Операція режиму «Такт»
13-14	Кількість тактів
15	Код типу ФБ 4
16	Операція режиму «Такт»
17-18	кількість тактів
19	Код типу ФБ 5
20	Операція режиму «Такт»
21-22	Кількість тактів
23	Код типу ФБ 0xFF - встановити режим у всіх списках
24	Операція режиму «Такт»
25-26	Кількість тактів

Рисунок 2.39- Структура інформаційної частини повідомлення для виконання операцій режиму «Такт»

Структура інформаційної частини повідомлення-відповіді показана на рис. 2.40.

	7 … 0
1	Код запиту (0x10)
2	Код завершення
3	0x7
4	Код типу ФБ 1
5	Операція режиму «Такт»
6-7	Кількість тактів
8	Код типу ФБ 2
9	Операція режиму «Такт»
10-11	Кількість тактів
12	Код типу ФБ 3
13	Операція режиму «Такт»
14-15	Кількість тактів
16	Код типу ФБ 4
17	Операція режиму «Такт»
18-19	Кількість тактів
20	Код типу ФБ 5
21	Операція режиму «Такт»
22-23	Кількість тактів
24	Код типу ФБ 0xff - операція режиму для всіх ФБ
25	Операція режиму «Такт»
26-27	Кількість тактів

Рисунок 2.40- Структура інформаційної частини повідомлення-відповіді

Код завершення може приймати значення, описані в 2.3.1.

Операція режиму «Такт» може набувати таких значень:

0 – скасувати режим «Такт»;

1 – встановити режим «Такт»;

2 – зупинити режим «Такт»;

3 – не змінювати режим «Такт».

При виконанні операції відпрацьовується алгоритм роботи одного із заданих режимів «Такт» в даній операції.

2.3.20 Операція установки роботи за сценарієм в режимі «Такт»

Структура інформаційної частини повідомлення для операції установки роботи за сценарієм в режимі «Такт» показана на рис. 2.41.

	7	0
1	Код запиту (0x10)	
2	0x8	

Рисунок 2.41- Структура інформаційної частини повідомлення для операції установки роботи за сценарієм в режимі «Такт»

Структура інформаційної частини повідомлення-відповіді показана на рис. 2.42.

	7	0
1	Код запиту (0x10)	
2	Код завершення	
3	0x8	

Рисунок 2.42- Структура інформаційної частини повідомлення-відповіді

Код завершення може приймати значення, описані в 2.3.1.

При виконанні операції відпрацьовується алгоритм роботи, заданий в сценарії, і завдання записано в далеку пам'ять за адресою 0x600000.Структура завдання описана в 2.3.3.

2.3.21 Операція отримання стану режиму «Такт»

Структура інформаційної частини повідомлення для операції отримання стану режиму «Такт» показана на рис. 2.43.

	7	0
1	Код запиту (0x10)	
2	0x9	

Рисунок 2.43- Структура інформаційної частини повідомлення для операції отримання стану режиму «Такт»

Структура інформаційної частини повідомлення-відповіді показана на рис. 2.44.

	7 … 0
1	Код запиту (0x10)
2	Код завершення
3	0x9
4	Код типу ФБ 1
5	Код стану режиму «Такт»
6-7	Залишок тактів
8	Код типу ФБ 2
9	Код стану режиму «Такт»
10-11	Залишок тактів
12	Код типу ФБ 3
13	Код стану режиму «Такт»
14-15	Залишок тактів
16	Код типу ФБ 4
17	Код стану режиму «Такт»
18-19	Залишок тактів
20	Код типу ФБ 5
21	Код стану режиму «Такт»
22-23	Залишок тактів
24	Код типу ФБ 0xff
25	Код стану режиму «Такт»
26-27	Залишок тактів

Рисунок 2.44- Структура інформаційної частини повідомлення-відповіді

Код завершення може приймати значення, описані в 2.3.1.

Код стану режиму «Такт» може набувати таких значень:

0 – не заданий режим «Такт»;

1 – виконується режим «Такт»;

2 – зупинений режим «Такт»;

3 – завершено режим «Такт».

При виконанні операції формується стан відпрацювання режиму «Такт» і видається по мережі Ethernet HP повідомлення.

2.3.22 Операція установки / скасування ознаки імітації введення даних від МЗО

Структура інформаційної частини повідомлення для операції установки / скасування ознаки імітації введення даних від МЗО показана на рис. 2.45.

	7	0
1	Код запиту (0x10)	
2	0xA	
3	Режим	

Рисунок 2.45- Структура інформаційної частини повідомлення для операції установки / скасування ознаки імітації введення даних від МЗО

Структура інформаційної частини повідомлення-відповіді показана на рис. 2.46.

	7	0
1	Код запиту (0x10)	
2	Код завершення	
3	0xA	
4	Режим	

Рисунок 2.46- Структура інформаційної частини повідомлення-відповіді

Код завершення може приймати значення, описані в 2.3.1.

Код режиму може набувати таких значень:

0 – скасувати ознаку імітації введення даних від МЗО;

1 – встановити ознаку імітації введення даних від МЗО.

При виконанні операції встановлюється або скасовується ознака імітації введення даних від МЗО.

2.3.23 Операція установки / скасування ознаки імітації повідомлень з мережі Ethernet HP

Структура інформаційної частини повідомлення для операції установки / скасування ознаки імітації повідомлень з мережі Ethernet HP показана на рис. 2.47.

	7 … 0
1	Код запиту (0x10)
2	0xB
3	Режим

Рисунок 2.47- Структура інформаційної частини повідомлення для операції установки / скасування ознаки імітації повідомлень з мережі Ethernet HP

Структура інформаційної частини повідомлення-відповіді показана на рис. 2.48.

	7 … 0
1	Код запиту (0x10)
2	Код завершення
3	0xB
4	Режим

Рисунок 2.48- Структура інформаційної частини повідомлення-відповіді

Код завершення може приймати значення, описані в 2.3.1.

Код режиму може набувати таких значень:

0 – скасувати ознаку імітації повідомлень з мережі Ethernet HP;

1 – встановити ознаку імітації повідомлень з мережі Ethernet HP.

При виконанні операції встановлюється або скасовується ознака імітації повідомлень з мережі Ethernet HP.

2.3.24 Операція установки / скасування / отримання стану холостого ходу

Структура інформаційної частини повідомлення для операції установки/скасування/отримання стану холостого ходу показана на рис. 2.49.

	7	0
1	Код запиту (0x10)	
2	0xD	
3	Режим	

Рисунок 2.49- Структура інформаційної частини повідомлення для операції установки/скасування/отримання стану холостого ходу

Структура інформаційної частини повідомлення-відповіді показана на рис. 2.50.

	7	0
1	Код запиту (0x10)	
2	Код завершення	
3	0xD	
4	Режим	
5	0 – холостий хід не встановлено; 1 – холостий хід встановлено	

Рисунок 2.50- Структура інформаційної частини повідомлення-відповіді

Код завершення може приймати значення, описані в 2.3.1.

Код режиму може набувати таких значень:

0 – скасувати холостий хід;

1 – встановити холостий хід;

2 – отримати стан холостого ходу.

При виконанні операції встановлюється або скасовується ознака відпрацювання холостого ходу і по операції «отримати стан холостого ходу» у відповіді передається стан холостого ходу.

Висновок до розділу 2

У данному розділі огянуто логічну структуру інструментальних засобів налагодження та структури даних, що використовуються засобами підтримки налагодження.

3 РОЗРОБКА ПРОГРАМНИХ ЗАСОБІВ НАЛАГОДЖЕННЯ ПРОГРАМ КС МСКУ

3.1 Опис програм

3.1.1 Загальні відомості

Монітор призначений для забезпечення роботи комплекту програмних модулів, що викликаються в міру необхідності виконання обраної операції.

Призначена для забезпечення роботи з контролером МСКУ в режимі налагодження.

Програма представляє собою набір взаємозв'язаних модулів, перелік яких наведено в табл.3.1.

Табл.3.1 – Список програм ІЗН і їх функціональне призначення

Ім'я файлу	Ім'я програми	Виконувані функції
otl4: main.cpp, mainwindow.ui.h	main(), selectButton_clicked(), selectButton_clickedup(), changeData(), debugPrg()	Головна програма ІЗН, що виконує: -створення потоку для прийому повідомлень з МСКУ; -створення потоку для прийому повідомлень з УП ОД; -виклик основного вікна програми; -виклик програмних модулів для виконання виконання функцій налагодження прикладних програм
fetherfile.cpp, fetherfile.h	openEth(), recvEth(), sendMesEth(), closeEth(), readMemEth()	Функції роботи з МСКУ по мережі Ethernet HP
fetherfileup.cpp, fetherfileup.h	openEthup(), recvEthup(), sendMesEthup(), closeEthup(), readMemEthup()	Функції роботи з КП ОД по мережі Ethernet HP

Продовження Табл. 3.1

Ім'я файлу	Ім'я програми	Виконувані функції
simForm.ui.h	procFiles(), KTButton_clicked(), stepButton_clicked(), DbgTimerDone(), VarChg()	Функції режиму налагодження по контрольним точкам
taktform.ui.h	goQuery(), stopQuery(), cancelQuery(), getSostQuery(), setNewTakt(), getVars(), taktForm::scrMode(), taktForm::ScnLoad(), taktForm::ScnNextTakt(), taktForm::CreateScoInit()	Функції режиму «Такт»
schload.cpp, schload.h	main(), VarAddrGet(), ParamValGet(), readMapFile(), procXmpFile(), ScnFile::Save(), ScnFile::Load()	Функції формування бінарного файлу сценарію
chnk32.c, chnk32.h	ChkWrite(), ChkPass(), ChkRead()	
scnedit: scnedit.ui.h	XmpLoad, ProjectDirSet, VarInputAdd, VarOutputAdd, fileNew, fileOpen, fileSave	Програма редагування сценаріїв
testfb: scnverif.c	scnMode,scnLoad, CompareProto	Програма виконання сценарію налагодження й формування звіту порівняння протоколу налагодження з еталоном
prt_diff: prt_diff.c	main	Програма порівняння протоколу результату тестування функціональних блоків із зразками

3.1.2 Опис програми main ()

Функціональне призначення

Програма призначена для виконання наступних дій:

- створення і ініціалізації програми ІЗН;

- виклику програмних модулів для виконання функцій налагодження прикладних програм;

- завершення виконання.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Створення програми ІЗН.

Крок 2

Ініціалізація змінних.

Крок 3

Створення, виведення на екран головної форми.

Вхідні дані

Вхідні дані відсутні.

Вихідні дані

Вихідними даними є основне вікно, призначене для надання оператору можливості вибору режиму роботи.

3.1.3 Опис функції вибору МСКУ selectButton_clicked ()

Функціональне призначення

Функція призначена для визначення адреси МСКУ й вибору проекту для подальшої роботи.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Виклик діалогового вікна для введення адреси МСКУ.

Крок 2

Виклик діалогового вікна для вибору проекту.

Крок 3

Відкриття точки доступу до МСКУ.

Крок 4

Запуск процесу прийому повідомлень із МСКУ.

Вхідні дані

Вхідні дані відсутні.

Вихідні дані

Вихідними даними є адреса МСКУ і повний шлях до проекту.

3.1.4 Опис функції вибору КП ОД selectbutton_clickedup()

Функціональне призначення

Функція призначена для визначення адреси КП ОД і вибору проекту для подальшої роботи.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Виклик діалогового вікна для введення адреси ЕП ОД.

Крок 2

Виклик діалогового вікна для вибору проекту.

Крок 3

Відкриття точки доступу до КП ОД.

Крок 4

Запуск процесу прийому повідомлень з КП ОД.

Вхідні дані

Вхідні дані відсутні.

Вихідні дані

Вихідними даними є адреса КП ОД і повний шлях до проекту.

3.1.5 Зміни даних changeData()

Функціональне призначення

Функція призначена для переходу в режим оперативної зміни даних.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Виклик діалогового вікна для вибору проекту ОБД/ШЛЮЗ.

Крок 2

Розбір файлу-дескриптора оперативно змінюваних даних.

Крок 3

Відображення вікна, що містить список груп параметрів.

Вхідні дані

Вхідними даними є файл-дескриптор оперативно змінюваних даних.

Вихідні дані

Вихідними даними є діалогове вікно, призначене для роботи з оперативно змінними даними.

3.1.6 Опис функції вибору режиму оперативного зміни даних changedata()

Функціональне призначення

Функція призначена для переходу в режим оперативної зміни даних.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Виклик діалогового вікна для вибору проекту ОБД/ШЛЮЗ.

Крок 2

Розбір файлу-дескриптора оперативно змінюваних даних.

Крок 3

Відображення вікна, що містить список груп параметрів.

Вхідні дані

Вхідними даними є файл-дескриптор оперативно змінюваних даних.

Вихідні дані

Вихідними даними є діалогове вікно, призначене для роботи з оперативно змінними даними.

3.1.7 Опис функції вибору режиму відладки debugPrg()

Функціональне призначення

Функція призначена для переходу в режим налагодження.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Виклик діалогового вікна для вибору режиму «Такт» або налагодження в режимі контрольних точок.

Крок 2

Якщо обраний режим «Такт», виконується відображення вікна для роботи в режимі «Такт», в іншому випадку відображається вікно для роботи в режимі налагодження по контрольним точкам.

Вхідні дані

Вхідними даними є повний шлях до проекту, визначений на етапі вибору МСКУ.

Вихідні дані

Вихідними даними є вікно, призначене для роботи в режимі налагодження.

3.1.8 Опис функції відкриття точки доступу до МСКУ openEth()

Функціональне призначення

Функція призначена для відкриття точки доступу до МСКУ.

Опис алгоритму

Алгоритм являє собою виклик функції отримання доступу до мережевого сервісу.

Вхідні дані

Вхідними даними є адреса МСКУ.

Вихідні дані

Вихідними даними є дескриптор мережевого інтерфейсу або значення, що дорівнює -1 у разі виникнення помилки при виконанні операції.

3.1.9 Опис функції прийняття повідомлень з МСКУ recvEth()

Функціональне призначення

Функція призначена для прийняття повідомлень із МСКУ.

Опис алгоритму

Алгоритм являє собою виклик функції прийняття повідомлення з мережі разом із записом адреси відправника.

Вхідні дані

Вхідними даними є дескриптор мережевого інтерфейсу.

Вихідні дані

Вихідними даними є кількість прийнятих байтів або значення, що дорівнює -1 у разі виникнення помилки при виконанні операції.

3.1.10 Опис функції видачі повідомлення в МСКУ sendMesEth()

Функціональне призначення

Функція призначена для видачі даних в МСКУ.

Опис алгоритму

Алгоритм являє собою виклик функції видачі повідомлення в мережу.

Вхідні дані

Вхідними даними є дескриптор мережевого інтерфейсу.

Вихідні дані

Вихідними даними є значення 0 в разі успішного завершення операції або значення, що дорівнює -1 у разі виникнення помилки при виконанні операції.

3.1.11 Опис функції закриття доступу до МСКУ closeEth()

Функціональне призначення

Функція призначена для закриття точки доступу до мережевого інтерфейсу.

Опис алгоритму

Алгоритм являє собою виклик функції закриття точки доступу до мережевого інтерфейсу.

Вхідні дані

Вхідними даними є дескриптор мережевого інтерфейсу.

Вихідні дані

Вихідними даними є значення 0 в разі успішного завершення операції або значення, що дорівнює -1 у разі виникнення помилки при виконанні операції.

3.1.12 Опис функції читання даних з оперативної пам'яті МСКУ readMemEth()

Функціональне призначення

Функція призначена для читання заданої кількості байтів з оперативної пам'яті МСКУ за вказаною адресою.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Видача запиту читання пам'яті.

Крок 2

Очікування відповіді на електронний лист і перевірка прийнятого повідомлення на відповідність заданим у запиті ідентифікатором повідомлення, коду запиту й адресою в оперативній пам'яті.

Вхідні дані

Вхідними даними є дескриптор мережевого інтерфейсу, сформований буфер запиту читання оперативної пам'яті.

Вихідні дані

Вихідними даними є буфер відповідного повідомлення або значення, що дорівнює -1 у разі виникнення помилки при виконанні операції.

3.1.13 Опис функції відкриття точки доступу до КП ОД openethup()

Функціональне призначення

Функція призначена для відкриття точки доступу до КП ОД.

Опис алгоритму

Алгоритм являє собою виклик функції отримання доступу до мережевого сервісу.

Вхідні дані

Вхідними даними є адреса КП ОД.

Вихідні дані

Вихідними даними є дескриптор мережевого інтерфейсу або значення, що дорівнює -1 у разі виникнення помилки при виконанні операції.

3.1.14 Опис функції прийому повідомлень з КП ОД recvEthup()

Функціональне призначення

Функція призначена для прийому повідомлень з КП ОД

Опис алгоритму

Алгоритм являє собою виклик функції прийому повідомлення з мережі із записом адреси відправника.

Вхідні дані

Вхідними даними є дескриптор мережевого інтерфейсу.

Вихідні дані

Вихідними даними є кількість прийнятих байтів або значення, що дорівнює -1 у разі виникнення помилки при виконанні операції.

3.1.15 Опис функції видачі повідомлення в КП ОД sendMesEthup()

Функціональне призначення

Функція призначена для видачі даних в КП ОД.

Опис алгоритму

Алгоритм являє собою виклик функції видачі повідомлення в мережу.

Вхідні дані

Вхідними даними є дескриптор мережевого інтерфейсу.

Вихідні дані

Вихідними даними є значення 0 в разі успішного завершення операції або значення, що дорівнює -1 у разі виникнення помилки при виконанні операції.

3.1.16 Опис функції закриття доступу до КП ОД closeethup()

Функціональне призначення

Функція призначена для закриття точки доступу до мережевого інтерфейсу.

Опис алгоритму

Алгоритм являє собою виклик функції закриття точки доступу до мережевого інтерфейсу.

Вхідні дані

Вхідними даними є дескриптор мережевого інтерфейсу.

Вихідні дані

Вихідними даними є значення 0 в разі успішного завершення операції або значення, що дорівнює -1 у разі виникнення помилки при виконанні операції.

3.1.17 Опис функції читання даних з оперативної пам'яті КП ОД readMemEthup()

Функціональне призначення

Функція призначена для читання заданої кількості байтів із оперативної пам'яті МСКУ за вказаною адресою.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Видача запиту читання пам'яті.

Крок 2

Очікування відповідв на електронний лист і перевірка прийнятого повідомлення на відповідність заданим у запиті ідентифікатором повідомлення, коду запиту й адресою в оперативній пам'яті.

Вхідні дані

Вхідними даними є дескриптор мережевого інтерфейсу, сформований буфер запиту читання оперативної пам'яті.

Вихідні дані

Вихідними даними є буфер відповідного повідомлення або значення, що дорівнює -1 у разі виникнення помилки при виконанні операції.

3.1.18 Опис функції розбору допоміжних файлів procfiles()

Функціональне призначення

Функція призначена для проведення аналізу допоміжних файлів (конфігураційні файли *.xml, файли *.itx, *.map, *.xmp, *.asm проекту) для формування списку ФБ налагоджують програму.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Розбір файлу *.map для отримання адрес сегментів змінних і ФБ в оперативній пам'яті.

Крок 2

Розбір файлу *.xmp для отримання типів змінних і значень зсуву адрес змінних в оперативній пам'яті.

Крок 3

Розбір файлу *.asm для визначення зміщення ФБ в оперативній пам'яті.

Крок 4

Розбір настроювальних файлів *.xml для визначення правил розбору файлів *.itx.

Крок 5

Розбір файлу *.itx для формування списку ФБ програму, що налагоджують.

Крок 6

Відображення тексту програми, що налагоджують на екрані.

Вхідні дані

Вхідними даними є шлях до каталогу проекту.

Вихідні дані

Вихідними даними є список ФБ програми, що налагоджують.

3.1.19 Опис функції інсталяції/скасування контрольної точки KTButton_clicked()

Функціональне призначення

Функція призначена для інсталяції/скасування контрольної точки для обраного ФБ.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Перевірка, чи встановлена контрольна точка для обраного ФБ.

Крок 2

Якщо контрольна точка встановлена, то здійснюється видача запиту скасування контрольної точки для обраного ФБ і виділення рядка, що містить ФБ, сірим кольором.

Крок 3

Якщо контрольна точка не встановлена, то здійснюється видача запиту установки контрольної точки для обраного ФБ і виділення рядка, що містить ФБ, червоним або зеленим кольором.

Вхідні дані

Вхідними даними є список ФБ програми, що налагоджують і номер рядка з обраним ФБ.

Вихідні дані

Вихідними даними є інсталяція/скасування контрольної точки.

3.1.20 Опис функції поетапного виконання програми stepButton_clicked()

Функціональне призначення

Функція призначена для поетапного виконання програми, що налагоджують.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Пошук поточного ФБ у списку ФБ.

Крок 2

Видача запиту інсталяції режиму «Крок».

Вхідні дані

Вхідними даними є список ФБ програми, що налагоджують і номер рядка з поточним ФБ.

Вихідні дані

Вихідними даними є перехід в режим «Крок».

3.1.21 Опис функції DbgTimerDone()

Функціональне призначення

Функція призначена для відображення поточного стану програми, що налагоджують.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Видача запиту отримання списку контрольних точок.

Крок 2

Виділення рядків, що містять ФБ із встановленими контрольними точками, червоним або зеленим кольором.

Крок 3

Читання з оперативної пам'яті й виведення на екран значень обраних змінних.

Вхідні дані

Вхідними даними є дескриптор мережевого інтерфейсу, список ФБ програми, що налагоджують.

Вихідні дані

Вихідними даними є дескриптор мережевого інтерфейсу, список ФБ програми, що налагоджують.

3.1.22 Опис функції зміни значення змінної VarChg()

Функціональне призначення

Функція призначена для запису в оперативну пам'ять введеного значення змінної.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Видача запиту інсталяції режиму холостого ходу.

Крок 2

Перевірка стану режиму холостого ходу, якщо режим встановлений, виконується перехід до кроку 3, в іншому випадку - вихід з функції.

Крок 3

Формування буфера запиту запису в оперативну пам'ять введеного значення змінної.

Крок 4

Видача запиту запису в оперативну пам'ять.

Крок 5

Видача запиту скасування режиму холостого ходу.

Вхідні дані

Вхідними даними є дескриптор мережевого інтерфейсу, сформований буфер запиту читання оперативної пам'яті.

Вихідні дані

Вихідними даними є запис введеного значення в оперативну пам'ять.

3.1.23 Опис функції пуску режиму «Такт» goQuery()

Функціональне призначення

Функція призначена для видачі запиту пуску режиму «Такт» після натискання кнопки «Пуск».

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Формування буфера запиту пуску режиму «Такт» для обраних диспетчерів ФБ із заданою кількістю тактів.

Крок 2

Видача запиту пуску режиму «Такт».

Вхідні дані

Вхідними даними є вибрані диспетчери ФБ і кількість тактів.

Вихідні дані

Вихідними даними є встановлений режим «Такт».

3.1.24 Опис функції зупинки режиму «Такт» stopQuery()

Функціональне призначення

Функція призначена для видачі запиту зупинки режиму «Такт» після натискання кнопки «Стоп».

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Формування буфера запиту зупинки режиму «Такт» для обраних диспетчерів ФБ.

Крок 2

Видача запиту зупинки режиму «Такт».

Вхідні дані

Вхідними даними є вибрані диспетчери ФБ.

Вихідні дані

Вихідними даними є зупинений режим «Такт».

3.1.25 Опис функції зупинки режиму «Такт» cancelQuery()

Функціональне призначення

Функція призначена для видачі запиту скасування режиму «Такт» щодо закриття форми.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Формування буфера запиту скасування режиму «Такт» для всіх диспетчерів ФБ.

Крок 2

Видача запиту скасування режиму «Такт».

Вхідні дані

Вхідними даними є диспетчери ФБ.

Вихідні дані

Вихідними даними є скасування режиму «Такт».

3.1.26 Опис функції отримання стану режиму «Такт» getSostQuery()

Функціональне призначення

Функція призначена для отримання стану режиму «Такт».

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Формування буфера запиту отримання стану режиму «Такт».

Крок 2

Видача запиту отримання стану режиму «Такт».

Крок 3

Виконання розбору буфера відповіді на запит, відображення кількості відпрацьованих тактів і стану режиму «Такт» для кожного обраного диспетчера.

Крок 4

Читання з оперативної пам'яті й виведення на екран значень обраних параметрів, якщо виконання режиму «Такт» завершено для всіх обраних диспетчерів.

Вхідні дані

Вхідні дані відсутні.

Вихідні дані

Вихідними даними є поточний стан режиму «Такт».

3.1.27 Опис функції установки нового значення кількості тактів setNewTakt()

Функціональне призначення

Функція призначена для інсталяції нового значення кількості тактів.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Виведення на екран діалогового вікна для введення нового значення кількості тактів.

Крок 2

Якщо встановлена ознака зміни кількості тактів для всіх диспетчерів, необхідно відобразити нове значення для кожного диспетчера у відповідному полі.

Крок 3

Якщо зміна кількості тактів виконується для певного диспетчера, необхідно відобразити нове значення кількості тактів у відповідному полі для заданого диспетчера.

Вхідні дані

Вхідні дані відсутні.

Вихідні дані

Вихідними даними є нове значення кількості тактів.

3.1.28 Опис функції отримання значень параметрів getVars()

Функціональне призначення

Функція призначена для читання з оперативної пам'яті й виведення на екран значень обраних параметрів.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Формування запиту читання даних з оперативної пам'яті.

Крок 2

Видача запиту читання даних із оперативної пам'яті.

Крок 3

Розбір буфера відповіді й виведення на екран отриманого значення параметром відповідно до його типу.

Вхідні дані

Вхідними даними є список параметрів, обраних для перегляду.

Вихідні дані

Вихідними даними є поточні значення параметрів, обраних для перегляду.

3.1.29 Опис функції установки значень в режимі сценарію taktForm::scrMode()

Функціональне призначення

Функція призначена для підготовки до роботи в режимі сценарію.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Вибір файлу сценарію проекту.

Крок 2

Якщо для файлу сценарію не існує файлу *.bin сценарію, то виконується виклик програми scn2bin для формування файлу *.bin сценарію.

Крок 3

Завантаження файлу * .bin сценарію.

Вхідні дані

Вхідними даними є файл сценарію.

Вихідні дані

Вихідними даними є долучення *.bin сценарію.

3.1.30 Опис функції завантаження секції сценарію taktForm::ScnLoad()

Функціональне призначення

Функція призначена для завантаження в пам'ять КМп секції сценарію й відображення на основній формі вхідних і вихідних змінних проекту, заданих у сценарії.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Відображення на формі вхідних параметрів, заданих у сценарії.

Крок 2

Відображення на формі вихідних параметрів для протоколювання/контролю.

Крок 3

Завантаження секції сценарію в пам'ять КМп.

Вхідні дані

Вхідними даними є номер завдання.

Вихідні дані

Вихідними даними є змінні для контролю на формі.

3.1.31 Опис функції отримання стану поточного такту в режимі сценарію taktForm::ScnNextTakt()

Функціональне призначення

Функція призначена для виконання чергового такту сценарію.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Виконання чергового такту сценарію.

Крок 2

Відображення значень контрольованих змінних, відображення поточного такту й стану диспетчерів сценарію.

Крок 3

Протоколювання значень змінних.

крок 4

Якщо поточне завдання завершено, то виконується завантаження чергового завдання.

Вхідні дані

Вхідними даними є номер завдання.

Вихідні дані

Вихідними даними є значення змінних на основній формі, протокол зміни значень змінних.

3.1.32 Опис функції автоматичного формування сценарію налагодження taktForm;:CreateScoInit()

Функціональне призначення

Функція призначена для збереження дій оператора в файл сценарію для подальшого виконання в автоматичному режимі.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Аналіз режиму функціонування. Якщо операція запису сценарію не була запущена, то перехід до кроку 2. Якщо операція запису сценарію запущена й не виконано жодного завдання, то перехід до кроку 3. Якщо операція запису сценарію запущена і є виконані завдання, то перехід до кроку 4.

Крок 2

Ініціалізація структур для запису нового сценарію. Режим функціонування встановлений «операція запису сценарію запущена, не виконана жодного завдання».

Крок 3

Режим функціонування встановлений «операція запису сценарію не запущена».

Крок 4

Збереження файлу зі сценарієм налагодження в файл, вказаний користувачем. Режим функціонування встановлений «операція запису сценарію не запущена».

Вхідні дані

Вхідними даними є режим функціонування.

Вихідні дані

Вихідними даними є режим функціонування й файл із сценарієм налагодження.

3.1.33 Опис головної програми формування файлу сценарію main()

Функціональне призначення

Програма призначена для виконання наступних дій:

- створення файлу сценарію в бінарному форматі з текстового сценарію;
- формування даних для контролю змінних при роботі сценарію;
- завершення виконання.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Ініціалізація змінних, обробка параметрів командного рядка.

Крок 2

Завантаження файлу *.map проекту.

Крок 3

Завантаження файлу з конфігурацією блоків/каналів системи.

Крок 4

Завантаження файлу *.xmp проекту.

Крок 5

Завантаження файлу сценарію.

Крок 6

Створення файлу *.bin сценарію.

Вхідні дані

Вхідними даними є:

- файл сценарію в текстовому форматі;
- файл *.map проекту;
- файл *.xmp проекту;
- файл з конфігурацією блоків/каналів системи.

Вихідні дані

Вихідними даними є файл сценарію в бінарному форматі.

3.1.34 Опис функції отримання адреси і типу змінної проекту VarAddrGet()

Функціональне призначення

Функція призначена для отримання адреси й типу змінної на ім'я змінної.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Пошук імені змінної у масиві змінних проекту.

крок 2

Визначення адреси(сегмент і зсув) змінної.

Крок 3

Визначення типу змінної.

Вхідні дані

Вхідними даними є:

- ім'я шуканої змінної;

- масив змінних проекту.

Вихідні дані

Вихідними даними є:

- адреса(сегмент і зсув) змінної;

- тип змінної.

3.1.35 Опис функції отримання значення параметра з файлу сценарію ParamValGet()

Функціональне призначення

Функція призначена для отримання значення параметра з файлу сценарію.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Порівняння поточного параметра з файлу сценарію з шуканим. Якщо параметри збігаються, відбувається перехід до кроку 2, при розбіжності функція завершується.

Крок 2

Збереження даних параметра, перехід до обробки наступного параметра.

Вхідні дані

Вхідними даними є:

- файл сценарію, відкритий в режимі читання;

- поточний нерозібраний рядок сценарію;

- номер поточного рядка в файлі сценарію;

- шуканий параметр.

Вихідні дані

Вихідними даними є значення параметра.

3.1.36 Опис функції розбору файлу *.map проекту readMapFile()

Функціональне призначення

Функція призначена для отримання адрес сегментів диспетчерів проекту.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Визначення адреси зміщення диспетчера за іменем диспетчера.

Крок 2

Заповнення таблиці з адресами сегментів диспетчерів проекту.

Вхідні дані

Вхідними даними є ім'я файлу *.map проекту.

Вихідні дані

Вихідними даними є таблиця з адресами сегментів диспетчерів проектів.

3.1.37 Опис функції розбору файлу *.xmp проекту procXmpFile()

Функціональне призначення

Функція призначена для створення списку змінних проекту.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Завантаження файлу *.xmp.

Крок 2

Розбір файлу *.xmp, пошук змінних проекту.

Крок 3

Формування списку змінних проекту.

Вхідні дані

Вхідними даними є:

- ім'я файлу *.xmp проекту;

- таблиця з адресами сегментів диспетчерів проекту.

Вихідні дані

Вихідними даними є список змінних проекту.

3.1.38 Опис функції створення файлу сценарію ScnFile::Save()

Функціональне призначення

Функція призначена для створення файлу сценарію в бінарному форматі.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Формування адрес у таблиці сценарію відповідно до початкової адреси розміщення таблиці в оперативній пам'яті КМп за межами 1 Mbyte (далі - дальна пам'ять).

Крок 2

Збереження контрольної суми файлу сценарію в текстовому форматі.

Крок 3

Збереження змінних для контролю/протоколювання в процесі налагодження сценарію.

Крок 4

Збереження сценарію в бінарному форматі у файлі *.bin.

Вхідні дані

Вхідними даними є об'єкт ScnFile із завантаженим файлом сценарію в текстовому форматі.

Вихідні дані

Вихідними даними є файл *.bin сценарію.

3.1.39 Опис функції завантаження файлу сценарію ScnFile::Load()

Функціональне призначення

Функція призначена для завантаження файлу сценарію в текстовому форматі.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Формування контрольної суми файлу сценарію.

Крок 2

Завантаження й розбір файлу сценарію.

Крок 3

Переведення значень параметрів із фізичних одиниць у код.

Вхідні дані

Вхідними даними є файл сценарію в текстовому форматі.

Вихідні дані

Вихідними даними є об'єкт ScnFile з завантаженим файлом сценарію.

3.1.40 Опис функції запису блоку до файлу сценарія ChkWrite()

Функціональне призначення

Функція призначена для запису блоку даних в файл *.bin.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Розбиття блоку даних на секції по 0x3FFE байт.

Крок 2

Запис секцій в файл.

Вхідні дані

Вхідними даними є:

- блок даних для запису;
- розмір блоку даних.

Вихідні дані

Вихідними даними є записаний у файл блок даних, розбитий на секції.

3.1.41 Опис функції переходу до наступного блоку файлу сценарію ChkPass()

Функціональне призначення

Функція призначена для переходу до наступного блоку даних у файлі *.bin сценарію.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Отримання розміру поточного блоку даних файлу *.bin сценарію.

Крок 2

Перехід до першої секції наступного блоку даних.

Вхідні дані

Вхідними даними є файл *.bin сценарію.

Вихідні дані

Вихідними даними є файл *.bin сценарію.

3.1.42 Опис функції читання блоку з файлу сценарію ChkRead()

Функціональне призначення

Функція призначена для читання блоку даних з файлу *.bin сценарію.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Читання секцій поточного блоку в файлі сценарію до заповнення буфера.

Крок 2

Якщо був прочитаний і записаний у буфер не весь блок, встановлюється ознака незавершеної операції читання блоку.

Крок 3

Запис кількості прочитаних байтів блоку.

Вхідні дані

Вхідними даними є:

- буфер для запису блоку;

- розмір буфера.

Вихідні дані

Вихідними даними є:

- буфер з прочитаними даними блоку файлу *.bin сценарію;

- кількість прочитаних байтів;

- ознака незавершеної операції читання блоку.

3.1.43 Опис функції завантаження xmp-файлу карти пам'яті КМп XmpLoad

Функціональне призначення

Функція призначена для отримання інформації про розташування параметрів в оперативній пам'яті КМп.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Якщо xmp-файл не був завантажений, то перехід до кроку 4.

Крок 2

Аналіз режиму завантаження xmp-файлу. Якщо режим завантаження «завантажити без видалення попередніх даних», то завершення функції. Інакше перехід до кроку 3.

Крок 3

Видалення інформації про розташування параметрів в оперативній пам'яті КМп, завантажених раніше.

Крок 4

Розбір xmp-файлу, збереження інформації про розташування параметрів в оперативній пам'яті КМп в робочих структурах.

Вхідні дані

Вхідними даними є xmp-файл.

Вихідні дані

Вихідними даними є робочі структури з інформацією про розташування параметрів в оперативній пам'яті КМп.

3.1.44 Опис функції вибору каталогу з проектом ProjectDirSet

Функціональне призначення

Функція призначена для вибору каталогу проекту для налагодження за сценарієм.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Збереження змін сценарію в файл.

Крок 2

Вибір каталогу з проектом.

Крок 3

Виклик функції створення сценарію налагодження (fileNew).

Вхідні дані

Вхідними даними є ім'я каталогу проекту.

Вихідні дані

Вихідними даними є робочі структури з інформацією про розташування проекту для налагодження за сценарієм.

3.1.45 Опис функції додавання сигналу в сценарій налагодження VarInputAdd

Функціональне призначення

Функція призначена для додавання вхідного сигналу в обрану секцію сценарію налагодження.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Визначення поточної секції сценарію налагодження.

Крок 2

Пошук сигналу в списку вхідних параметрів поточної секції сценарію налагодження. Якщо сигнал знайдений - завершення функції. Інакше перехід до кроку 3.

Крок 3

Додавання сигналу до сценарію налагодження.

Вхідні дані

Вхідними даними є ім'я сигналу.

Вихідні дані

Вихідними даними є вхідні параметри секції сценарію налагодження.

3.1.46 Опис функції додавання сигналу до сценарію налагодження VarOutputAdd

Функціональне призначення

Функція призначена для додавання вихідного сигналу в обрану секцію сценарію налагодження.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Визначення поточної секції сценарію налагодження.

Крок 2

Пошук сигналу в списку вхідних параметрів поточної секції сценарію налагодження. Якщо сигнал знайдений - завершення функції. Інакше перехід до кроку 3.

Крок 3

Додавання тонів в сценарій налагодження.

Вхідні дані

Вхідними даними є ім'я сигналу.

Вихідні дані

Вихідними даними є вхідні параметри секції сценарію налагодження.

3.1.47 Опис функції створення сценарію налагодження fileNew

Функціональне призначення

Функція призначена для створення нового сценарію налагодження обраного проекту.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Якщо поточний сценарій не був збережений, виконується виклик функції збереження поточного сценарію до файлу (FileSave).

Крок 2

Перевірка каталогу проекту для налагодження за сценарієм. Якщо не заданий, виконується виклик діалогу вибору каталогу проекту. Інакше перехід до кроку 3.

Крок 3

Ініціалізація робочих структур для створення нового сценарію налагодження обраного проекту.

Вхідні дані

Вхідними даними є ім'я каталогу проекту налагодження за сценарієм.

Вихідні дані

Вихідними даними є робочі структури з інформацією про проект, що налагоджують.

3.1.48 Опис функції редагування сценарію налагодження fileOpen

Функціональне призначення

Функція призначена для завантаження сценарію налагодження обраного проекту з файлу зі сценарієм (*.sco).

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Виклик діалогу вибору файлу зі сценарієм налагодження обраного проекту.

Крок 2

Завантаження сценарію налагодження з файлу зі сценарієм налагодження в робочі структури.

Крок 2

Завантаження значень вихідних параметрів сценарію налагодження з файлу з еталонними значеннями в робочі структури.

Крок 3

Відображення сценарію налагодження в основному вікні редактора сценаріїв.

Вхідні дані

Вхідними даними є ім'я файлу зі сценарієм.

Вихідні дані

Вихідними даними є робочі структури з інформацією про сценарії налагодження.

3.1.49 Опис функції збереження сценарію налагодження fileSave

Функціональне призначення

Функція призначена для збереження сценарію налагодження обраного проекту в файл.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Якщо сценарій не був збережений раніше, виконується виклик діалогу вибору імені файлу зі сценарієм. Інакше перехід до кроку 2.

Крок 2

Збереження значень вихідних параметрів сценарію в файл з еталонними значеннями (*.pre).

Крок 3

Збереження сценарію налагодження в файл (*.sco).

Вхідні дані

Вхідними даними є ім'я файлу зі сценарієм.

Вихідні дані

Вихідними даними є файли зі сценарієм налагодження й еталонними значеннями.

3.1.50 Опис функції завантаження сценарію налагодження scnMode

Функціональне призначення

Функція призначена для завантаження сценарію налагодження з файлу зі сценарієм в бінарному форматі (*.bin).

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Завантаження інформації про сценарії налагодження в робочі структури.

Крок 2

Формування списків сценаріїв за типом ФБ, що перевіряються.

Вхідні дані

Вхідними даними є ім'я файлу зі сценарієм в бінарному форматі.

Вихідні дані

Вихідними даними є робочі структури з інформацією про сценарії налагодження.

3.1.51 Опис функції виконання сценарію налагодження scnLoad

Функціональне призначення

Функція призначена для завантаження і виконання сценарію налагодження в КМп.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Завантаження сценарію налагодження в КМп.

Крок 2

Покрокове виконання сценарію в КМп.

Крок 3

Формування протоколу сценарію налагодження.

Вхідні дані

Вхідними даними є сценарій налагодження в бінарному форматі.

Вихідні дані

Вихідними даними є протокол сценарію налагодження.

3.1.52 Опис функції порівняння протоколу сценарію налагодження CompareProto

Функціональне призначення

Функція призначена для порівняння протоколу сценарію налагодження з еталонними значеннями.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Запуск програми порівняння файлу з протоколом налагодження з еталонними значеннями (prt_diff).

Крок 2

Аналіз коду завершення програми порівняння, формування протоколу з результатом порівняння.

Вхідні дані

Вхідними даними є файл з протоколом налагодження й файл з еталонними значеннями.

Вихідні дані

Вихідними даними є протокол порівняння файлів протоколу налагодження і еталонних значень.

3.1.53 Опис функції порівняння протоколу результату тестування ФБ із еталонами main

Функціональне призначення

Програма призначена для порівняння отриманого протоколу тестування з еталонними значеннями й формування протоколу з повідомленнями про помилки.

Опис алгоритму

Алгоритм представляє собою наступну послідовність кроків.

Крок 1

Розбір і перевірка на коректність вхідних параметрів.

Крок 2

Завантаження файлу з еталоном тестування ФБ, якщо сталася помилка, формується виведення повідомлення про помилку й перехід до кроку 5.

Крок 3

Завантаження файлу з результатом тестування ФБ, якщо сталася помилка, формується виведення повідомлення про помилку і перехід до кроку 5.

Крок 4

Порівняння значення змінних еталона тестування з результатом тестування, при виявленні незрівняннь виконується формування коду завершення й звіту з помилками тестування ФБ.

Крок 5

Звільнення дескрипторів, займаних даними про ФБ, значень змінних із еталона тестування й результату тестування.

Вхідні дані

Вхідними даними є файл із протоколом налагодження й файл із еталонними значеннями.

Вихідні дані

Вихідними даними є протокол порівняння файлів протоколу налагодження й еталонних значень.

Висновок до розділу 3

У данному розділі виконано реалізацію інструментальних засобів налагодження у вигляді алгоритмів.

4 ІНСТРУКЦІЯ ОПЕРАТОРУ

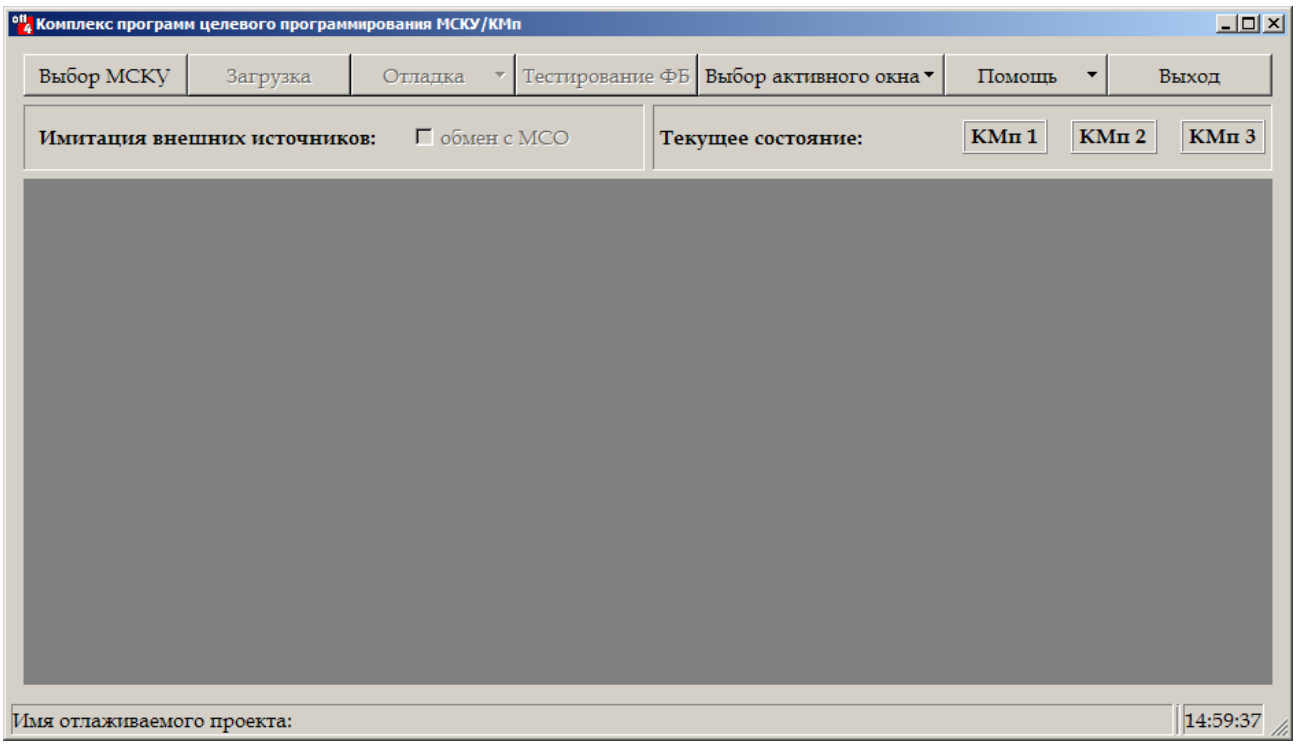


Рисунок 4.1 - Вид основного кадру, що видається при запуску зневаджувача

Пункт меню «Вибір МСКУ»

В даному режимі виконується визначення МСКУ, в якому необхідно вести процес налагодження. Для указання шляху до групи проектів необхідно задати повний шлях до проекту.

Після того, як було вказано шлях до групи проектів для продовження роботи, слід натиснути на кнопку «**Згоден**».

Кнопка «**Скасувати**» призначена для скасування дій вибору проекту і повернення в основне меню.

Після вказівки шляху до групи проектів на екран виводиться вікно зі списком проектів.

Для вибору проекту необхідно за допомогою клавіш на клавіатурі <Вгору> і <Вниз> виділити рядок, що містить ім'я проекту, і натиснути кнопку «**Згоден**».

В результаті виконання дій закріплюється вибір МСКУ. При цьому виконується повернення в основне вікно і встановлюється зв'язок з МСКУ, з адресою, відповідною вибраному проекту.

Кнопка «**Скасувати**» призначена для скасування дій вибору.

Пункт меню «Завантаження»

При виборі пункту меню «**Завантаження**» на екран виводиться вікно для виконання процесу завантаження.

У рядку «**Номер крейта**» виводиться номер завантажуємого крейта, в рядках «**Адреса МСКУ**» і «**Ім'я проекту**» відображається адреса МСКУ і ім'я проекту, задані при виборі пункту меню «**Вибір МСКУ**».

Кнопка «**Налаштування**» призначена для вибору інтерфейсу доступу до МСКУ для виконання процесу завантаження.

Для початку процесу завантаження необхідно натиснути кнопку «**Завантажити**». Якщо МСКУ знаходиться не в стартовому режимі, на екран виводиться вікно з попередженням.

Для виконання скидання МСКУ з наступним завантаженням необхідно вимкнути, а потім вімкнути живлення МСКУ і натиснути кнопку «**Згоден**». Щоб скасувати завантаження необхідно натиснути кнопку «**Скасувати**».

Відсоток виконання роботи відображається в рядку «**0% 100%**».

При завершенні на екран буде виведено додаткове вікно з повідомленням «**Операція завантаження успішно завершена!**».

Кнопка «**Скасувати**» призначена для скасування завантаження МСКУ і повернення в основне меню.

Режим «Налагодження програми»

У режимі «**Налагодження програми**» надаються наступні можливості:

- вибір і зміна вхідних змінних, в тому числі і значень вхідних аналогових і дискретних сигналів;
- моніторинг зміни вихідних змінних;
- виконання програми в режимі «Такт»;
- виконання програми в режимі контрольних точок;
- послідовне виконання програми.

Програми, що налагоджуються повинні бути представлені в файлах * .itx.

При виборі пункту меню «**Налагодження**» виводиться вікно для вибору режиму налагодження.

В полі «**Номер крейта**» необхідно вказати номер крейта МСКУ, в якому необхідно налагоджувати програми. Для цього слід за допомогою клавіатури ввести цифри 1, 2 або 3. Якщо вказано неприпустимий номер або з крейтом не встановлений зв'язок, то буде видане повідомлення про порушення.

В полі «**Режим**» необхідно вибрати режим налагодження програми.

За замовчуванням встановлюється режим «**Такт**» для всіх типів прикладних програм з кількістю циклів виконання, рівним 1.

Вибір режиму налагодження по контрольним точкам або в кроці здійснюється кнопкою «**КТ, Пуск, Крок**».

Поле «**Диспетчери**» заповнюється тільки в режимі «**Такт**». Елементи «**Кількість тактів**» за замовчуванням визначають режим виконання програми за один цикл.

Поле «**Імітація зовнішніх джерел**» служить для визначення додаткових джерел (ПЗО, мережі МСКУ-4, ліній оптичного UART), дані від яких будуть задаватися значеннями відповідних параметрів в списку вхідних параметрів.

Кнопка «**Вибрати**» призначена для підтвердження завдання режиму.

Кнопка «**Скасувати**» призначена для виходу з цього вікна.

Режим «Такт»

Режим «**Такт**» використовується в тому випадку, коли необхідно в кінці кожного робочого циклу (20 ms) або через задану користувачем кількість циклів мати можливість аналізувати результати виконання програм, змінювати вхідні сигнали в ручному режимі або в режимі сценарію.

В ручному режимі значення вхідних сигналів і змінних ОБД вказуються в діалозі і описані далі. У режимі сценарію значення вхідних сигналів, інтервали і правила їх зміни вказуються в спеціальному файлі.

В полі «**Диспетчери**» необхідно вказати типи прикладних програм, результати яких необхідно аналізувати в режимі «**Такт**».

За замовчуванням режим «**Такт**» вказується для всіх типів прикладних програм: первинної обробки, формування розрахункових параметрів, контролю за технологічними вставками, логічних функцій і формування вихідних параметрів. Слід встановити курсор на елементі «**Кількість тактів**» в рядку, що відповідає елементу «**Все**», і за допомогою клавіатури ввести кількість тактів.

Установка режиму «**Такт**» для прикладної програми будь-якого типу виконується вибором рядка з описом диспетчера відповідного типу. Потім необхідно за допомогою клавіатури ввести кількість тактів в елементі «**Кількість тактів**» цього ж рядка.

Кнопка «**Вибрати**» призначена для підтвердження завдання.

Кнопка «**Скасувати**» призначена для скасування завдання і виходу з цього вікна.

У кожному рядку «**Номер такту**» відображає задану кількість тактів, а також поточний номер виконуваного або виконаного такту. Для введення нового значення кількості тактів (тільки після переходу в стан «**Зупинено**» або «**Завершено**») слід натиснути кнопку «**Змінити**» для відповідного рядка «**Номер такту**» і в діалоговому вікні вказати нове значення кількості тактів у вигляді десяткового числа. Зміна кількості тактів можлива тільки в стані «**Зупинено**» або «**Завершено**».

У кожному рядку «**Стан**» у вигляді тексту відображається поточний стан відповідного диспетчера управління прикладною програмою: «**Завершено**» - програма виконалася протягом зазначеної кількості тактів і більше не викликається на виконання; «**Виконується**» - програма виконується; «**Зупинено**» - програма зупинена до закінчення зазначеної кількості тактів і більше не викликається на виконання.

Для додавання змінних в список для перегляду необхідно вибрати поле «**Вхідні змінні**» або «**Вихідні змінні**». Для цього необхідно встановити «**мишу**» на будь-яку точку обраного поля і спочатку натиснути ліву клавішу «**миші**», а потім на кнопку «**Список змінних**». При виборі пункту меню «**Список змінних для перегляду**» виводиться додаткове вікно з пунктами меню «**Додати**», «**Видалити**», «**Зберегти**», «**Оновити**».

Пункт меню «**Список змінних**» -> «**Список змінних для перегляду**» -> «**Додати**» дозволяє додати змінну (із загального списку, з файлу, за шаблоном, по атрибуту, по типу диспетчера прикладних програм).

При виборі пункту меню «**Список змінних**» -> «**Установка значень**» -> «**Сценарій**» на екран буде виведено вікно для вибору файлу, що містить список і сценарій зміни значень вхідних змінних.

При виборі пункту меню «**Список змінних**» -> «**Установка значень**» -> «**Ручний режим**» на екран буде виведено додаткове вікно для вибору та задання змінних.

Режим «КТ, Пуск, Крок»

Режим «**КТ, Пуск, Крок**» використовується в тому випадку, коли необхідно виконувати програму поетапно з зупинкою в кінці етапу для аналізу результатів і установки нових вхідних значень. Будь-який етап виконання програми обмежується контрольними точками. Перехід від однієї контрольної точки до іншої здійснюється кнопкою «**Пуск**». Крім того, в цьому режимі надається можливість виконання програми на обраній ділянці в покроковому режимі.

Вибір режиму «**КТ, Пуск, Крок**» здійснюється кнопкою з написом «**КТ, Пуск, Крок**». В результаті на екран виводиться допоміжне вікно для вибору файлу з програмою, яку треба налагодити.

Поле «**Список файлів**» призначено для вибору файлу з програмою і містить список файлів з розширенням * .itx, що входять до складу обраного для налагодження проекту.

Для вибору програми, що налагоджують, або декількох програм (якщо прикладні програми розміщені в декількох файлах і потрібно налагодження всіх програм відразу) слід встановити курсор на рядку з відповідним файлом, що містить її текст, і натиснути кнопку «**Вибрати**». В результаті на екран виводиться вікно «**Налагодження програми**».

Поле «**Імітація зовнішніх джерел**» служить для визначення додаткових джерел (ПЗО, мережі МСКУ-4, ліній оптичного UART), дані від яких будуть задаватися значеннями відповідних параметрів в списку вхідних параметрів.

Вікно «**Налагодження програми**» містить наступні елементи:

- поле «Програма», що містить текст програми;
- поле «Вхідні змінні функціонального блоку», призначене для виведення списку вхідних змінних, обраних для перегляду. Значення вхідних змінних можуть бути змінені;
- поле «Вихідні змінні функціонального блоку», призначене для виведення списку вихідних змінних, обраних для перегляду;
- поле «Локальні змінні функціонального блоку», призначене для виведення списку локальних змінних, обраних для перегляду.

У верхній частині екрану вікна «**Налагодження програми**» розташовані кнопки управління ходом налагодження програми.

Кнопка «**Контрольна точка**» (або клавіша <F1>) призначена для установки контрольної точки.

Кнопка «**Список змінних**» призначена для формування списку вхідних і вихідних змінних для зміни і перегляду.

Кнопка «**Крок**» призначена для виконання програми в шаговом режимі.

Кнопка «**Пуск**» (або клавіша <F9>) призначена для пуску програми з поточного адреси зупинки.

Кнопка «**Вихід**» призначена для виходу з режиму відладки.

Для установки контрольної точки необхідно за допомогою клавіш < Вгору > і < Вниз > або лівою кнопкою < миші > вибрати потрібний рядок в полі «**Програма**» (обраний рядок виділяється сірим кольором) і натиснути кнопку «**Контрольна точка**» або клавішу < F1>. При цьому рядок зі встановленою контрольної точкою виділяється червоним кольором.

При виході програми в встановлену контрольну точку рядок, що містить опис контрольної точки (ім'я функції), виділяється зеленим кольором.

Пуск програми із зазначеної адреси виконується після натискання клавіші <F9> або кнопки «**Пуск**».

Для покрокового виконання програми необхідно натиснути клавішу <F1> або кнопку «**Крок**». Після чого рядок, що містить адресу поточної точки зупину, буде виділений блакитним кольором, якщо за цією адресою контрольна точка не встановлена, в іншому випадку - зеленим.

Для додавання змінних в список для перегляду необхідно вибрати відповідне поле («**Вхідні змінні функціонального блоку**», «**Вихідні змінні функціонального блоку**» або «**Локальні змінні функціонального блоку**») і натиснути на кнопку «**Список змінних**». На екран виводиться додаткове меню, що дозволяє додати змінну (із загального списку, з файлу, за шаблоном, по атрибуту, на ім'я функції, для поточної обраної функції), видалити змінну або зберегти список змінних в файл.

Режим «Перегляд змінних ОБД»

При виборі пункту меню «**Перегляд змінних ОБД**» виводиться вікно, що містить список проектів групи, що включають налаштування сервера архівування та діагностування (СДА) і опис ОБД, заданої при виборі пункту меню «Вибір МСКУ».

Для вибору потрібного проекту необхідно за допомогою покажчика «**миші**» або клавіш <**Вгору**> і <Вниз> вибрати рядок, що містить ім'я потрібного проекту, і

натиснути кнопку «**Згоден**». В результаті на екран виводиться вікно, що містить список блоків.

Для вибору всіх змінних блоку необхідно в поле «**Вибір змінних**» вибрати елемент «**по блокам**», а потім в полі «**Список блоків**» відзначити потрібні блоки.

Для вибору змінних по атрибуту необхідно в поле «**Вибір змінних**» вибрати елемент «**по атрибуту**», а потім в рядку введення ввести атрибут. Якщо в полі «**Список блоків**» не зазначено жодного блоку, будуть обрані змінні з відповідним атрибутом з усіх блоків, в іншому випадку - тільки для обраних блоків.

Для вибору змінних по імені необхідно в полі «**Вибір змінних**» вибрати елемент «**по імені**», а потім в рядку введення ввести ім'я змінної. Якщо в полі «**Список блоків**» не зазначено жодного блоку, будуть обрані змінні з відповідним ім'ям з усіх блоків, в іншому випадку - тільки для обраних блоків.

Для виведення списку обраних змінних ОБД необхідно натиснути кнопку «**Перегляд**», при цьому на екран виводиться вікно, що містить список змінних.

В полі «**Список змінних**» відображається список змінних, що відповідають заданим параметрам вибору.

Для перегляду значень обраних змінних необхідно натиснути кнопку «**Дані**» і вибрати режим відображення значень: одноразовий прийом (пункт меню «**Дані**» -> «**Оновити**») або відображення даних кожен раз по прийому блоку (пункт меню «**Дані**» -> «**Прийом**»). Значення оновлюються тільки для обраних змінних. При натисканні кнопки «**Фільтр**» можна вибрати як всі змінні, так і знайти потрібні змінні по імені, а також зняти виділення і продовжити пошук.

Отримані значення зберігаються в файл протоколу, якщо натиснута кнопка «**Протокол**». При цьому на екран виводиться додаткове вікно, в якому необхідно вибрати повний шлях до файлу протоколу на диску ПЕОМ і задати ім'я файлу протоколу.

Режим «Завдання оперативно змінюваних даних»

Режим «**Завдання оперативно змінюваних даних**» надає можливість зміни значень оперативно змінюваних даних для аналізу поведінки програм користувача.

При виборі пункту меню «**Завдання оперативно змінюваних даних**» виводиться вікно, призначене для формування списку змінних параметрів з можливістю його збереження і редагування.

Вікно містить наступні елементи:

- поле «**Список груп параметрів**», призначене для виведення списку груп параметрів. Група з ім'ям «**Всі параметри**» створюється за замовчуванням і містить всі параметри, доступні для запису. Під групою розуміється об'єднання параметрів (одного або трьох крейтів в разі резервованого МСКУ) в один список з заданим ім'ям;
- поле «**Вибір крейта**», що містить елементи для відображення / приховування параметрів відповідного крейта;
- поле «**Найменування групи**», призначене для відображення імені групи, обраної в полі «**Список груп параметрів**», і ввести нову назву групи в режимі редагування;
- поля «**Список параметрів групи (Крейт *i*)**», де *i* приймає значення від 1 до 3, призначені для відображення і вибору параметрів для зміни;
- кнопку «**Створити групу**», призначену для переходу в режим створення нової групи параметрів. Для формування нової групи параметрів необхідно в поле «**Найменування групи**» ввести ім'я групи, а в полях «**Список параметрів групи (Крейт *i*)**», де *i* приймає значення від 1 до 3, вибрати необхідні параметри. Вибір параметра здійснюється за допомогою натискання лівої клавіші «**миші**» на рядку, що містить ім'я параметра. Вихід з режиму створення групи зі збереженням змін виконується після натискання кнопки «**Зберегти групу**»;
- кнопку «**Видалити групу**», призначену для видалення обраної групи зі списку груп в полі «**Список груп параметрів**»;
- кнопку «**Редагувати групу**», призначену для переходу в режим редагування обраної групи. Режим редагування надає можливість зміни імені та

списку параметрів обраної групи. Вихід з режиму редагування із збереженням змін виконується після натискання кнопки «**Зберегти групу**»;

- кнопку «**Зберегти групу**», призначену для виходу з режимів створення і редагування групи зі збереженням внесених змін;
- кнопку «**Скасувати**», призначену для виходу з режимів створення і редагування групи без збереження внесених змін;
- кнопку «**Фільтр**», яка надає можливість установки / зняття мітки вибору для всіх параметрів, що відображаються в полях «**Список параметрів групи (Крейт *i*)**», де *i* приймає значення від 1 до 3, а також пошуку параметра в списку по імені;
- кнопку «**Закрити**», призначену для завершення роботи в режимі «Завдання оперативно змінюваних даних»;
- кнопку «**Зміна даних**», призначену для переходу в режим безпосереднього зміни даних після вибору групи і змінюваних параметрів.

При натисканні кнопки «**Перегляд / зміна даних**» на екран виводиться вікно для перегляду і зміни даних.

Вікно містить наступні елементи:

- поле «**Найменування групи**», призначене для відображення імені групи, обраної в полі «**Список груп параметрів**»;
- поле «**Список параметрів групи**», призначене для відображення параметрів, обраних в полях «**Список параметрів групи (Крейт *i*)**»;
- поле «**Формат**», призначене для вибору формату даних (десятичний або шістнадцятковий);
- поле «**Стан (Крейт *i*)**», де *i* приймає значення від 1 до 3, призначене для відображення наявності вирівнювання і незрівняннь оперативно змінюваних даних відповідного крейта. Елемент «**Вирівнювання**» дозволяє відключити / включити вирівнювання;
- кнопку «**Змінити**», призначену для передачі заданих значень в обраний крейт МСКУ. При цьому елементом «**Вирівнювання**» в МСКУ повинен бути відключений процес вирівнювання;

- кнопку «**Оновити**», призначену для відображення поточного стану обраних змінних;
- кнопку «**Закрити**», призначену для виходу з режиму зміни даних.

Для завдання нових значень необхідно за допомогою клавіш <Вгору>, <Вниз> або натисканням лівої кнопки «**миші**» вибрати рядок з ім'ям параметра і двічі натиснути ліву кнопку «миші» або клавішу <Enter>. Далі у вікні ввести значення для запису і натиснути кнопку «**Записати**». Після завдання всіх необхідних значень необхідно натиснути кнопку «**Змінити**».

Пункт меню «Вибір активного вікна»

Пункт меню «**Вибір активного вікна**» містить перелік усіх відкритих вікон.

При виборі пункту меню, відповідного відчиненого вікна, це вікно стає активним (тобто відображається поверх всіх вікон).

Пункт меню «Допомога»

При виборі пункту «**Допомога**» -> «**Довідка**» на екран виводиться вікно, яке містить перелік можливих операцій налагодження програм в МСКУ. Для отримання інформації про правила роботи при виконанні необхідної операції налагодження необхідно встановити покажчик «миші» на потрібний рядок і натиснути праву клавішу. При цьому на екрані буде виведена текстова інформація про правила використання необхідної операції, яку можна переглядати, використовуючи елемент правою прокрутки.

За допомогою пункту «**Допомога**» -> «**Налаштування шрифту основного додатка**» можна задати параметри шрифту для основного додатка.

За допомогою пункту «**Допомога**» -> «**Налаштування шрифту виведення даних**» можна задати параметри шрифту для відображення виводу даних.

Пункт меню «Вихід»

При виборі пункту меню «**Вихід**» або натисканні комбінації клавіш <Alt> <F4> завершується робота відладчика.

Висновок до розділу 4

У данному розділі розроблена інструкція з користування розробленою підсистемою налагодження.

5 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯЇХ

В даному розділі проведено аналіз потенційних небезпечних та шкідливих виробничих факторів, причин пожеж. Розглянуті заходи, які дозволяють забезпечити гігієну праці і виробничу санітарію. На підставі аналізу розроблені заходи з техніки безпеки та рекомендації з пожежної профілактики.

Так як завданням на дипломне проектування є розробка інструментальних засобів налагодження прикладних програм МСКУ-4, то аналіз потенційно небезпечних і шкідливих виробничих чинників виконується для персонального комп'ютера на якому буде знаходитися підсистема налагодження.

5.1 Аналіз потенційно небезпечних і шкідливих виробничих факторів, що впливають на персонал

У дипломному проекті розробляється програмне забезпечення. Розроблюване програмне забезпечення орієнтоване на роботу з персональним комп'ютером. Використовувані для вирішення завдань ПЕОМ типу IBM PC мають такі характеристики:

- споживана потужність 220 Вт;
- робоча напруга 220 В;
- напруга джерел живлення +12 В; - 12 В, 5 В.

На користувачів під час роботи з комп’ютерною технікою можуть діяти такі види небезпек:

– ураження електричним струмом;

– енергетична небезпека (виникає через коротке замикання: опіки, електрична дуга, викид розплавленого металу);

– небезпека загоряння; термонебезпека (дія високих температур через нагрівання конструктивних елементів);

– механічна небезпека (травми через падіння, дію рухомих частин, поріз за гострі частини конструктивних елементів);

– небезпека випромінювання (дія звукового (акустичного), високочастотного, інфрачервоного, ультрафіолетового й іонізуючого випромінювання, а також видимого світла когерентної високої інтенсивності (лазерного випромінювання);

– хімічна небезпека (контакт із деякими хімікатами, які використовують для того, щоб обслуговувати обладнання, або від вдихання їх парів).

Згідно з ГОСТ 12.0.003-74 «ССБТ. Опасные и вредные производственные факторы. Классификация» [18], при обслуговуванні ПЕОМ мають місце фізичні та психофізичні небезпеки, а також шкідливі виробничі фактори:

- підвищений значення напруги в електричному ланцюзі, замикання якого може відбутися через тіло людини;
- підвищений рівень статичної електрики;
- підвищений рівень електромагнітних випромінювання;
- підвищений або знижена температура повітря робочої зони;
- підвищений або знижена рухливість повітря;
- підвищений або знижена вологість повітря;
- відсутність або нестача природного світла;
- підвищений пульсація світлового потоку;
- недостатність освітлення робочого місця;
- підвищений рівень шуму на робочому місці;
- розумове перенапруження;
- емоційне навантаження;
- монотонність праці.

5.2 Заходи з охорони праці

Основним небезпечним фактором при роботі з ЕОМ є небезпека ураження людини електричним струмом, яка посилюється тим, що органи чуття людини не можуть на відстані знайти наявності електричної напруги на обладнанні.

Проходячи через тіло людини, електричний струм чинить на нього складний вплив, що є сукупністю термічної (нагрів тканин і біологічних середовищ), електролітичної (розкладання крові і плазми) і біологічної (роздратування і збудження нервових волокон та інших органів тканин організму) дій.

Тяжкість ураження людини електричним струмом залежить від цілого ряду чинників:

- значення сили струму;
- електричного опору тіла людини і тривалості протікання через нього струму;
- частоти струму;
- індивідуальних властивостей людини і навколишнього середовища.

Розроблений дипломний проект передбачає такі технічні засоби і засоби, що попереджають людини від ураження електричним струмом:

- заземлення електроустановок;
- занулення;
- захисне відключення;
- електричне поділ мереж;
- використання малої напруги;
- ізоляція частин, які проводять струм;
- огородження електроустановок.

Занулення зменшує напругу дотику і обмежує час, протягом якого людина, доторкнувшись до корпусу, може потрапити під дію напруги.

5.3 Заходи, що забезпечують виробничу санітарію і гігієну праці

Трудова діяльність людини завжди протікає в певних метеорологічних умовах, які визначаються поєднанням температури повітря, швидкості його руху і відносної вологості, тиском і тепловим випромінюванням від нагрітих поверхонь. Оскільки експлуатація проектованого програмного засобу відбувається в приміщенні, то ці показники в сукупності (за винятком тиску) називаються мікрокліматом виробничого приміщення. В даний час основним нормативним документом нормалізації мікроклімату є ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень» [19] та ГОСТ 12.1.005-88 «ССБТ.Общие санитарно-гигиенические требования к воздуху рабочей зоны» [20].

Тяжкість праці характеризує сукупну дію всіх елементів, що складають умови праці, на працездатність людини, його здоров'я, життєдіяльність і відновлення робочої сили. У такому представлені поняття тяжкості праці однаково застосовне як до розумової, так і до фізичної праці. Відповідно з ГОСТ 12.1.005-88 «ССБТ.Общие санитарно-гигиенические требования к воздуху рабочей зоны» [20] тяжкість роботи персоналу, який обслуговує ЕОМ, відноситься до легкої категорії 1а (роботи, що виконуються сидячи, не вимагаючи систематичного фізичного напруження і перенесення важкостей). Оптимальні норми мікроклімату в робочій зоні, забезпечувані для робіт легкої категорії 1а приведені в таблиці 5.1

Таблиця 5.1– Оптимальні норми мікроклімату

Період року	Температура, ºС	Відносна вологість, %	Швидкість руху повітря, м/с, не більш
Холодний	22 – 24	60 – 40	0,1
Теплий	23 – 25	60 – 40	0,1

У приміщенні, де знаходяться ЕОМ, повітрообмін реалізується за допомогою природної організованої вентиляції (вентиляційні шахти) і установки кондиціонера. Цей метод забезпечує приток потрібної кількості свіжого повітря, що визначається в ДБН (30 кубічних метрів на годину на одного працюючого).

Для захисту від електромагнітного випромінювання передбачаються наступні заходи:

–застосування нових плазмових моніторів;

–віддалення робочого місця не менше, ніж на 0,4 – 0,5 м, оскільки напруженість електричного поля зменшується при віддаленні від джерела поля;

–встановлення раціональних режимів роботи персоналу (обмеження часу перебування);

–раціональне розміщення в робочому приміщенні устаткування, що випромінює електромагнітну енергію.

Оскільки рівень шуму не перевищує гранично допустимих величин [21][22], які встановлені санітарними нормами, заходи для зниження шуму не проводяться.

Для зниження стомлюваності обслуговуючого персоналу в приміщеннях, де розташовані обчислювальні засоби, передбачається використовувати спокійні колірні поєднання і покриття, що не дають відблисків.

Світло є природною умовою існування людини. Воно впливає на стан вищих психічних функцій і фізіологічні процеси в організмі. Хороше освітлення діє

тонізуюче, створює гарний настрій, покращує протікання основних процесів вищої нервової діяльності.

Збільшення освітленості сприяє поліпшенню працездатності навіть в тих випадках, коли процес праці практично не залежить від зорового сприйняття. При поганому освітленні людина швидко втомлюється, працює менш продуктивно, виникає потенційна небезпека помилкових дій і нещасних випадків.

У проекті, що розробляється, передбачається використовувати суміщене освітлення. У світлий час доби використовуватиметься природне освітлення приміщення через віконні отвори, в решту часу використовуватиметься штучне освітлення. Штучне освітлення створюється газорозрядними лампами.

Розрахунок штучного освітлення виробляється по коефіцієнтах використання світлового потоку, яким визначається потік, необхідний для створення заданої освітленості при загальному рівномірному освітленні. Розрахунок кількості світильників *N* виробляється по формулі (4.1):,

$$N=\frac{E\cdot l\cdot m\cdot Z\cdot K}{U\cdot M\cdot F} \qquad (5.1)$$

де *E* – нормована освітленість – 200 лк;

l – довжина кімнати – 8 м;

m – ширина кімнати – 4 м;

Z – поправочний коефіцієнт світильника (для стандартних світильників Z = 1.1 - 1.3) приймаємо рівним 1,2;

K – коефіцієнт запасу, що враховує зниження освітленості в процесі експлуатації – 1,5;

U – коефіцієнт використання, залежний від типу світильника, показника індексу приміщення і т.п. – 0,55

M – число люмінесцентних ламп в світильнику – 1;

F – світловий потік лампи – 5400 лм.

Підставивши числові значення у формулу (5.1), отримуємо:

$$N=\frac{200\cdot 8\cdot 4\cdot 1{,}2\cdot 1{,}5}{5400\cdot 0{,}55\cdot 1}\approx 3{,}8$$

Вибирається кількість світильників N, що дорівнює 4. Схема розташування світильників показана на рис. 5.1.

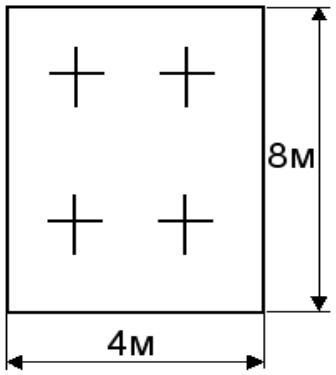


Рисунок 5.1 – Схема розташування світильників

5.4 Рекомендації з пожежної профілактики

Пожежі в робочому приміщенні становлять небезпеку, тому що пов'язані як з матеріальними втратами, так і з відмовою засобів обчислювальної техніки.

Пожежа може виникати при внесенні джерела запалювання в горючу середу. Горючими матеріалами в приміщенні, де розташовані обчислювальні засоби є будівельні матеріали, віконні рами, двері, підлоги, меблі, ізоляція силових і сигнальних кабелів, радіотехнічні деталі, конструктивні елементи з пластичних матеріалів, рідини для очищення елементів і вузлів ЕОМ від забруднень:

1) поліамід – матеріал корпусу мікросхем, горюча речовина, температура самозаймання 420 °С,

2) полівінілхлорид – ізоляційний матеріал, горюча речовина, температура запалювання 335 °С, температура самозаймання 530 °С,

3) стеклотекстоліт ДЦ – матеріал друкарських плат, важкогорючий матеріал, показник горючості 1.74, не схильний до температурного самозаймання,

4) пластикат кабельний №.489 – матеріал ізоляції кабелів, горючий матеріал, показник горючості більше 2.1,

5) деревина – будівельний і обробний матеріал, з якого виготовлені меблі, горючий матеріал, показник горючості більше 2.1, температура запалювання 255 °С, температура самозаймання 399 °С.

Згідно НАПБ Б. 03.002-2007 [23] таке приміщення відноситься до категорії "В" (пожежонебезпечної) .

Пожежа може виникнути в результаті утворення джерела запалювання (іскри і дуги короткого замикання, порушення ізоляції, що приводить до короткого замикання, перегріву радіодеталей внаслідок тривалого перевантаження) і внесення його в горючу середу.

При повному згорянні органічних сполук утворюється (CO_2, SO_2, H_2O, N_2), а при згорянні неорганічних сполук - оксиди. Залежно від температури плавлення продукції, реакції диму можуть знаходитися у вигляді розплаву (Al_2O_3, TiO_2), або підніматися в повітря у вигляді диму (P_2O_5, Na_2O, MgO). Розплавлені тверді частинки створюють світність полум'я. Склад продуктів неповного згоряння горючих речовин складний і різноманітний. Це можуть бути горючі речовини - H_2, CO, CH_4 та інші; атомарний водень і кисень; різні радикали - OH, CH та інші. Продуктами неповного згорання можуть бути також оксиди азоту, спирти альдегіди, кетони і високотоксичні з'єднання, наприклад, синильна кислота.

Для захисту персоналу від дії небезпечних і шкідливих чинників пожежі проектом передбачається застосування промислового протигаза, що фільтрує, з коробкою марки В (жовтий).

Небезпека розвитку пожежі на обчислювальному центрі обумовлюється застосуванням розгалужених систем вентиляції і кондиціювання, розвиненою системою електроживлення ЕОМ. Небезпека загоряння в ЕОМ пов'язана із значною кількістю щільно розташованих на монтажній платі і блоках електронних вузлів і схем, електричних і комутаційних кабелів, резисторів, конденсаторів, напівпровідникових діодів і транзисторів. Висока щільність елементів в електронних схемах призводить до значного підвищення температури окремих вузлів (80…100 °С), що може служити причиною запалювання ізоляційних матеріалів. Слабкий опір

ізоляційних матеріалів дії температури може викликати порушення ізоляції і привести до короткого замикання.

Пожежна безпека при застосуванні ЕОМ забезпечується:

1) системою запобігання пожежі;

2) системою протипожежного захисту;

3) організаційно-технічними заходами.

Запобігти утворенню горючого середовища (замінити горючі речовини і матеріали на негорючі і важкогорючі) не надається технічно можливим. Тому проектом передбачаються способи і засоби запобігання утворення (або внесення) в горюче середовище джерел запалювання, таких як:

1) застосування електроустаткування, відповідної пожежонебезпечної і вибухонебезпечної зонам відповідно до ПУЕ;

2) застосування в конструкції швидкодійних засобів захисного відключення можливих джерел запалення;

3) виключення можливості появи іскрового розряду в горючому середовищі з енергією, рівної і вище мінімальної енергії запалення.

Для протипожежного захисту проектом передбачається використання автоматичну пожежну сигналізацію із застосуванням датчика-сповіщувача РІД-1 (сповіщувач димовий ізоляційний) в кількості 1 шт. і застосуванням первинних засобів пожежогасіння. Площа контрольована сповіщувачем 150 м^2. Відповідно до норм первинних засобів пожежогасінні пропонується використовувати:

–ручний вуглекислий вогнегасник ОУ-5 в кількості 1 шт.

–хімічний пінний ОП-10 – 1 шт;

–повсть 1×1 м^2, кошму 2×1,5 м^2 або азбестове полотно 2×2 м^2 в кількості 1 шт.

В якості організаційно-технічних заходів рекомендується проводити навчання робочого персоналу на тему пожежної безпеки.

Висновок до розділу 5

У розділі "Охорона праці" проаналізовано потенційні небезпеки при роботі з засобами обчислювальної техніки, на підставі якого розроблено заходи з техніки безпеки, заходи, що забезпечують виробничу санітарію та гігієну праці, рекомендації з пожежної профілактики, які підтверджені відповідними розрахунками.

ВИСНОВКИ

У даному дипломному проекті були розроблені інструментальні засоби налагодження прикладних програм мікропроцесорних субкомплексів контролю й управління.

В поданих розділах містяться загальні відомості про досліджуваний об’єкт, розкривається актуальність виконаного завдання. Наведена й описана логічна структура сервісних інструментальних засобів та допоміжних засобів налагодження. Також представлені програмні рішення й реалізація інструментальних засобів налагодження у вигляді алгоритмів. Розроблено інструкцію оператору.

У розділі "Охорона праці" проаналізовано потенційні небезпеки при роботі з засобами обчислювальної техніки, на підставі якого розроблено заходи з техніки безпеки, заходи, що забезпечують виробничу санітарію та гігієну праці, рекомендації з пожежної профілактики, які підтверджені відповідними розрахунками.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. «Програмно-технічні комплекси АСУ ТП: Навч. Посібник-К» - Єлісєєв В.В., Ларгін В.А., Пивоваров Г.Ю. Видавничо-поліграфічний центр «Київський університет», 2003. - 429 с.

2. «Системи контролю та управління технологічними процесами: Збірник наукових статей» Єлісєєв В. В. - Луганськ: Світлиця, 2006. - 440 с.

3. http://www.imp.lg.ua

4. 0229767.40400-03 13 05 Комплекс программ технологического программирования «ЯРУС 4.0». Архив компонент поставки и компонент сопровождения. Сервисные средства отладки

5. 0229767.40400-03 35 02 Комплекс программ технологического программирования «ЯРУС 4.0». Текстовый язык программирования прикладных программ. Описание языка.

6. 0229767.40400-03 35 01 Комплекс программ технологического программирования «ЯРУС 4.0». Описание конфигурации технических и программных средств. Описание языка.

7. 0229767.40400-03 34 01 Комплекс программ технологического программирования «ЯРУС 4.0». Инструментальные средства подготовки целевых управляющих программ и настроек ОБД. Руководство оператора.

8. 0229767.40400-03 13 02 Комплекс программ технологического программирования «ЯРУС 4.0». Архив компонент поставки и компонент сопровождения. Библиотека функциональных блоков

9. 0229767.40400-03 34 02 Комплекс программ технологического программирования «ЯРУС 4.0». Сервисные средства отладки

10. 0229767.40400-03 13 01 Комплекс программ технологического программирования «ЯРУС 4.0». Архив компонент поставки и компонент сопровождения. Резидентное ядро управляющей программы. Описание программ.

11. 0229767.00446-01 30 01 Системное ПО сети МАПС/Ethernet для ОС Windows. Формуляр.

12. 0229767.00447-01 30 01 Системное ПО сети МАПС/Ethernet для ОС Linux. Формуляр.

13. 0229767.00446-01 31 01 Системное ПО сети МАПС/Ethernet для ОС Windows. Описание применения.

14. 0229767.00447-01 31 01 Системное ПО сети МАПС/Ethernet для ОС Linux. Описание применения.

15. 0229767.40400-03 30 01 Комплекс программ технологического программирования «ЯРУС 4.0». Формуляр.

16.0229767.40400-03 13 02 Комплекс программ технологического программирования «ЯРУС 4.0». Архив компонент поставки и компонент сопровождения. Библиотека функциональных блоков. Описание программ.

17. Методичні вказівки до виконання і захисту дипломного проекту (роботи) бакалавра спеціальностей 122 "Комп'ютерні науки та інформаційні технології", 123 "Комп'ютерна інженерія", 125 "Кібербезпека" (за напрямами 6.050101 "Комп’ютерні науки", 6.050102 "Комп’ютерна інженерія") для здобувачів вищої освіти денної і заочної форм навчання / Уклад.: Скарга-Бандурова І.С., Барбарук В.М., Кардашук В.С. – Сєродонецьк: СНУ ім. В. Даля, 2018. – 60 с.

18. ГОСТ 12.0.003-74 ССБТ. Опасные и вредные производственные факторы. Классификация

19. ДСН 3.3.6.042-99 Санітарні норми мікроклімату виробничих приміщень

20. ГОСТ 12.1.005-88 ССБТ.Общие санитарно-гигиенические требования к воздуху рабочей зоны

21. ГОСТ 12.1.003-83. Шум. Общие требования безопасности.

22. ДСН 3.3.6.037-99. Санітарні норми виробничого шуму, ультразвуку та інфразвуку.

23. НАПБ Б. 03.002-2007 «Норми визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою»

Додаток А

Програмна реалізація

Файл main.cpp

```
#include <qapplication.h>
#include <stdlib.h>
#include <qsettings.h>
#include <qdir.h>
#include <qmessagebox.h>
//#include <qsplashscreen.h>
#ifdef _DEBUG
#ifdef WIN32
//#define _CRTDBG_MAP_ALLOC
//#include <mfc32.h>
//#define new DEBUG_NEW
//#include <crtdbg.h>
#endif
#endif
#include "mainwindow.h"
#include "proto.h"
QString MainFontFamily;
QString FixFontFamily;
int MainFontSize;
int FixFontSize;
int segmOffset;
#ifdef __cplusplus
extern "C" {
#endif
unsigned long workMemOffset = 0;
#ifdef __cplusplus
}
#endif
unsigned char verRCore;
Protocol proto;
int main( int argc, char ** argv )
{
    int r;
    QSettings* settings;
    QApplication a( argc, argv );
#ifdef _DEBUG
#ifdef WIN32
//AfxEnableMemoryTracking(TRUE);
// Get current flag
int tmpFlag = 0;/////_CrtSetDbgFlag( _CRTDBG_REPORT_FLAG );
// Turn on leak-checking bit
tmpFlag |= _CRTDBG_LEAK_CHECK_DF;
// Turn off CRT block checking bit
tmpFlag &= ~_CRTDBG_CHECK_CRT_DF;
// Set flag to the new value
/////_CrtSetDbgFlag( tmpFlag );
// +Ё ър фы  юяЁхфхыхэш  ьхёЄр тччютр new {198}
//_CrtSetBreakAlloc(908);
// фы  ASSERT()
/////_CrtSetReportMode(_CRT_ASSERT, _CRTDBG_MODE_FILE);
/////_CrtSetReportFile(_CRT_ASSERT, _CRTDBG_FILE_STDOUT);
#endif
#endif
//    _set_purecall_handler(mypurecall);
    /*QPixmap pixmap( "qt.png" );
    QSplashScreen *splash = new QSplashScreen( pixmap );
    splash->show();
    splash->message( "Loaded obd information" );//*/

```

```
60            settings = new QSettings(QSettings::Ini);
61            settings->setPath( QDir::homeDirPath(), QString("/."__COMPANY__),
62    QSettings::User);
63            settings->beginGroup(QFileInfo(qApp->applicationFilePath()).baseName());
64
65            settings->beginGroup("/mainfonts");
66            MainFontFamily = settings->readEntry("/family", _MAINAPP_FONT_STYLE_);
67            MainFontSize = settings->readNumEntry("/pointSize", _MAINAPP_FONT_SIZE_);
68            settings->endGroup();
69
70            settings->beginGroup("/fixedfonts");
71            FixFontFamily = settings->readEntry("/family", _MY_FONT_STYLE);
72            FixFontSize = settings->readNumEntry("/pointSize", _MY_FONT_SIZE);
73            settings->endGroup();
74
75            settings->beginGroup("/progsettings");
76            segmOffset = settings->readNumEntry("/segm_offset", 0x40);
77        verRCore = settings->readNumEntry("/ver_rcore", 0x1);
78            settings->endGroup();
79
80            qApp->setFont(QFont(MainFontFamily, MainFontSize), TRUE);
81
82            MainWindow w;
83            w.showMaximized();
84        w.connect(&proto, SIGNAL(ProtoAppend(const QString&)), &w,
85    SLOT(ProtoAppend(const QString &)));// connect protocol form
86        QString bmpPath;
87        #ifdef linux
88            bmpPath = "/usr/share/p40400/tool/log.bmp";
89        #else
90            char *asd;
91        QTextCodec::setCodecForLocale(QTextCodec::codecForName("IBM 866"));
92
93            if((asd=getenv(MSKU))==NULL)
94            {
95                QMessageBox::warning(0, ruCodec->toUnicode (_PRG_NAME),
96                         ruCodec->toUnicode ("Отсутствует переменная окружения
97    %1").arg(MSKU),
98                         ruCodec->toUnicode ("Закрыть"),  0,0,1);
99              return -1;
100           }
101           else
102           {
103             qApp->addLibraryPath(QString("%1\\tool").arg(QString(asd)));
104             bmpPath =
105   QString("%1%2TOOL%3%4").arg(QString(asd)).arg(QDir::separator()).arg(QDir::separator(
106   )).arg("log.bmp");
107           }
108       #endif
109       QPixmap p(bmpPath);
110       //p.resize(wspace->size());
111           wspace->setPaletteBackgroundPixmap(p);
112
113           settings->beginGroup("/rs422server");
114       if(cthread)strcpy(cthread->zs.config, QString("dp_super://%1/%2").arg(settings-
115   >readEntry("/ip", "127.0.0.1")).arg(settings->readEntry("/serial",
116   "serial1")).ascii());
117           settings->endGroup();
118           settings->endGroup();
119           delete settings;
120
121           /*splash->finish( &w );
122           delete splash;//*/
```

```
123             a.connect( &a, SIGNAL( lastWindowClosed() ), &a, SLOT( quit() ) );
124             r = a.exec();
125
126             settings = new QSettings(QSettings::Ini);
127             settings->setPath( QDir::homeDirPath(), QString("/."__COMPANY__),
128     QSettings::User);
129             settings->beginGroup(QFileInfo(qApp->applicationFilePath()).baseName());
130
131             settings->beginGroup("/mainfonts");
132             settings->writeEntry("/family", MainFontFamily);
133             settings->writeEntry("/pointSize", MainFontSize);
134             settings->endGroup();
135
136             settings->beginGroup("/fixedfonts");
137             settings->writeEntry("/family", FixFontFamily);
138             settings->writeEntry("/pointSize", FixFontSize);
139             settings->endGroup();
140
141             settings->beginGroup("/progsettings");
142             settings->writeEntry("/segm_offset", segmOffset);
143             settings->writeEntry("/ver_rcore", verRCore);
144             settings->endGroup();
145
146             settings->endGroup();
147             delete settings;
148             return r;
149         }
```

Файл fetherfile.h

```
1         #ifndef  __FETHERFILE_H_
2         #define __FETHERFILE_H_
3         #ifdef WIN32
4         #include <windows.h>
5         #elif defined (linux)
6         #include <errno.h>
7         #endif
8         #include <qthread.h>
9         #include <qevent.h>
10        #include <qmutex.h>
11        #include <qsettings.h>
12        //#include <qvaluevector.h>
13        #include <qvaluestack.h>
14        #include <qsemaphore.h>
15        #include <qwaitcondition.h>
16        //#ifdef WIN32
17        //#ifdef ETHER
18        //#include "./include/win/maps.h"
19        //#else
20        //#include "./include/win/dp_422.h"
21        //#endif
22        //#else
23        //#include "./include/lin/maps.h"
24        //#endif
25        #define NET_SENDANSW
26
27        #include "libnetclass.h"
28        #include "memrw.h"
29
30        extern int segmOffset;
31        #ifdef __cplusplus
```

```
32          extern "C" {
33          #endif
34          extern unsigned long workMemOffset;
35          #ifdef __cplusplus
36          }
37          #endif
38
39          #define EventSetCrNum    QEvent::User + 410
40          #define EventErrorThread  QEvent::User + 420
41          #define EventEndLoad      QEvent::User + 430
42          #define EventErrLoad      QEvent::User + 440
43
44          /*
45          #pragma pack (push, 1)
46          struct stdescriptor
47          {
48                long file_offst;
49                unsigned long size;
50                unsigned long mem_offst;
51                unsigned char crc;
52                unsigned char code;
53                unsigned char _rezerv[2];
54          };
55          struct mdescriptor
56          {
57                unsigned short cnt;
58                struct stdescriptor des[0];
59          };
60          #pragma pack (pop)
61          static unsigned char msg_cnt = 1;
62
63          int sendMesEth(unsigned char* buf, int len, unsigned char addr, unsigned char
64    nummag, unsigned short id = 0xd0d0, bool useMsgCnt = true);
65
66          int sendMesEthOPID(unsigned char* buf, int len, unsigned char addr, unsigned
67    char nummag);
68
69          class cstateThread : public QThread
70          {
71          public:
72                cstateThread(unsigned short _recvto = 250, unsigned short _msgWaitTimeOut
73    = 640):readcnt(0),recvto(_recvto),msgWaitTimeOut(_msgWaitTimeOut), large(false),
74    emula(FALSE), tip(3), mrw(NULL), netLibName(NULL),
75                      mode(0), paused(FALSE), wgtptr(NULL), lwgtptr(NULL), idshk(0),
76    nummag(0), stopped(false), cnt(0), compat(FALSE)
77                {sem = new QSemaphore(32);memset(&zs2, 0, sizeof zs2);memset(&zs, 0,
78    sizeof zs);memset(&isKMPansv, 0, sizeof(isKMPansv));};
79              ~cstateThread(){if(sem) delete sem; sem = NULL; if(mrw) delete mrw; mrw =
80    NULL;};
81              void run();
82              void stop();
83              int checkStart();
84
85              QWidget* wgtptr;
86              QWidget* lwgtptr;
87              unsigned char idshk;
88              unsigned char nummag;
89                unsigned char tip;
90                QString netLibName;
91                memrw *mrw;
92              zagrstruct zs;
93              zagrstruct zs2;
94                      unsigned char isKMPansv[3];
```

```
95                  unsigned int recvto;
96                  unsigned int msgWaitTimeOut;
97                  bool large;
98                  bool emula;
99        #ifdef DEBUG
100                 inline int BlkVectorSize(void) const {return blkVector.count();};
101       #endif
102                 inline int sendMesAnsw(void* send_buf, int slen, void* answ_buf,
103   int alen, int *rlen = NULL) {return (mrw ? mrw->sendMesAnsw(send_buf, slen, answ_buf,
104   alen, rlen) : -1);};
105                 inline int sendMesTo(void* send_buf, int slen, netmaps_address *nma
106   = NULL) {return (mrw ? mrw->sendMes(send_buf, slen, nma) : -1);};
107                 inline int sendMesRecv(void* send_buf, int slen, void* answ_buf,
108   int alen, int *rlen = NULL) {return (mrw ? mrw->sendMesRecv(send_buf, slen, answ_buf,
109   alen, rlen) : -1);};
110       //          void send65(void) {mode = 1; mwk.wakeOne();};
111                 void send65(void) {mwk.wakeOne();};
112                 inline void pause(bool b) {paused = b;if(!paused) mwk.wakeOne();};
113                 inline void CompatMode(bool b) {compat = b;};
114       private:
115                 bool compat;// признак работы в режиме совместимости
116                 bool paused;
117           volatile int mode;
118           volatile bool stopped;
119           int cnt;
120
121                 QValueStack <datastruct> blkVector;
122                 QSemaphore *sem;
123           unsigned char readcnt;
124                 QWaitCondition mwk;
125       };
126
127       //событие для передачи состояния крейтов в основную форму
128       class CEvent_SetCrNum : public QCustomEvent
129       {
130       public:
131          CEvent_SetCrNum():QCustomEvent(EventSetCrNum), numCr(0), colCr(0) {}
132
133           unsigned char getNumCr(){return numCr;};
134           unsigned char getColCr(){return colCr;};
135           void setNumCr(unsigned char n){numCr = n;};
136           void setColCr(unsigned char c){colCr = c;};
137
138       private:
139           unsigned char numCr;
140           unsigned char colCr;
141       };
142
143       class CEvent_ErrorThread : public QCustomEvent
144       {
145       public:
146           CEvent_ErrorThread():QCustomEvent(EventErrorThread){};
147       };
148
149       class CEvent_EndLoad : public QCustomEvent
150       {
151       public:
152           CEvent_EndLoad():QCustomEvent(EventEndLoad){};
153       };
154
155       class CEvent_ErrLoad : public QCustomEvent
156       {
157       public:
```

```
158             CEvent_ErrLoad():QCustomEvent(EventErrLoad){};
159         };
160
161         extern cstateThread* cthread;
162         extern NetworkInterface * instance;
163
164         int CRCCheck(const char *fpath, unsigned long *crc, unsigned long *offset,
165   unsigned short *len);
166
167         int CRCCheck(const NetworkInterface *instance, unsigned char *crc);
168         int CRCCheck(unsigned char idshk, unsigned long crc, unsigned long offset,
169   unsigned short len);
170         #endif
```

Додаток Б

Презентація

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

Дипломний проект на тему:

«ІНСТРУМЕНТАЛЬНІ ЗАСОБИ НАЛАГОДЖУВАННЯ ПРИКЛАДНИХ ПРОГРАМ МСКУ-4»

виконав ст. групи КІ-14ад:
Коверга Марк
Наук. керівник:
доц. Ларгін В. А.

Сєвєродонецьк 2018

1

Розробка ІЗН для прикладних програм МСКУ-4 обумовлена:

- можливістю пошуку, локалізації та усунення помилок у програмах;
- необхідністю впровадження на інженерні станції.

2

Мета:

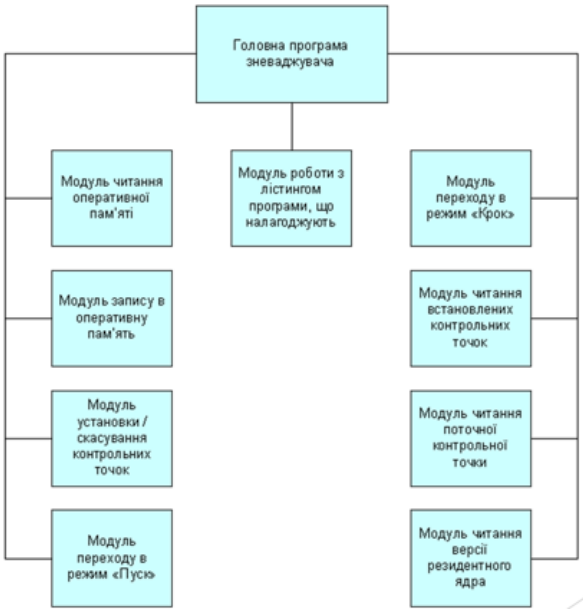
Метою даної дипломної роботи є розробка інструментальних засобів налагодження прикладних програм МСКУ-4 в ПЕОМ.

Основні задачі:

- завантаження КС із прикладної програми;
- читання-запис глобальних змінних оперативної бази даних;
- читання-запис локальних змінних ОБД;
- читання-запис даних оперативної пам'яті МСКУ по фізичним адресам пам'яті й значень названих елементів КС МСКУ по ідентифікаторам елементів;
- передачі в МСКУ на вимогу оператора вихідних повідомлень операцій типу «запит-відповідь», прийом і виведення оператору вмісту відповідних повідомлень від МСКУ на дані операції;
- пуску програми в покроковому, тактовому або безперервному режимі;
- встановлення/скасування контрольних точок зупину;
- контролю й зміни значень вхідних дискретних і аналогових параметрів у режимі діалогу або в режимі сценарію.

3

Логічна структура підсистеми налагодження



4

Інструментальні засоби налагодження

Підсистема представляє собою набір взаємозв'язаних модулів, перелік яких наведено у таблиці нижче.

Ім'я файлу	Ім'я програми	Виконувані функції
main.cpp, mainwindow.ui.h	main(), selectButton_clicked(), selectButton_clickedup(), changeData(), debugPrg()	Головна програма ІЗН, що виконує: -створення потоку для прийому повідомлень з МСКУ; -створення потоку для прийому повідомлень з УП ОД; -виклик основного вікна програми; -виклик програмних модулів для виконання виконання функцій налагодження прикладних програм
fetherfile.cpp, fetherfile.h	openEth(), recvEth(), sendMesEth(), closeEth(), readMemEth()	Функції роботи з МСКУ по мережі Ethernet HP

5

Підсистема представляє собою набір взаємозв'язаних модулів, перелік яких наведено у таблиці нижче.

Ім'я файлу	Ім'я програми	Виконувані функції
fetherfileup.cpp, fetherfileup.h	openEthup(), recvEthup(), sendMesEthup(), closeEthup(), readMemEthup()	Функції роботи з КП ОД по мережі Ethernet HP
simForm.ui.h	procFiles(), KTButton_clicked(), stepButton_clicked(), DbgTimerDone(), VarChg()	Функції режиму налагодження по контрольним точкам
taktform.ui.h	goQuery(), stopQuery(), cancelQuery(), getSostQuery(), setNewTakt(), getVars(), taktForm::scrMode(), taktForm::ScnLoad(), taktForm::ScnNextTakt(), taktForm::CreateScoInit()	Функції режиму «Такт»

6

Підсистема представляє собою набір взаємозв'язаних модулів, перелік яких наведено у таблиці нижче.

Ім'я файлу	Ім'я програми	Виконувані функції
schload.cpp, schload.h	main(), VarAddrGet(), ParamValGet(), readMapFile(), procXmpFile(), ScnFile::Save(), ScnFile::Load()	Функції формування бінарного файлу сценарію
chnk32.c, chnk32.h	ChkWrite(), ChkPass(), ChkRead()	
scnedit.ui.h	XmpLoad, ProjectDirSet, VarInputAdd, VarOutputAdd, fileNew, fileOpen, fileSave	Програма редагування сценаріїв

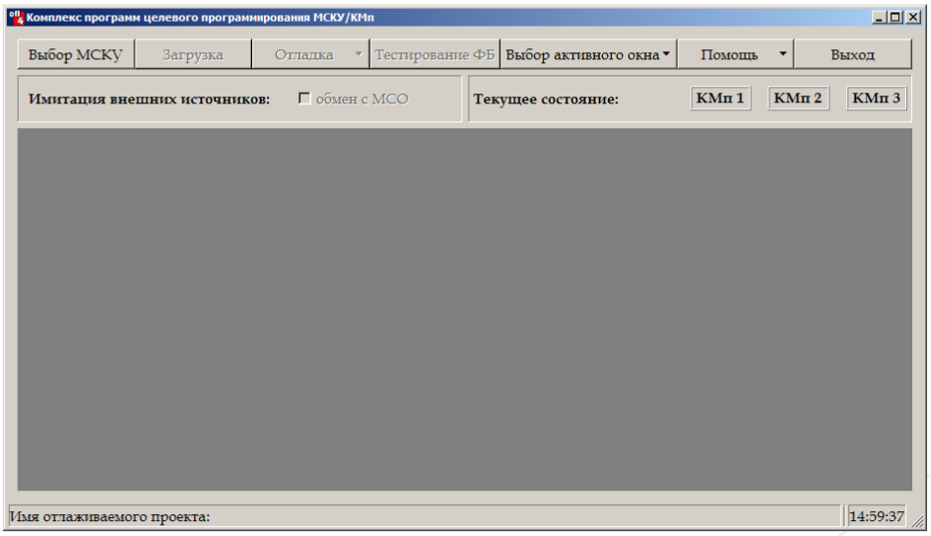
7

Підсистема представляє собою набір взаємозв'язаних модулів, перелік яких наведено у таблиці нижче.

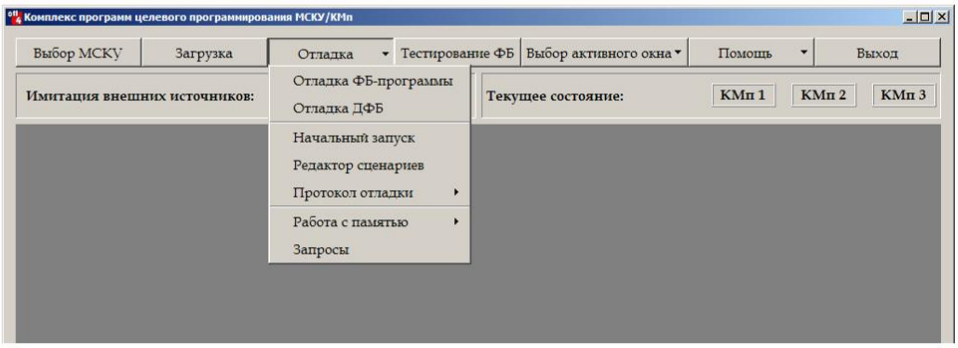
Ім'я файлу	Ім'я програми	Виконувані функції
scnverif.c	scnMode.scnLoad, CompareProto	Програма виконання сценарію налагодження й формування звіту порівняння протоколу налагодження з еталоном
prt_diff.c	main	Програма порівняння протоколу результату тестування функціональних блоків із зразками

8

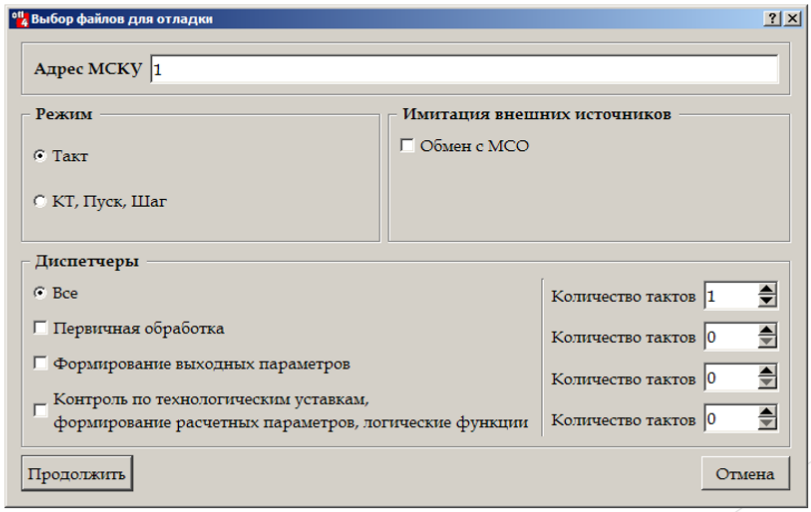
Вид основного кадру, що видається при запуску зневаджувача



Вид вікна з розкритими пунктами меню «Налагодження»



Вид вікна, виведеного на екран при виборі пункту меню«Налагодження» -> «Налагодження ФБ-програми»



Висновки

У даному дипломному проекті були розроблені інструментальні засоби налагодження прикладних програм мікропроцесорних субкомплексів контролю й управління.

Містяться загальні відомості про досліджуваний об’єкт, розкривається актуальність виконаного завдання. Наведена й описана логічна структура сервісних інструментальних засобів та допоміжних засобів налагодження. Також представлені програмні рішення й реалізація інструментальних засобів налагодження у вигляді алгоритмів.