НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:

Завідувач кафедри

_____ Сергій СТІРЕНКО

«___»_____2024 р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Комп'ютерна інженерія»
спеціальності 123 «Комп'ютерна інженерія»
на тему: «Апаратно-програмний комплекс експрес-контролю стану двигунів літака»

Виконала:
студентка IV курсу, групи ІО-з01
Кузьменко Юлія Андріївна _____

Керівник:
доцент, к.т.н.
Марковський О.П. _____

Консультант (нормконтроль)
ст. викладач
Виноградов Ю.М. _____

Рецензент:
Декан ФПМ, професор, д.т.н.
Дичка І.А. _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студентка Кузьменко Ю.А. _____

Київ – 2024 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Комп'ютерні інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій СТІРЕНКО
«___»_____2024 р.

ЗАВДАННЯ
на дипломний проєкт студентці
Кузьменко Юлії Андріївни

1. Тема проєкту «Апаратно-програмний комплекс експрес-контролю стану двигунів літака», керівник проєкту Марковський Олександр Петрович, доцент, к.т.н., затверджені наказом по університету від «31» травня 2024 р. № 2101-с

2. Термін подання студентом проєкту: "10" червня 2024 р.

3. Вихідні дані до проєкту: *див.технічне завдання*

4. Зміст пояснювальної записки: Завдання комп'ютерного тестування авіаційних двигунів, огляд наявних систем та обґрунтування вибору структурної схеми. Розробка функціональної схеми. Розробка алгоритмів і програм роботи.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо): Система моніторингу технологічного процесу. Процедура формування сигналів управління. Контролер тестування авіаційних двигунів.

6. Консультант проєкту, з вказівкою розділів проєкту, які до них вносяться

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормконтроль	Виноградов Ю.М.		

7. Дата видачі завдання “20” грудня 2024 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Затвердження теми проєкту	20.12.2023-25.12.2023	
2.	Вивчення та аналіз завдання	25.12.2023-16.03.2024	
3.	Розробка архітектури та загальної структури системи	16.03.2024-26.03.2024	
4.	Розробка алгоритмів роботи	26.03.2024-07.04.2024	
5.	Програмна реалізація системи	7.04.2024-17.04.2024	
6.	Оформлення пояснювальної записки	17.04.2024-20.05.2024	
7.	Захист програмного продукту	25.04.2024	
8.	Предзахист	23.05.2024	
9.	Захист		

Студентка Юлія КУЗЬМЕНКО

Керівник Олександр МАРКОВСЬКИЙ

АНОТАЦІЯ

Дипломний проект присвячено розробці комп'ютерної системи тестування авіаційних двигунів. Пристрій для тестування спроектований на базі PIC мікроконтролера, який видає керуючі сигнали на канали управління двигуном та приймає сигнали з датчиків. Особливістю розробки є аналіз динамічних характеристик авіаційних двигунів. Розробка виконана до рівня принципових схем. Була розроблена управляюча програма. Розраховані показники надійності та технічні характеристики.

ANNOTATION

Diploma project devoted to developing computer system for testing of aeroengine. The testing device has been designed on the base of PIC microcontroller, which issues control signals for aero-engine control and accept the signal from pickups. Feature of development is that the implementation of encryption software is analysis of dynamic characteristics of aero-engine. The elaboration has been done up to principal schemes level. The control program was developed. The parameters and reliability specifications. has been calculated.

Номер рядка	Формат	Позначення	Найменування	Кільк. аркушів	Номер екзем.	Примітка
1			Документація загальна			
2			Знову розроблена			
3						
4	А4	ІАЛЦ 468243.002 ТЗ	Апаратно-програмний	5		
5			комплекс експрес-контролю			
6			стану двигунів літака.			
7			Технічне завдання			
8						
9	А4	ІАЛЦ 468243.003 ПЗ	Апаратно-програмний	65		
10			комплекс експрес-контролю			
11			стану двигунів літака.			
12			Пояснювальна записка			
13						
14	А1	ІАЛЦ 468243.004 Е1	Система моніторингу	1		
15			технологічного процесу.			
16			Схема електрична структурна			
17						
18	А1	ІАЛЦ. 468243.005 Д1	Процедура формування	1		
19			сигналів управління.Схема			
20			алгоритму роботи			
21						
22	А1	ІАЛЦ. 468243.006 Е3	Контролер тестування	1		
23			авіаційних двигунів. Схема			
24			Електрична принципова			
25						
26						
27						
28						

Зм.	Аркуш	№ докум.	Підпис	Дата	ІАЛЦ.468243.001 ОА		
Розробила		Кузьменко Ю.А.			Апаратно-програмний комплекс експрес-контролю стану двигунів літака. Опис альбому	Літ.	Аркуш 1 / Аркушів 1
Керівн.		Марковський О.					
Н.контр.		Виноградов Ю.				КПІ ім. Ігоря Сікорського ФІОТ, ІО-з01	
Затв.							

Технічне завдання до дипломного проєкту на тему: “Апаратно-програмний комплекс експрес-контролю стану двигунів літака”

Київ - 2024 р.

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Технічне завдання поширюється на розробку апаратно-програмного комплексу експрес контролю стану двигуна літака.

Область застосування – системи технічної діагностики та контролю стану авіаційних газотурбінних двигунів цивільних та військових літаків, що забезпечують їх безпечне функціонування.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання бакалаврського проєкту по освітньо-професійній програмі “Комп'ютерні системи та мережі” спеціальності 123 “Комп'ютерна інженерія”, затверджене кафедрою Обчислювальної техніки Національного технічного Університету України “Київський Політехнічний інститут імені Ігоря Сікорського”.

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Мета проекту полягає в підвищенні надійності та повноти контролю стану газотурбінних авіаційних двигунів з використанням сучасних комп’ютерних засобів в процесі передполітної підготовки для забезпечення безпеки транспортних перевезень повітряним транспортом за рахунок можливості аналізу суттєво більших об’ємів інформації про поточний стан вузлів та агрегатів двигуна.

Розробка призначена для проведення експрес контролю стану вузлів та агрегатів авіаційних двигунів безпосередньо перед польотом в процесі їх запуску шляхом комп’ютерного аналізу акустичних сигналів в різних точках двигуна.

4. ДЖЕРЕЛА РОЗРОБКИ

4.1 Дмитрієв С.О. Методика применения акустического метода идентификации неисправностей ТРДД / С.О. Дмитрієв, Ю.М.Чоха // Науч.-техн.журн. ”Вісник двигунобудування”. – Запоріжжя.: ЗНТУ, 2004, №2. – С.173-176.

4.2 Дмитрієв С.О. Методичні основи діагностування турбореактивного двоконтурного двигуна з використанням засобів штучного інтелекту/ Дмитрієв С.О., Попов О.В., Якушенко О.С., Потапов В.Е.// Авіаційно-космічна техніка і технологія. – Х.: – 2015. – № 5/122. – С. 69 - 73

4.4 Лозня С.В. Автоматизированная диагностика неисправностей газотурбинных двигателей / С.В. Лозня, М.И. Торхов, С.А.Пустовой, В.А. Седристий, Ю.В.Черкасов // Науч.-техн.журн. ”Вісник двигунобудування”. – Запоріжжя.: ЗНТУ, 2008, №3. – С.176-181.

4.4 Мазиди А.М. Микроконтроллеры PIC и встроенные схемы / А.М. Мазиди, Р. МакКинли, Д. Кусэй// К.: МК-Пресс.- 2009.- 784 с.

4.5 Барри Б. Применение микроконтроллеров PIC-18. Архитектура, программмирование и построение интерфейсов. К.: МК-Пресс.- 2008.- 576 с.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до продукту, що розробляється

5.1.1 Апаратно засоби експрес-контролю повинні включати такі основні вузли: мікроконтролер; блок введення характеристик роботи двигунів; блок видачі управляючих сигналів; вбудований блок живлення; вузол виведення і візуалізації інформації; вузол сполучення з USB базового комп’ютера.

5.1.2 Апаратно-програмні засоби контролю стану двигуна повинні реалізувати функції завдання режиму тестування, видачу керуючих сигналів на розгінний електродвигун турбіни і форсунки вприску палива згідно з режимом тестування. З часовим інтервалом 0.2 с. мають проводитися вимірювання миттєвих значень тиску на вході в турбіну, числа її обертів, тиск на вході камери згоряння, температури в камері згоряння і тиску на виході обох контурів авіаційного двигуна.

5.1.3 Апаратно-програмні засоби контролю стану двигуна повинні виконувати попередній аналіз стану авіаційного двигуна з видачею рекомендацій по вибору оптимальних режимів його роботи, видачею діагностики роботи його вузлів і необхідності в проведенні регламентних робіт. Також має забезпечуватися видача нформації про стан авіаційного двигуна в базовий комп'ютер для її подальшої обробки та аналізу.

5.2. Вимоги до програмного забезпечення

- Операційна система MS Windows 10
- Visual Studio 2017
- C++11

5.3. Вимоги до апаратної частини

- Мікропроцесор PIC - 18
- Процесор рівня Intel i5 і вище.
- Оперативна пам’ять не менше 500 МБ.
- Вільне місце на жорсткому диску не менше 100 МБ.

6. ЕТАПИ РОЗРОБКИ

	Дата
Вивчення літератури	20.12.2023
Створення та узгодження технічного завдання	15.01.2024
Вивчення літературних джерел	27.01.2024
Розробка апартної частини проекту	14.02.2024
Розробка алгоритмів та програм	01.05.2024
Відлагодження програми та виправлення помилок	15.05.2024
Оформлення документації дипломного проекту	04.06.2024

Пояснювальна записка
до дипломного проєкту
на тему: «Апаратно-програмний комплекс експрес-контролю стану двигунів літака»

Київ – 2024 року

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ - програмне забезпечення;

ЛА - летальный апаратів;

ГТД - газотурбінний двигун;

АЦП - аналого-цифрових перетворювачів;

АСУ ТП - автоматизована система управління технологічним процесом

ВСТУП

Початок XXI-го століття знаменується динамічним розвитком засобів авіаційного транспорту. За добу по всьому світу здійснюється близько 80 тисяч польотів пасажирських літаків, тобто щосекунди, в середньому, виконується зліт одного літака. При цьому, незважаючи на потенційний ризик, пов'язаний із виконанням польотів, авіаційний транспорт залишається одним із найнадійніших: серйозна аварія трапляється, в середньому, в одному з 2 млн. польотів.

Зазначений стан аварійності під час виконання авіаційних перевезень забезпечується постійно діючим комплексом заходів, спрямованих на мінімізацію ризику відмов технічних засобів літаків. До теперішнього часу на більшості літальних апаратів (ЛА) життєво важливі технічні системи дублюються, здійснюється постійний контроль за їхнім станом, суворо виконується комплекс профілактичних і регламентних робіт, спрямованих на безвідмовну роботу авіаційної техніки. Необхідно відзначити ефективність вжити х заходів для забезпечення безвідмовної роботи технічних компонентів: у сучасному літаку використовується до півтора мільйона деталей і компонентів, водночас частка катастроф, викликаних відмовою технічних компонентів, не перевищує 25% (60% припадає на людський чинник - помилки пілотів і диспетчерів, 10% - погодні умови, а 5%, що залишилися, - на інші).

Незважаючи на досягнуті успіхи в забезпеченні надійності авіаційних перевезень, існує необхідність у постійному вдосконаленні методів і засобів забезпечення безвідмовної роботи технічних комплексів сучасних ЛА. Обсяг уже укладених до теперішнього часу контрактів на постачання авіаційної техніки свідчить про те, що в найближчі 5 років обсяг перевезень, здійснюваних авіаційним транспортом, зросте на 30%.

Ще більш значне (до 8% на рік) збільшення обсягу перевезень прогнозується для вантажних авіаційних перевезень. При цьому слід врахувати, що більша частина таких перевезень здійснюється не на літаках спеціальної споруди, а на

пасажирських літаках, перероблених на вантажні. Статистика показує, що для таких ЛА, які мають значний наліт, збільшується потенційна небезпека виникнення технічних відмов. Тому, з огляду на наведені обставини, проблема забезпечення безвідмовної роботи складних технічних комплексів сучасних літальних апаратів є актуальною та практичною важливою.

Серед складного комплексу заходів і засобів забезпечення безвідмовної роботи технічних систем літаків і вертольотів важливе місце посідають системи контролю та тестування. Такі системи забезпечують виявлення як відмов, що виникають, так і потенційних передумов до відмов шляхом своєчасного контролю змін характеристик технічних засобів. Останній вид контролю є особливо важливим, оскільки дає змогу виявити потенційну небезпеку відмов за непрямими ознаками.

Розрізняють два види систем контролю: бортові, тобто ті, що знаходяться прямо на борту ЛА, і наземні. Основною перевагою бортових систем контролю є автономність, можливість виконання контролю як на землі, так і безпосередньо в польоті. Бортові системи контролю технічних засобів дають змогу оцінити параметри їхньої роботи безпосередньо в польоті, тобто з урахуванням впливу повітряного напору, різниці тиску, температури і діючих прискорень. Однак великим недоліком бортових систем контролю є обмеження на габарити та вагу, обмеження на можливий діапазон тестувальних впливів, а також на арсенал вимірювальних засобів, що використовується.

Наземні системи контролю технічних засобів дають змогу реалізувати ширшу програму тестування, вимірювати параметри, які неможливо виміряти безпосередньо в польоті (ультразвукове зондування деталей і несучих конструкцій ЛА). Важливість наземних засобів контролю зростає з триваючим в авіаційній техніці процесом уніфікації використовуваних технічних засобів. Це дає змогу проводити більш поглиблений аналіз параметрів технічних засобів без прив'язки до конкретного ЛА.

Особливо важливу роль для забезпечення безпеки польотів відіграє справність авіаційних двигунів. Згідно з даними статистики на них припадає до

75% відмов агрегатів ЛА. Агрегати сучасних газотурбінних двигунів працюють в умовах високих температур і екстремальних навантажень. Агрегати газотурбінних двигунів розташовані в негерметизованій частині літаків і безпосередньо схильні до кліматичних впливів, великих перепадів температур, забруднення частинками, які перебувають у повітряному просторі. Це може призвести до прискореного зносу механічних деталей, корозійних явищ, втрати мастилом своїх властивостей. Загальною властивістю відмов агрегатів авіаційних двигунів є те, що виникненню безпосередньої відмови передує зміна характеристик їхньої роботи, які можуть бути виявлені в процесі контролю та тестування.

Виходячи з цього, важливим і актуальним у теперішніх умовах видається розробка комп'ютеризованих засобів контролю характеристик газотурбінних авіаційних двигунів. Сучасні мікроконтролери дають змогу створювати гнучкі та доволі потужні системи контролю, що дають змогу з високою достовірністю й оперативно виявляти потенційні відмови. При цьому можна забезпечити компактність засобів контролю.

Розв'язанню цього важливого технічного завдання присвячено досправжній дипломний проєкт, спрямований на розроблення комп'ютеризованої системи тестування динамічних характеристик авіаційних двигунів.

РОЗДІЛ 1
ЗАВДАННЯ КОМП'ЮТЕРНОГО ТЕСТУВАННЯ АВІАЦІЙНИХ ДВИГУНІВ, ОГЛЯД НАЯВНИХ СИСТЕМ ТА ОБҐРУНТУВАННЯ ВИБОРУ СТРУКТУРНОЇ СХЕМИ

1.1 Завдання тестування авіаційних двигунів

Одним із ключових напрямів у покращенні безпеки та регулярності польотів авіаційної техніки є удосконалення структури та логічної організації експлуатаційно-технічної діагностики, а також її процесів, що спрямовані на ефективне раннє виявлення передвідмовних станів високонавантажених компонентів літальних апаратів. Цей аспект є основою методології діагностики. Безпека використання авіаційної техніки значною мірою залежить від надійності, закладеної під час проектування та виробництва, а також від ефективності методів і засобів діагностики технічного стану авіатехніки, які дозволяють своєчасно виявляти несправності та передвідмовні стани, що виникають під час експлуатації.

Планер, двигун та функціональні системи авіаційної техніки постійно піддаються якісним змінам. Ці зміни визначаються другим законом термодинаміки, який стверджує, що впорядковані системи, до яких належать усі технічні пристрої, з часом мають тенденцію самовільно руйнуватися, втрачаючи початковий порядок, закладений під час їх створення. Ця тенденція виникає під впливом численних дезорганізаційних чинників, які не можуть бути повністю враховані при проектуванні та виготовленні авіаційної техніки. Тому процеси зміни якості здаються нерегулярними та випадковими, а їхні наслідки часто є непередбачуваними.

Удосконалення систем діагностики авіаційної техніки вимагає інтеграції новітніх технологій і методів, таких як:

1. Моніторинг стану в реальному часі: Використання датчиків і систем моніторингу для безперервного контролю стану критичних компонентів. Це включає в себе технології Інтернету речей (IoT), які дозволяють збирати дані з численних точок в реальному часі та аналізувати їх для виявлення аномалій.

2. Прогнозування та аналітика на основі великих даних (Big Data): Аналіз великих обсягів даних, зібраних під час експлуатації літаків, дозволяє виявляти патерни та тенденції, які передують відмовам. Використання машинного навчання і штучного інтелекту (AI) може значно підвищити точність прогнозів та швидкість реакції на потенційні проблеми.

3. Неперервне удосконалення алгоритмів діагностики: Включення адаптивних алгоритмів, які можуть самооновлюватися на основі нових даних і зворотного зв'язку з експлуатації. Це дозволяє системам діагностики ставати більш точними та надійними з часом.

4. Моделювання та симуляція: Використання комп'ютерних моделей для симуляції різних сценаріїв експлуатації та відмов. Це допомагає в ідентифікації потенційних проблем ще на етапі проектування та у визначенні оптимальних стратегій технічного обслуговування.

5. Використання новітніх матеріалів та технологій: Інноваційні матеріали, такі як композити та легкі сплави, можуть збільшити надійність та зменшити вагу авіаційної техніки, що в свою чергу впливає на її експлуатаційні характеристики і довговічність. Технології 3D-друку також дозволяють створювати складні деталі з високою точністю, що покращує їхню надійність.

6. Система управління безпекою (Safety Management System, SMS): Інтеграція системи управління безпекою, яка включає в себе політики, процеси та процедури для управління ризиками, пов'язаними з технічною експлуатацією. SMS забезпечує систематичний підхід до моніторингу, аналізу та вдосконалення процедур безпеки.

7. Впровадження цих інновацій може значно підвищити ефективність діагностики технічного стану авіаційної техніки та забезпечити вищий рівень безпеки і надійності польотів.

Альтернативним підходом є використання фізичних методів діагностики, до яких відносяться відомі методи оптико-візуального контролю, трибодіагностики, аналізу продуктів згоряння, діагностика за віброакустичними параметрами та контроль термогазодинамічних параметрів. Однак постійно виникає питання: яке

поєднання методів діагностики дозволяє швидко та точно попередити відмову? Це питання досі не має вичерпної відповіді. Часто трапляються випадки необґрунтованого зняття двигунів з експлуатації або, що ще гірше, пропуску дефектів через неправильно поставлений діагноз, зазвичай пов'язаних з похибками в обробці діагностичної інформації або помилками під час її аналізу (так званий людський фактор). Крім того, не повністю розкрито інформаційний потенціал контрольованих параметрів, які містять важливу інформацію про об'єкт діагностики. Важливо звернути увагу на термін "інформаційний потенціал", що означає невикористану можливість врахування інформаційної значущості як контрольованих параметрів, так і методів діагностики, що дозволяє точніше визначити стан об'єкта та швидше наблизитися до виявлення дефекту.

Аналіз результатів досліджень, проведених у [1], продемонстрував, що несправності, які у своєму розвитку можуть призвести до руйнування газотурбінного двигуна (ГТД), можна умовно поділити на три групи:

а) несправності, які дуже швидко (протягом часток секунди або декількох секунд) переходять в аварію двигуна, або, що майже те саме - несправності, які занадто пізно виявляються за допомогою доступних засобів діагностики. До цієї групи входять, наприклад, "розкрутка" вала вільної турбіни двигуна, виникнення негативного крутного моменту на валу турбогвинтових двигунів, помпаж тощо;

б) несправності, здатні розвиватися в аварію протягом кількох хвилин, а також несправності, характер і темп розвитку яких не можна достовірно передбачити на основі досягнутого рівня знань. Виникнення подібних несправностей має супроводжуватися негайною видачею сигналу екіпажу літака (або персоналу випробувального стенда) для привернення уваги, оцінки ситуації та вжиття необхідних заходів. З цією метою двигуни забезпечуються певним набором аварійних сигналізаторів (пожежі, падіння тиску мастила, появи "стружки" в мастилі тощо).

в) несправності, що розвиваються відносно повільно або виявляються наявними діагностичними засобами на такій ранній стадії, що перехід їх в аварію в продовження даного польоту можна вважати практично виключеним. Раннє

виявлення саме таких несправностей і становить основу прогнозування станів двигунів.

Часовий інтервал від появи першого симптому несправності до її небезпечного розвитку визначається не стільки фізичними характеристиками самої несправності, скільки рівнем наших знань про її причини, ознаки та процеси розвитку. Зі зростанням таких знань та появою відповідної апаратури деякі види руйнування, наприклад, зубчастих передач і підшипників, перестали вважатися "раптовими" і стали прогнозованими.

Одне з практичних завдань досліджень динаміки розвитку несправностей ГТД полягає в тому, щоб максимально скорочувати кількість несправностей першої та другої груп і поступово "переводити" їх у третю групу, розширюючи таким чином можливості раннього діагностування та довгострокового прогнозування стану ГТД. Високий ступінь попередження діагнозу не тільки підвищує безпеку польотів, а й сприяє істотному зниженню експлуатаційних витрат, пов'язаних із порушенням регулярності польотів, ремонтом ГТД.

Досвід експлуатації газотурбінних двигунів (ГТД) показує, що для правильної постановки діагнозу спочатку потрібно знати всі можливі стани ГТД, враховуючи попередні статистичні дані та ймовірності виникнення різних ситуацій, а також набір діагностичних ознак, які реагують на ці стани. Як вже зазначалося, процес зміни технічних властивостей авіаційного ГТД відбувається постійно, що означає, що множина можливих його станів нескінченна і навіть незліченна. Завдання полягає в тому, щоб розділити цю множину станів на обмежену і невелику кількість класів станів, де кожен клас об'єднує стани з однаковими властивостями, обраними для класифікації. При цьому статистична база параметрів, отриманих за допомогою згаданих методів діагностики, повинна бути об'єктивною і достовірною.

Не всі параметри, які можуть бути використані в діагностиці, рівноцінні за змістовністю відомостей про функціонуючий ГТД. Одні з них приносять інформацію відразу про багато властивостей працюючих модулів двигуна, інші, навпаки, вкрай бідні. Безумовно, перевагу слід віддавати діагностичним

параметрам, що мають флуктуаційний характер, а не тим, які постійні або змінюються дуже повільно. Наприклад, шум ГТД і його вібрація за кількістю привнесеної інформації мають велику перевагу перед такими стійкими інертними сигналами, як температура охолоджувальної рідини, швидкість обертання вала та ін., хоча ці параметри так само, як шум і вібрація, залежать від стану ГТД, що працює. Тому, на другому етапі цікавим видається розглянути взаємозв'язок діагностичних параметрів, їхню зміну та можливий вплив один на одного, а також оцінити значущість ознак різних функціональних параметрів ГТД.

Відомо, що теорія постановки діагнозу добре описується загальною теорією зв'язку, яка є частиною теорії управління [1]. Для діагностики можна використовувати математичний і логічний апарати, систему освоєних понять і термінологію. Потрібно лише знайти фізичне пояснення абстрактних формул і способи практичного застосування запропонованих підходів. Отже, на третьому етапі слід підтвердити значущість діагностичних ознак, використовуючи відомі принципи інформаційної теорії, і на основі цього сформувати діагноз, а також здійснити прогноз передвідмовних станів. Ця частина роботи є найскладнішою, оскільки авіаційний двигун є багатопараметричною системою, але не всі параметри є однаково важливими (інформативними) в різних умовах.

1.2 Огляд засобів тестування та діагностики авіаційних двигунів

З огляду на важливість проблеми раннього діагностування несправностей авіаційних двигунів, до теперішнього часу в усіх країнах-виробниках авіаційних двигунів (США, Канада, Англія, Франція, Німеччина та Україна) розробляють системи їхнього тестування.

В контексті тестування авіаційних двигунів, важливим аспектом є застосування сучасних технологій для підвищення точності та ефективності діагностики. Розрізняють три різновиди таких систем. Кожен з трьох типів систем тестування має свої особливості та переваги, які допомагають у різних етапах життєвого циклу авіаційних двигунів.

1. Стендові системи:

- Комплексне тестування: Стендові системи дозволяють проводити всебічне тестування двигуна, включаючи вібраційний аналіз, термодинамічні випробування та оцінку продуктивності. Це допомагає виявити навіть найменші дефекти на ранніх стадіях.

- Технологічні рішення: Наприклад, комплекс "Салют" не лише вимірює вібрації, а й аналізує їхню природу, що дозволяє більш точно діагностувати причини можливих несправностей.

2. Рухомі аеродромні системи:

- Гнучкість та мобільність: Ці системи забезпечують швидку діагностику двигунів без необхідності демонтажу з літального апарату. Вони використовуються для проведення регламентних робіт, передпольотних перевірок та у випадку підозри на несправність.

- Приклади систем: Рухомі аеродромні комплекси часто включають портативні пристрої для аналізу вібрацій, тепловізори для виявлення перегрівів та інші інструменти для неінвазивної діагностики.

3. Бортові системи:

- Неперервний моніторинг: Бортові системи забезпечують постійний контроль за станом двигуна під час польоту. Вони інтегруються в систему управління літаком, надаючи екіпажу актуальну інформацію про технічний стан.

- Розвиток технологій: Сучасні бортові системи використовують методи предиктивної аналітики та машинного навчання для прогнозування можливих відмов ще до їх виникнення. Вони можуть виявляти тенденції, що вказують на деградацію компонентів двигуна.

Крім того, важливою є інтеграція систем діагностики з загальною системою управління технічним обслуговуванням (Maintenance Management System). Це дозволяє автоматизувати процеси планування ремонтів, управління запасами запасних частин та оптимізації робочих процесів.

Застосування сучасних матеріалів та технологій, таких як адитивне виробництво (3D-друк) для виготовлення та ремонту компонентів двигунів, також

є важливим напрямом розвитку. Це дозволяє створювати деталі з підвищеною точністю та зменшувати час простою техніки.

Важливість міжнародної співпраці в розробці та вдосконаленні систем діагностики також не можна переоцінити. Спільні дослідження та обмін досвідом між країнами-виробниками сприяють швидшому впровадженню інновацій та підвищенню загального рівня безпеки авіаційної техніки.Найвідомішою зі стендових систем, що використовуються в Росії і на Україні, є комплекс "Салют". Цей комплекс розрахований на дослідження вібрацій, створюваних агрегатами авіаційного двигуна.

Початково, для зниження загальної вібрації двигунів, використовували комплексний підхід, спрямований на визначення діагностичних ознак, що вказували на ті складові частини або деталі, вплив на які міг би покращити вібраційну поведінку двигуна. Вібраційна діагностика авіаційних газотурбінних двигунів на стендових випробуваннях мала свої особливості, оскільки оцінювалися не зношення двигуна або поява та розвиток дефектів, а технологічні відхилення під час виготовлення та складання двигуна. Тому вібраційна діагностика для кожного типу двигуна мала індивідуальний підхід. Під час вирішення цієї задачі була створена статистично усереднена спектральна модель двигуна, яка визначала діагностичні стандарти для амплітуд вібрацій окремих компонентів і агрегатів двигуна. Одночасно зі створенням моделі проводився пошук і накопичення діагностичних ознак - характерних особливостей вібраційних сигналів, пов'язаних з особливостями складання конкретного екземпляра двигуна.

Використання комплексу дало змогу також внести низку уточнень у технологію виготовлення двигунів. Так, наприклад, введено додатковий контроль дисбалансу окремих шестерень і агрегатів. Вібраційний стан усіх двигунів, що проходять пред'явницькі та приймально-здавальні випробування, контролюється і реєструється з використанням стендового комплексу вібраційної діагностики. У процесі експлуатації комплексу в міру необхідності нарощували функціональні можливості. Для того щоб відрізняти форсажні режими від безфорсажних, у конфігурацію вимірювальної системи комплексу було введено канал вимірювання

αРУД - кута положення важеля керування двигуном. Коли було поставлено завдання створення статистично усередненого спектра двигуна, до програмного забезпечення спектроаналізатора додали функцію передавання набору спектральних амплітуд в EXEL. В останній версії програмного забезпечення додано можливість автоматизованого протоколювання результатів випробувань. Цей програмний компонент дуже корисний у разі, коли необхідно контролювати одну або кілька спектральних амплітуд під час випробувань. Також розширено можливості конфігурування вимірювальної системи, що дають змогу не тільки встановити параметри реєстрації вібрацій за каналом вимірювань, а й одразу визначити набір контрольованих під час проведення випробувань амплітуд спектрів вібрацій двигуна.

Стендовий комплекс вібраційної діагностики авіаційних ГТД призначено для контролю вібрацій газотурбінних двигунів під час випробувань, а також для реєстрації вібраційного стану двигунів з метою подальшого аналізу та вироблення рекомендацій для вібраційного доведення двигуна. Велика частина функціональних можливостей комплексу забезпечується в автоматичному режимі без участі оператора, тому для роботи з комплексом практично немає необхідності мати спеціально підготовлений персонал. Мінімально від оператора потрібно запустити програмне забезпечення на виконання, задати номер двигуна, вид випробувань. Далі забезпечується автоматичний запуск засобів контролю вібрацій і реєстрації широкосмугових вібраційних сигналів одночасно із запуском двигуна. Також автоматично припиняється робота засобів контролю та реєстрації разом із зупинкою двигуна. На екрані дисплея постійно ведеться відображення вібраційного стану двигуна та режиму його роботи. У графічній і цифровій формі показано рівні вібрацій двигуна, частоти обертання турбіни. За необхідності можна безпосередньо стежити за конкретними спектральними складовими за будь-яким каналом вимірювань. Забезпечено видавання повідомлень, що характеризують ступінь перевищення контрольованих вібраційних параметрів порівняно з допустимим рівнем.

Автоматизована реєстрація вібраційних сигналів є однією з ключових особливостей цього комплексу. Відомо, що неперервний запис цифрових широкосмугових вібраційних сигналів за багатьма каналами вимірювання протягом півгодини-години випробувань потребує великих обсягів дискової пам'яті і значно уповільнює подальшу обробку. Оскільки реально використовувана частина цих даних для аналізу вібраційного стану двигуна є досить мала, примусове включення реєстрації широкосмугового буфера в певні моменти може призвести до великого впливу суб'єктивного фактора і, як наслідок, до неточних результатів аналізу.

Найефективнішим виявився режим напівавтоматичної реєстрації. Під час випробувань проводиться безперервна реєстрація значень частот обертання роторів двигуна і вібраційних характеристик для кожного каналу вимірювання згідно з технічними вимогами відносно вібрацій двигуна, тобто створюється циклограма випробувань.

Після завершення випробувань перегляд цієї інформації дозволяє зробити загальну оцінку вібраційної активності двигуна та визначити, чи потрібно проводити подальший аналіз вібрацій для діагностики. Реєстрація широкосмугових вібраційних сигналів відбувається в автоматичному режимі, що дозволяє зберігати дані тільки при стаціонарних режимах роботи двигуна в достатньому обсязі для проведення надійного аналізу. Таким чином, можна отримати всю необхідну інформацію для спектрального аналізу та діагностики, оптимізуючи при цьому обсяг даних, що реєструються під час випробувань.

Додатково автоматично вмикається режим безперервної реєстрації ("цифровий магнітофон") у двох випадках: по-перше, у разі перевищення будь-яких контрольованих вібраційних характеристик двигуна, по-друге, - на режимі вибігу, починаючи з моменту падіння частот обертання роторів нижче за малий газ і до повної зупинки роторів двигуна.

Стендовий комплекс оснащений вбудованими засобами спектрального аналізу та перегляду осцилограм. На відображуваному спектрі відзначаються амплітуди основних гармонік і максимально допустимі їхні рівні (якщо

визначено). Додатково за відповідних налаштувань програмного забезпечення спектральний аналіз в автоматичному режимі ведеться безперервно з метою обчислення і контролю заданих спектральних характеристик, що описують технічний стан і якість виготовлення окремих вузлів і агрегатів двигуна. Використання функцій спектроаналізатора в стендовому комплексі додатково дає змогу в процесі випробувань діагностувати такі явища, як, наприклад, відмова датчика вібрацій або обрив лінії зв'язку.

Після закінчення випробувань двигуна оператор може отримати протокол. Протокол випробувань - один із засобів автоматизації, він дає змогу в низці випадків скоротити час на опрацювання та аналіз результатів випробувань. Протокол випробувань формується в автоматичному режимі і являє собою таблицю усереднених значень вібраційних характеристик двигуна для всіх режимів його роботи. Формат таблиці - набір вібраційних характеристик, що входять до неї, - визначається заздалегідь. Може бути підготовлено кілька варіантів різних форматів, один з яких завантажується перед підготовкою протоколу.

Структурна схема комплексу, яка показана на рисунку 1.1, використовує персональний комп'ютер у промисловому виконанні класу Intel Pentium IV, а також адаптери аналого-цифрового перетворення для вимірювання швидкості обертання (тахометрії) від фірми "Динаміка" та від компанії Advantech. Крім того, комплекс включає програмне забезпечення.

Основу комплексу складає персональний комп'ютер від фірми Advantech, який включає системну плату PCA_6176F з процесором Pentium 4. Використання цієї високоінтегрованої системної плати дозволило зменшити кількість системних роз'ємів і використовувати компактний корпус IPC_6908P.

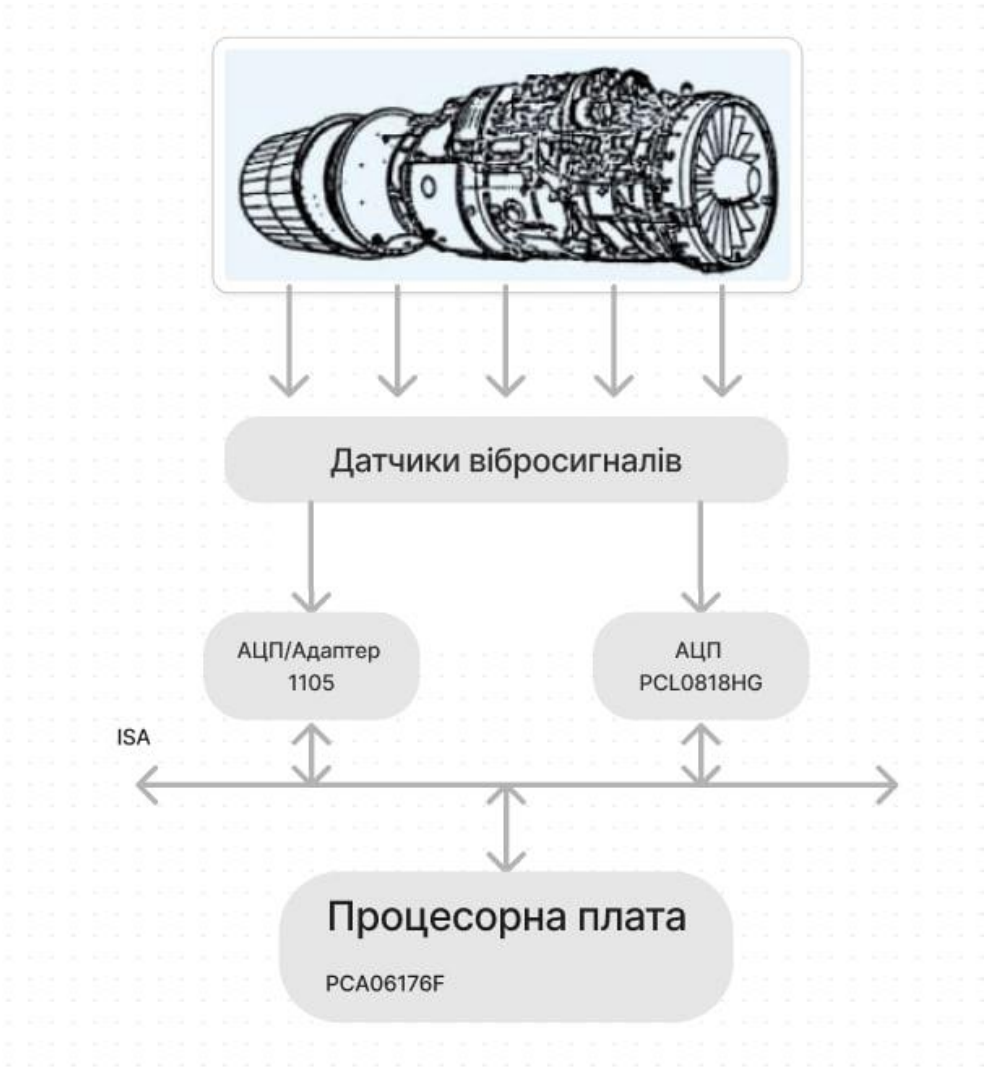


Рисунок 1.1— Структурна схема системи тестування авіадвигуна САЛЮТ

Необхідність використання комп'ютера в промисловому виконанні пояснюється рівнем зовнішніх впливів, яким він піддається, зокрема вібрацією. Корпус IPC_6908P забезпечує надійний захист встановленого обладнання від вібрації та звукового тиску, що робить необхідним використання промислового обладнання. Комп'ютер підключений до робочого місця групи обслуговування через Ethernet-мережу (10 Мбіт/с), де проводиться обробка та архівування зібраних даних.

Вимірювальне ядро комплексу складається з двох адаптерів: PCL_818HG і 1105, які встановлюються у з'єднувачі шини ISA персонального комп'ютера.

Швидкодія цієї шини достатня для передавання потоків даних від аналого-цифрових перетворювачів (АЦП) в оперативну пам'ять для подальшої обробки.

Вимірювання широкосмугової вібрації потребує встановлення точного кроку дискретизації, тобто визначення інтервалу часу, з яким проводяться вимірювання. Тому адаптери аналого-цифрових перетворювачів (АЦП), які використовуються для таких вимірювань, повинні мати апаратну функцію встановлення кроку дискретизації та достатньо великий буфер даних.

Особливістю вимірювання вібрацій газотурбінних двигунів є необхідність синхронного вимірювання частот обертання роторів разом із вібраційними сигналами. Без цього інтерпретувати спектри вібрацій було б надзвичайно складно.

Адаптери PCL_818HG і 1105, які використовуються в комплексі, забезпечують апаратне встановлення кроку дискретизації та буферизацію даних. Крім того, адаптер 1105 має вбудовані фільтри низької частоти і три канали для вимірювання частоти сигналу, що використовуються для визначення частот обертання роторів.

Адаптер PCL_818HG обробляє диференціальні вхідні сигнали і забезпечує аналого-цифрове перетворення по 7 каналах вимірювання вібрацій корпусу двигуна, а також вимірювання кута важеля керування двигуном - αРУД як супровідну інформацію. Адаптер PCL_818G містить:

- 12_розрядний АЦП, чого цілком достатньо для взаємодії його з датчиками вібрацій типу МВ_27, що використовуються для вимірювання вібрацій корпусу двигуна;

- таймер, що забезпечує апаратне встановлення частоти опитування (дискретизації);

- FIFO_буфер пам'яті 1000 слів, що дає змогу забезпечити передачу даних без втрат в оперативну пам'ять комп'ютера.

Адаптер 1105 забезпечує аналогоцифрове перетворення за 4 каналами вимірювання вібрацій виносної коробки агрегатів і турбостартера, а також вимірювання частот обертання роторів двигуна за двома каналами. Адаптер було

розроблено АТ "Динаміка" спеціально для високоточних вимірювань вібрацій авіаційних газотурбінних двигунів.

Адаптер містить:

- 16_розрядний АЦП із широким діапазоном коефіцієнтів посилення вхідного вимірюваного сигналу (до 160);

- таймер, що забезпечує апаратне встановлення частоти опитування (дискретизації);

- програмовані фільтри низької частоти;

- 2 FIFO_буфери пам'яті по 1000 слів кожен;

- три канали вимірювання частоти вхідного сигналу.

Такий функціональний набір дає змогу використовувати його під час вирішення практично будь-яких завдань вимірювання вібрацій газотурбінних двигунів.

Програмне забезпечення (ПЗ) комплексу розроблено мовою програмування С++ і працює під керуванням ОС Windows NT/2000.

Використання ОС Windows у таких системах цілком виправдане, оскільки не потребує жорстких вимог до реакції системи в реальному часі, забезпечує зручні для користувачів методи роботи, і не вимагає спеціальної підготовки персоналу, як у випадку з UNIX-системами. ПЗ реалізує всі функціональні можливості комплексу для контролю і реєстрації вібрацій газотурбінних двигунів, описані раніше.

Структура ПЗ дозволяє швидко налаштовувати його для виконання різних завдань вібраційної діагностики. Наприклад, у процесі відпрацювання в стендовий комплекс можуть бути інтегровані різні діагностичні процедури, а ПЗ спектроаналізатора можна доповнювати будь-якими алгоритмами цифрової обробки сигналів.

Як відомо, основними етапами типового підходу до розроблення регуляторів цифрових

САУ є:

- отримання вихідних даних об'єкта управління;

- ідентифікація лінійних моделей;

- аналітичний синтез або оптимізація регуляторів;

- програмна і апаратна реалізація;

- доведення під час напівнатурних і моторних випробувань.

Однак, недоліками типового підходу, на наш погляд, є:

- протиріччя між можливостями синтезу і якістю ідентифікації;

- відмінність між синтезованими регуляторами і реалізованими;

- великі витрати під час доведення на стендах.

Основні вимоги до стендів для випробувань авіаційних двигунів викладені в галузевому стандарті ОСТ 1 01021-93. Вони передбачають, що система випробувального стенда має включати спеціальну автоматизовану систему управління технологічним процесом випробувань (АСУ ТП). Ця система повинна забезпечувати керування режимом випробувань у режимі реального часу.

Основним підходом до розробки системи автоматичного управління турбореактивним двоконтурним двигуном форсажного режиму АІ-222К-25Ф, який дав змогу досягти успіху, став деякий відхід від традиційного методу розробки регуляторів двигуна. Традиційний метод зазвичай базується на попередньому спрощеному моделюванні з використанням лінійних моделей. Хоча цей метод значно полегшується завдяки використанню таких передових інтегральних пакетів аналізу та синтезу САУ, як MatLab (MathWorks Inc, США), LabView (National Instruments, США) та інших.

Використання в моторному випробувальному стенді реальних часів ППММ двигуна і ГМЧ САУ з їх відповідним налаштуванням на калібрувальні, проливні та інші характеристики стендової обв'язки двигуна і ГМЧ САУ.

Початкові дані для розробки детальних поелементних моделей двигуна включають характеристики турболопаткових машин, втрати і відбори в тракті двигуна, які надаються розробником двигуна. Для створення ППММ агрегатів ГМЧ САУ використовуються вихідні дані, надані розробником агрегатів - ХАКБ, а також геометричні та гідравлічні параметри агрегату.

Для забезпечення можливості реалізації моделі в реальному часі враховуються тільки диференціальні рівняння нерозривності, нехтуючи інерційністю рухомих золотників та ін. елементів. Під час реалізації в розрахункові схеми реального часу вводяться додаткові динамічні ємності агрегату, що забезпечують можливість переходу від ітераційних методів розв'язування алгебраїчних систем рівнянь до диференціальних рівнянь заповнення ємностей агрегату. Отриманий комплекс математичних моделей пов'язується між собою відповідно до реальних зв'язків двигуна і є, по суті, імітатором об'єкта управління.

Комплекс моделей взаємодіє з реальним регулятором через його вхідні сигнали вимірювальних каналів і вихідні керувальні сигнали, імітуючи роботу регулятора з реальним об'єктом керування. Верифікація поелементних моделей двигуна здійснюється шляхом перевірки відповідності розрахунковим параметрам на сталих режимах за допомогою статичної моделі розробника двигуна. Для цілей моделювання прийнятною вважається розбіжність до 0,2%. Верифікація динамічних моделей проводилася на основі експериментальних і розрахункових даних, наданих розробником двигуна і ГМЧ, які відображають динаміку зміни тисків у порожнинах сервопоршнів керуючих органів агрегату.

Для роботи моделі з реальним регулятором необхідно включити реальні калібрувальні характеристики вимірювальних каналів. Це забезпечує взаємодію моделей гідромеханічних агрегатів з моделлю двигуна. Наприклад, включають залежності площі критичного перерізу від положення гідроциліндрів сопла, що визначаються властивостями важільної системи приводу сопла. Остаточне налаштування контурів керування здійснюється введенням коригувальних залежностей, які дозволяють моделі відтворювати реальні проливні характеристики паливних колекторів з регуляторами положення керівних органів.

Остаточний вигляд алгоритмів інтеграції моделей двигуна та агрегатів ГМЧ САУ оформлюється у вигляді табличних залежностей, які коригують сигнали вимірювальних каналів моделі та керувальні сигнали регуляторів.

Незважаючи на те, що алгоритми реалізовані в графічному середовищі програмування LabView, більш повну інформацію оператору стенда про процес формування керувальних впливів і можливість змінювати параметри регуляторів у реальному часі під час випробувань надає інтерактивна панель оператора алгоритму. Наприклад, панель оператора ПІ-регулятора, інтегрована у випробувальний стенд. Ця технологія дозволяє представити всі алгоритми, що працюють у реальному часі.

Таким чином, наявні російські та вітчизняні системи тестування та діагностики авіаційних газотурбінних двигунів є складними і призначені для використання в стендових умовах, а не для безпосередньої підготовки перед польотом. Час тестування у цих системах перевищує півгодини, що потребує спеціальних витрат палива та моторесурсів.

Отже, існуючі системи тестування та діагностики газотурбінних двигунів не відповідають сучасним вимогам для оперативного тестування двигунів безпосередньо в аеродромних умовах під час експлуатації літальних апаратів.

1.3 Обґрунтування вибору структурної схеми

В умовах України більш ефективним є використання тестувальних комп'ютерних систем, які органічно пов'язані з безпосередньою експлуатацією авіадвигунів на літальних апаратах. Таке тестування дає змогу отримувати статистично повнішу картину роботи двигуна і не потребує додаткових ресурсів, оскільки процес тестування передбачається здійснювати безпосередньо в процесі запуску двигуна перед польотом з видачею рекомендації екіпажу або скасування польоту.

Для такого тестування корисно використовувати прості моделі авіадвигунів, які працюють на основі порівняння отриманих параметрів з еталонними. Щоб забезпечити мобільність системи тестування авіаційних двигунів на аеродромах, використання персональних комп'ютерів як базової платформи може бути неефективним. Замість цього, виглядає більш оптимальним використання сучасних однокристальних мікроконтролерів як базової системи.

Додаткових аспектів, які можуть покращити систему тестування:

Інтеграція з бортовими системами:

1. Автоматичний збір даних: Використання однокристальних мікроконтролерів дозволяє інтегрувати системи тестування безпосередньо з існуючими бортовими системами управління та моніторингу. Це забезпечує автоматичний збір і аналіз даних без участі екіпажу.

2. Реальні умови експлуатації: Тестування двигунів під час реальних умов експлуатації дає змогу отримувати більш точні та релевантні дані про їхній стан, що підвищує точність діагностики.

Простота та доступність:

1. Легкість обслуговування: Сучасні мікроконтролери є компактними та легкими у використанні. Вони не потребують складного обслуговування та можуть бути легко інтегровані у вже існуючу інфраструктуру.

2. Зниження витрат: Використання однокристальних мікроконтролерів замість персональних комп'ютерів значно знижує вартість системи тестування, що є важливим фактором в умовах обмежених ресурсів.

Порівняння з еталонними параметрами:

1. Алгоритми аналізу: Мікроконтролери можуть бути запрограмовані для порівняння отриманих параметрів з еталонними значеннями, які були встановлені під час виробництва двигуна або після останнього капітального ремонту. Це дозволяє швидко виявляти відхилення від норми.

2. Адаптивні моделі: Використання адаптивних моделей, які враховують історичні дані про роботу конкретного двигуна, дозволяє підвищити точність діагностики та зменшити ймовірність помилкових спрацьовувань.

Мобільність та гнучкість:

1. Мобільні діагностичні комплекси: Розробка мобільних діагностичних комплексів на базі мікроконтролерів дозволяє проводити тестування у будь-якому місці та у будь-який час, що є особливо важливим в польових умовах.

2. Бездротові технології: Використання бездротових технологій для передачі даних між двигуном та системою аналізу дозволяє проводити тестування навіть під час руху літального апарату по аеродрому.

Програмне забезпечення та інтерфейси:

1. Інтуїтивно зрозумілий інтерфейс: Розробка інтуїтивно зрозумілих інтерфейсів для системи діагностики забезпечує легкість використання навіть для персоналу з мінімальною технічною підготовкою.

2. Інтеграція з іншими системами: Системи тестування мають бути інтегровані з іншими інформаційними системами авіакомпанії або аеродрому, що дозволяє автоматично оновлювати дані про технічний стан двигунів та планувати їхнє обслуговування.

Впровадження таких підходів дозволить значно підвищити ефективність та точність діагностики авіаційних двигунів в умовах України, забезпечивши надійну експлуатацію літальних апаратів та підвищивши загальний рівень безпеки польотів.

Розроблювана система тестування призначена для передпольотного контролю розгінних характеристик авіаційних двигунів. Як основні параметри, що характеризують розгін двигуна в процесі його запуску використовуються:

- швидкість повітряного потоку на вході компресора;
- кількість обертів на хвилину компресора;
- тиск на виході компресора;
- температура в камері згоряння;
- температура на виході турбіни високого тиску;
- швидкість потоку на виході зовнішнього контуру двигуна.

Окрім основних параметрів, варто розглянути також наступні:

Додаткові параметри для комплексного аналізу:

1. Вібраційні характеристики: Моніторинг вібраційних сигналів на різних частотах може допомогти виявити механічні проблеми, такі як дисбаланс або зношування підшипників.

2. Паливна ефективність: Вимірювання витрати палива під час запуску може дати додаткові відомості про ефективність роботи двигуна і можливі проблеми з паливною системою.

3. Електричні параметри: Напруга та струм, споживані під час запуску, можуть вказувати на стан електричних систем двигуна, таких як стартер або системи запалювання.

Технології збору та обробки даних:

1. Сучасні датчики: Використання високоточних датчиків для вимірювання вказаних параметрів забезпечує більш надійні дані для аналізу. Наприклад, термопари для вимірювання температури або п'єзоелектричні датчики для моніторингу тиску.

2. Цифрова обробка сигналів: Використання алгоритмів цифрової обробки сигналів (DSP) дозволяє ефективно фільтрувати шум та витягувати корисну інформацію з вимірюваних сигналів.

Системи попередження та інтерфейси:

1. Автоматичне виявлення відхилень: Розробка алгоритмів для автоматичного виявлення відхилень від норми та попередження екіпажу про потенційні проблеми. Це може включати порогові значення для кожного параметра або більш складні моделі на основі машинного навчання.

2. Користувацький інтерфейс: Інтуїтивно зрозумілий користувацький інтерфейс для виводу даних про стан двигуна, що дозволяє екіпажу швидко приймати рішення про безпеку польоту. Це може бути графічний дисплей з візуалізацією параметрів у режимі реального часу.

Інтеграція з іншими системами авіаційного обслуговування:

1. База даних історичних даних: Зберігання історичних даних про запуски двигунів для проведення трендів аналізу та виявлення довгострокових змін у характеристиках двигуна.

2. Підтримка протоколів зв'язку: Забезпечення підтримки стандартних авіаційних протоколів зв'язку, таких як ARINC 429 або CAN bus, для інтеграції з іншими системами літального апарату.

Безпека та надійність системи:

1. Захист від збоїв: Реалізація механізмів захисту від збоїв, таких як резервування ключових компонентів системи та автоматичне перемикання на резервні системи у разі відмови основної.

2. Кібербезпека: Забезпечення захисту від кібератак та несанкціонованого доступу до системи. Це включає шифрування даних та автентифікацію користувачів.

Врахування цих додаткових аспектів допоможе розробити більш надійну та ефективну систему тестування передпольотного контролю розгінних характеристик авіаційних двигунів, що підвищить рівень безпеки польотів та знизить ризики несправностей під час експлуатації.

Усі ці параметри вимірюються штатними, вбудованими датчиками авіаційного двигуна, які одночасно використовуються для індикації в пілотській кабіні.

Крім зазначених параметрів, які традиційно використовуються для визначення поточного технічного стану авіаційного двигуна, використовують акустичні параметри роботи двигуна в процесі його розгону. Для цього необхідно вимірювати акустичний сигнал у ділянці компресора, камери згоряння та сопла.

Основна ідея тестування стану двигуна полягає в тому, що швидкість, з якою двигун набирає обороти до досягнення номінальних параметрів, суттєво залежить від його стану. Новий та безпроблемний двигун досягає робочих параметрів швидко, у той час як у випадку старого двигуна цей процес може бути значно сповільненим. Наприклад, за даними з дослідження, час, необхідний для набору оборотів двигуна Д-36, який має 10 тисяч годин експлуатації, в чотири рази більший, ніж у двигуна з 100 годинами робочого часу.

Багато сучасних систем діагностики авіаційних двигунів базуються на складних аналітичних моделях. Хоча побудова таких моделей може бути складною та вимагати багато часу, вони дозволяють тестувати двигун у різних режимах роботи.

У розробці системи тестування використовується підхід, за яким порівнюються розгінні характеристики справного нового двигуна з характеристиками досліджуваного. Для коректного порівняння обидві системи отримують однакові керівні впливи під час розгону та управління під час тестування.

Необхідно реалізувати два режими порівняльного аналізу розгінних характеристик - порівняння з еталонною розгінною характеристикою справного нового авіаційного двигуна, а також порівняння з розгінною характеристикою цього самого двигуна, для того щоб відстежувати на основі сучасної комп'ютерної техніки динаміку старіння кожного конкретного авіаційного двигуна.

Для реалізації викладених підходів пропонується наступна структурна схем комп'ютерної системи тестування стану авіаційного двигуна. Цю структуру показано на рис.1.2.

Розроблена система тестування авіаційного двигуна базується на спеціальному контролері, який встановлений у корпусі на технологічному візку. Цей контролер має канал введення параметрів роботи двигуна, який реалізований як послідовний канал у рамках вбудованих засобів PIC-контролера. До цього каналу під'єднано три акустичні датчики, які здатні сприймати звукові сигнали до 110 децибел. Також до цього каналу підключено датчики швидкості потоку на вході і виході компресора, а також аналоговий вихід датчика тиску на виході компресора. Вимірюється також температура на виході з камери згоряння і внутрішнього контуру двигуна. Крім того, контролер генерує сигнали керування розгінним двигуном і керування впорскуванням палива, щоб забезпечити однакові умови під час розгону двигуна, як під час еталонного тестування.

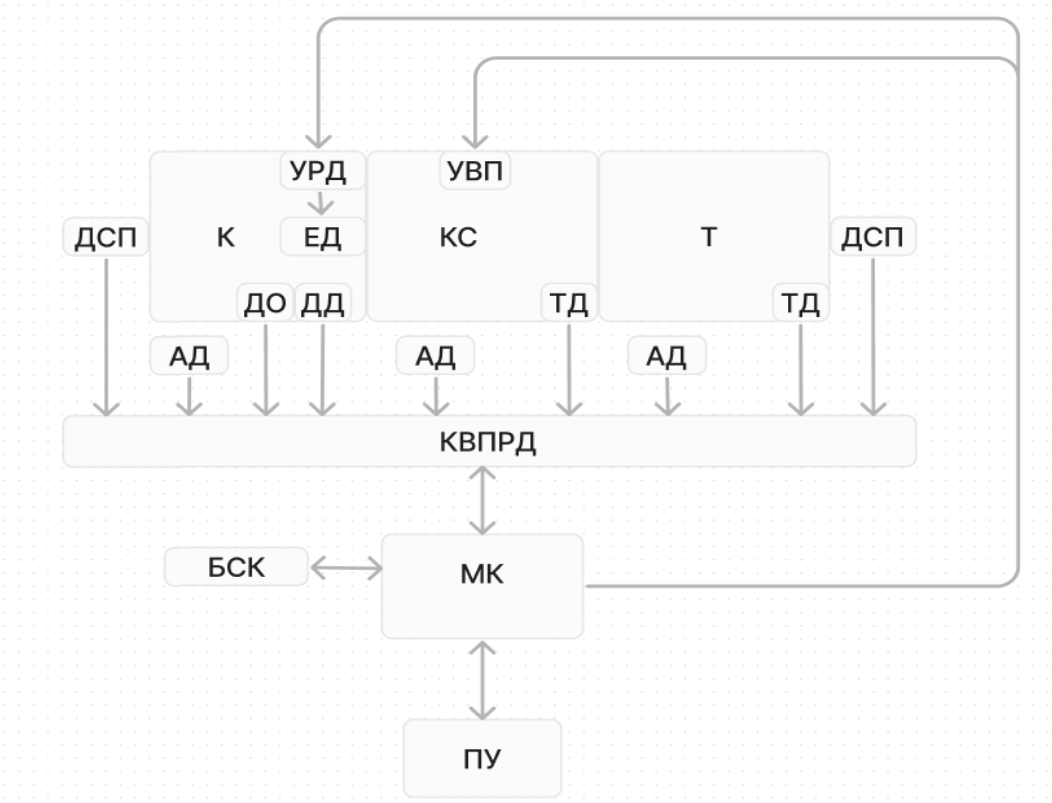


Рисунок 1.2 – Структурна схема системи тестування

Вибір підключення контролера тестування авіаційних двигунів до базового комп'ютера з великим обсягом пам'яті є стратегічно правильним рішенням з кількох причин. Додаткова інформація про переваги цього варіанту, а також можливі підходи до реалізації може бути корисною для забезпечення його ефективності:

1. Переваги використання базового комп'ютера:

- Необмежений обсяг пам'яті: Базовий комп'ютер з великим обсягом пам'яті дозволяє зберігати великі обсяги даних, включаючи численні діаграми розгону двигунів, історичні дані та результати аналізу.

- Потужні обчислювальні можливості: Використання потужного процесора та великої оперативної пам'яті базового комп'ютера забезпечує можливість проведення складного аналізу даних в режимі реального часу.

- Можливість глибокого аналізу: Застосування алгоритмів машинного навчання та інших сучасних методів обробки даних дозволяє здійснювати

глибокий аналіз динамічних характеристик двигунів, що сприяє виявленню прихованих дефектів та передвідмовних станів.

2. Підходи до реалізації:

- Інтерфейси зв'язку: Підключення контролера до базового комп'ютера можна реалізувати через різні інтерфейси зв'язку, такі як USB, Ethernet або Wi-Fi. Це забезпечить надійне та швидке передавання даних.

- Програмне забезпечення: Розробка спеціалізованого програмного забезпечення для базового комп'ютера, яке буде взаємодіяти з контролером, обробляти дані, зберігати їх та надавати користувачеві зручний інтерфейс для аналізу.

- Хмарні технології: Інтеграція з хмарними сервісами для зберігання та обробки даних може додатково підвищити гнучкість та доступність системи. Це дозволить отримувати доступ до даних та аналізувати їх з будь-якого місця.

3. Безпека та надійність:

- Резервне копіювання даних: Регулярне резервне копіювання даних на базовому комп'ютері та у хмарних сервісах забезпечить збереження інформації навіть у разі збою обладнання.

- Захист даних: Використання сучасних методів шифрування та автентифікації забезпечить захист даних від несанкціонованого доступу та кібератак.

4. Синхронізація даних:

- Автоматична синхронізація: Забезпечення автоматичної синхронізації даних між контролером та базовим комп'ютером дозволить уникнути ручного втручання та зменшить ризик втрати даних.

- Моніторинг у реальному часі: Система повинна мати можливість моніторингу параметрів двигуна в режимі реального часу з базового комп'ютера, що підвищить оперативність прийняття рішень.

5. Інтеграція з існуючими системами:

- Сумісність: Забезпечення сумісності з існуючими бортовими та наземними системами обслуговування, що дозволить використовувати вже наявні інфраструктуру та ресурси.

- Модульність: Модульний підхід до розробки системи дозволить легко оновлювати та розширювати її функціонал у майбутньому.

Ці додаткові аспекти підкреслюють переваги вибору підключення контролера до базового комп'ютера та допоможуть у розробці ефективної та надійної системи тестування авіаційних двигунів.Для забезпечення зв'язку системи з комп'ютером передбачено використання блоку сполучення з комп'ютером (БСК). Оскільки комп'ютер не бере участь у обробці даних, що відбувається на контролері, блок сполучення може бути реалізований за допомогою USB.

Висновки до розділу 1

В результаті виконання першого розділу бакалаврського проєкту, направленого на обляд вимог до систем тестування авіаційних двигунів та обґрунтування вибору апаратних та програмних засобів для її реалізації можуть бути зроблені наступні висновки:

1. Задача оперативного тестування стану авіаційних двигунів на основі вимірів динаміки розгінних характеристик при запуску та акустичних сигналів є важливою для перевірки стану двигуна безпосередньо в процесі його запуску та виходу на режим перед польотом. Ця задача може вирішуватися апаратно-програмним мобільним комплексом, підєднаним до мережі Інтернет. Основу комплексу складає термінальний мікроконтролер, який здійснює первинну обробку результатів вимірювань з використанням історії запусків кокретного двигуна, що зберігається на центральному комп'ютері.

2. З огляду на часові характеристики процедур первинної обробки результатів розгінних характеристик двигуна, а також об'єм потрібних ресурсів для виконання цих процедур обґрунтовано вибір структурної схеми комплекса, а також вибір типу термінального мікроконтролера для його практичної реалізації. Визначана номенклатура вимірювальних пристроїв, що підключаються до термінального мікроконтролера з використанням послідовних шин та USB.

РОЗДІЛ 2
РОЗРОБКА ФУНКЦІОНАЛЬНОЇ СХЕМИ

2.1 Розробка схеми підключень вимірювальних пристроїв

Контролер тестування авіаційних двигунів призначений для вимірювання з певним періодом даних датчиком стану авіаційного двигуна в процесі його розгону для виходу на польотний режим.

Контролер тестування авіаційних двигунів, що використовує вбудовані засоби контролю у вигляді датчиків, має враховувати кілька ключових аспектів для забезпечення ефективної та точної діагностики. Ось додаткова інформація, яка може бути корисною:

1. Частота вимірювання та синхронізація даних:

- Періодичність збору даних: Вибір оптимальної частоти збору даних є критичним. Занадто рідкісні вимірювання можуть пропустити короткочасні аномалії, тоді як надто часті можуть перевантажити систему обробки даних. Зазвичай частота вимірювань обирається залежно від динамічних характеристик двигуна, наприклад, від 1 до 10 Гц.

- Синхронізація: Синхронізація часу між різними датчиками забезпечує точне зіставлення даних для коректного аналізу. Це може бути реалізовано через використання загального часу або спеціальних алгоритмів синхронізації.

2. Обробка та аналіз даних:

- Попередня обробка даних: Фільтрація та нормалізація даних перед основним аналізом дозволяють видалити шум і зменшити вплив зовнішніх факторів. Використання цифрових фільтрів, таких як Кальманів фільтр, може значно покращити якість даних.

- Аналіз тенденцій: Виявлення довгострокових змін у параметрах двигуна дозволяє прогнозувати можливі відмови. Аналіз тенденцій може бути здійснений за допомогою алгоритмів машинного навчання, які виявляють відхилення від нормального режиму роботи.

3. Інтерфейс користувача та вивід даних:

- Графічний інтерфейс: Інтуїтивно зрозумілий графічний інтерфейс для відображення поточних параметрів двигуна та їх історії. Це може включати графіки, діаграми та індикатори стану.

- Система попередження: Інтеграція системи попередження, яка повідомляє екіпаж про виявлені аномалії або відхилення від норми. Це можуть бути звукові сигнали, повідомлення на дисплеї або світлові індикатори.

4. Взаємодія з іншими системами:

- Інтеграція з бортовими системами: Контролер повинен мати можливість взаємодії з іншими бортовими системами для обміну даними та інтегрованого аналізу. Це включає підтримку стандартних протоколів зв'язку, таких як ARINC 429, CAN bus, або Ethernet.

- Зв'язок із наземними системами: Передача даних на наземні системи обслуговування для подальшого аналізу та архівування. Це дозволяє наземному персоналу проводити детальніший аналіз та планувати технічне обслуговування.

5. Надійність та безпека системи:

- Резервування: Впровадження резервних систем для критичних компонентів контролера забезпечує безперебійність роботи у разі відмови основних компонентів.

- Кібербезпека: Захист системи від кіберзагроз через використання сучасних методів шифрування, автентифікації та контролю доступу.

6. Навчання персоналу:

- Навчальні програми: Розробка програм навчання для технічного персоналу та екіпажу, щоб вони могли ефективно використовувати систему тестування та розуміти отримані дані.

- Симуляції та тренування: Проведення симуляцій запуску та роботи двигуна з метою навчання та тренування персоналу у використанні системи діагностики.

Врахування цих аспектів забезпечить створення надійної та ефективної системи тестування авіаційних двигунів, яка зможе оперативно та точно діагностувати їхній стан, забезпечуючи високий рівень безпеки польотів.

Кожен датчик, який використовується в системі, має вбудований технологічний роз'єм для передачі аналогових даних про параметри роботи двигуна. Ці датчики надають аналогові дані з максимальною напругою 12 В. Для введення цих даних до схеми контролера потрібно використовувати аналого-цифрові перетворювачі і комутаційні схеми, які дозволяють підключати їх по черзі до базового мікроконтролера.

У якості мікроконтролера використовується PIC-контролер від фірми Microchip. Він має розширені інтерфейсні можливості, у тому числі вбудований USB і COM-порт для підключення до зовнішніх пристроїв, таких як модем касового апарата. Крім того, PIC-контролери дозволяють конфігурувати кілька послідовних синхронних інтерфейсів. У розробці налаштовано два послідовних синхронізованих інтерфейси: один для отримання даних від датчиків і другий для передавання даних на цифрові потенціометри, які керують роботою розгінного електродвигуна і системи впорскування пального. Однорозрядні порти мікроконтролера використовуються для керування керуючими перемикачами і індикаторами роботи світлодіодів.

При керуванні розгінним електродвигуном і системою вприскування палива необхідно забезпечити аналогові сигнали. Є два способи здійснення цього:

- Використання зовнішніх мікросхем для цифро-аналогового перетворення.
- Використання аналогових потенціометрів, які керуються цифровим кодом.

З урахуванням особливостей керування агрегатами авіаційних двигунів кращим варіантом є другий, оскільки він забезпечує більш плавне і безперервне керування.

Крім традиційних параметрів, які використовуються для оцінки технічного стану двигуна, також вимірюються акустичні параметри під час його розгону. Для цього встановлюються акустичні датчики на корпусі двигуна в певних точках. Датчики такого типу, як ДАД-26В, призначені для вимірювання звукового тиску до 120 ДБ і мають аналоговий вихід із максимальною напругою 12 В.

Для зменшення кількості мікросхем у контролері тестування авіаційних двигунів ми використовуємо мікроконтролер, який вже має вбудований 8-

розрядний аналого-цифровий перетворювач (АЦП). Єдиним мікроконтролером з такою можливістю є PIC18 від Microchip. Окрім цього, PIC18 має вбудовану підтримку USB і може побудувати послідовний інтерфейс.

Аналогові сигнали з дев'яти аналогових датчиків потребують посилення. Для посилення цих сигналів без спотворення найбільш доцільно використовувати операційні підсилювачі. Оскільки в схемі контролера використовуються сигнали з дев'яти датчиків, потрібно вирішити проблему комутації цих сигналів з акустичних датчиків. Є три можливих способи вирішення цієї задачі:

1. Комутація (мультиплексування) аналогових сигналів з датчиків до їх посилення;

2. Комутація (мультиплексування) аналогових сигналів після їх посилення до входу АЦП;

3. Комутація цифрових сигналів після аналого-цифрового перетворення.

Останній варіант пов'язаний з найбільшими апаратними витратами, оскільки необхідно мати 9 окремих АЦП. Оскільки мікросхема мікроконтролера має тільки один вбудований АЦП, цей варіант неприйнятний. Перший варіант пов'язаний з найменшими апаратними витратами, однак, його істотним недоліком є те, що слабкі сигнали з датчиків під час їхньої комутації піддаються великим спотворенням. Виходячи з наведених міркувань, найефективнішим способом розв'язання задачі комутації сигналів із датчиків слід визнати їхнє мультиплексування після їхнього посилення до входу АЦП.

2.2 Визначення часових характеристик

Головною проблемою з останнім варіантом є гарантування прийнятних часових характеристик, оскільки всі датчики повинні бути опитані послідовно, і коди даних також повинні передаватися послідовно. Більшість використовуваних датчиків видають аналоговий сигнал, який може бути представлений 8-розрядним кодом, що є достатнім для цілей нашого проєкту.

Проведемо швидкісний аналіз. Згідно з технічним описом PIC контролера, швидкість передавання даних через послідовний інтерфейс становить 256

Кбайт/с. Це означає, що можна зчитати дані з 9-ти байтових датчиків приблизно 10000 разів за секунду. З урахуванням швидкості реальної роботи вбудованих АЦП, темп опитування датчиків становить не менше 100 разів за секунду. Цей темп зчитування вважається прийнятним, враховуючи швидкість зміни стану агрегатів газотурбінного двигуна.

Однак обмеженням часу опитування датчиків є обсяг пам'яті. Якщо припустити, що об'єм вбудованої оперативної пам'яті PIC контролера моделі 18 складає 512 Кбайт, то максимальний обсяг даних для зберігання діаграми становить не більше 256 Кбайт. Таким чином, для збереження даних з 9-ти датчиків об'єм пам'яті, що призначається для кожного параметра, складає 256Кб / 9 ≈ 29000 байтів.

Враховуючи, що час розгону двигуна не перевищує 300 секунд, то темп опитування датчиків може становити:

$$t = \frac{V}{T} = \frac{29000}{300} =$$ 97 опитувань/сек.

Реально, кількість опитувань може бути значно меншою. За аналогією з наявними системами аналогічного призначення, темп опитування може становити не більше 10 опитувань за секунду.

Недоліком прийнятого рішення є малий обсяг пам'яті, необхідний для зберігання діаграм розгону двигуна. Є два можливих способи вирішення цієї проблеми:

- використання зовнішньої флеш-пам'яті для зберігання діаграм розгінних характеристик авіаційних двигунів;

- можливість підключення контролера тестування авіаційних двигунів до базового комп'ютера з достатньо великим об'ємом пам'яті.

Другий варіант виглядає більш привабливим через можливість використання потужних базових комп'ютерів для глибокого аналізу динамічних характеристик роботи двигунів. Цей варіант не має обмежень на кількість діаграм розгону двигунів, що можуть бути використані (як еталонних, так і для конкретних двигунів). Тому в розробці вирішили скористатися другим варіантом

- підключення контролера тестування авіаційних двигунів до базового комп'ютера з достатньо великим обсягом пам'яті. Для здійснення зв'язку між контролером керування авіаційним двигуном і базовим комп'ютером передбачається використання лінії USB. Це рішення мотивується двома факторами:

Основним вузлом системи для моніторингу технологічного процесу вимірювання розгінних характеристик авіаційного двигуна є однокристальний мікроконтролер. Він має вбудовану пам'ять для зберігання даних та програм, яку можна адресувати довільно, і можливість змінювати програми та параметри технологічного процесу. Також він має розвинені інтерфейсні можливості, зокрема, може налаштовувати кілька послідовних синхронних портів для передачі даних.

У складі контролера авіаційного двигуна передбачено використання термометрів для вимірювання температури. Ці термометри повинні мати вбудовані аналого-цифрові перетворювачі і можливість обміну даними через послідовний порт у цифровій формі. Однією з можливих мікросхем для цього застосування є TC665C від фірми Microchip, які призначені спеціально для використання з контролерами PIC. Ці мікросхеми дозволяють вимірювати температури до 1200°C та мають вбудований аналого-цифровий перетворювач. Вони також забезпечують можливість підключення через послідовний порт для обміну даними з мікроконтролером. Схема вимірювання температури включає в себе терморезистор, під'єднаний до цифрового термометра через опорну напругу. Зміна опору терморезистора відбувається в залежності від температури навколишнього середовища. Потік струму через терморезистор змінюється, і ця зміна вимірюється цифровим термометром через його вхід для отримання температурних значень. Технічні можливості передбачають відстань до 16 метрів для видаленого розташування терморезистора, що дозволяє ефективно використовувати такі датчики для тестування авіаційних двигунів. Контроль за цими датчиками передбачає проведення операцій, пов'язаних з опитуванням цифрових термометрів і датчиків стану авіаційного двигуна.

Для вимірювання температури в камері згоряння та турбінному відсіку авіаційного двигуна використовуються терморезистори, які розміщені прямо на їхніх стінах. Ці терморезистори підключені до входів цифрових термометрів. Кожен цифровий термометр містить в собі мікросхему, яка здійснює нелінійне перетворення вхідної напруги на 8-розрядний цифровий код. З практичної точки зору, кожен цифровий термометр є своєрідним аналого-цифровим перетворювачем з цифровим послідовним виходом.

До компонентів проектованого контролера тестування авіаційних двигунів входять три кнопкових перемикачі, які підключені до однобітових портів базового мікроконтролера.

Стан цих перемикачів програмно опитується для встановлення режиму роботи контролера. Два світлодіодні індикатори використовуються для відображення стану контролера і керуються відповідними сигналами з портів мікроконтролера.

Наявний USB-вихід на платі контролера дозволяє підключати його до зовнішнього персонального комп'ютера для завантаження еталонних діаграм та зчитування отриманих даних. Крім того, цей же вихід може використовуватися для розширення пам'яті за допомогою зовнішньої флеш-пам'яті.

Описані режими роботи контролера тестування авіаційних двигунів є основними для забезпечення надійної діагностики та прийняття рішень щодо їх експлуатації. Нижче наведено додаткову інформацію, яка може допомогти у вдосконаленні системи:

1. Завантаження еталонних діаграм:

- Формати даних: Забезпечення підтримки різних форматів даних для еталонних діаграм, що дозволяє використовувати дані з різних джерел та систем.

- Автоматичне оновлення: Можливість автоматичного оновлення еталонних діаграм через мережу або з зовнішніх носіїв для забезпечення актуальності даних.

2. Побудова та збереження діаграм:

- Реальний час: Відображення параметрів у режимі реального часу під час розгону двигуна, що дозволяє оперативно реагувати на будь-які відхилення.

- Збереження даних: Використання надійної пам'яті для збереження даних, зокрема, флеш-пам'яті або SSD-накопичувачів, що забезпечує тривале зберігання без втрати інформації.

3. Первинне порівняння та індикація:

- Алгоритми порівняння: Використання алгоритмів порівняння, які можуть враховувати різні типи відхилень, такі як постійні зміщення, коливання або пікові відхилення. Це може включати використання статистичних методів або методів машинного навчання.

- Індикація придатності: Чітка та інтуїтивно зрозуміла індикація придатності двигуна до виконання польоту, що може включати кольорові індикатори (зелене, жовте, червоне світло) або текстові повідомлення на дисплеї.

4. Зчитування діаграм:

- Зв'язок з базовим комп'ютером: Використання сучасних інтерфейсів зв'язку, таких як USB, Wi-Fi, або Ethernet, для зчитування даних з контролера на базовий комп'ютер. Це забезпечує швидкий та зручний доступ до даних для подальшого аналізу.

- Формати експорту: Підтримка різних форматів експорту даних, таких як CSV, XML або спеціалізовані формати для подальшої обробки в програмному забезпеченні для аналізу.

5. Додаткові функції:

- Журнал подій: Ведення журналу подій, який фіксує всі запуски двигуна, їхні параметри та результати порівняння. Це допоможе в аналізі історії роботи двигуна та виявленні довгострокових тенденцій.

- Самодіагностика: Вбудована функція самодіагностики контролера, яка перевіряє справність всіх компонентів системи та повідомляє про можливі несправності.

6. Безпека та захист даних:

- Резервне копіювання: Регулярне резервне копіювання даних на зовнішні носії або в хмарне сховище для захисту від втрати даних.

- Шифрування: Використання шифрування для захисту даних під час зберігання та передачі, що запобігає несанкціонованому доступу до конфіденційної інформації.

Розробка такої системи з урахуванням наведених аспектів дозволить забезпечити високу надійність та ефективність контролю авіаційних двигунів, підвищуючи загальний рівень безпеки польотів.

Висновки до розділу 2

В результаті виконання другого розділу бакалаврської роботи, ціллю якого визначена розробка функціональної схеми програмно-апаратного мобільного комплексу отримані наступні результати:

1. Виходячи із номенклатури вбудованих в авіаційний двигун засобів контролю параметрів його роботи визначані вимоги для портів підключення відповідних сигналів до термінального мікроконтролера, визначені потрібні для цього інтерфейсні засоби, такі як аналого-цифрові перетворювачі. Визначена номенклатура мікросхем, на яких реалізується інтерфейс між вимірювальними приладами та термінальнім мікроконтролером типу PIC.

2. Проведено розрахунок часових характеристик роботи термінального мікроконтролера для визначення можливості реалізації прийому сигналів від виміріювалних пристроїв і їх первинної обробки в режимі реального часу або зі збереженням частини результатів вимірювань в оперативній пам’яті термінального мікроконтролера. На основі розрахунків часових характеристик, тактової частоти і потрібного об’єму внутрішньої памяті, а також виходячи з інтерфейс них можливостей здійснено вибір типу мікроконтролера.

РОЗДІЛ 3
РОЗРОБКА АЛГОРИТМІВ І ПРОГРАМ РОБОТИ

3.1 Розробка алгоритму формування сигналів управління

Проектований контролер тестування призначений для перевірки характеристик роботи авіаційного двигуна під час його розгону шляхом їх вимірювання та порівняння з еталонними діаграмами розгону. Для того, щоб порівняння розгінних характеристик еталонних і досліджуваних двигунів було коректним, при розгоні двигуна необхідно надати такі ж самі керуючі впливи на електродвигун розгону двигуна та систему впорскування палива, як при еталонному тестуванні. Тому контролером видаються сигнали керування розгінним двигуном (УРД) та впорскуванням палива (УВП).

Процедура видачі керуючих сигналів починається з сигналу готовності, який генерується першим блоком алгоритму. Наступним етапом є опитування стану кнопкових перемикачів (блок 2 алгоритму), щоб визначити режим роботи контролера тестування авіаційних двигунів. Якщо встановлений режим тестування, третій блок алгоритму активує розгінний двигун і одночасно обнуляє таймер мікроконтролера. Четвертий блок алгоритму збільшує час регулювання на певний інтервал Δt. П'ятий блок опитує перший вбудований датчик, що вимірює швидкість повітряного потоку на вході компресора. Аналоговий сигнал підсилюється, перетворюється в цифрову форму та зберігається в оперативній пам'яті мікроконтролера. Шостий блок вводить дані про кількість обертів компресора, опитуючи другий вбудований датчик, який вимірює оберти нагнітаючого компресора. Аналоговий сигнал підсилюється, перетворюється в цифрову форму та зберігається в оперативній пам'яті. Так само, тринадцятий блок алгоритму вводить дані про тиск на виході компресора, опитуючи третій вбудований датчик, який вимірює цей параметр. Аналоговий сигнал підсилюється, перетворюється в цифрову форму та зберігається в оперативній пам'яті мікроконтролера.

Після цього блоком 14 алгоритма ініціюється ввод результатів вимірювання трьох акустичних датчиків, що встановлені, відповідно, на компресорі, камері

згорання та турбіні високого тиску. Аналогові сигнали акустического давления усиливаются, коммутируются и преобразуются в цифровую форму. В таком виде показания трех акустических датчиков фиксируются в специально выделенной области оперативной памяти микроконтроллера.

Блок 15 алгоритму порівнює виміряний тиск на виході компресора з допустимим рівнем. Якщо тиск перевищує встановлений для поточної стадії розгону двигуна, то блок 16 алгоритму встановлює необхідний рівень впорскування палива в двигун. Це робиться для того, щоб уникнути помпажу двигуна під час запуску.

Блок 19 алгоритму здійснює введення і фіксацію температури в камері згоряння. Дані зчитуються з цифрового термометра і зберігаються в пам'яті. Блок 20 алгоритму вводить дані про акустичний тиск у камері згоряння. Аналоговий сигнал акустичного тиску підсилюється, комутується і перетворюється в цифрову форму, після чого зберігається в спеціально виділеній області оперативної пам'яті мікроконтролера. Наступний, блок 21 алгоритму, здійснює введення швидкості потоку газів на виході внутрішнього контуру двигуна. Для цього опитується відповідний вбудований датчик двигуна, який вимірює швидкість витікання продуктів згоряння на виході турбіни високого тиску. Аналоговий сигнал підсилюється, комутується і перетворюється в цифрову форму. У такому вигляді показання цього датчика зберігаються в спеціально виділеній області оперативної пам'яті мікроконтролера.

Блок 22 алгоритму порівнює швидкість потоку на вході і виході двигуна. Якщо швидкість потоку на вході перевищує швидкість на вході в компресор на задану величину, блок 23 алгоритму вимикає розгінний двигун. Потім блок 24 алгоритму опитує датчик температури на виході турбінного відсіку двигуна. Аналоговий сигнал підсилюється, комутується і перетворюється в цифрову форму. У такому вигляді показання цього датчика зберігаються в спеціально виділеній області оперативної пам'яті мікроконтролера. Блок 25 алгоритму зчитує дані про акустичний тиск у турбінному відсіку. Аналоговий сигнал підсилюється,

комутується і перетворюється в цифрову форму, після чого зберігається в спеціально виділеній області оперативної пам'яті мікроконтролера.

Блок 27 алгоритму аналізує отримані показники параметрів, щоб визначити, чи був досягнутий нормальний режим зльоту двигуна. Блок 28 алгоритму контролює і аналізує час, який був потрібний для досягнення нормального режиму зльоту. Якщо цей час перевищує допустимий критичний рівень, то блок 29 алгоритму відображає несправність двигуна, який не може бути використаний для польоту. У протилежному випадку, блоком 30 алгоритму видається світловий сигнал, що двигун з точки зору його розгінних характеристик може вважатися справним і готовим для польоту.

3.2 Розробка алгоритму процедури аналізу розгінних характеристик двигуна.

Робота алгоритму розпочинається з прийому сигналу готовності контролером, що надходить через блок 1. Якщо такий сигнал отримано (перевіряється блоком 2), то блок 3 вмикає розгінний двигун. Наступним кроком, блок 4 алгоритму встановлює таймер мікроконтролера на нуль. Блок 5 відповідає за очікування сигналу від внутрішнього таймера після встановленої затримки. Після завершення затримки другий біт регістру прапорців встановлюється в одиницю. Це встановлення контролюється блоком 6 алгоритму.

Якщо видається сигнал таймера, то блок 7 проводить зчитування датчика обертів компресора двигуна. Передача даних з датчика відбувається через комутатор у послідовному коді. Блок 8 контролює момент завершення передачі. Коли передача завершена, блок 9 алгоритму ініціює зчитування еталонного значення обертів на поточний момент часу з таблиці, завантаженої з основного комп'ютера. Блок 10 обчислює різницю між фактичним і еталонним значеннями обертів компресора для заданого моменту часу розгону двигуна. Блок 11 алгоритму контролює завершення часу розгону двигуна. Якщо двигун не розігнався, то відбувається повернення до повторного виконання блоку 5 алгоритму.

Блок 22 алгоритму здійснює зчитування значення положення заслонки вприскування палива на поточний момент часу запуску двигуна з таблиці еталонної діаграми розгону двигуна. На основі отриманих даних з таблиці, блок 16 алгоритму видає управляючий сигнал на потенціометр, напруга з якого керує положенням заслонки. Далі, блок 17 здійснює зчитування датчика обертів компресора двигуна. Передача даних з датчика виконується через комутатор у послідовному коді. Блок 18 контролює момент завершення передачі. Після завершення передачі, блок 19 алгоритму ініціює зчитування еталонного значення обертів на поточний момент часу з таблиці, завантаженої з основного комп'ютера. Блок 20 алгоритму обчислює різницю між фактичним і еталонним значеннями обертів компресора для заданого моменту часу розгону двигуна. Блок 21 алгоритму контролює завершення часу розгону двигуна.

Якщо двигун не розігнався, то виконується перехід до виконання блоку 27 алгоритму, який ініціює індикацію недопустимої затримки виходу двигуна на номінальний режим зльоту. У протилежному випадку, блоком 23 алгоритму виконується індикація виходу двигуна на номінальний режим зльоту в межах встановленого часового допуску.

3.3 Розробка алгоритму видачі даних на комп'ютер.

Процедура видачі даних на комп'ютер починається з перевірки, чи встановлений режим читання даних з контролера за допомогою клавіатури. Цей контроль здійснюється блоком 1 алгоритму. Якщо через клавіатуру встановлено режим читання, то блок 2 алгоритму індикаторами світлодіодів вказує на цей режим. Далі, блоком 3 алгоритму розпочинається цикл ініціалізації читання даних з вбудованої флеш-пам'яті мікроконтролера. Цей процес включає встановлення вказівника адреси в нуль за допомогою блоку 4 алгоритму та ініціалізацію читання через відповідну команду до вбудованої флеш-пам'яті.

Після цього, блоком 6 алгоритму очікується сигнал готовності. Якщо він отриманий, то блоком 7 алгоритму дані зберігаються в буфері з одночасним збільшенням адреси пам'яті. Блок 8 алгоритму перевіряє, чи досягнута максимальна адреса блоку даних. Якщо максимальна адреса блоку в пам'яті не

досягнута, то блоком 9 алгоритму починається очікування опитування мікроконтролера через шину USB. При цьому розроблений контролер для тестування авіаційних двигунів виступає у ролі пристрою-веденого в ієрархії пристроїв, підключених до USB-хабу персонального комп'ютера. При отриманні запиту від персонального комп'ютера через шину USB, виконується контроль відповідної адреси через шину (блок 12 алгоритму). Якщо адреса контролера для тестування авіаційних двигунів вказана в пакеті через USB, то блоком 14 алгоритму починається передача блоку даних із буфера. При цьому довжина блоку визначається відповідно до формату кадру передачі даних. Після передачі блоку очікується прийом підтвердження прийому блоку даних хабом персонального комп'ютера.

Якщо підтвердження отримано, то адреса збільшується на 128 блоком 17 алгоритму.

Процес введення характеристик роботи двигуна розпочинається з видачі сигналу готовності. Потім відбувається перевірка надходження сигналу початку циклу вимірювання. Після отримання такого сигналу починається видача сигналу фіксації на цифрові термометри. Після отримання цих сигналів цифрові термометри фіксують поточний рівень вхідного аналогового сигналу і розпочинають процес його перетворення в цифрову форму. Потім виконується установка режиму послідовного введення.

Для непосереднього введення температурних даних спочатку видається адреса термометра. Після отримання сигналу готовності від цифрового термометра відбувається послідовне читання бітів байта температури. При цьому здійснюється контроль правильності передачі шляхом обчислення біту парності та порівняння його з переданим від цифрового термометра. Якщо помилок немає, то відбувається видача адреси датчика температури камери згоряння. Також для непосереднього введення температурних даних в камері згоряння авіаційного двигуна видається адреса термометра камери згоряння. Після отримання сигналу готовності від цифрового термометра знову відбувається послідовне читання бітів байта температури. Тут також проводиться контроль правильності передачі

шляхом обчислення біту парності та порівняння його з переданим від цифрового термометра. Якщо помилок немає, то ініціюється видача сигналу фіксації вимірів акустичних датчиків. Після отримання цих сигналів акустичні датчики фіксують поточний рівень вхідного аналогового сигналу звукового тиску та починають процес його перетворення в цифрову форму. Після цього встановлюється режим послідовного введення. При цьому лічильник акустичного датчика встановлюється в нуль.

Мікроконтролер відправляє адресу поточного акустичного датчика. Після отримання від акустичного датчика сигналу готовності проводиться послідовне читання бітів байта акустичного тиску.

Якщо блоком 19 алгоритму виявлено завершення блоку даних, то мікроконтролер формує та видає байт закінчення пакету даних для хаба персонального комп'ютера (блок 20 алгоритму). Передача байту закінчення пакету повторюється мікроконтролером до отримання від персонального контролера пакета підтвердження всього блоку даних. Після цього, блоком 22 алгоритму виконується відключення режиму читання. Блок 23 алгоритму ініціалізує відключення відповідної індикації. Наступний блок 24 алгоритму ініціалізує включення опитування дозволу на очищення зчитаних даних з флеш-пам'яті. Якщо дозвіл на очищення отримано (перевіряється блоком 25 алгоритму), то наступним блоком 26 алгоритму виконується очищення переданих даних з вбудованої флеш-пам'яті мікроконтролера.

3.4 Розробка програми взаємодії контролера з базовим комп'ютером через шину USB.

Інтерфейс USB у зовнішньому пристрої можна реалізувати двома способами:

1. Існують два способи реалізації інтерфейсу USB у зовнішньому пристрої. Перший - це використання мікроконтролера, у якого вбудований апаратний USB-інтерфейс. В цьому випадку потрібно розуміти, як працює USB, щоб написати програмне забезпечення для мікроконтролера. Також, якщо операційна система не підтримує стандартні USB-класи, то потрібно написати драйвер для комп'ютера. Основний недолік цього методу - обмежена доступність

таких мікроконтролерів і їх вища вартість порівняно зі звичайними мікроконтролерами, які використовуються для зв'язку через "RS232".

2. Цей підхід полягає у використанні універсального перетворювача інтерфейсів, який може працювати з USB та іншими інтерфейсами, такими як RS232, 8-розрядна шина даних або шина TWI. У цьому випадку не потрібно розробляти спеціальне програмне забезпечення або драйвер, оскільки виробник перетворювача надає свій власний драйвер. Однак це може призвести до збільшення вартості системи і розмірів готового виробу. Замість цього, розглядається використання недорогого мікроконтролера і програмного емуляції USB-протоколу в ньому. Основним недоліком цього підходу є складність досягнення високої швидкості передачі даних через USB. USB має різні режими швидкості: низька (1.5 Мбіт/с), повна (12 Мбіт/с) та висока (480 Мбіт/с). Мікроконтролери можуть використовуватися для низькоскоростних режимів USB, але не рекомендуються для використання на вищих швидкостях USB.

Обмін даними з мікроконтролером відбувається через інтерфейс SPI, що використовує три лінії зв'язку. Лінія MISO (Master In Slave Out) використовується для передачі інформації від пристрою-раба до мікроконтролера. Лінія MOSI (Master Out Slave In) відповідно передає дані від мікроконтролера до пристрою-раба. Лінія SCLK (Serial Clock) використовується для синхронізації передачі даних, надсилаючи тактовий сигнал для кожного біту, який передається.

Годинник реального часу живиться від напруги +5 В, а при вимкненому живленні пристрою використовується живлення від батареї. Обмін даними з мікроконтролером відбувається через інтерфейс SPI.

Робота з мікроконтролером через інтерфейс SPI використовує три лінії зв'язку. Необхідне живлення складає +3,3 В. Вибір мікросхеми для роботи через інтерфейс здійснюється за допомогою сигналів CS0, CS1 і CS2.

Блок розширення пам'яті № 1 працює з мікроконтролером через паралельну шину.

USB-шина підтримує чотири типи передачі даних:

- Control (керуюча передача) — для керування передачею даних;

- Bulk (масова передача) — для передачі великих обсягів даних;
- Interrupt (передача по перериванню) — для передачі даних по запитах переривань;
- Isochronous (ізохронна передача) — для передачі даних у режимі реального часу.

Тип керуючої передачі (Control) для кінцевої точки існує у всіх USB-пристроях і використовується для визначення дескрипторів пристрою та інших налаштувань. Цей тип передачі не слід використовувати для завдань, які не описані у стандарті USB.

Тип масової передачі (Bulk) для кінцевої точки забезпечує надійну доставку великих обсягів даних (до 1024 байт). Цей режим ідеально підходить для передачі команд та отримання відповідей на команди від пристрою.

Список основних функцій драйвера режиму ядра включає:

- «Відправлення URB» — функція, необхідна для створення пакета запиту USB відповідно до операції, який потім передається наступному драйверу в стеку.
- «Конфігурація» — функція, яка створює розширення пристрою, де зберігається контекст даних.
- - «Читання дескриптора конфігурації» — функція, яка зчитує дескриптор пристрою, що містить опис усіх кінцевих точок пристрою.
- - «Вибір конфігурації» — функція, яка вибирає конфігурацію з переліку, наданого пристроєм.
- - «Відключення пристрою» — функція, яка очищає розширення пристрою та звільняє пам'ять.
- - «Функція запиту читання/запису» — функція, яка обробляє запити введення/виведення, ініційовані функціями Winapi32 - Readfile і Writefile.

Алгоритм роботи функції буде розглянутий далі.

- «Функція завершення читання/запису» — функція, яка обробляє подію завершення операції введення/виведення. Вона необхідна для обробки функції Winapi32-Cancelio.

Алгоритм драйвера режиму ядра виглядає наступним чином. Спочатку виконується початкова ініціалізація змінних: файлового об’єкта, розширення пристрою та визначення типу операції — читання або запис. Виконується перевірка стану пристрою, і якщо пристрій у робочому стані, то виконується вихід з функції. Далі перевіряється активований режим введення/виведення та очікується його активація. Перевіряється дескриптор файлового об’єкта та отримується опис каналу введення/виведення. Перевіряється підтримка режиму введення/виведення кінцевими точками. Виділяється пам’ять для контексту введення/виведення з перевіркою. Отримується розмір даних для обміну, якщо довжина запиту не перевищує максимальне задане значення. Встановлюється дозвіл на передачу неповного пакета даних.

Отримується адреса буфера даних і встановлюється прапор читання/запису. Перевіряється розмір переданих даних і встановлюється початкове значення розміру. Розміщується блок MDL з перевіркою. Виділяється пам’ять та формується запит URB. Буфери пам’яті зберігаються в контексті введення/виведення та встановлюється функція зворотного виклику.

Передача URB драйвера виконується через стек з контролем статусу передачі. Після успішного завершення передачі встановлюється статус успішного виконання.

У функції зворотного виклику розраховується розмір наступного пакета даних і формується наступний URB до повної передачі всього масиву даних.

Для забезпечення обміну між пристроєм і ПК необхідно розробити протокол інформаційного обміну.

Розроблений протокол складається з чотирьох основних груп:

- Службові - виконують основні сервісні функції для налаштування роботи з пам'яттю.
- Робота з пам'яттю** - містить команди для зчитування та запису даних у пам'ять.

- Робота з числовими коефіцієнтами та лічильниками подій - забезпечує команди для роботи з коефіцієнтами.
- Годинник реального часу - включає команди для встановлення та зчитування значень годин реального часу.

Робота з пам'яттю охоплює групу команд, які забезпечують можливість зчитування або запису даних у пам'ять. Одна з таких команд - це "отримання таблиці розміщення файлів".

Ця команда дозволяє отримати інформацію про те, які файли знаходяться у пам'яті і де саме вони розташовані.

Таблиця розміщення файлів - це інформаційна структура, яка містить описи файлів та їх розташування у пам'яті пристрою. Вона включає такі поля, як номер файлу, початкова адреса і кінцева адреса.

Очищення основної пам'яті - це процес, за допомогою якого вся інформація, збережена в основній пам'яті, видаляється, щоб вільною могла бути використана для нових даних.

Операція "Повне очищення пам'яті пристрою" видаляє всі дані, що зберігаються у пам'яті пристрою, забезпечуючи її повне очищення.

Операція "Очищення пам'яті з картою стану основної пам'яті" полягає у повному очищенні пам'яті, в якій зберігається карта стану основної пам'яті.

Операція "Зчитування інформаційних даних" виконує зчитування інформаційних даних з основної пам'яті пристрою.

Операція "Зчитування службових даних" виконує зчитування службових даних з основної пам'яті пристрою.

Сервісні функції виконують основні сервісні операції для налаштування роботи з пам'яттю.

"Визначення ідентифікатора пристрою" - це операція, що виконується для визначення ідентифікатора конкретного пристрою.

Ідентифікатор пристрою - це унікальний номер, який визначається для кожного типу пристрою, що використовує даний протокол обміну даними.

Операція "Передача таблиці настроювання" передбачає передачу пристроєм таблиці, яка містить основні параметри налаштування.

Операція "Зчитування таблиці настроювання" дозволяє пристрою отримати таблицю налаштувань.

Таблиця налаштувань містить основні характеристики пам'яті, що використовується для зберігання інформаційного потоку ARINC-429.

Операція "Зчитування інформації про мікросхеми пам'яті" дозволяє пристрою передати персональному комп'ютеру інформацію про кількість мікросхем пам'яті та їх ідентифікатори.

Операція "Одержання карти стану основної пам'яті" передбачає передачу пристроєм персональному комп'ютеру карти стану основної пам'яті. Карта стану представляє собою послідовність бітів, кожен з яких відображає стан певного блоку пам'яті.

Операція "Запис таблиці розподілу пам'яті експлуатаційних характеристик" передбачає запис у пристрій таблиці, що містить розподіл пам'яті для експлуатаційних характеристик.

Операція "Зчитування таблиці розподілу пам'яті експлуатаційних характеристик" дозволяє пристрою отримати таблицю розподілу пам'яті для експлуатаційних характеристик.

Таблиця розподілу пам'яті експлуатаційних характеристик містить інформацію про кількість пам'яті, необхідну для зберігання різних типів параметрів. Вона включає опис чотирьох типів параметрів:

• Лічильник, що фіксує тривалість події. Цей лічильник зберігає значення, що вказує на тривалість певної події. Це значення не втрачається після вимкнення живлення пристрою.

• Лічильник, що фіксує тривалість події під час роботи пристрою. Він записує задану кількість останніх значень лічильника, що вказують на тривалість події.

• Лічильник, що збільшується на одиницю при наявності певної події. Цей лічильник також зберігає своє значення навіть після вимкнення живлення.

• Збереження числових коефіцієнтів. В таблиці також вказується, скільки пам'яті потрібно для зберігання числових коефіцієнтів.

Функції, пов'язані з годинником реального часу, дозволяють зчитувати поточне значення годин і встановлювати нові значення. Ось дві основні операції:

• Зчитування поточного значення годин реального часу. Ця операція дозволяє отримати інформацію про поточний час з годинника реального часу.

• Установка годин реального часу. Ця команда дає можливість встановити нове значення годин у годиннику реального часу, щоб синхронізувати його з поточним часом.

Схема взаємодії програми від користувача до рівня драйверів така: спочатку користувач за допомогою миші вибирає пристрій USB зі списку, який був сформований під час підключення до комп'ютера. Потім він обирає операцію, яку потрібно виконати. Після цього формується команда відповідно до протоколу обміну, і цей запит передається драйверу у користувацькому режимі.

Потім створюється запит для драйвера режиму ядра, який передає дані до пристрою через драйвер шини USB. Інформаційний потік ARINC-429 зберігається у файлі, а дані про стан лічильників записуються в інший файл. Якщо виникає помилка на будь-якому рівні, генерується відповідне повідомлення з кодом помилки.

Ця програма обміну розділена на кілька блоків для зручності.

Перший блок - процедури переривань - відповідає за обробку подій, які виникають в програмі. Другий блок - процедури дешифрації - відповідає за розшифрування отриманих даних. USB-прийом та USB-передача - це блоки, які забезпечують прийом і передачу даних через USB-порт. Дешифрація запитаної дії - це блок, який розшифровує запитані дії або команди. І, нарешті, виконання треба дії відповідає за виконання відповідних дій або команд.

Ці блоки разом утворюють комплексну програму обміну, яка дозволяє взаємодіяти з USB-пристроєм, обробляти дані, передавати їх і виконувати відповідні дії.

Процедура обробки переривань "EXT_INT0" викликається при активації зовнішнього переривання 0, яке залишається активним протягом усього часу виконання програми. Ця процедура починає прийом даних, які послідовно передаються через USB-шину. Зовнішнє переривання виникає при появі високої напруги на виводі INT0. Цей високий рівень сигналізує про початок синхронізації інтервалів і викликає процедуру прийому даних через USB.

Спочатку процес оцифрування даних повинен бути синхронізований з серединою біта. Це досягається за допомогою образцової синхронізації. Оскільки тривалість передачі одного біта становить лише 8 періодів синхронізації XTAL, а виникнення переривання може бути затримано (+/- 4 періоди), то синхронізацію з фронтами образцової синхронізації потрібно виконати дуже точно. Завершення передачі образцової синхронізації і початок передачі бітів даних визначається двома бітами з низьким рівнем, що наступають за образцовою синхронізацією.

Після синхронізації починається фактичне цифрування даних. Це проводиться в середині кожного біта. Зважаючи на швидкість передачі даних 1,5 Мбіт/с (1.5 МГц) та частоту синхронізації мікроконтролера 12 МГц, ми маємо лише 8 тактів для цифрування даних, їх запису в однобайтовий буфер, зсуву даних у цьому буфері, перевірки на отримання повного байта, запису байта в статичну ОЗП і визначення закінчення пакета (EOP). Це найкритичніша частина програми; все має відбуватися синхронно і в межах встановлених часових рамок. Після отримання всього USB-пакета необхідно виконати дешифрацію пакета. Спочатку ми швидко визначаємо тип пакета (SETUP, IN, OUT, DATA) та отриману USB-адресу. Швидка дешифрація повинна відбуватися всередині процедури обробки переривань, оскільки потрібно дуже швидко відповісти після отримання USB-пакета (пристрій має відповісти пакетом ACK, якщо пакет був адресований пристрою, та NAK, якщо пакет адресований пристрою, але відповідь ще не готова). По закінченні процедури прийому (після відправки пакета ACK/NAK) дані, записані у буфер, повинні бути скопійовані в інший буфер, де відбувається дешифрація даних. Це потрібно для звільнення приймального буфера для отримання нового пакета.

Під час прийому пакета дешифрується його тип, і відповідне значення встановлюється для прапорця. Цей прапорець перевіряється в головному циклі програми, і залежно від його значення виконується відповідна дія, підготовлюється відповідь, не накладаючи жодних вимог щодо швидкодії мікроконтролера.

Для забезпечення високої швидкості обробки зовнішнього переривання INT0 воно має залишатися активним постійно, навіть при обробці інших переривань (наприклад, переривання від послідовного прийому). Швидкість прийому в процедурі обробки переривання INT0 має велике значення, тому програму необхідно оптимізувати з точки зору швидкодії та часу виконання. Важливою проблемою є оптимізація резервування регістрів у процедурах обробки переривань.

Основний програмний цикл є досить простим. У ньому лише потрібно перевіряти стан прапорця дії і визначати, що робити після отримання даних. Крім того, в основному циклі перевіряється, чи скинута USB-інтерфейс (обидва лінії даних знаходяться на низькому рівні протягом тривалого часу), і в разі визначення такого стану виконується переініціалізація пристрою. Якщо прапорець дії активний, то викликається відповідна дія: дешифрація коду NRZI у пакеті, видалення поразрядного заповнення та підготовка запитаної відповіді у буфері передавача (з поразрядним заповненням та кодуванням NRZI). Після цього прапорець активується для сигналізації про готовність відповіді до відправлення.

Фізична передача вихідного буфера через лінії USB виконується у процедурі прийому як відповідь на пакет IN. Нижче наведено короткий опис підпрограм.

Скидання: Здійснює ініціалізацію ресурсів мікроконтролера: стеку, вводу-виводу, буферів USB, переривань.

Основний: Основний цикл програми. Перевіряє стан прапорця дії; якщо прапорець встановлений, то виконує відповідну дію. Крім того, дана процедура перевіряє скидання ліній даних USB-каналу і в разі виявлення скидання повторно ініціалізує USB-інтерфейс.

Int0Handler: Процедура обробки зовнішнього переривання INT0, або Int0Handler, є основною частиною програми обміну даними. Вона відповідає за прийом та передачу даних через USB. Під час роботи ця процедура здійснює запис

даних у буфер, визначення отримувача USB-пакета (адреси USB), розпізнавання типу пакета та відправлення відповіді на комп'ютер. Оскільки Int0Handler є основою USB-пристрою, вона відіграє ключову роль у забезпеченні правильної роботи пристрою та обміну даними між пристроєм та комп'ютером.

SetMyNewUSBAddresses: Процедура SetMyNewUSBAddresses призначена для зміни USB-адреси пристрою. Під час виконання цієї процедури адрес змінюється і кодується у вигляді NRZI (Non-Return-to-Zero Inverted) еквіваленту. Це необхідно для швидкого дешифрування адреси під час прийому USB-пакета. У процесі обміну даними з комп'ютером може виникнути потреба в зміні адреси пристрою, і ця процедура дозволяє здійснити цю зміну ефективно, забезпечуючи при цьому правильну обробку адресованих пакетів.

FinishReceiving: Процедура FinishReceiving виконує важливу функцію у програмі, пов'язану з обробкою отриманих даних через USB-пакети. Коли дані надходять через USB, вони спочатку зберігаються в спеціальному буфері. Після цього FinishReceiving бере ці неперевірені дані та починає процес їх дешифрування. Дешифрація включає перетворення кодування NRZI (напівретельного кодування з інверсією нуля) та видалення поразрядного заповнення, яке може бути застосоване для забезпечення сталої довжини передаваного сигналу. Після завершення цієї процедури дані будуть готові для подальшої обробки або використання в програмі.

USBreset: Процедура USBreset відповідає за повернення інтерфейсу USB до початкових значень, аналогічних тим, що встановлюються при включенні живлення. Це означає, що всі параметри і налаштування, які можуть змінюватися під час роботи програми, будуть скинуті до значень за замовчуванням. Після виклику цієї процедури інтерфейс USB повернеться до початкового стану, готового до нового циклу роботи.

SendPreparedUSBAnswer: Процедура SendPreparedUSBAnswer відповідає за відправлення підготовленого вмісту з вихідного буфера через лінії USB. Під час передачі даних здійснюється NRZI-кодування і додавання поразрядного заповнення. Крім того, після передачі даних додається сигнал завершення пакету (End Of Packet, EOP), що вказує на завершення передачі інформації.

ToggleDATAPID: Процедура ToggleDATAPID відповідає за перемикання ідентифікатора пакета DATAPID (PID) між значеннями DATA0 і DATA1 PID. Це перемикання необхідне під час передачі відповідно до технічних вимог USB.

ComposeZeroDATA1PIDAnswer: формує нульову відповідь для передачі. Нульова відповідь не містить даних і використовується у деяких випадках як відповідь, коли в пристрої немає доступних додаткових даних.

InitACKBuffer: Процедура InitACKBuffer виконує ініціалізацію буфера в оперативній пам'яті даними ACK (підтвердження прийому). Цей буфер часто використовується для відправки в якості відповіді, тому він завжди має бути готовий у пам'яті.

SendACK (Відправка ACK):

- Опис: SendACK є процедурою або функцією, яка відправляє пакет ACK (Acknowledge, підтвердження прийому) через лінії USB або інший комунікаційний канал.

- Застосування: Використовується для підтвердження успішного прийому даних. Наприклад, в ситуаціях, де важливо підтверджувати доставку або прийом інформації, SendACK відіграє ключову роль в комунікаційному протоколі.

InitNAKBuffer (Ініціалізація буфера NAK):

- Опис: InitNAKBuffer - це процедура або функція, яка ініціалізує буфер у відомій області пам'яті даними NAK (Negative Acknowledge, відмова в прийманні).

- Застосування: Буфер з даними NAK використовується для відправлення сигналу про відмову в прийманні як відповідь на отриману інформацію. Це може бути корисно в ситуаціях, коли система не може прийняти чи обробити отримані дані і вона повідомляє про це через NAK.

Застосування в системах зв'язку:

- USB комунікація: Обидва ці процедури часто застосовуються в USB-протоколах для забезпечення надійного обміну даними між пристроями.

- Мережеві протоколи: В інших типах мережевих протоколів SendACK і InitNAKBuffer можуть використовуватись для управління потоком даних і підтвердженнями передачі.

Роль в системах реального часу:

- Системи реального часу: SendACK та інші подібні процедури є критичними для систем реального часу, де важлива точність і швидкість обміну даними.

- Управління відмовами: InitNAKBuffer допомагає ефективно керувати відмовами в прийманні, забезпечуючи відповідність у випадках, коли не можливо обробити отримані дані.

Ці процедури інтегруються в комунікаційні протоколи для забезпечення надійності та ефективності обміну даними між пристроями і системами. Вони є важливими складовими в будь-якій системі, де критична точність і надійність комунікаційного процесу.SendNAK: Процедура SendNAK відправляє пакет NAK (відмова в прийманні) через лінії USB. Вона використовується для відправлення сигналу, що пристрій не може прийняти або обробити отримані дані.

ComposeSTALL: Процедура ComposeSTALL ініціалізує буфер у пам'яті даними, які відповідають пакету STALL (зупинка). Цей буфер часто використовується для надсилання сигналу, який вказує, що пристрій не може виконати запит, і додається до пам'яті для негайної відправки.

DecodeNRZI: Процедура DecodeNRZI виконує дешифрацію методом NRZI (Non-Return-to-Zero Inverted). Дані, які надходять по лініях USB, кодуються за допомогою цього методу, де кожен біт представлений як зміна стану лінії (піднята або опущена). Під час дешифрації ці дані перетворюються назад у вихідний бітовий потік.

BitStuff: Процедура BitStuff відіграє ключову роль у забезпеченні правильної передачі даних через USB або інші комунікаційні канали, де використовується поразрядне заповнення. Ось детальніше про цю процедуру:

Опис BitStuff:

- Функціональність: BitStuff видаляє або додає поразрядне заповнення в прийнятих даних відповідно до вимог USB протоколу. Поразрядне заповнення (bit stuffing) використовується для забезпечення правильної синхронізації при передачі

даних, зокрема в умовах, коли виникають послідовності бітів, що відповідають специфічним керуючим символам.

- Мета: Головна мета поразрядного заповнення полягає в тому, щоб уникнути можливих помилок у виявленні спеціальних символів (які могли б виникнути як частина самого потоку даних). Відповідно до стандартів USB, наприклад, додавання додаткових бітів допомагає уникнути переплутання між даними та керуючими символами.

- Реалізація: Зазвичай поразрядне заповнення виконується на апаратному рівні комп'ютера або іншого пристрою, який обробляє передачу даних, щоб забезпечити відповідність протоколу передачі і забезпечити надійну комунікацію.

2. Застосування в USB протоколі:

- USB комунікація: У стандарті USB поразрядне заповнення використовується для забезпечення правильної синхронізації між передавачем і приймачем. Наприклад, у USB 2.0 і 3.0 додавання і видалення бітів відбувається відповідно до специфікацій, що дозволяє підтримувати стабільну передачу даних з високою швидкістю.

Процес BitStuff:

- Вставка бітів: Під час вставки поразрядного заповнення, апаратне забезпечення вставляє додаткові біти в потік даних у випадках, коли зустрічаються послідовності, які можуть бути сприйняті як керуючі символи.

- Видалення бітів: При прийомі даних, процедура BitStuff видаляє ці додаткові біти перед тим, як передати чисті дані додатку або користувачу.

4. Ефективність і надійність:

- Уникнення помилок: Поразрядне заповнення забезпечує високий рівень надійності передачі даних, оскільки дозволяє системі правильно ідентифікувати спеціальні символи і уникнути їх переплутування з даними.

Процедура BitStuff є важливою складовою в комплексі технологій, що забезпечують стабільну і надійну передачу даних через різні комунікаційні канали, зокрема через USB, де вона грає ключову роль у забезпеченні відповідності протоколам передачі.

Висновки до розділу 3

В рамках третього розділу бакалаврського проєкту, ціль якого полягала в створенні частини програмного забезпечення апаратно-програмного комплексу тестування стану авіадвигунів, отримані результати, що можуть бути сформульовані у вигляді наступних висновків:

1. Розроблено алгоритми взаємодії ядра апаратно-програмного комплексу тестування стану авіадвигунів – термінального мікроконтролера з зовнішніми схемими інтерфейсу з вбудованими та термінальними вимірівальними приладами стану двигуна. Розроблений алгоритм враховує синхронізацію роботи ядра апаратно-програмного комплексу з зовнішніми вимірювальними приладами по послідовних шинах, а також по аналоговим входам термінального мікроконтролера. Алгоритм однозначно визначає організацію розподілення ресурсів процесора для виконання великої кількості паралельних процесів по вимірюванню параметрів роботи двигуна, їх первинної обробки і взаємодії з зовнішнім центральним комп'ютером, в якому зберігається історія динаміки зміни параметрів двигуна в процесі всіх його запусків при різній зовнішній температурі.

2. Розроблена програма, яка реалізує інтерфейсні функції ядра апаратно-програмного комплексу тестування стану авіадвигунів – термінального мікроконтролера по взаємодії з зовнішніми схемими інтерфейсу, що з'єднують мікроконтролер з вбудованими та термінальними акустичними вимірівальними приладами стану двигуна. Розроблена програма протестована на програмному емуляторі термінального мікроконтролера. Результати роботи програми на емуляторі показали, що виконана розробка повною мірою задовольняє вимогам технічного завданню.

ВИСНОВКИ

В результаті виконання бакалаврського проєкту, направленого на створення апаратно-програмного комплексу тестування стану авіадвигунів отримані певні результати, які можуть бути узагальнено виклдадені у вигляді наступних висновків:

1. Задача оперативного тестування стану авіаційних двигунів на основі вимірів динаміки розгінних характеристик при запуску та акустичних сигналів є важливою для перевірки стану двигуна безпосередньо в процесі його запуску та виходу на режим перед польотом. Ця задача може вирішуватися апаратно-програмним мобільним комплексом, підєднаним до мережі Інтернет. Основу комплексу складає термінальний мікроконтролер, який здійснює первинну обробку результатів вимірювань з використанням історії запусків кокретного двигуна, що зберігається на центральному комп'ютері.

2. З огляду на часові характеристики процедур первинної обробки результатів розгінних характеристик двигуна, а також об'єм потрібних ресурсів для виконання цих процедур обґрунтовано вибір структурної схеми комплекса, а також вибір типу термінального мікроконтролера для його практичної реалізації. Визначана номенклатура вимірювальних пристроїв, що підключаються до термінального мікроконтролера з використанням послідовних шин та USB.

3. Виходячи із номенклатури вбудованих в авіаційний двигун засобів контролю параметрів його роботи визначані вимоги для портів підключення відповідних сигналів до термінального мікроконтролера, визначені потрібні для цього інтерфейсні засоби, такі як аналого-цифрові перетворювачі. Визначена номенклатура мікросхем, на яких реалізується інтерфейс між вимірювальними приладами та термінальнім мікроконтролером типу PIC.

4. Проведено розрахунок часових характеристик роботи термінального мікроконтролера для визначення можливості реалізації прийому сигналів від виміріювалних пристроїв і їх первинної обробки в режимі реального часу або зі збереженням частини результатів вимірювань в оперативній пам'яті термінального мікроконтролера. На основі розрахунків часових характеристик, тактової частоти і

потрібного об’єму внутрішньої памяті, а також виходячи з інтерфейс них можливостей здійснено вибір типу мікроконтролера

5. Розроблено алгоритми взаємодії ядра апаратно-програмного комплексу тестування стану авіадвигунів – термінального мікроконтролера з зовнішніми схемими інтерфейсу з вбудованими та термінальними вимірівальними приладами стану двигуна. Розроблений алгоритм враховує синхронізацію роботи ядра апаратно-програмного комплексу з зовнішніми вимірювальними приладами по послідовних шинах, а також по аналоговим входам термінального мікроконтролера. Алгоритм однозначно визначає організацію розподілення ресурсів процесора для виконання великої кількості паралельних процесів по вимірюванню параметрів роботи двигуна, їх первинної обробки і взаємодії з зовнішнім центральним комп’ютером, в якому зберігається історія динаміки зміни параметрів двигуна в процесі всіх його запусків при різній зовнішній температурі.

6. Розроблена програма, яка реалізує інтерфейсні функції ядра апаратно-програмного комплексу тестування стану авіадвигунів – термінального мікроконтролера по взаємодії з зовнішніми схемими інтерфейсу, що з’єднують мікроконтролер з вбудованими та термінальними акустичними вимірівальними приладами стану двигуна. Розроблена програма протестована на програмному емуляторі термінального мікроконтролера. Результати роботи програми на емуляторі показали, що виконана розробка повною мірою задовольняє вимогам технічного завданню.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Kong Ch. A study on practical condition monitoring system for 2-spool Turbofan Engineusing artificial intelligent algorithms / Kong Ch., Kang M., Koh S. and Park G. //, American Institute of Aeronautics and Astronautics,-2013.- Vol.9 – P.82-90.
2. Кеба І.В. Діагностика авіаційних газотурбінних двигунів. К. Транспорт.1980 р., 248 с.
3. Vladov S. Onboard parameter identification method of the TV3-117 aircraft engine of the neural network technologies / Vladov S., Shmelov Yu., Kotli-arov K., Hrybanova S., Husarova O., Derevyanko I. and Chyzhova L.// Transactions of Kremenchuk Mykhailo Ostrohradskyi National University,-2019.- № 5 (118), - P. 90–96.
4. Kobayashi T. Integration of on-line and off-line diagnostic algorithms for aircraft engine health management / Kobayashi T. Simon D.L. //Proceedings of ASME Turbo Expo. - 2007,- P.11-19.
5. Дмитрієв С.О., Чоха Ю.М. Методика применения акустического метода идентификации неисправностей ТРДД. // Науч.-техн.журн. ”Вісник двигунобудування”. – Запоріжжя.: ЗНТУ, 2004, №2. – С.173-176.
6. Campbell L. Database Reliability Engineering: Designing and Operating Re-silient Database Systems.- O'Reilly Media, 2017, 294 p. ISBN-13:978-1491925942.
7. Tovkach S.S, Self-similarity of Operating Modes of Aviation Engine with the use of Wireless Data Transmission, Advances in Science and Technology Research Journal,-2019, № 13(2), P. 176–185. DOI: 10.12913/22998624/105887.
8. Лозня С.В. Автоматизированная диагностика неисправностей газотурбинных двигателей / Лозня С.В., Торхов М.И., Пустовой С.А., Седристый В.А., Черкасов Ю.В.// Науч.-техн.журн. ”Вісник двигунобудування”. – Запоріжжя.: ЗНТУ, 2008, №3. – С.176-181.

9. Мазиди А.М., МакКинли Р., Кусэй Д. Микроконтроллеры PIC и встроенные схемы. К.: МК-Пресс.- 2009.- 784 с.
10. Барри Б. Применение микроконтроллеров PIC-18. Архитектура, программирование и построение интерфейсов. К.: МК-Пресс.- 2008.- 576 с.
11. Цирульник С. М. Проектування мікропроцесорних систем / С. М. Цирульник, Г. Л. Лисенко. – Вінниця : ВНТУ, 2012. – 191 с.
12. Ревич Ю. В. Практическое программирование микроконтроллеров Atmel AVR на языке асемблера / Ю. В. Ревич. – СПб. : БХВ-Петербург, 2008. – 384 с.
13. Boroumand P. Mismatched Binary Hypothesis Testing Error Exponents Sensitivity / P. Boroumand, A.G. Fabregas //IEEE Transaction on Information Theory,- 2022,- Vol.68, № 10, - P. 6738-6761.
14. Дансмор Б., Скандьер Т. Справочник по телекоммуникационным технологиям. М.: Вильямс.- 2004.- 640 с.
15. Дженнингс Ф. Практическая передача данных: Модемы, сети и протоколы: Пер. с англ.-М.: Мир, 1989 – 272 с.
16. Bos J.N. Additional chain heuristics / J.N. Bos, M. Coster // Cryptographic Hardware and Embedded System- CHES'2014. LNCS-2116, Springer-Verlag. – 2014. – P.143-151.
17. Ирвин Дж., Харль Д. Передача данных в сетях: инженерный подход. СПб.: БХВ-Петербург, 2002.- 448 с.
18. Мелкозьорова О. Програмування мікроконтролерів з використанням рішень Simplisity studio / Мелкозьорова О., Гальцева І.// Компютерні науки та кібербезпека – 20121- № 2.- P.15-21 // DOI: 10.26565/2519-2310-2021-2-02

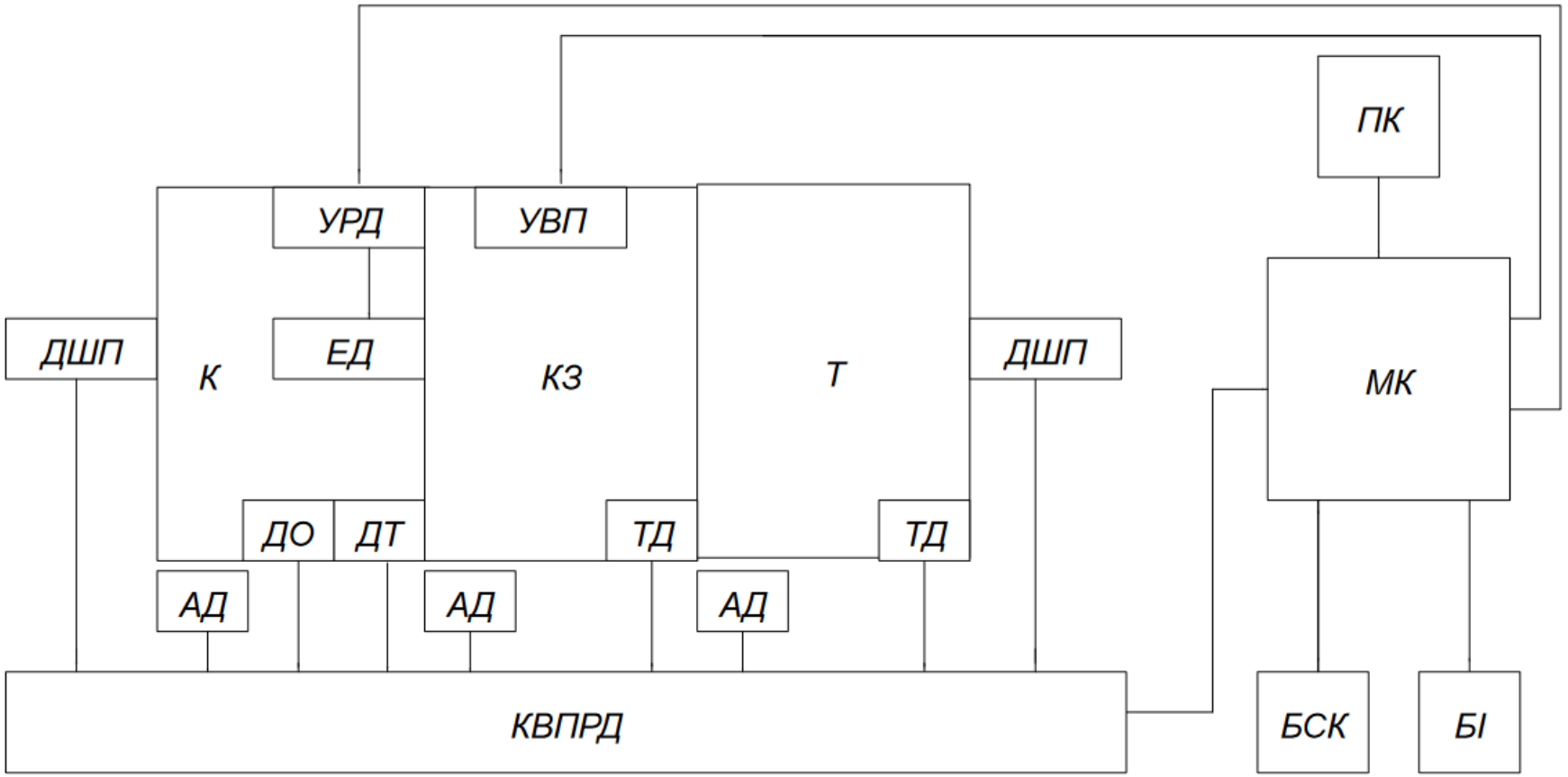
ДОДАТОК 1

Апаратно-програмний комплекс експрес-контролю стану двигунів літака

Схема електрична структурна
ІАЛЦ.468243.004 Д1

Аркушів 1

Київ 2024 р



АД - акустичний датчик
БІ - блок індикації
БСК - блок сполучення з комп'ютером
ДТ - датчик тиску
ДО - датчик обертів

ДШП - датчик швидкості потоку
К - компресор двигуна
КВПРД - канал введення параметрів роботи двигуна
КЗ - камера згоряння двигуна
МК - мікроконтролер

ПК - пульт керування
ТД - температурний датчик
ЕД - розгінний електродвигун компресора
УВП - управління впорскуванням палива
УРД - управління розгінним двигуном

ІАЛЦ.468243.004.Е1

Система моніторингу технологічного процесу
Схема електрична структурна

Літ. Маса Масштаб
Е1

Аркуш Аркушів 1

"КПІ ім. Ігоря Сікорського", ФІОТ, ІО-401

ДОДАТОК 2

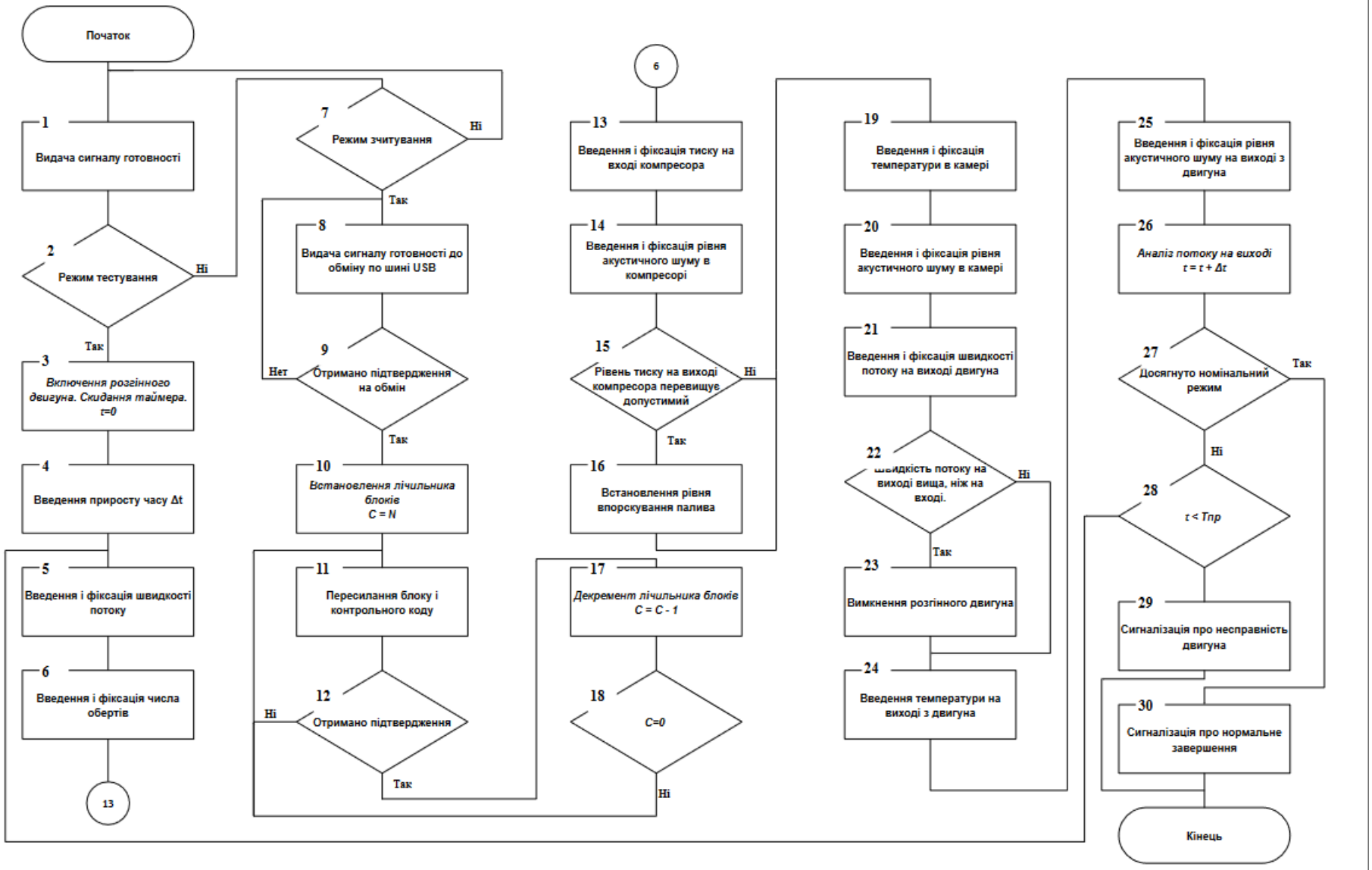
Апаратно-програмний комплекс експрес-контролю стану двигунів літака

Процедура формування сигналів управління

ІАЛЦ.468243.005 Д2

Аркушів 1

Київ 2024 р



						ІАЛЦ.468243.005,Д1	
Розробив	Прізвище І.О.				Процедура формування сигналів управління Схема алгоритму роботи	Літ.	Маса	Масштаб
Перевірив						Д		
Т.контр.						Аркуш 1	Аркушів 1	
Н.контр.	Прізвище І.О.					ЧПІ ім. Ігоря Сікорського, ФІОТ, ІО-ад1		
Затв.	Прізвище І.О.							

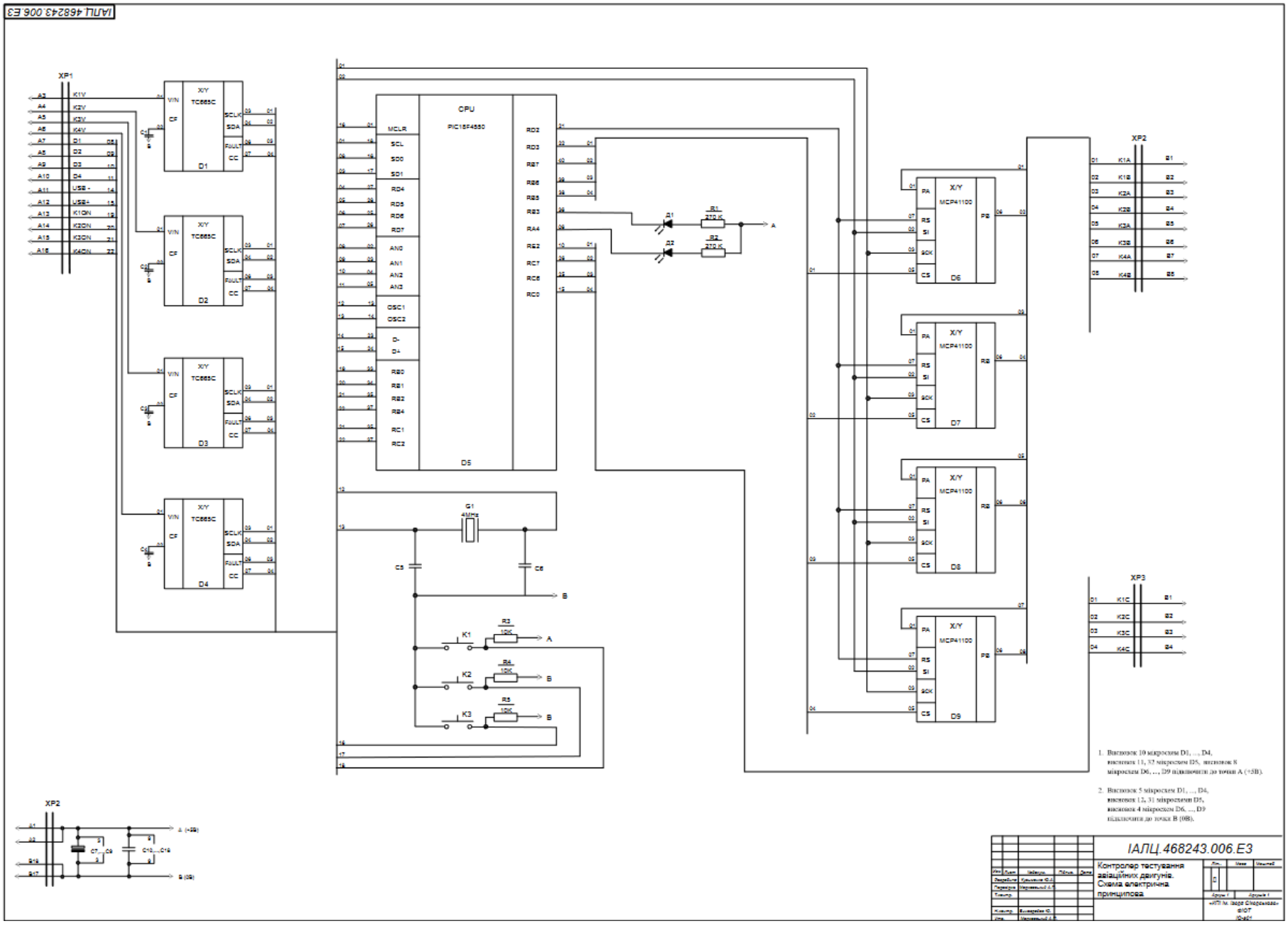
ДОДАТОК 3

Апаратно-програмний комплекс експрес-контролю стану двигунів літака

Контролер тестування авіаційних двигунів
ІАЛЦ.468243.006 Д3

Аркушів 1

Київ 2024 р



1. Виводи 10 мікросхем D1, ..., D4, виводи 11, 32 мікросхем D5, виводи 8 мікросхем D6, ..., D9 підключити до точки А (+5В).

2. Виводи 5 мікросхем D1, ..., D4, виводи 12, 31 мікросхеми D5, виводи 4 мікросхем D6, ..., D9 підключити до точки В (0В).

Змн.	Арк.	№докум.	Підпис	Дата			
Розробив		Кравченко Ю.І.			Контролер тестування авіаційних двигунів. Схема електрична принципова	Літ.	Маса	Масштаб
Перевірив		Мирошниченко А.В.						
Т.контр.						Аркуш 1	Аркушів 1	
Н.контр.		Кравченко Ю.І.				«КПІ ім. Ігоря Сікорського» ел-07 ІО-401		
Утв.		Мирошниченко А.В.						

ІАЛЦ.466243.006.Е3

ДОДАТОК 4

Апаратно-програмний комплекс експрес-контролю стану двигунів літака

Текст програмного коду

ІАЛЦ.468243.007 Д4

Аркушів 1

Київ 2024 р

ЗАСТОСУНОК. Лістинг програми

```
unit MainFormUnit;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  StdCtrls, ExtCtrls, Grids, Spin, mmsystem, shellapi, ComCtrls;

const
  WM_USBTIMERMESSAGE = WM_USER + 401;

  NO_ERROR=0;
  DEVICE_NOT_PRESENT=1;
  NO_DATA_AVAILABLE=2;
  INVALID_BAUDRATE=3;
  OVERRUN_ERROR=4;
  INVALID_DATABITS=5;
  INVALID_PARITY=6;
  INVALID_STOPBITS=7;
type
  TMainForm = class(TForm)
    LEDRadioGroup: TRadioGroup;
    EEPROMStringGrid: TStringGrid;
    IRCodeImage: TImage;
    DirectionGroupBox: TGroupBox;
    DataDirectionCheckBox0: TCheckBox;
    DataDirectionCheckBox1: TCheckBox;
    DataDirectionCheckBox2: TCheckBox;
    DataDirectionCheckBox3: TCheckBox;
    DataDirectionCheckBox4: TCheckBox;
    DataDirectionCheckBox5: TCheckBox;
    DataDirectionCheckBox6: TCheckBox;
    DataDirectionCheckBox7: TCheckBox;
    InfraTimer: TTimer;
    OutDataGroupBox: TGroupBox;
    DataOutCheckBox0: TCheckBox;
    DataOutCheckBox1: TCheckBox;
    DataOutCheckBox2: TCheckBox;
    DataOutCheckBox3: TCheckBox;
    DataOutCheckBox4: TCheckBox;
    DataOutCheckBox5: TCheckBox;
    DataOutCheckBox6: TCheckBox;
    DataOutCheckBox7: TCheckBox;
    DataInGroupBox: TGroupBox;
    DataInCheckBox0: TCheckBox;
    DataInCheckBox1: TCheckBox;
    DataInCheckBox2: TCheckBox;
    DataInCheckBox3: TCheckBox;
    DataInCheckBox4: TCheckBox;
```

```
    DataInCheckBox5: TCheckBox;
    DataInCheckBox6: TCheckBox;
    DataInCheckBox7: TCheckBox;
    DataInTimer: TTimer;
    InfraIntervalLabel: TLabel;
    RemoteCodeLabel: TLabel;
    DeviceNotPresentLabel: TLabel;
    CopyRightLabel: TLabel;
    RS232BufferTimer: TTimer;
    RS232BufferMemo: TMemo;
    TerminalMemo: TMemo;
    RS232GroupBox: TGroupBox;
    RS232ReadTimer: TTimer;
    RS232BaudLabel: TLabel;
    RS232ReadBaudLabel: TLabel;
    TransmittingLabel: TLabel;
    ParityLabel: TLabel;
    StopBitsLabel: TLabel;
    DataBitsLabel: TLabel;
    ReceivingLabel: TLabel;
    RS232SendButton: TButton;
    RS232BaudrateSpinEdit: TSpinEdit;
    ParityComboBox: TComboBox;
    StopBitsComboBox: TComboBox;
    DataBitsComboBox: TComboBox;
    EEPROMGroupBox: TGroupBox;
    EEPROMReadButton: TButton;
    EEPROMWriteButton: TButton;
    EEPROMSizeSpinEdit: TSpinEdit;
    EEPROMSizeLabel: TLabel;
    PortChoiceRadioGroup: TRadioGroup;
    DataInIntervalLabel: TLabel;
    RS232ReadIntervalLabel: TLabel;
    procedure LEDRadioGroupClick(Sender: TObject);
    procedure InfraTimerTimer(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure DataDirectionCheckBox0Click(Sender: TObject);
    procedure DataOutCheckBox0Click(Sender: TObject);
    procedure DataInTimerTimer(Sender: TObject);
    procedure EEPROMReadButtonClick(Sender: TObject);
    procedure EEPROMWriteButtonClick(Sender: TObject);
    procedure RS232SendButtonClick(Sender: TObject);
    procedure RS232BaudrateSpinEditChange(Sender: TObject);
    procedure RS232ReadTimerTimer(Sender: TObject);
    procedure CopyRightLabelClick(Sender: TObject);
    procedure RS232BufferTimerTimer(Sender: TObject);
    procedure RS232SendEditKeyPress(Sender: TObject; var Key: Char);
    procedure DataBitsComboBoxChange(Sender: TObject);
    procedure ParityComboBoxChange(Sender: TObject);
    procedure StopBitsComboBoxChange(Sender: TObject);
    procedure FormResize(Sender: TObject);
    procedure EEPROMSizeSpinEditChange(Sender: TObject);
    procedure PortChoiceRadioGroupClick(Sender: TObject);
  private
    { Private declarations }
  public
```

```
      { Public declarations }
      procedure DrawOsciloscope(var InputData:array of byte; InputLength:integer);
    protected
      procedure WMUSBMeasure(var Message: TMessage); message WM_USBTIMERMESSAGE;
    end;

  var
    MainForm: TMainForm;

  implementation
  {$R *.DFM}

    {$WARN UNSAFE_CODE OFF}
    {$WARN UNSAFE_TYPE OFF}
    {$WARN UNSAFE_CAST OFF}
  const
    ClosedApp:boolean=false;
    PreciseTimer:integer=0;
    DLLPath='AVR309.dll';
  var
    InputInfraData:array[0..255]of byte;
    InputRS232Data:array[0..10000]of byte;

  function DoGetInfraCode(var TimeCodeDiagram:array of byte; var DiagramLength:inte-
ger):integer;  stdcall external DLLPath name 'DoGetInfraCode';
  function DoSetDataPortDirection(DirectionByte:byte):integer;  stdcall external DLL-
Path name 'DoSetDataPortDirection';
  function DoGetDataPortDirection(var DataDirectionByte:byte):integer;  stdcall exter-
nal DLLPath name 'DoGetDataPortDirection';
  function DoSetOutDataPort(DataOutByte:byte):integer;  stdcall external DLLPath name
'DoSetOutDataPort';
  function DoGetOutDataPort(var DataOutByte:byte):integer;  stdcall external DLLPath
name 'DoGetOutDataPort';
  function DoGetInDataPort(var DataInByte:byte):integer;  stdcall external DLLPath name
'DoGetInDataPort';
  function DoEEPROMRead(Address:word; var DataInByte:byte):integer;  stdcall external
DLLPath name 'DoEEPROMRead';
  function DoEEPROMWrite(Address:word; DataOutByte:byte):integer;  stdcall external
DLLPath name 'DoEEPROMWrite';
  function DoRS232Send(DataOutByte:byte):integer;  stdcall external DLLPath name
'DoRS232Send';
  function DoRS232Read(var DataInByte:byte):integer;  stdcall external DLLPath name
'DoRS232Read';
  function DoSetRS232Baud(BaudRate:integer):integer;  stdcall external DLLPath name
'DoSetRS232Baud';
  function DoGetRS232Baud(var BaudRate:integer):integer;  stdcall external DLLPath name
'DoGetRS232Baud';
  function DoGetRS232Buffer(var RS232Buffer:array of byte; var RS232BufferLength:inte-
ger):integer;  stdcall external DLLPath name 'DoGetRS232Buffer';
  function DoRS232BufferSend(var RS232Buffer:array of byte; var RS232BufferLength:inte-
ger):integer;  stdcall external DLLPath name 'DoRS232BufferSend';

  function DoSetRS232DataBits(DataBits:byte):integer;  stdcall external DLLPath name
'DoSetRS232DataBits';
  function DoGetRS232DataBits(var DataBits:byte):integer;  stdcall external DLLPath name
'DoGetRS232DataBits';
  function DoSetRS232Parity(Parity:byte):integer;  stdcall external DLLPath name
'DoSetRS232Parity';
```

```
   function DoGetRS232Parity(var Parity:byte):integer;  stdcall external DLLPath name
'DoGetRS232Parity';
   function DoSetRS232StopBits(StopBits:byte):integer;  stdcall external DLLPath name
'DoSetRS232StopBits';
   function DoGetRS232StopBits(var StopBits:byte):integer; stdcall external DLLPath name
'DoGetRS232StopBits';

   function DoSetDataPortDirections(DirectionByteB, DirectionByteC, DirectionByteD:byte;
UsedPorts:byte):integer;  stdcall external DLLPath name 'DoSetDataPortDirections';
   function DoGetDataPortDirections(var DataDirectionByteB, DataDirectionByteC, Data-
DirectionByteD:byte; var UsedPorts:byte):integer;  stdcall external DLLPath name 'DoGet-
DataPortDirections';
   function    DoSetOutDataPorts(DataOutByteB,    DataOutByteC,    DataOutByteD:byte;
UsedPorts:byte):integer;  stdcall external DLLPath name 'DoSetOutDataPorts';
   function DoGetOutDataPorts(var DataOutByteB, DataOutByteC, DataOutByteD:byte; var
UsedPorts:byte):integer;  stdcall external DLLPath name 'DoGetOutDataPorts';
   function  DoGetInDataPorts(var  DataInByteB,  DataInByteC,  DataInByteD:byte;  var
UsedPorts:byte):integer;  stdcall external DLLPath name 'DoGetInDataPorts';


   procedure    TemperatureTimerProc(uID:integer;    uMsg:integer;dwUser,dw1,dw2:inte-
ger);stdcall;
   var
     LocalMainForm:TMainForm;
   begin
     LocalMainForm:=TMainForm(dwUser);
     if ClosedApp then Exit;
     PostMessage(LocalMainForm.Handle,WM_USBTIMERMESSAGE,0,0);
   end;

   procedure TMainForm.WMUSBMeasure(var Message: TMessage);
   var
     MyMsg:TMsg;
   begin
     Message.Result:=1;
     InfraTimerTimer(self);
     while PeekMessage(MyMsg,Handle,WM_USBTIMERMESSAGE,WM_USBTIMERMESSAGE,PM_REMOVE) do
     begin end; //removing of all waiting WM_USBTIMERMESSAGE messages
     inherited;
   end;

   procedure TMainForm.LEDRadioGroupClick(Sender: TObject);
   var
     PortValue:byte;
   begin
     PortValue:=1 shl (LEDRadioGroup.ItemIndex);
     DoSetOutDataPorts(PortValue,PortValue,PortValue,1 shl PortChoiceRadioGroup.ItemIn-
dex);
     DataInTimerTimer(self);
   end;

   procedure TMainForm.InfraTimerTimer(Sender: TObject);
   var
     DataLength:integer;
   begin
     InfraIntervalLabel.Caption:=IntToStr(InfraTimer.Interval)+'ms';
     if (DoGetInfraCode(InputInfraData,DataLength)<>NO_ERROR) then
       begin
```

```
        //DeviceNotPresentLabel.Visible:=true;
        Exit;
      end;
    DeviceNotPresentLabel.Visible:=false;
    if (DataLength=0) then Exit;
    DrawOsciloscope(InputInfraData,DataLength);
  end;

  procedure TMainForm.DrawOsciloscope(var InputData:array of byte; InputLength:inte-
ger);
  var
    i:integer;
    x,y,yHi,yLo:integer;
    TotalLength:integer;
    OscRect:TRect;
  const
    vyska=50;
    Xzaciatok=10;
    Xkoniec=10;
  begin
    TotalLength:=Xzaciatok+Xkoniec;
    for i:=0 to InputLength-1 do
      TotalLength:=TotalLength+InputData[i];

    IRCodeImage.Picture.Bitmap.Width:=TotalLength;
    IRCodeImage.Picture.Bitmap.Height:=IRCodeImage.Height;
    IRCodeImage.Picture.Bitmap.Canvas.Brush.Color:=clBlack;
    IRCodeImage.Picture.Bitmap.Canvas.Brush.Style:=bsSolid;
    IRCodeImage.Picture.Bitmap.Canvas.Pen.Color:=clYellow;
    IRCodeImage.Picture.Bitmap.Canvas.Pen.Style:=psSolid;
    IRCodeImage.Picture.Bitmap.Canvas.Pen.Mode:=pmCopy;

    OscRect.Top:=0;
    OscRect.Left:=0;
    OscRect.Right:=IRCodeImage.Picture.Bitmap.Width;
    OscRect.Bottom:=IRCodeImage.Picture.Bitmap.Height;
    IRCodeImage.Picture.Bitmap.Canvas.FillRect(OscRect);
    yHi:=IRCodeImage.Picture.Bitmap.Height-vyska;
    yLo:=IRCodeImage.Picture.Bitmap.Height-5;
    x:=IRCodeImage.Picture.Bitmap.Width;
    y:=yHi;
    IRCodeImage.Picture.Bitmap.Canvas.MoveTo(x,y);
    x:=x-Xkoniec;
    IRCodeImage.Picture.Bitmap.Canvas.LineTo(x,y);
    y:=yLo;
    IRCodeImage.Picture.Bitmap.Canvas.LineTo(x,y);

    for i:=InputLength-1 downto 0 do
      begin
        x:=x-InputData[i];
        IRCodeImage.Picture.Bitmap.Canvas.LineTo(x,y);
        if y=yLo then y:=yHi
        else y:=yLo;
        IRCodeImage.Picture.Bitmap.Canvas.LineTo(x,y);
      end;
    IRCodeImage.Picture.Bitmap.Canvas.LineTo(x-Xzaciatok,y);
    IRCodeImage.Repaint;
  end;
```

```
  procedure TMainForm.FormCreate(Sender: TObject);
  begin
    Application.HintColor:=clYellow;
    Application.HintPause:=0;
    Application.HintHidePause:=20000;
    //************ Uncomment this next 3 lines if you want higher speed in IR code
polling ******************
    //PreciseTimer:=timeSetEvent(InfraTimer.Interval,1,  TemperatureTimerProc,  inte+
ger(self), TIME_PERIODIC);
    //InfraTimer.Enabled:=false;
    //SetThreadPriority(GetCurrentThread,THREAD_PRIORITY_TIME_CRITICAL);
      RS232BaudrateSpinEditChange(self);
    DataBitsComboBox.ItemIndex:=3;
    ParityComboBox.ItemIndex:=0;
    StopBitsComboBox.ItemIndex:=0;
    DataBitsComboBoxChange(self);
    ParityComboBoxChange(self);
    StopBitsComboBoxChange(self);
    EEPROMSizeSpinEditChange(self);

    DataInIntervalLabel.Parent:=PortChoiceRadioGroup;
    DataInIntervalLabel.Left:=PortChoiceRadioGroup.Width-DataInIntervalLabel.Width-5;
    DataInIntervalLabel.Top:=PortChoiceRadioGroup.Height-DataInIntervalLabel.Height-5;
  end;

  procedure TMainForm.FormClose(Sender: TObject; var Action: TCloseAction);
  begin
    ClosedApp:=true;
    if PreciseTimer<>0 then
      begin
        timeKillEvent(PreciseTimer);
        PreciseTimer:=0;
      end;
  end;

  procedure TMainForm.DataDirectionCheckBox0Click(Sender: TObject);
  var
    DirectionByte:byte;
    i:integer;
    MyComponent:TComponent;
  begin
    if DataInTimer.Tag<>0 then Exit;
    DirectionByte:=0;
    for i:= 0 to 7 do
      begin
        MyComponent:=FindComponent('DataDirectionCheckBox'+IntToStr(i));
        if (MyComponent as TCheckBox).Checked then
          DirectionByte:=DirectionByte+MyComponent.Tag;
      end;
    DoSetDataPortDirections(DirectionByte,DirectionByte,DirectionByte,1 shl PortChoic+
eRadioGroup.ItemIndex);
    DataInTimerTimer(self);
  end;


  procedure TMainForm.DataOutCheckBox0Click(Sender: TObject);
  var
```

```
    OutByte:byte;
    i:integer;
    MyComponent:TComponent;
  begin
    if DataInTimer.Tag<>0 then Exit;
    OutByte:=0;
    for i:= 0 to 7 do
      begin
        MyComponent:=FindComponent('DataOutCheckBox'+IntToStr(i));
        if (MyComponent as TCheckBox).Checked then
          OutByte:=OutByte+MyComponent.Tag;
      end;
    DoSetOutDataPorts(OutByte,OutByte,OutByte,1 shl PortChoiceRadioGroup.ItemIndex);
    DataInTimerTimer(self);
  end;

  procedure TMainForm.DataInTimerTimer(Sender: TObject);
  var
    InByte,InByteB,InByteC,InByteD:byte;
    UsedPorts:byte;
    i:integer;
    MyComponent:TComponent;
  begin
    DataInTimer.Tag:=1;
    DataInIntervalLabel.Caption:=IntToStr(DataInTimer.Interval)+'ms';
    if DoGetInDataPorts(InByteB,InByteC,InByteD,UsedPorts)<>NO_ERROR then
      begin
        DeviceNotPresentLabel.Visible:=true;
        DataInTimer.Tag:=0;
        Exit;
      end;
    if UsedPorts<=1 then PortChoiceRadioGroup.ItemIndex:=0;
    case PortChoiceRadioGroup.ItemIndex of
      1:InByte:=InByteC;
      2:InByte:=InByteD;
    else
      InByte:=InByteB;
    end;
    DeviceNotPresentLabel.Visible:=false;
    for i:= 0 to 7 do
      begin
        MyComponent:=FindComponent('DataInCheckBox'+IntToStr(i));
        (MyComponent as TCheckBox).Checked:=((MyComponent as TCheckBox).Tag and In+
Byte)<>0;
      end;

    if DoGetOutDataPorts(InByteB,InByteC,InByteD,UsedPorts)<>NO_ERROR then
      begin
        DeviceNotPresentLabel.Visible:=true;
        DataInTimer.Tag:=0;
        Exit;
      end;
    if UsedPorts<=1 then PortChoiceRadioGroup.ItemIndex:=0;
    case PortChoiceRadioGroup.ItemIndex of
      1:InByte:=InByteC;
      2:InByte:=InByteD;
    else
      InByte:=InByteB;
```

```
    end;
    DeviceNotPresentLabel.Visible:=false;
    for i:= 0 to 7 do
      begin
        MyComponent:=FindComponent('DataOutCheckBox'+IntToStr(i));
        (MyComponent as TCheckBox).Checked:=((MyComponent as TCheckBox).Tag and In+
Byte)<>0;
      end;

    if DoGetDataPortDirections(InByteB,InByteC,InByteD,UsedPorts)<>NO_ERROR then
      begin
        DeviceNotPresentLabel.Visible:=true;
        DataInTimer.Tag:=0;
        Exit;
      end;
    if UsedPorts<=1 then PortChoiceRadioGroup.ItemIndex:=0;
    case PortChoiceRadioGroup.ItemIndex of
      1:InByte:=InByteC;
      2:InByte:=InByteD;
    else
      InByte:=InByteB;
    end;
    DeviceNotPresentLabel.Visible:=false;
    for i:= 0 to 7 do
      begin
        MyComponent:=FindComponent('DataDirectionCheckBox'+IntToStr(i));
        (MyComponent as TCheckBox).Checked:=((MyComponent as TCheckBox).Tag and In+
Byte)<>0;
      end;
    DataInTimer.Tag:=0;
  end;


  procedure TMainForm.EEPROMReadButtonClick(Sender: TObject);
  var
    i:integer;
    DataByte:byte;
  begin
    EEPROMStringGrid.Col:=1;
    for i:=0 to EEPROMStringGrid.RowCount-1 do
      begin
        if DoEEPROMRead(i,DataByte)=0 then
          begin
            EEPROMStringGrid.Cells[1,i]:=IntToStr(DataByte);
            EEPROMStringGrid.Row:=i;
          end
        else
          begin
            raise Exception.Create('Unable to read from EEPROM!'+#13+#10+'(maybe device
is not present)');
          end;
      end;
  end;

  procedure TMainForm.EEPROMWriteButtonClick(Sender: TObject);
  var
    i:integer;
    DataByte:integer;
```

```
  begin
    EEPROMStringGrid.Col:=1;
    try
      for i:=0 to EEPROMStringGrid.RowCount-1 do
        begin
          EEPROMStringGrid.Row:=i;
          DataByte:=StrToInt(EEPROMStringGrid.Cells[1,i]);
          if (DataByte>255)or(DataByte<0) then
            raise Exception.Create('Number is not in BYTE range: 0-255 !');
          if DoEEPROMWrite(i,DataByte)<>0 then
            begin
              raise Exception.Create('Error to write to EEPROM!'+#13+#10+'(maybe device
is not present)');
            end;
        end;
    except on EConvertError do
      begin
        ActiveControl:=EEPROMStringGrid;
        MessageBox(Handle,'Invalid BYTE number'+#13+#10+'(unable to convert to numeric
value)',PChar(Caption),MB_ICONERROR);
      end
    else
      begin
        ActiveControl:=EEPROMStringGrid;
        raise;
      end;
    end;
  end;


  procedure TMainForm.RS232SendButtonClick(Sender: TObject);
  type
    MyBuffer=array [0..100000] of byte;
  var
    j,i:integer;
    P:MyBuffer;
    MyStr:string;
  begin
    MyStr:=TerminalMemo.Text;
    i:=Length(MyStr);
    Screen.Cursor:=crHourGlass;
    TransmittingLabel.Visible:=true;
    TransmittingLabel.Repaint;
    try
      for j:=1 to i do
        begin
          P[j-1]:=byte(MyStr[j]);
        end;
      DoRS232BufferSend(P, i);
    finally
      Screen.Cursor:=crArrow;
      TransmittingLabel.Visible:=false;
      TransmittingLabel.Repaint;
    end;
  end;

  procedure TMainForm.RS232BaudrateSpinEditChange(Sender: TObject);
  var
```

```
  RetVal:integer;
begin
  try
    RetVal:=DoSetRS232Baud(RS232BaudrateSpinEdit.Value);
    RS232BaudrateSpinEdit.Color:=clWindow;
    RS232BaudrateSpinEdit.Font.Style:=[];
    case RetVal of
      DEVICE_NOT_PRESENT:RS232BaudrateSpinEdit.Color:=clRed;
      INVALID_BAUDRATE: RS232BaudrateSpinEdit.Font.Style:=[fsStrikeOut];
    end;
  except
  end;
end;

procedure TMainForm.RS232ReadTimerTimer(Sender: TObject);
var
  BaudRate:integer;
  Databits:byte;
  Parity:byte;
  Stopbits:byte;
begin
  RS232ReadIntervalLabel.Caption:=IntToStr(RS232ReadTimer.Interval)+'ms';
  if DoGetRS232Baud(BaudRate)=NO_ERROR then
    begin
      RS232ReadBaudLabel.Caption:='='+IntToStr(BaudRate);
      DeviceNotPresentLabel.Visible:=false;
    end
  else
    begin
      DeviceNotPresentLabel.Visible:=true;
    end;
  if not DataBitsComboBox.DroppedDown then
    if DoGetRS232DataBits(Databits)=NO_ERROR then
      begin
        DataBitsComboBox.ItemIndex:=Databits-5;
        DeviceNotPresentLabel.Visible:=false;
      end
    else
      begin
        //DeviceNotPresentLabel.Visible:=true;
      end;
  if not ParityComboBox.DroppedDown then
    if DoGetRS232Parity(Parity)=NO_ERROR then
      begin
        ParityComboBox.ItemIndex:=Parity;
        DeviceNotPresentLabel.Visible:=false;
      end
    else
      begin
        //DeviceNotPresentLabel.Visible:=true;
      end;
  if not StopBitsComboBox.DroppedDown then
    if DoGetRS232StopBits(Stopbits)=NO_ERROR then
      begin
        StopBitsComboBox.ItemIndex:=Stopbits;
        DeviceNotPresentLabel.Visible:=false;
      end
    else
```

```
        begin
          //DeviceNotPresentLabel.Visible:=true;
        end;
  end;

  procedure TMainForm.CopyRightLabelClick(Sender: TObject);
  begin
    ShellExecute(Handle,'open','http://www.cesko.host.sk',nil,'.',0);
    ShellExecute(Handle,'open','http://www.atmel.com',nil,'.',0);
  end;

  procedure TMainForm.RS232BufferTimerTimer(Sender: TObject);
  var
    DataLength:integer;
    i:integer;
  begin
    DataLength:=SizeOf(InputRS232Data);
    if (DoGetRS232Buffer(InputRS232Data,DataLength)<>NO_ERROR) then
      begin
        //DeviceNotPresentLabel.Visible:=true;
        Exit;
      end;
    if (DataLength=0) then Exit;
    Screen.Cursor:=crAppStart;
    ReceivingLabel.Visible:=true;
    ReceivingLabel.Repaint;
    try
      TerminalMemo.SelStart:=Length(TerminalMemo.Text);
      TerminalMemo.SelLength:=0;
      for i:=0 to DataLength-1 do
        begin
          RS232BufferMemo.Lines.Add(IntToStr(InputRS232Data[i])+'   ->   '+IntToHex(In+
putRS232Data[i],2)+' -> '+chr(InputRS232Data[i]));
          TerminalMemo.SelText:=chr(InputRS232Data[i]);
        end;
    finally
      Screen.Cursor:=crArrow;
      ReceivingLabel.Visible:=false;
      ReceivingLabel.Repaint;
    end;
  end;

  procedure TMainForm.RS232SendEditKeyPress(Sender: TObject; var Key: Char);
  begin
    DoRS232Send(byte(Key));
  end;

  procedure TMainForm.DataBitsComboBoxChange(Sender: TObject);
  var
    RetVal:integer;
  begin
    RetVal:=DoSetRS232DataBits(StrToInt(DataBitsComboBox.Text));
    DataBitsComboBox.Color:=clWindow;
    DataBitsComboBox.Font.Style:=[];
    case RetVal of
      DEVICE_NOT_PRESENT: DataBitsComboBox.Color:=clRed;
      INVALID_DATABITS: DataBitsComboBox.Font.Style:=[fsStrikeOut];
    end;
```

```
end;

procedure TMainForm.ParityComboBoxChange(Sender: TObject);
var
  RetVal:integer;
begin
  RetVal:=DoSetRS232Parity(ParityComboBox.ItemIndex);
  ParityComboBox.Color:=clWindow;
  ParityComboBox.Font.Style:=[];
  case RetVal of
    DEVICE_NOT_PRESENT: ParityComboBox.Color:=clRed;
    INVALID_PARITY: ParityComboBox.Font.Style:=[fsStrikeOut];
  end;
end;

procedure TMainForm.StopBitsComboBoxChange(Sender: TObject);
var
  RetVal:integer;
begin
  RetVal:=DoSetRS232StopBits(StopBitsComboBox.ItemIndex);
  StopBitsComboBox.Color:=clWindow;
  StopBitsComboBox.Font.Style:=[];
  case RetVal of
    DEVICE_NOT_PRESENT: StopBitsComboBox.Color:=clRed;
    INVALID_STOPBITS: StopBitsComboBox.Font.Style:=[fsStrikeOut];
  end;
end;

procedure TMainForm.FormResize(Sender: TObject);
begin
  TerminalMemo.Width:= ClientWidth-EEPROMStringGrid.Width-TerminalMemo.Left;
  EEPROMStringGrid.Left:=ClientWidth-EEPROMStringGrid.Width;
  TerminalMemo.Width:= EEPROMStringGrid.Left-TerminalMemo.Left;
  TerminalMemo.Height:= ClientHeight-TerminalMemo.Top;
  RS232BufferMemo.Height:= ClientHeight-RS232BufferMemo.Top;
end;

procedure TMainForm.EEPROMSizeSpinEditChange(Sender: TObject);
var
  i:integer;
begin
  EEPROMStringGrid.RowCount:=EEPROMSizeSpinEdit.Value;
  for i:=0 to EEPROMStringGrid.RowCount-1 do
    EEPROMStringGrid.Cells[0,i]:=IntToStr(i);
end;


procedure TMainForm.PortChoiceRadioGroupClick(Sender: TObject);
begin
  DataInTimerTimer(self);
end;


end.
```