СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ЕЛЕКТРОНІКИ КАФЕДРА
ПРОГРАМУВАННЯ ТА ІНФОРМАТИКИ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної випускної роботи

освітній ступінь_____
(бакалавр, магістр)

спеціальність _____
(шифр і назва спеціальності)

спеціалізація _____
(назва спеціалізації)

на тему_____
_____
_____

Виконав: студент групи _____ _____ _____
(підпис) (ініціали і прізвище)

Керівник _____ _____
(підпис) (ініціали і прізвище)

Завідувач кафедри _____ _____
(підпис) (ініціали і прізвище)

Рецензент_____

Сєвєродонецьк - 2020

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) _____
(повне найменування інституту, факультету) .

_____

Кафедра _____
(повна назва кафедри)

Освітній ступінь _____
(бакалавр, магістр)

спеціальність _____
(шифр і назва спеціальності)

спеціалізація _____
(назва спеціалізації)

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____

“____”

_____20___року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ ВИПУСКНУ РОБОТУ СТУДЕНТУ

_____
(прізвище, ім’я, по батькові)

1. Тема роботи_____
_____
_____
_____

Керівник роботи_____,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджений наказом університету від “___”_____20__року №___

2. Строк подання студентом роботи_____

3. Вихідні дані до роботи_____
_____
_____
_____

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

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслеників)
_____
_____
_____
_____
_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7.Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання кваліфікаційної випускної роботи	Строк виконання етапів	Примітка

Студент _____ _____
(підпис) (ініціали і прізвище)

Керівник роботи _____ _____
(підпис) (ініціали і прізвище)

РЕФЕРАТ

Пояснювальна записка складається з 92 сторінок, 16 рисунків, 5 таблиць та 5 джерел літератури.

Дана робота представляє собою розробку модулю вводу-виводу дискретних сигналів. Проведено аналіз можливих рішень досягення поставленої задачі та обрано найбільш доцільний варіант. Проведено розробку як апаратної, так і програмної частини пристрою.

Ключові слова: АВТОМАТИЗАЦІЯ, ДИСКРЕТНІ СИГНАЛИ, ДРУКОВАНА ПЛАТА, МІКРОКОНТРОЛЕР, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

ЗМІСТ

ВСТУП

Сучасні системи автоматизації виробництва активно використовують в своїй роботі багаточисленні модулі управління, в тому числі і модулі вводу і формування дискретних сигналів. Область їх використання дуже різноманітна, й від того не менш затребувана. На прикладі своєї курсової роботи хочу розглянути і розробити модуль вводу-виводу дискретних сигналів.

Актуальність даної дипломної роботи зумовлена широкою уживаністю та поширенням технологій автоматизації, використання котрих можна знайти майже на будь-якому виробництві. Модуль, що буде розроблятися, розрахований для використання в залізно-дорожньому управлінні та на АЕС і нестиме в собі ціль полегшити та автоматизувати керування.

Об’єктом дослідження, відповідно, буде ввід і формування дискретних сигналів.

Предметом досліджень є різні можливості розробки бажаного пристрою й вибір найбільш підходящого з них для поточних цілей.

Завдянням дослідження є розробка модулю вводу, обробки і формування вихідного дискретного сигналу.

Практичне застосування даного модулю може бути таким:

- Керування залізно-дорожніми перегонами;
- Керування виробництвом, яке застосовує в своїй роботі релейні перемикачі;
- Керування обладнанням на АЕС.

1. ОГЛЯД МОЖЛИВИХ РІШЕНЬ

У системах автоматизації дуже поширені виконавчі сигнали, які надходять від кінцевих вимикачів, датчиків охоронної або пожежної сигналізації, датчиків заповнення ємностей, датчиків збігання стрічки на конвеєрі, датчиків наближення і т. п. Такі сигнали називаються "дискретними".

Модулі введення дискретних сигналів в промисловій автоматизації мають кілька різних типів входів:

- вхід типу "сухий контакт";
- дискретний вхід для логічних сигналів у формі напруги;
- вхід дискретних сигналів 110 ... 220 В.

"Сухим" контактом в системах автоматизації називають джерело інформації, що не має вбудованого джерела енергії, наприклад, контакти реле або дискретні виходи типу "відритий колектор". Для передачі інформації про стан такого контакту необхідне зовнішнє джерело струму або напруги.

Мікроконтролер модуля введення виконує періодичне сканування входів або за запитом ПЛК. Мікроконтролер виконує також усунення ефекту "брязкоту" "сухих" контактів. Команди опитування входів, встановлення адреси, швидкості обміну, формату даних і ін. надсилаються в модуль через послідовний інтерфейс, зазвичай RS-485.

Для правильного застосування модулів дискретного вводу необхідно знати структуру і характеристики вхідних каскадів.

Дискретні входи гальванічно розв'язані від іншої частини модуля введення. Розв'язка виконується, як правило, за допомогою оптронів з двома випромінюючими діодами, ввімкненими зустрічно. Це забезпечує можливість підключення до входів дискретних сигналів будь-якої полярності. Гальванічна ізоляція може бути поканальною або груповою. Найчастіше використовується

групова ізоляція, оскільки при цьому майже вдвічі зменшується кількість вхідних клем модуля.

Рівень логічної одиниці дискретних сигналів становить зазвичай від 3В до 30В, рівень логічного нуля - від 0 до 2 В.

Каскади для введення високої напруги можуть бути із загальним проводом або незалежні.

Для відображення стану дискретних входів (ввімкнено / вимкнено) використовують світлодіоди, які включають або до оптрона, або після нього.

Вивід дискретних сигналів використовується для управління станом ввімкнено / вимкнено виконавчих пристроїв. Пристрої виведення відрізняються великим різноманіттям. Знання структури вихідних каскадів необхідно для правильного їх застосування.

Вихідні каскади зі стандартними ТТЛ або КМОП логічними рівнями в промисловій автоматизації використовуються рідко. Це пов'язано з тим, що навантаженням дискретних виходів є не логічні входи електронних пристроїв, а найчастіше електромеханічні реле, пускачі, крокові двигуни та ін. Дискретні виходи зазвичай будуються на основі потужних біполярних транзисторів з відкритим колектором або польових транзисторів (зазвичай МОП) з відкритим стоком. З точки зору схемотехніки застосування цих каскадів еквівалентні, тому ми будемо їх називати "каскади ВК". Каскади з ВК забезпечують більшу гнучкість, дозволяючи отримати необхідні для навантаження струм або напругу за допомогою зовнішнього джерела живлення.

Найкращим рішенням для побудови дискретних виходів є мікросхеми інтелектуальних ключів, які містять в собі не тільки потужний транзистор з відкритим стоком, а й ланцюг його захисту від перевантаження по струму, напрузі, короткого замикання, переполюсовки і перегріву, а також електростатичних розрядів. При перегріві вихідного каскаду або перевищенні струму навантаження інтелектуальний ключ вимикається.

Найбільш широко поширені вихідні каскади ВК модулів виводу двох типів: для впадаючого струму і випливаючого. Різниця між ними полягає в тому, який висновок є загальним для декількох навантажень: заземлений або з'єднаний з шиною живлення.

Каскади з відкритим колектором (стоком) зручні тим, що дозволяють використовувати зовнішнє джерело живлення з напругою, відмінною від напруги живлення модулів виводу. Крім того, в цих схемах замість джерела живлення можна використовувати те ж саме джерело, що і для живлення модулів виводу.

Для управління навантаженнями, що живляться від великого струму або від джерела напруги 110...220В використовують вихідні каскади з електромагнітними або твердотільними (напівпровідниковими) реле, тиристорами, симисторами.

Основною перевагою електромагнітних реле є дуже низьке падіння напруги на замкнутих контактах, що виключає необхідність їх охолодження. Недоліком є обмежена кількість спрацьовувань.

Пристрої (модулі) введення-виведення є інтерфейсом між процесором ПЛК і реальним світом. В ідеальному випадку було б бажано мати в процесорі значення виміряних сигналів в будь-який момент часу. Однак, оскільки кількість каналів вводу-виводу в деяких системах може досягати тисяч, а вимірювальні канали завжди мають обмежену пропускну здатність, виміряні значення надходять в процесор в дискретні моменти часу.

Існує кілька рівнів і способів опитування безлічі каналів введення. Сучасний модуль введення має свій власний мікроконтролер, який виконує циклічне опитування всіх своїх каналів і поміщає отримані дані в буфер. Якщо за алгоритмом роботи системи автоматизації використовуються тільки кілька каналів модуля, то невикористовувані канали можна замаскувати (виключити їх з процедури опитування), якщо це потрібно для збільшення швидкодії системи. При надходженні в модуль команди зчитування значень зі входів зібрані дані

передаються з буфера модуля в ПЛК, де поміщаються в буфер OPC сервера або в певну область ОЗУ.

Опитування модулів може виконуватися циклічно з однаковою частотою для всіх модулів, або з різною частотою. Другий варіант дозволяє зменшити завантаженість шини, по якій виконується обмін даними між модулями введення і процесорним модулем.

Циклічне опитування всіх модулів з заздалегідь заданою частотою сильно завантажує шину, по якій модулі введення зв'язуються з процесором. Це особливо очевидно, якщо процесор сканує входи для виявлення сигналу від аварійного датчика, який може спрацювати один раз в 10 років, або якщо вводяться дані від датчика температури в умовах, коли температура постійна. У подібних випадках більш ефективні многомайстерні шини (наприклад, CAN або Profibus), які дозволяють використовувати режим підписки, при якому процесор модуля введення, в якому відбулася зміна стану входу, є ініціатором обміну даними.

Найбільшого поширення в промисловій автоматизації знайшли одномайстерні шини і циклічне опитування (полінг - від "polling") модулів введення в силу своєї простоти і порівняно низьку вартість.

Модулі введення і виведення в промисловій автоматизації мають гальванічну ізоляцію між вхідними (вихідними) зажимами і шиною контролера. Напруга ізоляції становить від 2500В (рідше від 500В) до 4000 В.

Іноді потрібно виконати одночасне опитування входів всіх модулів введення або вивести дані одночасно в канали всіх модулів виводу. Для вирішення цієї проблеми використовують широкомовні команди, які сприймаються усіма модулями одночасно і вони виконують введення або виведення даних в свої буферні регістри в один і той же час. Після цього звичайним циклічним опитуванням дані по черзі вводяться в процесорний модуль.

Модулі введення з'єднуються з процесором послідовною або паралельною шиною. У магістрально-модульних системах використовуються паралельні шини ISA, PCI, Compact PCI, PCI Express, PC / 104, SpeedBus, VME і ін., В модульних ПЛК - нестандартні послідовні і паралельні шини. У контролерах з розподіленими (віддаленими) модулями вводу-виводу найбільш поширені послідовні шини на основі інтерфейсів RS-485 і CAN.

Перевагою паралельної шини є висока пропускна здатність, що дозволяє виконувати сканування модулів введення з високою частотою і використовувати модулі аналогового вводу з тактовою частотою АЦП до 100 кГц. Однак невелика довжина паралельної шини, обмежена розсинхронізацією окремих біт в переданому слові, не дозволяє підключити до одного контролера більше 32 модулів. Контролери з послідовною шиною мають протилежні властивості.

Більшість паралельних і послідовних шин контролерів є одномайстерними, оскільки багатомайстерні шини істотно складніше і дорожче.

Обмін даними з модулем виконується за адресою, який зазвичай записується в ПЗУ модуля. Іноді адресою є номер слота, в який вставляється модуль або положення мікроперемикача.

Ланцюги входів і виходів модулів введення повинні мати гальванічну ізоляцію. Гальванічна ізоляція може бути по канальна, коли кожен канал ізольований від інших, або групова. Зазвичай використовується групова ізоляція. У віддалених модулях розподілених ПЛК може бути використана індивідуальна гальванічна ізоляція інтерфейсу RS-485 кожного модуля або групова ізоляція інтерфейсів декількох модулів за допомогою одного модуля розв'язує повторювача інтерфейсів. Для передачі напруги живлення в ізольовану частину модуля використовуються DC-DC перетворювачі, побудовані із застосуванням розв'язуючих мініатюрних трансформаторів.

Сучасні модулі введення-виведення можуть виконувати крім функцій введення деяку обробку інформації, що вводить і додаткові функції: компенсацію температури холодного спаю термопар, лінеаризацію нелінійних

датчиків, діагностику обриву датчика, автоматичне калібрування, регулювання, управління рухом. Перенесення частини функцій контролера в модулі вводу-виводу є сучасною тенденцією, спрямованої на збільшення ступеня розпаралелювання завдань управління, забезпечення незалежності локальних модулів (які за своїми функціями наближаються до ПЛК) та зменшення потоку інформації між паралельно працюючими процесорами в модулях введення-виведення.

Основною частиною модуля введення є аналого-цифровий перетворювач (АЦП). Зазвичай використовують один АЦП для введення декількох (зазвичай 8 або 16) аналогових сигналів. Для підключення джерел сигналу до АЦП використовується аналоговий комутатор на МОП-транзисторах. Введення декількох сигналів виконується послідовно в часі. У випадках, коли необхідне одночасне введення, використовують модулі, в яких кожен канал має свій АЦП.

У модулях введення зазвичай використовують диференціальні входи, які дозволяють виконати більш перешкодозахищений канал передачі аналогового сигналу в порівнянні з поодинокими (НЕ диференціальними) входами. Деякі модулі дозволяють програмно задавати конфігурацію входів: диференціальні або поодинокі.

Вхідні ланцюги пристроїв введення прийнято захищати від статичної електрики, від підвищеної напруги, від зміни полярності. Для захисту використовують спеціальні мікросхеми захисту, в яких активним елементом є МОП-транзисторний ключ. При підвищенні напруги вище допустимого ключ замикається, оберігаючи чутливі входи від підвищеної напруги. Вимірювання ланцюга будують проводитись таким чином, щоб опір відкритого МОП ключа не вносив похибку в результат вимірювання. Для цього ключ використовують або для передачі потенціалу, коли струм, що протікає через відкритий ключ, дуже малий, або для передачі струму, коли інформація переноситься в формі струму і тому падіння напруги на ключі не вносить похибка в переданий сигнал.

Модулі введення можуть мати діапазони вхідних сигналів, що програмно перемикаються. Діапазони вимірювань зазвичай задаються для всіх входів однаковими.

Сучасна елементна база дозволяє будувати недорогі модулі аналогового вводу з похибкою вимірювань ± 0,05%, що ще 10 років тому можна було реалізувати тільки в стаціонарних і дорогих вольтметрах.

Для комутації вхідних ключів модуля використовується програма, що виконується мікро контролером. Ця процедура досить проста і для її виконання можна використовувати мікроконтролер, що входить до складу деяких АЦП. Це дозволяє зменшити кількість каналів гальванічної розв'язки між аналоговими входами і портом RS-485.

Мікропроцесор типового модуля введення виконує наступні функції:

- реалізує протокол обміну з ПЛК;
- виконує команди, що посилаються ПЛК в модуль;
- реалізує виконання функцій автоматичного калібрування, діагностики обриву або к. З. в ланцюзі датчика;
- перетворює формати даних, що вводяться (інженерний формат - в одиницях вимірюваної величини, шістнадцятковий формат, відсотки від діапазону вимірювань);
- встановлює швидкість обміну з ПЛК (для ПЛК з розподіленими модулями вводу-виводу);
- виконує цифрову фільтрацію вхідного сигналу.

В постійній пам'яті ЕППЗУ модуля зберігаються калібрувальні коефіцієнти, адреса модуля, програма, таблиці лінеаризації нелінійних характеристик термопар і термоперетворювачів опору. Сторожовий таймер виконує перезавантаження (скидання) мікроконтролера в разі його зависання.

Живлення внутрішніх вузлів модуля виконується від внутрішнього стабілізатора напруги, який дозволяє подавати зовнішню напругу живлення в

широкому діапазоні, зазвичай від 10 до 30 В. Великий діапазон напруг живлення дуже корисний в розподілених системах, коли модулі введення можуть перебувати на значній відстані один від одного і тому падіння напруги на опорі проводів харчування досягає 10 ... 20 В.

Ланцюги живлення модулів захищаються від неправильної полярності напруги живлення і від перевищення напруги живлення над допустимим значенням. Захист виконується діодами, стабілітронами, позисторами і плавкими запобіжниками.

Для інтерфейсу RS-485 використовується захист від статичної електрики, від електромагнітних імпульсів, від короткого замикання і перегріву вихідного каскаду.

2. РОЗРОБКА АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Вимоги безпеки, ізоляції та кліматичних умов

В першу чергу, так як розроблюваний пристрій може застосовуватися на АЕС, він має відповідати певним вимогам безпеки:

- використання в якості елементу системи аварійного електрозабезпечення:
    1) клас безпеки 2 або 3 згідно НП 306.2.141 (НТД України);
    2) клас безпеки 2(А) або 3(В) згідно НП 306.2.202 (НТД України);
- використання в якості елементу мережі власних потреб нормальної експлуатації:
    1) клас безпеки 3 згідно НП 306.2.141 (НТД України);
    2) клас безпеки 3(В) згідно НП 306.2.202 (НТД України);
- використання в якості елементу інших систем нормальної експлуатації, що не впливають на безпеку:
    1) клас 4 згідно НП 306.2.141, НП 306.2.202 (НТД України).

По стійкості до зовнішніх впливових факторів пристрій має відповідати наступним вимогам:

- вид кліматичного виконання має відповідати ПХЛ 3.1 згідно ГОСТ 15150 ;
- пристрій має бути стійким до електромагнітних перешкод, має бути працездатним в експлуатації в умовах суворої електромагнітної обстановки.

За стійкістю електричної ізоляції модуль має відповідати вимогам ДСТУ EN 60255-5 (НТД України) з наступними параметрами:

- категорія перенапруги - III;

- ступінь забруднення середовища – 2 (частіше за все присутнє лише непровідне забруднення, але час від часу виникає тимчасова провідність, котру викликає конденсація);
- номінальна напруга ізоляції - 250 V;
- номінальна імпульсна напруга – 4000 V;
- мінімальний повітряний проміжок – 3 мм.

Кліматичні умови для коректної роботи мають відповідати наступним вимогам:

- температура зовнішнього середовища від 15 до 35 °C;
- відносна вологість від 45 до 75 %;
- атмосферний тиск від 80 до 106 kPa.

2.2 Технічні вимоги

Пристрій призначений для:

- прийому дискретних потенційних сигналів й видачі інформації про їх стан за запитом до центрального процесора (далі – ЦП);
- формування (комутація) керуючих впливів на виконавчі пристрої постійного і змінного струму і / або дискретних сигналів в зовнішні схеми управління / сигналізації відповідно до команд, які надходять від ЦП.

Електроживлення має здійснюватися від джерела постійного струму (5±0,25) V. Струм, споживаний ланцюгом 5 V не має перевищувати 1, 07 А. Час готовності модулю до роботи після ввімкнення електроживлення має бути не більше 5 с. Не має виникати помилкових дискретних вихідних сигналів (замикань контактів реле) при подачі живлення на модуль або при рестарті МК.

В таблиці 1 вказано вимоги до характеристик вхідних каналів модулю.

Таблиця 1 – Вимоги до характеристик вхідних каналів модулю.

Найменування та одиниця виміру	**Значення параметру**
Кількість каналів входу	8
Тип ДВ	Потенційний
Номінальна напруга ДВ: - постійного струму, V - змінного струму, V	 110 -
Поріг спрацювання ДВ, V	Від 80 до 86
Поріг відпускання ДВ, V	Від 66 до 77
Максимальна тривала напруга ДВ, V	154
Імпульс режекції струму ДВ	В наявності
Тип вихідного дискретного сигналу	Релейний, нормально розімкнутий «сухий контакт»
Кількість каналів формування	8
Напруга постійного струму, що комутується, V	Від 19,2 до 264
Максимальна тривала напруга постійного струму, V	264
Струм, що тривалий час протікає через замкнуті контакти, А (не менше ніж)	5
Електрична зносостійкість для навантажених контактів, циклів (не менше ніж)	10 000
Час спрацювання кожного вихідного каналу, ms (не більше ніж)	10

2.3 Архітектурний опис

На рисунку 1 зображено структурну схему модулю, на якій визначено основні вузли, що мають бути представлені в модулі:

- МК, що керує роботою модулю;
- канали вводу потенційних дискретних сигналів (U1 - U8) з вузлами формування струмів режекції;
- канали формування дискретних сигналів (U9 - U16) з елементами контролю та діагностики реле;
- вузол зв’язку з ЦП;
- вузол живлення;
- з’єднувачі для підключення зовнішніх ланцюгів (X1 – X4);

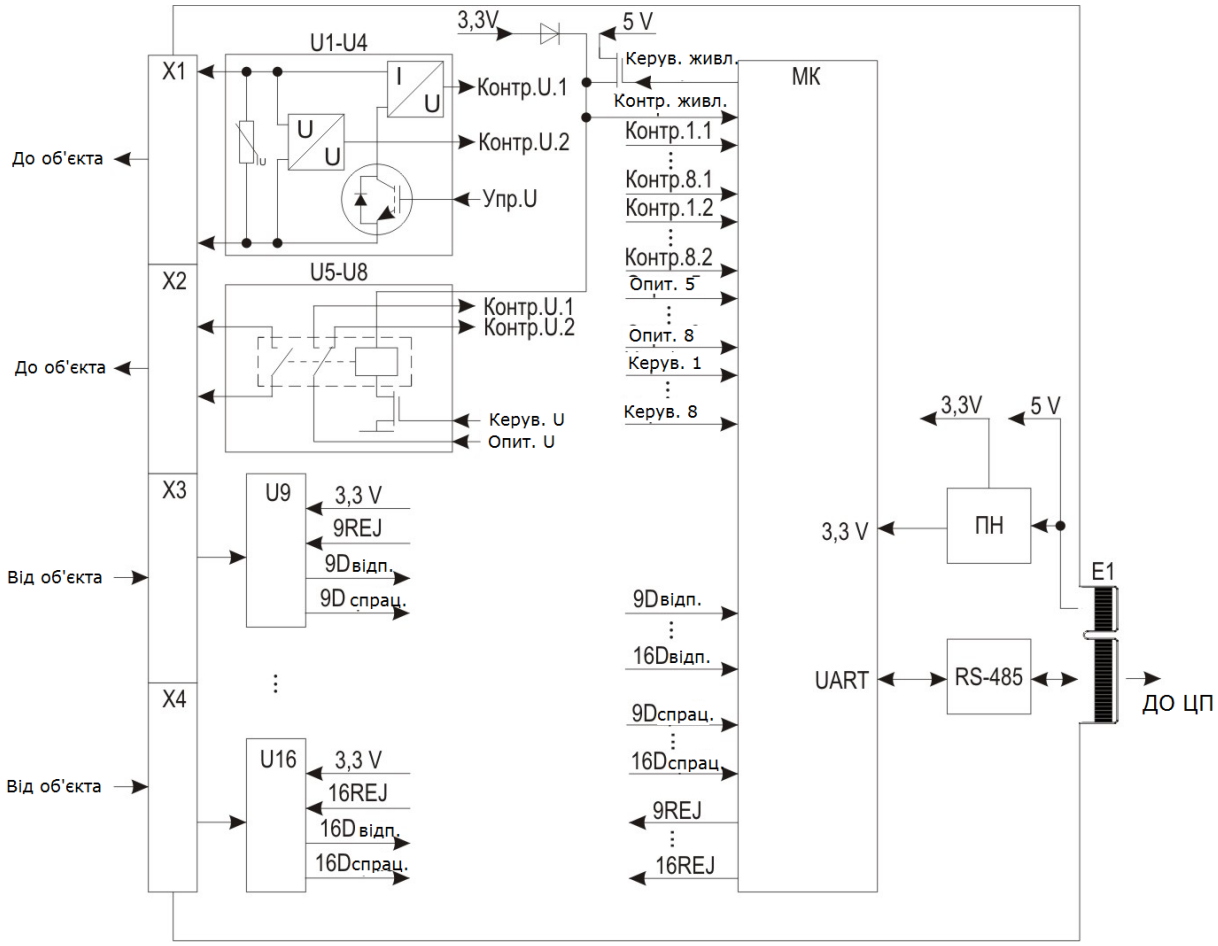


Рисунок 1. Структурна схема модулю

- ламель Е1 для підключення живлення та з’єднання з інтерфейсом RS-485.

Модуль має здійснювати діагностування:

- справності програмних засобів МК;
- зв’язку з ЦП (відсутність зв’язку, отримання даних про помилки та ін.);
- стану вихідних каналів формування дискретних сигналів.

Канали дискретного виводу U1 - U8 забезпечують формування керуючих впливів (комутацію) на виконуючі пристрої відповідно до команд, котрі поступають від ЦП.

Канали дискретного вводу U9 - U16 здійснюють ввід вхідних сигналів, а також формують імпульс режекції струму.

Також кожен канал дискретного вводу виконує такі функції:

- визначення вхідних потенційних сигналів зі значеннями напруги вище порогового рівня спрацювання;
- формування імпульсу режекції струму по тим каналам дискретного вводу, по яким з’явились потенційні сигнали зі значеннями вище порогового рівня відпускання на час, заданий в запиті від ЦП;
- контроль дієздатності каналів;
- формування затримки спрацювання каналу на час, заданий в запиті від ЦП (від 0 до 50 ms) з кроком 1 ms.

Кожен канал дискретного виводу типу RDO виконує наступні функції:

- формування вихідних дискретних сигналів;
- контроль стану контактів реле вихідних дискретних сигналів за станом суміжних пар контактів реле (з установкою (контакти реле замкнуті)/скиданням (контакти реле розімкнуті));
- контроль дієздатності каналів (з установкою параметру RL_ER*j* в разі відмови каналу);

- діагностування наявності напруги електроживлення 5 V, справності джерела напруги 3,3 V, справності загального ключа перемикання електроживлення обмоток реле шляхом виміру значень напруги 5 V/3,3 V.

2.4 Вимоги до ПЗ

ПЗ модулю має постійно виконувати наступні функції:

- вимірювання значення температури датчиком, вмонтованим у МК;
- формування і передачу в ЦП відповіді;
- самодіагностика програмних засобів (контроль ЗСД МК);
- програмну підтримку роботи внутрішньо контролерного Watchdog-таймера для боротьби з відмовами при «зависанні» виконання ПО (на час 100 ms);
- виявлення вхідних потенціальних сигналів зі значеннями напруги вище порогу відпускання (згідно таблиці 1);
- формування імпульсу режекції за вхідними каналами, по котрим з’явились потенційні сигнали зі значеннями вище порогу відпускання;
- виявлення вхідних потенціальних сигналів зі значеннями напруги вище порогу спрацювання (згідно таблиці 1);
- формування затримки спрацювання дискретного входу на час, заданий в запиті від ЦП (від 0 до 50 ms). У вказаному діапазоні затримка має регулюватися з кроком 1 ms;
- комутація вихідних релейних сигналів і контроль стану контактів реле вихідних каналів за станом суміжних пар контактів реле, діагностика справності загального ключа перемикання електроживлення обмоток реле шляхом вимірювання значень напруги 5 V/3 V.

На рисунку 2 зображено блок-схему мікроконтролеру. Для реалізації вищевказаних функцій ПЗ необхідно використовувати наступні апаратні засоби МК:

- порти вводу - виводу: PA, PB, PC, PD, PE, PG;
- асинхронний приймально-передавач USART1;
- модуль прямого доступу до пам’яті (далі - DMA);
- таймери TIM2–TIM4;
- незалежний сторожовий таймер IWDG;
- процесор переривань NVIC;
- аналого-цифрові перетворювачі ADC1 (далі – ADC1), ADC2 (далі – ADC2);
- внутрішній температурний датчик МК;
- системний годинник RCC.

Занесення інформації в ЗСД МК здійснюється за допомогою виводів PA13 (SWDIO) та PA14 (SWCLK), що реалізують апаратний інтерфейс налагодження і програмування SWD (Serial Wire Debug).

Для контролю значень напруги електроживлення обмоток реле 5 V/3 V використовується порт PA2 (канал 2 ADC1 МК).

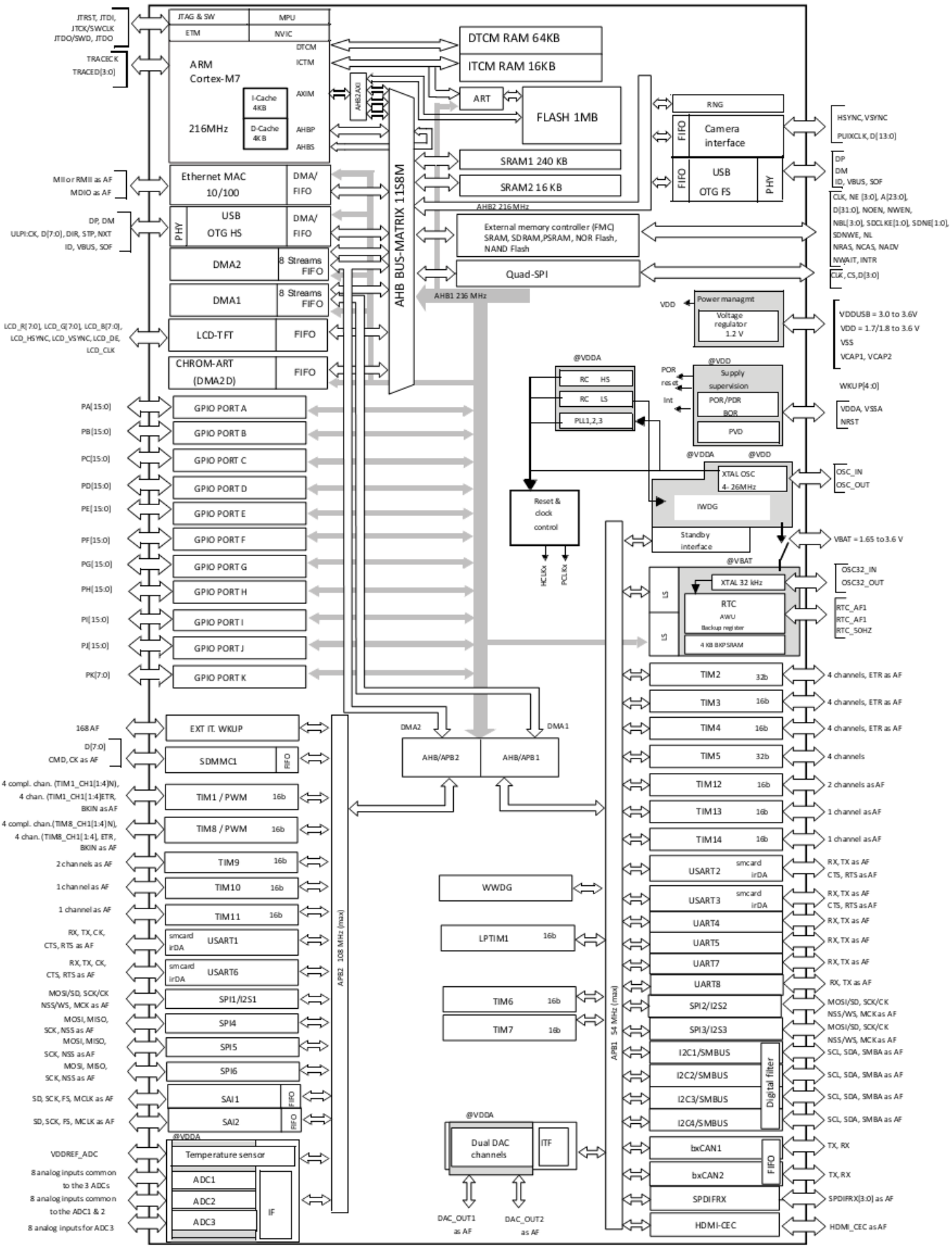


Рисунок 2. Блок-схема МК

Для вводу станів дискретних потенційних сигналів використовуються порти МК згідно таблиці 2.

Таблиця 2 – Порти вводу станів дискретних потенційних сигналів

Найменування вхідного сигналу	**Позначення сигналу**	**Порт МК**	**Стан сигналу на порту МК**	
			Логічний «0»	**Логічна «1»**
Стан каналу 9 відносно порогу відпускання	Doff9	PC8	Значення напруги на вході каналу вище порогу відпускання	Значення напруги на вході каналу нижче порогу відпускання
Стан каналу 10 відносно порогу відпускання	Doff10	PC9		
Стан каналу 11 відносно порогу відпускання	Doff11	PC10		
Стан каналу 12 відносно порогу відпускання	Doff12	PC11		
Стан каналу 13 відносно порогу відпускання	Doff13	PC12		
Стан каналу 14 відносно порогу відпускання	Doff14	PC13		
Стан каналу 15 відносно порогу відпускання	Doff15	PC14		
Стан каналу 16 відносно порогу відпускання	Doff16	PC15		
Стан каналу 9 відносно порогу спрацювання	Don9	PE8	Значення напруги на вході каналу вище порогу спрацювання	Значення напруги на вході каналу нижче порогу спрацювання
Стан каналу 10 відносно порогу спрацювання	Don10	PE9		

Закінчення таблиці 2

Стан каналу 11 відносно порогу спрацювання	Don11	PE10	Значення напруги на вході каналу вище порогу спрацювання	Значення напруги на вході каналу нижче порогу спрацювання
Стан каналу 12 відносно порогу спрацювання	Don12	PE11		
Стан каналу 13 відносно порогу спрацювання	Don13	PE12		
Стан каналу 14 відносно порогу спрацювання	Don14	PE13		
Стан каналу 15 відносно порогу спрацювання	Don15	PE15		
Стан каналу 16 відносно порогу спрацювання	Don16	PE14		

Для керування режекцією дискретних вхідних каналів використовуються порти згідно таблиці 3.

Таблиця 3 – Порти керування режекцією вхідних каналів

Найменування сигналу	**Позначення сигналу**	**Порт МК**	**Стан порту МК**	
			Логічний «0»	**Логічна «1»**
Імпульс режекції каналу 9	REJ9	PD8	Режекцію ввімкнено	Режекцію вимкнено
Імпульс режекції каналу 10	REJ10	PD9		
Імпульс режекції каналу 11	REJ11	PD10		
Імпульс режекції каналу 12	REJ12	PD11		
Імпульс режекції каналу 13	REJ13	PD12		

Закінчення таблиці 3

Імпульс режекції каналу 14	REJ14	PD13	Режекцію ввімкнено	Режекцію вимкнено
Імпульс режекції каналу 15	REJ15	PD15		
Імпульс режекції каналу 16	REJ16	PD14		

Для подачі керуючого впливу на обмотки реле і для подачі керуючого впливу на затвор транзисторного ключа використовуються порти МК згідно таблиці 4.

Таблиця 4 – Порти керування дискретними вихідними каналами

Найменування сигналу	**Позначення сигналу**	**Порт МК**	**Стан порту МК**	
			Логічний «0»	**Логічна «1»**
Керування каналом 1	EnCh1	PD0	Вимкнути реле (розімкнути канал)	Ввімкнути реле (замкнути канал)
Керування каналом 2	EnCh2	PD1		
Керування каналом 3	EnCh3	PD2		
Керування каналом 4	EnCh4	PD3		
Керування каналом 5	EnCh5	PD4		
Керування каналом 6	EnCh6	PD5		
Керування каналом 7	EnCh7	PD6		
Керування каналом 8	EnCh8	PD7		

Налаштування асинхронних приймально-передавачів: швидкість прийому/передачі має бути 16 384 000 bit/s, формат прийнятих/переданих байтів – один стартовий біт, вісім інформаційних бітів (передаються молодшими бітами вперед), один стоповий біт.

Таймер TIM2 (дискретність відліку 1 ms) використовувати для:

- відліку інтервалів контролю стану реле;
- відліку інтервалів виміру значень температури в пристрої;

- відліку тайм-ауту 5 s по включенню живлення пристрою.

Таймер TIM3 (дискретність відліку 100 μs) використовувати для:

- відліку часу між запитами від ЦП в основному циклі ПЗ (тайм-аут 5 ms);
- відліку часу режекції вхідних сигналів;
- відліку часу програмної затримки спрацювання дискретного входу;
- підрахунку часу відновлення каналу після імпульсу режекції.

Для налаштування переривань виконати ініціалізацію процесора переривань NVIC: для виключення вкладення переривань пріоритети переривань призначити рівними нулю.

Період роботи сторожового таймеру IWDG завдати рівним 100 ms.

Для підключення і тактування використовуваної вбудованої апаратури необхідно налаштувати і підключити системний годинник RCC:

- використовувати зовнішній генератор з частотою 25 MHz;
- активізувати PLL генератор для отримання системної частоти МК 214 MHz;
- виконати підключення всієї використовуваної периферії до модулю системного годинника.

Для контролю наявності напруги 5 V/3 V, виміру температури в пристрої, контролю струму у вхідних транзисторних каналах використовувати канал ADC1.

Для захисту від КЗ транзисторних ключів встановити верхній поріг спрацювання віконного компаратора ADC2 рівним 35 А для всіх чотирьох каналів з транзисторними ключами.

3. КОНСТРУЮВАННЯ ПРИСТРОЮ

3.1 Вимоги друкованої плати

Друкована плата має мати ламель Е1 (крайовий з'єднувач) з 64-ма контактами для з'єднування з джерелом електроживлення та для забезпечення зв'язку пристрою з ЦП.

Ланцюги GND пристрою мають складатися з двох електрично не зв'язаних частин:

- ланцюг GND, електрично зв'язаний з лицьовою панеллю пристрою, підключається тільки до елементів перешкодозахисту, котрі знаходяться біля лицьової панелі;
- ланцюг GND, електрично зв'язаний з іншими елементами схеми пристрою (див. додаток А), підключається через друковані контакти ламелі Е1.

Розділення ланцюга GND дозволяє забезпечити протікання перешкоди через конструктивні елементи та мінімізувати її вплив на пристрій.

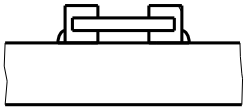
Під час розводки диференціальних пар (далі – дифпар) потрібно дотримуватись наступних вимог:

- для сигналів дифпар довжина провідника не більше 150 mm;
- друковані провідники дифпар мають бути симетричними;
- довжина друкованих провідників всередині дифпари має відрізнятися не більше, ніж на 0,127 mm;
- уникати гострих вигинів провідника;
- мінімізувати використання перехідних отворів для дифпар, так як вони приводять до втрати сигналу. Розмір отвору не більше 1, 78 mm, розмір контактної площадки отвори не більше 3, 175 mm;

- уникати трасування дифпар над контактними площадками інших ланцюгів в інших шарах.

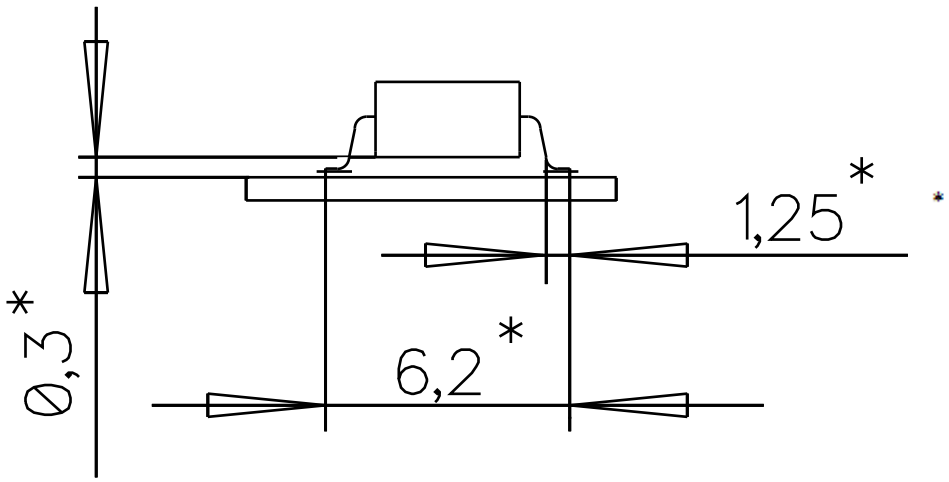
3.2 Розводка плати та монтування елементів

Електронні компоненти пристрою, котрі наведені в додатку Б, монтуються на друковану плату згідно рисункам 3 – 14.



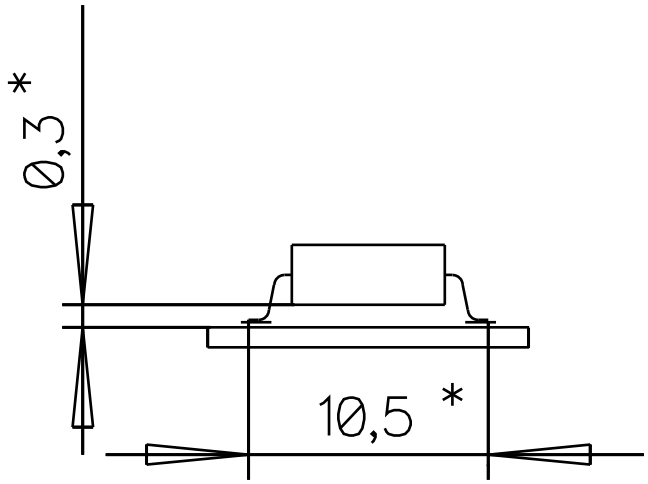
Встановлення C1-C31, F1, L2, R1-R9, 1C1-16C1, 1C2-16C2, 9C3-16C3, 1R1-8R1, 1R2-16R2, 1R3-8R3, 1R4-16R4, 9R5-16R5, 9R6-16R6, 9R7-16R7, 9R8-16R8, 9R9-16R9, 9R11-16R11, 9R12-16R12, 9R13-16R13, 9R14-16R14, 9R15-16R15, 9R16-16R16, 9R17-16R17, 9R18-16R18

Рисунок 3. Встановлення конденсаторів та резисторів



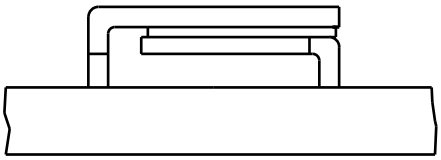
Встановлення D2, D4-D6

Рисунок 4. Встановлення мікросхем



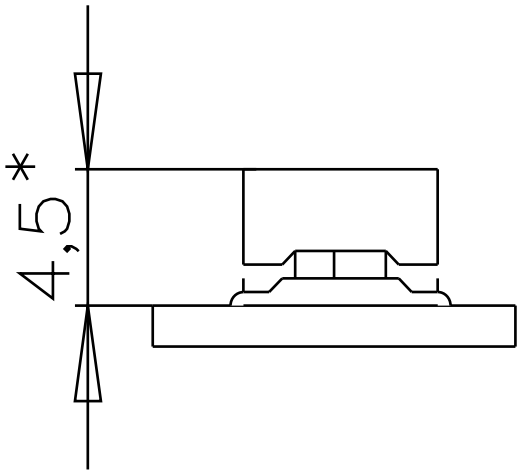
Встановлення 9V4-16V4, 9V5-16V5

Рисунок 5. Встановлення оптопари



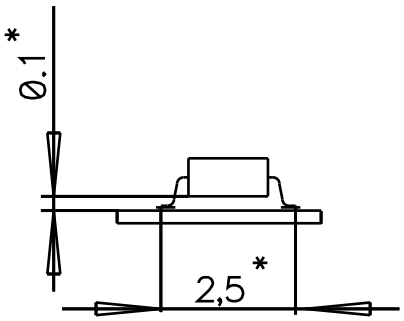
Встановлення F2

Рисунок 6. Встановлення запобіжника



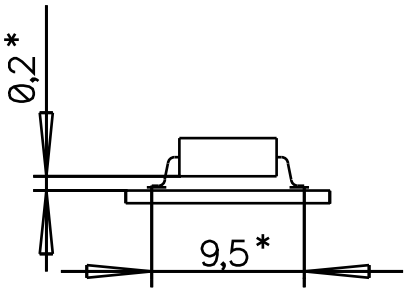
Встановлення L1

Рисунок 7. Встановлення дроселю



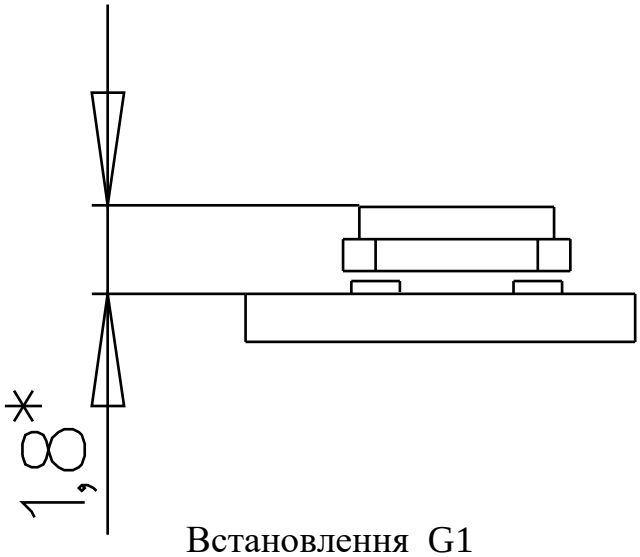
Встановлення V2, 9D1-16D1,

9D2-16D2,1V1 - 8V1, 9V3-16V3



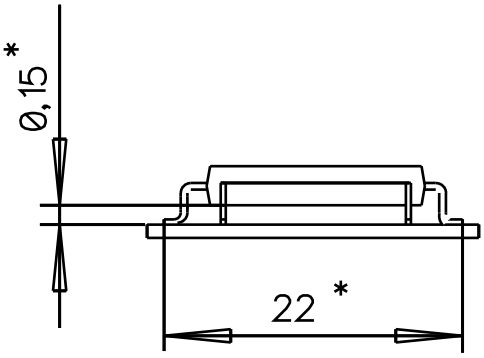
Встановлення 9V2-16V2

Рисунок 8. Встановлення транзисторів

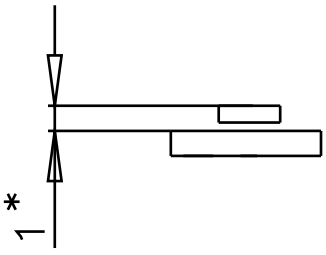


Встановлення G1

Рисунок 9. Встановлення генератора

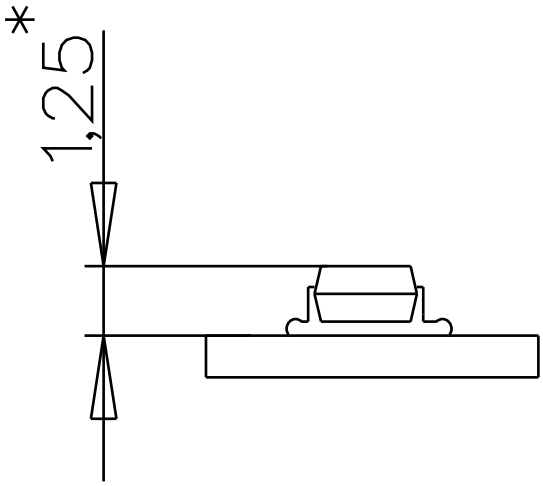


Встановлення D1



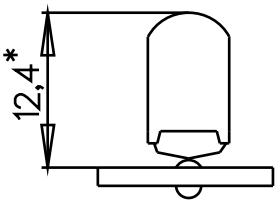
Встановлення D3

Рисунок 10. Встановлення мікросхем

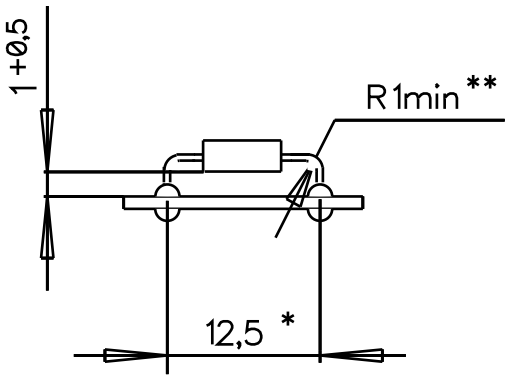


Встановлення V1, 1V2-8V2

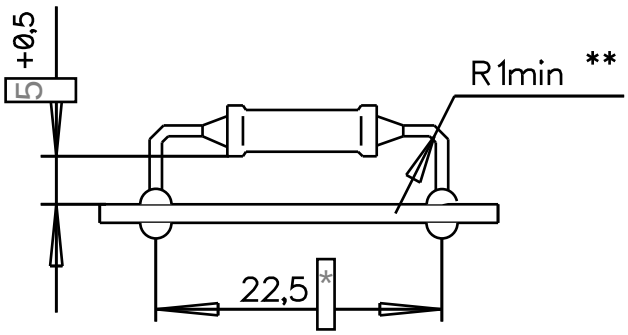
Рисунок 11. Встановлення діоду та транзистору



Встановлення 9R1-16R1

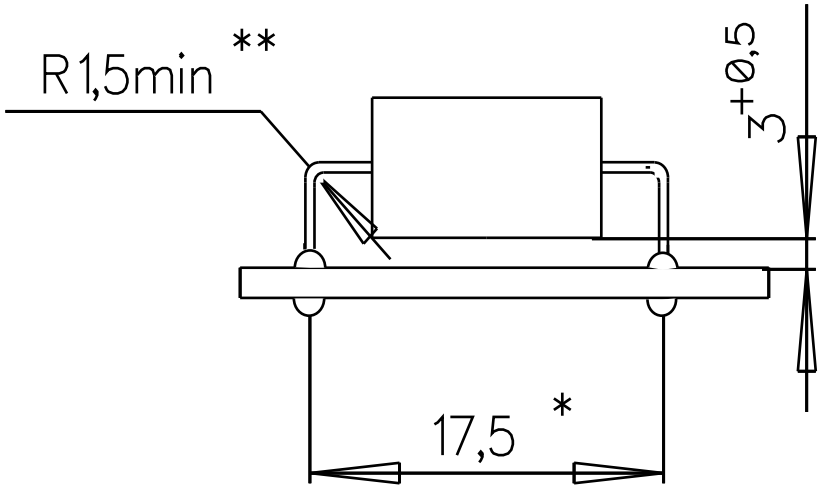


Встановлення 9R10-16R10



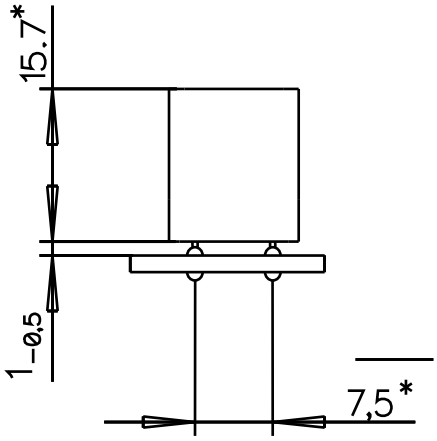
Встановлення 9R3-16R3

Рисунок 12. Встановлення резисторів

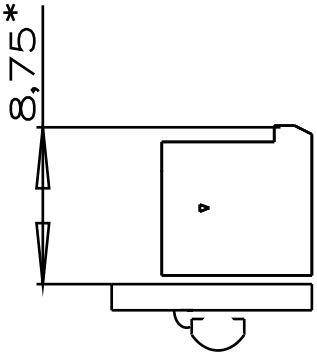


Встановлення 9V1-16V1

Рисунок 13. Встановлення діодів



Встановлення 1K1-8K1



Встановлення X1-X4

Рисунок 14. Встановлення реле та вилок

Розводка друкованої плати, з позначеннями наявних на ній вузлів наведена на рисунку 15.

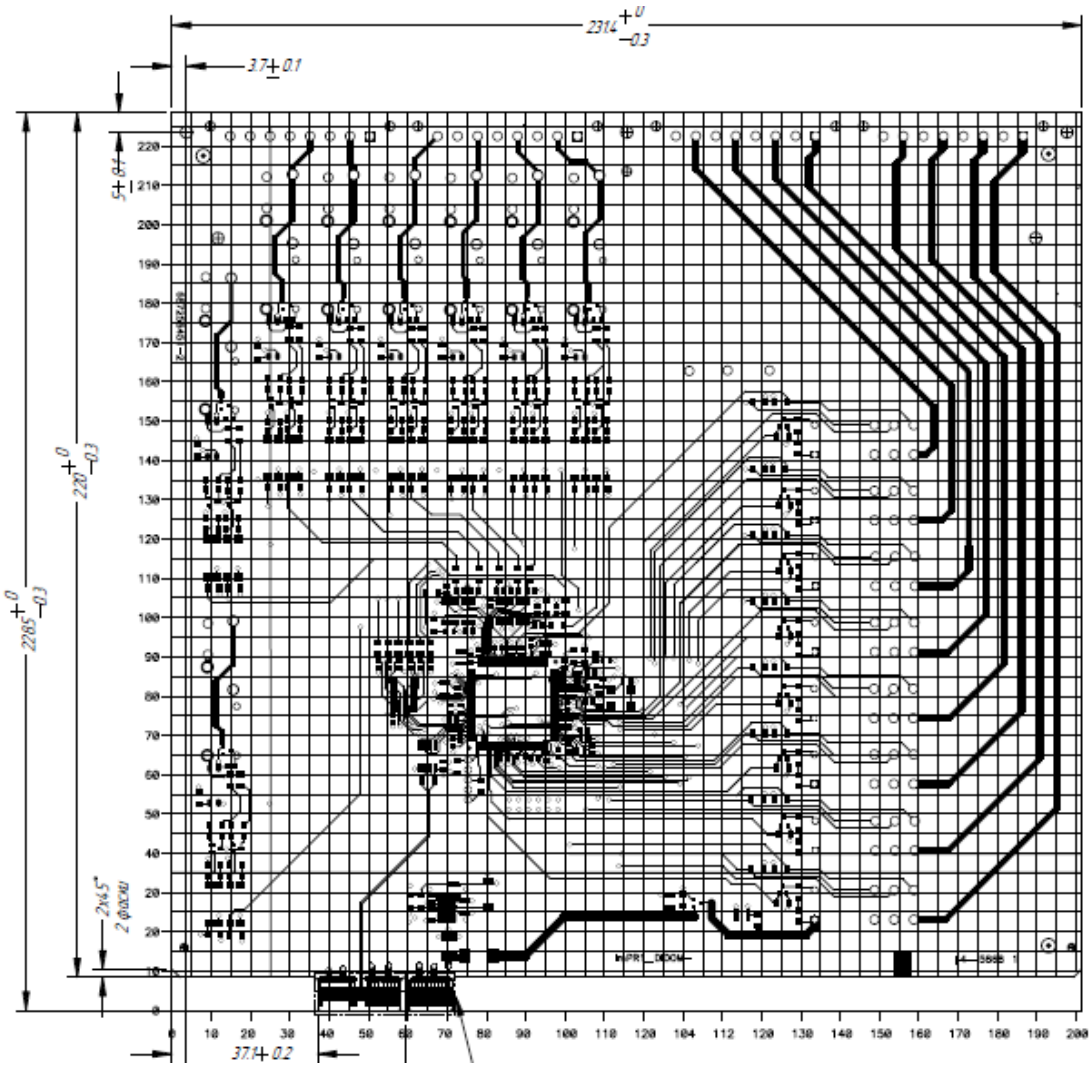


Рисунок 15. Розводка друкованої плати

Розміри отворів, їх кількість та характеристики наведено в таблиці 5.

Таблиця 5 – Характеристики отворів друкованої плати

Розмір (мм)	Позначення	Кількість	Металізація
0,300		3	Є
0,600		310	Є
1,000		32	Є
1,200		5	Є

Продовження таблиці 5

Розмір (мм)	Позначення	Кількість	Металізація
1,400	⊕	115	Є
1,600	⊖	16	Є
2,000	⊠	2	Немає
2,400	⊻	9	Немає
3,000	⊠	5	Немає

Схема друкованої плати з встановленими електронними елементами наведена на рисунку 16.

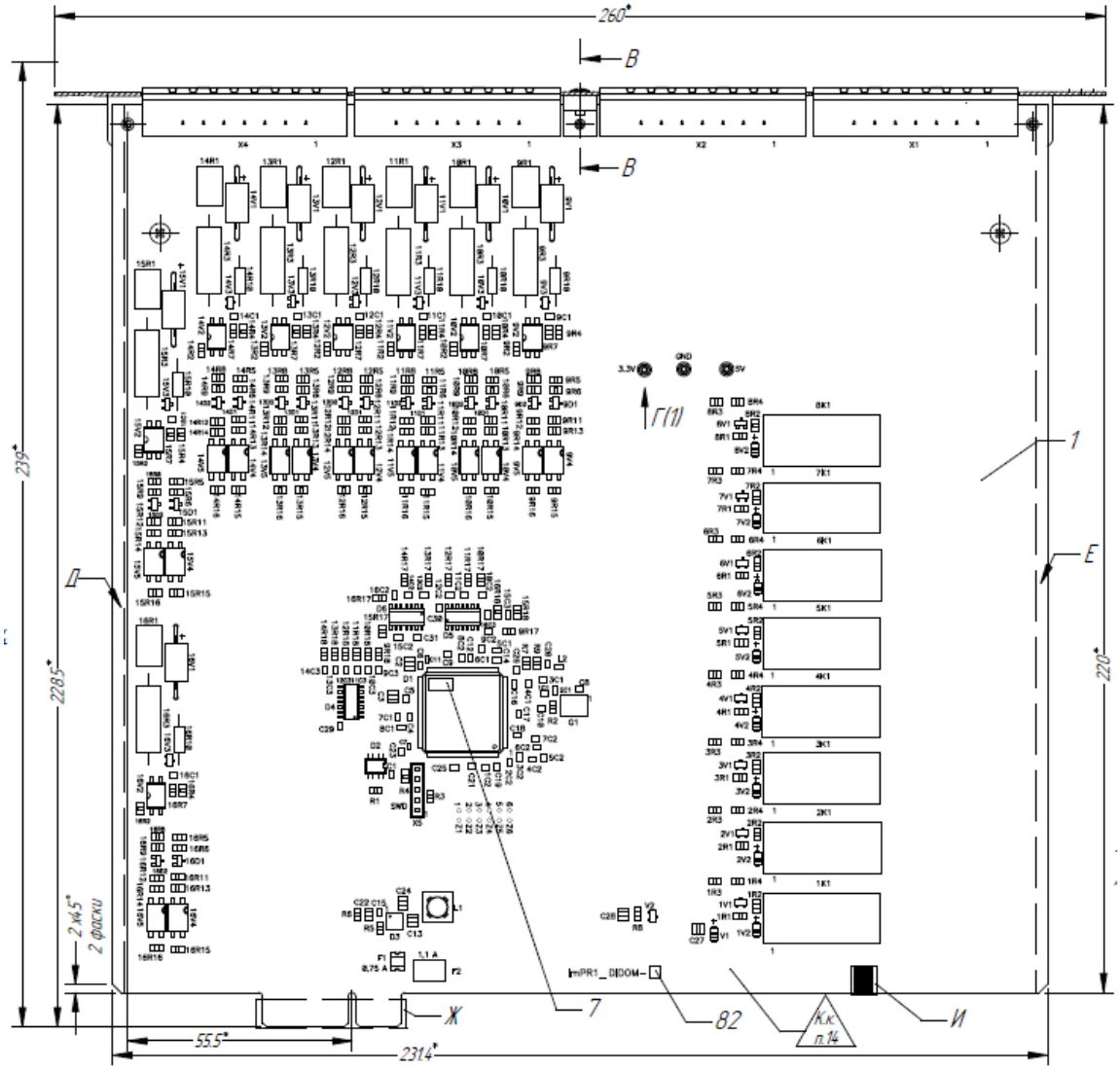


Рисунок 16. Схема друкованої плати

4. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Структура програми

Програма призначена для МК STM32F745ZGT6 STMicroelectronics, системна частота котрого 163,815789 MHz (отримана шляхом використання зовнішнього генератора з частотою 25 MHz та PLL - множника).

Програма записується у внутрішньому ЗСД МК і запускається на виконання після подачі живлення з адреси 0x08000000. Організація вектору переходу на виконання головної функції програми main відбувається автоматично за допомогою програм Linker/Locator и Startup.

Згідно рисунку 17 кроки 1 – 7 МК виконуються один раз (секція ініціалізації), кроки 8-22 виконуються в циклі.

Вхідними даними є байти запиту від ЦП, дані виміру ADC1, ADC2, значення портів вводу, котрі повідомляють МК про стан апаратної частини пристрою.

Вихідними даними є дані відповідного повідомлення від ЦП (помилки, результат контролю обладнання, стан дискретних каналів), значення портів виводу, керуючі апаратною частиною пристрою.

4.2 Опис підпрограм

Підпрограма SystemClock_Config виконує ініціалізацію генератора системної частоти шляхом використання зовнішнього генератора з частотою 25 MHz (RCC_PLLSOURCE_HSE) и PLL-множника (PLLM_DIV_19, PLLN_MUL_249, PLLP_DIV_2, AHBPrescaler_DIV_1, APB1Prescaler_DIV_4, APB2Prescaler_DIV_16). Системна частота робота МК SYSCLK=163,815789 MHz, частота на AHB1 HCLK=163,815789 MHz, частота роботи пристрою, підключеного до APB1, рівна 40,953947 MHz (для таймерів TIM2, TIM3, TIM4 –

81,907895 MHz), частота роботи пристрою, підключеного до APB2, рівна 10,238487 MHz (для USART1 – 163,815789 MHz).

Підпрограма MX_GPIO_Init виконує дозвіл роботи підключених до APB1 портів PA, PH, PD, PG и ініціалізацію портів PG10–PG15 (PG10, PG11, PG12, PG13, шостий старший розряд (порт PG15) є їх контрольним бітом парності).

Підпрограма Init_GPIO_MODULE в залежності від значення MODULE викликає підпрограму InitGPIO_CALL, яка викликає ініціалізацію портів, які використовує пристрій.

Підпрограма MX_DMA_Init виконує дозвіл роботи підключеного до AHB1 DMA2 та ініціалізацію каналу Channel 4 DMA2 потоку Stream 2 для прийому запиту от МЦП и Stream 7 для видачі йому відповіді.

Підпрограма MX_USART1_UART_Init виконує дозвіл роботи підключеного до APB2 USART1, ініціалізацію портів PA9 USART1_TX – вихід передавача, PA10 USART1_RX – вхід приймача, PA12 USART1_DE – керування режимом прийом/передача по USART1, ініціалізацію и налаштування USART1 для зв’язку с ЦП. Механізм передачі байту – асинхронний, старт-стоповий. Швидкість обміну – 16 384 000 bit/s, формат прийнятих/переданих байтів – один стартовий біт, вісім інформаційних бітів (передаються молодшими бітами вперед), один стоповий біт. Дозвіл переривання від USART1.

Підпрограма USART1_Begin_Init виконує ініціалізацію автоматичного перемикання режиму прийому/передачі (в регістрі CR3 встановлюється біт DEM=1 – Driver enable mode) та дозволяє роботу USART1 (в регістрі CR1 UE=1).

Підпрограма MX_ADC1_Init виконує дозвіл роботи підключеного до APB2 ADC1, ініціалізацію портів PA2 и PA3 и ADC1 (канал 2 и канал 3 ADC1 відповідно), ініціалізацію температурного датчика (канал 18 ADC1).

Підпрограма MX_ADC2_Init виконує підключення та ініціалізацію каналів ADC2: канал 4 ADC2 МК – порт PA4, канал 5 ADC2 МК – порт PA5, канал 6 ADC2 МК– порт PA6, канал 7 ADC2 МК – порт PA7.

Підпрограма CRC_Config виконує дозвіл роботи підключеного до AHB1 апаратного модуля підрахунку контрольної суми CRC32 з інверсією вхідних даних (32 розряду) та результату (в регістрі керування CRC_CR встановити Bits 6:5 REV_IN[1:0]=11 и Bit 7 REV_OUT=1).

Підпрограми MX_TIM2_Init, MX_TIM3_Init, MX_TIM4_Init виконують дозвіл роботи підключених до APB2 таймерів, ініціалізацію і налаштування таймерів TIM2, TIM3, TIM4 відповідно на переривання по відліку 1 ms, 100 μs, 5 ms.

Підпрограма MX_IWDG_Init виконує підключення і ініціалізацію «сторожового» таймеру IWDG на період роботи 100 ms.

Підпрограми TIM2_Start, TIM3_Start, TIM4_Start виконують дозвіл переривання і роботи таймерів TIM2, TIM3, TIM4 відповідно.

Підпрограма ADC2_start_cont_mode виконує запуск неперервних вимірів струму в транзисторних каналах.

Підпрограми викликаються в секції ініціалізації головної функції main.

Вхідні та вихідні дані відсутні.

4.2.1 Підпрограма самодіагнування test_full_ROM

Підпрограма test_full_ROM виконує само діагностування програмних засобів (контроль цілісності програмного коду ЗСД) в повному обсязі (початкова адреса 0x08000000, кінцева адреса 0x080FFFFF), використовуючи модуль CRC (апаратний метод).

Алгоритм підпрограми test_full_ROM складає наступну послідовність кроків:

1) циклічний запис змісту чотирьох байтів ЗСД в 32-розрядний регістр даних CRC_DR модулю апаратного розрахунку CRC по 16 байт за цикл (початкова адреса FLASHAXI_BASE=0x08000000, кінцева адреса FLASH_END=0x080FFFFF);

2) якщо адреса перевірки досягла кінцевого значення (0x080FFFFF), то виконується перевірка отриманого результату (регістру даних CRC_DR). якщо CRC_DR=0 (розрахована та прочитана з ЗСД контрольні суми CRC32 зрівнялись), то ErrROM=0 (немає помилки ПЗУ МК), інакше – ErrROM=1 (помилка ЗСД МК).

Підпрограма test_full_ROM викликається в секції ініціалізації головної функції main.

Вхідними даними є FLASHAXI_BASE, FLASH_END.

Вихідними даними є ErrROM.

4.2.2 Підпрограма test_part_ROM

Підпрограма test_part_ROM призначена для циклічного розрахунку (словами по чотири байти) чотирьох байтової контрольної суми фрагменту ЗСД із 16 byte, використовуючи модуль CRC (апаратний метод).

Привласнення вихідному стану in1 вказівника початкової адреси FLASHAXI_BASE (0x08000000), кожному із наступних вказівників in2, in3, in4 – на одиницю більше попереднього, вихідному стану CRC32 tmpCRC=0xFFFFFFFF виконується при визначенні цих змінних.

Алгоритм підпрограми скаладає наступну послідовність кроків:

1) якщо адреса перевірки за вказівником in4 не досягла кінцевого значення FLASH_END (0x080FFFFF), виконується привласнення регістру вихідного стану CRC_INIT=tmpCRC та запис вмісту 16 byte ЗСД послідовно по чотири байти в 32-розрядний регістр даних CRC_DR модулю апаратного розрахунку CRC (початкова адреса за вказівником in1, кінцева адреса за вказівником in4). Збільшення кожного із вказівників in1, in2, in3, in4 на чотири. Збереження результату підрахунку CRC32 фрагменту в глобальній змінній tmpCRC, вихід із підпрограми;

2) якщо адреса перевірки за вказівником in4 досягла кінцевого значення (0x080FFFFF), то виконується перевірка отриманого результату tmpCRC. Якщо tmpCRC=0 (розрахована та прочитана з ЗСД контрольні суми CRC32 зрівнялись), то ErrROM=0 (немає помилки ПЗУ МК), інакше – ErrROM=1 (помилка ПЗУ МК). Привласнення стану in1 вказівника початкової адреси FLASHAXI_BASE (0x08000000), кожному із наступних вказівників in2, in3, in4 – на одиницю більше поперднього, tmpCRC=0xFFFFFFFF (вихідний стан), вихід із підпрограми.

Підпрограма викликається в підпрограмі TIM3_IRQHandler (кожні 100 μs).

Вхідними даними є in1, in2, in3, in4, tmpCRC, FLASHAXI_BASE, FLASH_END.

Вихідними даними є in1, in2, in3, in4, tmpCRC, ErrROM.

4.2.3 Підпрограма дозволю прийому запиту RxDMA

Підпрограма RxDMA виконує налаштування USART1 на прийом за каналом DMA та дозвіл прийому запиту від ЦП (перемикання лінії на прийом/передачу виконується автоматично).

Алгоритм підпрограми складає наступну послідовність кроків:

1) вимикання і заборона переривання USART1, заборона роботи каналу DMA2 Stream2 на прийом;

2) ініціалізація USART1 на прийом 400 byte до буфер RxBuf;

3) завдання до Receiver timeout register (USART_RTOR) тайм-ауту тиші (1,875 μs час прийому трьох байтів (30 bit) при заданій швидкості обміну);

4) дозвіл переривання по тайм-ауту приймача USART1, ввімкнення USART1 та дозвіл роботи каналу DMA2 Stream2.

Підпрограма викликається в головній функції main після ввімкнення живлення та в циклі за встановленим прапором RX_START_FLAG=1 (після завершення передачі відповіді ЦП або після прийому запиту з помилковою довжиною ASK_ER=1 або помилковою контрольною сумою CRC32_ER=1).

Вхідні дані відсутні.

Вихідні дані відсутні.

4.2.4 Підпрограма запису прикладних даних до буферу Write_Data_Buf

Підпрограма Write_Data_Buf виконує занесення прикладних даних з буферу запиту RxBuf (с 8 по 39 byte) до буферу даних DataBuf для подальшої обробки значень команд керування (релейними та транзисторними) вихідними дискретними каналами EnCh_tek, часу режекції вхідних дискретних каналів Tr, часу затримки вхідних дискретних каналів Td, команд діагностики транзисторних ключів Diag_Ch.

Підпрограма викликається в USART1_IRQHandler після отримання безпомилкового запиту від ЦП з новими прикладними даними (NEW_DATA_FLAG=1).

Вхідними даними є RxBuf.

Вихідними даними є DataBuf.

4.2.5 Підпрограма опиту вхідних дискретних каналів модулю Control_Diskret

Підпрограма Control_Diskret виконує опит дискретних каналів пристрою.

Алгоритм підпрограми складає наступну послідовність кроків:

1) визначення загальної кількості каналів total_number_ch=16 (починаючи з восьмого);

2) виконується опит стану оптрону порогу відпускання кожного каналу (змінна Doff_tek[number_ch]). Якщо режекція каналу пройшла (FLAG_REG=0) і збіг тайм-аут після режекції (FLAG_TIME_OUT_REG=0), то зчитується поточний стан порогу відпускання каналу number_ch (port input data register GPIOC_IDR) в змінну Doff_tek[number_ch].;

3) якщо поточне значення стану порту каналу вводу (порога відпускання) Doff_tek[number_ch] змінилось в порівнянні з попереднім Doff_pred[number_ch], то виконується перехід до кроку 4, інакше – перехід до кроку 7;

4) якщо Doff_tek[number_ch]=0 (рівень сигналу на вході каналу вище порогу відпускання), то у відповіді ЦП Doff[number_ch]=0x01 (напруга на вході каналу вище порогу відпускання). Якщо в запиті заданий час режекції, то ініціалізується таймер часу режекції Trej[number_ch] і тайм-аут після режекції Timeout_diskret[number_ch] на час режекції плюс 2 ms (дискретність рахунку 100 μs) і подається імпульс режекції, встановлюється прапор режекції FLAG_REG[number_ch]=1 і прапор тайм-ауту режекції FLAG_TIME_OUT_REG[number_ch]=1, попередньому присвоюється поточне значення Doff_pred[number_ch]=Doff_tek[number_ch], перехід до кроку 23;

5) якщо Doff_tek[number_ch]=1 (рівень сигналу на вході каналу нижче порогу відпускання), то аналізується останнє зафіксоване значення каналу спрацювання SD[number_ch]. Якщо SD[number_ch]=1 (ввімкнений) і в запиті заданий час програмної затримки перемикання Td[number_ch], то ініціалізується таймер часу затримки каналу Time_delay_diskret[number_ch] (ддискретність рахунку 100 μs). Якщо SD[number_ch]=0 (вимкнено), то в відповіді ЦП

Doff[number_ch]=0x00. Попередньому присвоюється значення Doff_pred[number_ch]=Doff_tek[number_ch], виконується перехід до кроку 23;

6) якщо в запиті не заданий час програмної затримки перемикання Td[number_ch], то в відповіді ЦП встановлюється канал вимкнути (SD[number_ch]=0) напругу на вході нижче порогу відпускання (Doff[number_ch]=0x00), перехід до кроку 23;

7) поточне значення стану порту каналу вводу (порога відпускання) Doff_tek[number_ch] не змінилось в порівнянні з попереднім Doff_pred[number_ch], перехід до кроку 13;

8) якщо Doff_tek[number_ch]=0 (рівень сигналу на вході каналу вище порога відпускання - ввімкнений), то виконується перехід до кроку 9інакше (вимкнений) – перехід до кроку 12;

9) якщо Don_tek[number_ch]=0 (рівень сигналу на вході каналу вище порога спрацювання) і він змінився по відношенню до попереднього значення (Don_pred[number_ch]), то виконується перехід до кроку 10, інакше – перехід до кроку 11. Если Don_tek[number_ch]=1, то виконується перехід до кроку 23;

10) якщо в запиті задано значення часу затримки перемикання Td[number_ch], то ініціалізується таймер Time_delay_diskret[number_ch] (дискретність рахунку 100 μs), інакше – у відповіді ЦП встановлюється ознака ввімкнення каналу (SD[number_ch]=1), попередньому присвоюється поточне значення Don_pred[number_ch]=Don_tek[number_ch], перехід до кроку 23;

11) значення Don_tek[number_ch] не змінилось по відношенню до попереднього. Якщо час програмної затримки перемикання каналу збіг (Time_delay_diskret[number_ch]=0), то в відповіді до ЦП встановлюється SD[number_ch]=1 (канал ввімкнути), попередньому присвоюється поточне значення Don_pred[number_ch]= Don_tek[number_ch], перехід до кроку 23;

12) Doff_tek[number_ch]=1 (рівень сигналу на вході каналу нижче порогу відпускання – вимкнений. Якщо час програмної затримки перемикання каналу збіг (Time_delay_diskret[number_ch]=0), то в відповіді МЦ встановлюється SD[number_ch]=0 (канал вимкнений), Doff[number_ch]=0x00, попередньому присвоюється поточне значення Doff_pred[number_ch]=Doff_tek[number_ch], перехід до кроку 23;

14) якщо Doff_tek[number_ch]=0 (рівнь сигналу на входу каналу вище порогу відпускання – ввімкнений), то виконується перехід до кроку 15, інакше (вимкнений) – перехід до кроку 12;

15) якщо час режекції каналу Trej[number_ch] збіг, то виконується перехід до кроку 16, інакше – перехід до кроку 23;

16) вимкнення (зняття) імпульсу режекції (відповідний каналу порт PD встановлюється рівним 1), зкидання прапору режекції (FLAG_REG[number_ch]=0 – немає режекції);

17) якщо час режекції плюс 2 ms збіг (Timeout_diskret[number_ch]=0), то виконується зкидання прапору тайм-ауту режекції FLAG_TIME_OUT_REG[number_ch]=0;

18)виконується опит стану оптрону порогу спрацювання кожного каналу (змінна Don_tek[number_ch]).;

19)якщо рівень сигналу на вході каналу вище порогу спрацювання (Don_tek[number_ch]=0 – канал ввімкнений), то виконується перехід до кроку 20, інакше – перехід до кроку 23;

20) якщо поточне значення стану порту каналу вводу (поріг спрацювання) Don_tek[number_ch] змінилось в порівнянні з попереднім Don_pred[number_ch], то виконуєтся перехід до кроку 21, інакше – перехід до кроку 22;

21) якщо в запиті задано значення часу затримки перемикання каналу Td[number_ch], то ініціалізується таймер Time_delay_diskret[number_ch] (дискретність рахунку 100 μs), інакше – в відповіді ЦП встановлюється канал «ввімкнений» (SD[number_ch]=1), попередньому присвоюється поточне значення Don_pred[number_ch]=Don_tek[number_ch], перехід до кроку 23;

22) значення Don_tek[number_ch] не змінилось по відношенню до попереднього. Якщо час програмної затримки перемикання каналу збіг (Time_delay_diskret[number_ch]=0), то в відповіді ЦП встановлюється SD[number_ch]=1 (канал ввімкнути), оновлюється Don_pred[number_ch], перехід до кроку 23;

23) запис результатів контролю стану дискретних входів відносно порогу відпускання Doff[number_ch] і стану дискретних входів SD[number_ch] до буферу видачі в ЦП OutBuf (відповідний каналу біт Doff и SD).

Підпрограма викликаєтеся в підпрограмі Main.

Вхідними даними є MODULE, Doff_pred[number_ch], Don_pred[number_ch], Doff_tek[number_ch], Don_tek[number_ch], FLAG_REG[number_ch], FLAG_TIME_OUT_REG[number_ch], Time_delay_diskret[number_ch].

Вихідними даними є Doff_pred[number_ch], Don_pred[number_ch], FLAG_REG[number_ch], OutBuf, FLAG_TIME_OUT_REG[number_ch], Time_delay_diskret[number_ch].

4.2.6 Підпрограма виміру напруги ADC_voltage_measurement

Підпрограма ADC_voltage_measurement виконує виміри значень ннапруги 5 V/3 V по заздалегідь обраному каналу 2 АЦП (ADC1_SQR2) або каналу 3 АЦП (ADC1_SQR3).

Алгоритм підпрограм складає наступну послідовність кроків:

1) дозвіл роботи і старт перетворення ADC1 (HAL_ADC_Start);

2) після завершення перетворення визначається напруга електроживлення пристрою або електроживлення обмотки реле voltage_result за формулою в mV;

3) зупинка перетворення ADC1 (HAL_ADC_Stop).

Підпрограма викликається в головній підпрограмі main.

Вхідними даними є номер каналу АЦП.

Вихідними даними є uint_16 voltage_result.

4.2.7 Підпрограма опрацювання буферу запиту від ЦП ObrZapros

Підпрограма ObrBufZapr виконує контролб прийнятого запиту (буфер RxBuf) на наявність помилок.

Алгоритм підпрограми складає наступну послідовність кроків:

1) перевірка байтів запиту (RxBuf) на відповідність:

– довжині запиту. Якщо довжина запиту (в байтах) кратна чотирьом та знаходиться в діапазоні від 12 до 400, то в відповіді (ERRS_ASK) ASK_ER=0, інакше – ASK_ER=1 і встановлюється прапор пеермикання USART1 на прийом RX_START_FLAG=1 (відповіді на запит немає);

– контрольній сумі запиту (підпрограма calc_CRC32). Якщо CRC32 рівна нулю, то в відповіді (ERRS_ASK) CRC32_ER=0, інакше – CRC32_ER=1 і встановлюється прапор перемикання USART1 на прийом RX_START_FLAG=1 (відповіді на запит немає);

– прийнятому типу повідомлення. Якщо тип повідомлення Type_message=1 (запит), то в відповіді (ERRS_ASK) Z_ER=0, інакше – Z_ER=1 і встановлюється ознака видачі відповіді без прикладних даних FULL_OTVET_FLAG=0 довжиною 0 byte;

– номеру запиту зсередини 1 s. Якщо прийнятий номер запиту Cnt_message не рівний попередньому, то в відповіді (ERRS_ASK) CNT_ER=0, інакше – CNT_ER=1 і встановлюється ознака видачі відповіді без прикладних даних FULL_OTVET_FLAG=0 довжиною 0 byte;

– довжині прикладних даних запиту. Якщо прийнята довжина прикладних даних в запиті рівна встановленій довжині або рівна нулю, то в відповіді (ERRS_ASK) L_ER=0, інакше – L_ER=1 і встановлюється ознака видачі відповіді без прикладних даних FULL_OTVET_FLAG=0 довжиною 0 byte;

2) якщо довжина прикладних даних запиту рівна встановленій довжині, то встановлюється ознака нових прикладних даних NEW_DATA_FLAG=1 і вони переписуються до буферу RxBuf_out для виконання, а також встановлюється ознака видачі відповіді з прикладними даними FULL_OTVET_FLAG=1 довжиною 16 byte;

3) визначення в відповіді (ERRS_ASK) інтегральної ознаки помилки діагностики запитів при обміні Int_ER складанням по «АБО» всіх ознак помилок, котрі визначились згідно першому кроку.

Підпрограма ObrZapros викликається в підпрограмі опрацювання переривань від USART1 USART1_IRQHandler.

Вхідними даними є RxBuf.

Вихідними даними є ERRS_ASK, FULL_OTVET_FLAG, NEW_DATA_FLAG, RxBuf_out.

4.2.8 Підпрограма формування буферу відповіді в ЦП FormBufOtvet

Підпрограма FormBufOtvet виконує підготовку буферу відповіді ЦП.

Алгоритм підпрограми складає наступну послідовність кроків:

1) запис байтів 1–12 відповіді до буферу видачі відповіді TxBuf (заголовка, помилок самодіагностики ERRS_MODULE, результату контролю запиту від МЦП ERRS_ASK, версії програми модулю VerPO);

2) налаштування USART1 на передачу по DMA2 Stream7 и старт передачі (підпрограма TxDMA);

3) якщо довжина виданих прикладних даних нульова (FULL_OTVET_FLAG=0), то виконується підрахунок CRC32_Otv 12 byte TxBuf (підпрограма calc_CRC32) і запис її до TxBuf, починаючи з молодшого байту;

4) якщо довжина виданих прикладних даних не нульова (FULL_OTVET_FLAG=1), то виконується запис до TxBuf 16 byte прикладних даних (OutBuf) і підрахунок CRC32_Otv 28 byte TxBuf (підпрограма calc_CRC32) і запис її до TxBuf, починаючи з молодшого байту (передача вже запущена).

Підпрограма FormBufOtvet викликається в підпрограмі опрацювання переривань від USART1 USART1_IRQHandler (по закінченню тайм-ауту після прийому останнього байту від ЦП).

Вхідними даними є FULL_OTVET_FLAG, OutBuf.

Вихідними даними є TxBuf.

4.2.9 Підпрограма розрахунку контрольної суми масиву даних calc_CRC32

Підпрограма calc_CRC32 виконує підрахунок 32-разрядної контрольної суми масиву заданої довжини (кратної чотирьом байтам), використовуючи модуль CRC (апаратний метод).

Алгоритм підпрограми складає послідовність наступних кроків:

1) підрахунок вказівника кінцевої адреси масиву pEndAdr (вказівник початкової адреси масиву збільшується на довжину масиву, поділену на чотири);

2) присвоєння вихідного стану контрольної суми CRC_INIT=0xFFFFFFFF;

3) циклічний запис змісту чотирьох байтів масиву (вказівник поточної адреси pAdr інкрементується у кожному циклі) у 32-розрядний регістр даних CRC_DR модуля апаратного розрахунку CRC. Якщо вказівник поточної адреси pAdr більше вказівника кінцевої адреси масиву pEndAdr, то виконується вихід із підпрограми.

Підпрограма викликається в підпрограмах ObrZapros и FormBufOtvet (в підпрограмі опрацювання переривань від USART1_IRQHandler).

Вхідними даними є вказівник початкової адреси pAdr, довжина масиву nmbr.

Вихідними даними є CRC_DR (підрахована контрольна сума, рівна нулю, якщо немає помилки CRC32, в підпрограмі ObrZapros і значення CRC32, що записується до буферу відповіді, в підпрограмі FormBufOtvet).

4.2.10 Підпрограма налаштування USART1 на передачу по DMA2 і старт передачі TxDMA

Підпрограма TxDMA виконує налаштування USART1 на передачу по каналу DMA і старт передачі відповіді до ЦП (перемикання лінії на прийом/передачу виконується автоматично).

Алгоритм підпрограми складає послідовність наступних кроків:

1) вимикання і заборона переривання USART1, заборона роботи каналу DMA2 Stream7 на передачу;

2) ініціалізація USART1 на передачу 16 (FULL_OTVET_FLAG=0) або 32 byte (FULL_OTVET_FLAG=1) із буферу TxBuf, дозвіл передачі USART1 через DMA2 Stream7, дозвіл переривання по закінченню передачі USART1.

Підпрограма викликається в функції опрацювання переривання USART1_IRQHandler по закінченню тайм-аута після прийому останнього байта від ЦП USART_ISR_RTOF (в функції FormBufOtvet).

Вхідними даними є FULL_OTVET_FLAG.

Вихідних даних немає.

4.2.11 Підпрограма управління реле за командою, прийнятою в запиті, Enable_Disable_Relay

Підпрограма Enable_Disable_Relay виконує управління реле за командою, прийнятою в запиті від ЦП.

Алгоритм підпрограми складає послідовність наступних кроків:

1) якщо від ЦП для каналу j отримано команду EnChj_tek, що відрізняється від попередньої EnCh_pred та вона EnChj_tek=1 (ввімкнути канал), то встановлюється напруга обмотки 5V для каналів j=(1–8) PB14=0, для каналів j=(9–16) PB15=0, ініціалізується лічильник кількості вимірів після зміни напруги живлення реле з 3 V на 5 V, що дорівнює 3 (для каналів j=(1–8) voltage_number_conversion_V1=3, для каналів j=(9–16) voltage_number_conversion_V2=3, рахунок ведеться в підпрограмі Control_Relay), вмикається реле (відповідному номеру каналу порту PD присвоюється логічна «1») і встановлюється час затримки на вмикання реле

даного каналу Time_delay_relay_ON= 200 (20 ms з дискретністю рахунку 100 μs), протягом котрого не виконується контроль стану реле;

2) якщо поточна команда EnChj_tek=0 (вимкнути канал), то відповідному номеру каналу порту PD присвоюється логічний «0» і встановлюється час затримки на вимкнення реле даного каналу Time_delay_relay_OFF=150 (15 ms з дискретністю рахунку 100 μs), протягом котрого не виконується контроль стану реле;

Підпрограма викликається в підпрограмі опрацювання переривання USART1_IRQHandler по закінченню прийому запиту від ЦП.

Вхідними даними є MODULE, EnChj_tek, EnCh_pred.

Вихідними даними є порти PD0–PD7, Time_delay_relay_ON,

Time_delay_relay_OFF, voltage_number_conversion_V1,

voltage_number_conversion_V2.

4.2.12 Підпрограма опиту стану контактів реле Control_Relay

Підпрограма Control_Relay виконує опит стану контактів реле пристрою.

Алгоритм підпрограми складає послідовність наступних кроків:

1) визначається загальна кількість релейних каналів total_number_ch=8 і перехід до кроку 2;

2) якщо номер каналу number_ch менше 8, то перейти до кроку 3, інакше перейти до кроку 9;

3) якщо час на перемикання всіх каналів збіг (Time_delay_relay_ON[number_ch]=0 и Time_delay_relay_OFF[number_ch]=0), то виконується перехід до кроку 4, інакше – до кроку 5;

4) виконати вимір напруги 3 V на шині V1 зі допомогою підпрограми voltage_result, якщо виміряне значення входить в допустимий діапазон (від 2,7 до 3,5 V), то в змінній ERR_Napr біт Err3V1=0 (немає помилки електроживлення 3 V реле каналів 1–8), інакше– Err3V1=1 (є помилка), перехід до кроку 6;

5) якщо час затримки на перемикання всіх каналів не збіг (не всі канали ввімкнулись), то виконується вимір напруги 5 V на шині V1 (підпрограма ADC_voltage_measurement), в відповідь для ЦП записується значення напруги електроживлення 5 V з ціної одинці молодшого розряду 22 mV, інакше – перехід до кроку 6;

6) записати до буфер відповіді значення станів релейних каналів ChSt[number_ch], отримані в результаті виконання підпрограми Control_Contact в TIM3_IRQHandler і перейти до кроку 7;

7) записати до буферу відповіді значення справності релейних каналів RLER[number_ch], отримані в результаті мажорування результатів діагностики каналів за 10 ms в TIM2_IRQHandler і перейти до кроку 8;

8) збільшити значення номеру каналу number_ch++ і перейти до кроку 1;

9) вихід із підпрограми.

Підпрограма викликається в головній функції main кожну мілісекунду.

Вхідними даними є MODULE, Time_delay_relay_ON[number_ch], Time_delay_relay_OFF[number_ch],RlEr[2][number_ch],RlEr[1][number_ch], RlEr[0][number_ch],voltage_number_conversion_V1,voltage_number_conversion_V2.

Вихідними даними є U5V1, U3V1, U5V2, U3V2, ERR_Napr (Err5V1, Err3V1, Err5V2,Err3V2),OutBuf,voltage_number_conversion_V1,voltage_number_conversion_V2,RlEr[2][number_ch],RlEr[1][number_ch],RlEr[0][number_ch],RlEr[number_ch].

4.2.13 Підпрограма виміру температури ADC_temperature_measurement

Підпрограма ADC_temperature_measurement виконує вимір температури пристрою..

Алгоритм підпрограми складає послідовність наступних кроків:

1) вибір каналу 18 (вимір температурного датчика), дозвіл роботи і старт перетворення ADC1 (HAL_ADC_Start);

2) після завершення зчитується результат и виконується розрахунок температури temper в градусах Цельсію;

3) зупинка перетворення ADC1 (HAL_ADC_Stop).

Підпрограма викликається в головній функції main кожну мілісекунду.

Вхідними даними є номер каналу АЦП.

Вихідними даними є float temper.

4.2.14 Підпрограма опрацювання переривань від USART1 USART1_IRQHandler

Підпрограма USART1_IRQHandler виконує опрацювання переривань USART1 по завершенню передачі відповіді до ЦП і по завершенні прийому запиту від ЦП.

Алгоритм підпрограми складає послідовність наступних кроків:

1) якщо виникло переривання по завершенню передачі останнього байту відповіді в ЦП (прапор переривання TC=1), то виконується вимкнення USART1 і заборона переривання по закінченню передачі USART1;

2) скидання прапору переривання TC, заборона роботи каналу DMA2 Stream7;

3) встановлення прапору перемикання USART1 на налаштування на прийом RX_START_FLAG=1 і вихід із підпрограми;

4) якщо виникло переривання по закінченню прийому запиту від ЦП по закінченню тайм-ауту «тиші» прийому (прапор переривання RTOF=1), то виконується заборона переривання по тайм-ауту приймача USART1, скидання прапору переривання RTOF=0, заборона роботи каналу DMA2 Stream2;

5) опрацювання прийнятого від МЦП запиту (підпрограма ObrZapros);

6) якщо прийнятий запит без помилки довжини (ASK_ER_FLAG=0) і без помилки контрольної суми (CRC32_ER_FLAG=0), то виконується формування і видача буферу відповіді (підпрограма FormBufOtvet);

7) скидання помилок ERRS_ASK після видачі відповіді (підпрограма ERRS_ASK_RESET);

8) якщо отримано безпомилковий запит від ЦП з новими прикладними даними (NEW_DATA_FLAG=1), то втконується занесення прикладних даних з буферу запиту RxBuf (з 8 по 39 byte) до буферу даних DataBuf (підпрограма Write_Data_Buf);

9) виконується управління реле за командою, прийнятою в запиті від ЦП (підпрограма Enable_Disable_Relay);

Виклик підпрограми відбувається за вектором переривання USART1.

Вхідними даними є прапори переривання TC, RTOF.

Вихідними даними є RX_START_FLAG, TC, RTOF, ERRS_ASK, NEW_DATA_FLAG.

4.2.15 Підпрограма опрацювання переривання від таймеру TIM2 TIM2_IRQHandler

Підпрограма TIM2_IRQHandler виконує обробку переривання від таймеру TIM2 (відлік 1 ms).

Підпрограма TIM2_IRQHandler виконує скидання прапору переривання Update interrupt Flag UIF, встановлюються прапори контролю напруги на обмотках реле і виміри температури CONTROL_RELAY_FLAG=1, CONTROL_TEMPERATURE_FLAG=1, флаг контролю дієздатності реле CONTROL_CONTACT_FLAG (контроль дієздатності відбувається в підпрограмі TIM2_IRQHandler кожні 10 ms), а також відлік тайм-ауту 5 s після подачі живлення (декремент TIME_OUT_5s).

В підпрограмі TIM2_IRQHandler інкрементується значення лічильника counter_contact_flag, котрий відраховує часові інтервали для контролю справності реле (контроль справності реле з подальшим мажоруванням по «2 із 3» виконується кожні 10 ms). Всі реле опитуються по черзі, при цьому опитування кожного наступного реле відбувається в TIM3 (сто мікросекундна мітка).

В підпрограмі TIM2_IRQHandler здійснюється перемикання напруги електроживлення обмоток реле, якщо вийшов час затримки на ввімкнення всіх каналів в групі (всі Time_delay_relay_ON рівні нулю).

Виклик підпрограми відбувається за вектором переривання таймеру TIM2.

Вхідними даними є UIF, і час затримки на ввімкнення каналів Time_delay_relay_ON.

Вихідними даними є CONTROL_RELAY_FLAG, CONTROL_TEMPERATURE_FLAG, TIME_OUT_5s, стан портів управління напруги (PB14, PB15).

4.2.16 Підпрограма опрацювання переривання від TIM3 TIM3_IRQHandler

Підпрограма TIM3_IRQHandler виконує опрацювання переривання від таймеру TIM3 (відлік 100 μs).

Алгоритм підпрограми складає послідовність наступних кроків:

1) скидання прапору переривання Update interrupt Flag UIF;

3) якщо канал ввімкнено (Doff_temp[number_ch]=0), то виконується інкремент лічильника вмикань Cnt_OFF_AC[number_ch];

4) опитування 16 оптронів порогу спрацювання в Don_temp[number_ch], де номер каналу number_ch від нуля до 15;

5) якщо канал вимкнено (Don_temp[number_ch]=0), то виконується інкремент лічильника вмикань Cnt_ON_AC[number_ch];

6) для 16 каналів декремент лічильників часу режекції дискретних каналів Trej, часу затримки дискретних каналів Time_delay_diskret, часу тайм-ауту після режекції (на відновлення каналу) Timeout_diskret, часу затримки на перемикання реле Time_delay_relay_ON (на ввімкнення) та Time_delay_relay_OFF (на вимкнення);

8) контроль тайм-ауту 5 ms між безпомилковими запитами TIME_OUT_5ms (біт TIME_OUT_ER в ERRS_ASK).

Підпрограма TIM3_IRQHandler виконує для кожного релейного каналу з'ясування стану реле (за контрольним контактом) і відповідність цього стану поточній команді за допомогою підпрограми Control_Contact. Алгоритм складається з наступних кроків::

1)якщо час на перемикання збіг, то виконується опитування поточного стану контрольного контактного реле (порт PE). Стан контакту контролюється згідно діям 2)–4);

2) якщо NO_state[Relay_Contr_number]=0, то ChSt[Relay_Contr_number]=1 (ввімкнений стан каналу), якщо при цьому встановлена команда на ввімкнення реле EnCh_tek[Relay_Contr_number]=1, то лічильник помилок RlErr_Counter[Relay_Contr_number] залишається незмінним, інакше лічильник помилок інкрементується RlErr_Counter[Relay_Contr_number]++);

3) якщо NO_state[Relay_Contr_number]=1, то ChSt[Relay_Contr_number]=0 (вимкнений стан каналу), якщо при цьому встановлена команда на вимкнення реле (EnCh_tek[Relay_Contr_number]=0, то лічильник помилок RlErr_Counter[Relay_Contr_number] залишається незмінним, інакше лічильник помилок інкрементується RlErr_Counter[Relay_Contr_number]++);

Виклик підпрограми відбувається за вектором переривання таймеру TIM3.

Вхідними даними є UIF, Cnt_OFF_AC, Cnt_ON_AC, Trej, Time_delay_diskret, Timeout_diskret, Time_delay_relay_ON, Time_delay_relay_OFF, TIME_OUT_5ms, NO_state[Relay_Contr_number], EnCh_tek[Relay_Contr_number], RlErr_Counter[Relay_Contr_number].

Вихідними даними є Cnt_OFF_AC, Cnt_ON_AC, Trej, Time_delay_diskret, Timeout_diskret, Time_delay_relay_ON, Time_delay_relay_OFF, TIME_OUT_5ms,

ERRS_ASK, ChSt[Relay_Contr_number], RlEr[number_ch] (при динамічний діагностиці каналу).

4.2.17 Підпрограма опрацювання переривань від таймеру TIM4 TIM4_IRQHandler

Підпрограма TIM4_IRQHandler виконує опрацювання переривань від таймеру TIM4 (відлік 5 ms). Алгоритм складає наступну послідовність кроків:

1) скидання прапору переривання Update interrupt Flag UIF;

2) якщо Delay10ms_OFF[number_ch]=0 (пройшло 10 ms), то канал вимкнений (Doff_tek[number_ch]=1), інакше – декремент лічильника Delay10ms_OFF[number_ch];

5) скидання лічильника Cnt_OFF_AC[number_ch];

6) якщо тривалість сигналу на виході оптрона (порога спрацювання) рівна 1 ms і більше (Cnt_ON_AC[number_ch]≥10), то канал ввімкнений (Don_tek[number_ch]=0), інакше – канал вимкнений (Don_tek[number_ch]=1);

7) скидання лічильника Cnt_ON_AC[number_ch].

Підпрограма TIM4_IRQHandler виконує контроль справності оптрона порога відпускання дискретного входу. Алгоритм складає наступну послідовність кроків:

1) якщо на вході порту порогу відпускання PCj є логічна «1», то кожні 5 ms в підпрограмі TIM4_IRQHandler контрорлюється поріг спрацювання (на вході порту порогу спрацювання PEj також має бути рівень логічної «1»). Якщо є невідповідність, то нарощується лічильник помилкових ситуацій RlErrDiskret_DIDOM (поканально);

2) один раз в 100 ms (відлік тайм-ауту ведеться так само, як і в підпрограмі TIM4_IRQHandler) встановлюється прапор CONTROL_OPTO_FLAG;

3) за прапором CONTROL_OPTO_FLAG в основному циклі виконується контроль лічильника RlErrDiskret_DIDOM. Якщо його значення більше 16 (16*5=90 ms), то стан оптрону порогу спрацювання не відповідає правильному і встановлюється помилка (відмова) дискретного входу в поточному циклі контроля;

4) виконується мажорування результату по «2 із 3». Якщо помилка каналу тримається 180 ms і більше, то встановлюється відмова каналу, якщо менше 180 ms, то канал дієздатний. За дієздатність каналу відповідає біт RL_ERj (де j– номер каналу).

Виклик підпрограми відбувається за вектором переривання таймеру TIM4 кожні 5 ms.

Вхідними даними є MODULE, Cnt_OFF_AC[number_ch], Cnt_ON_AC[number_ch], Delay10ms_OFF[number_ch], Temp_Number_Doff_DIDOM [num-ber_ch].

Вихідними даними є Doff_tek[number_ch], Don_tek[number_ch], Cnt_OFF_AC[number_ch], Cnt_ON_AC[number_ch], Delay10ms_OFF[number_ch], RlErrDis-kret_DIDOM[number_ch].

4.2.18 Підпрограма формування поточної команди управління транзисторним каналом для динамічної діагностики каналу Read_Transistor_Command

Підпрограма формування поточної команди управління транзисторним каналом для динамічної діагностики каналу Read_Transistor_Command інвертує стан поточної команди управління в випадку наявності в запиті команди на динамічну діагностику (за умови, що поточна команда управління не змінювалась. Алгоритм складає наступну послідовність кроків:

1) якщо номер каналу number_ch менее четырёх, то перейти до кроку 2, інакше перейти до кроку 8;

2) якщо поточна команда не змінювалась EnCh_tek[number_ch]=EnCh_pred[number_ch], то перейти до кроку 3, інакше перейти до кроку 7;

3) якщо встановлена команда динамічної діагностики каналу Diag_Ch[number_ch]=1, то перейти до кроку 3, інакше перейти до кроку 7;

4) якщо поточна команда управління каналом – канал ввімкнений (EnCh_tek[number_ch]=1), то проінвертувати команду EnCh_tek[number_ch]=0, зберегти значення команди до початку діагностики EnCh_do_dinamic[number_ch]=1 і перейти до кроку 6, інакше перейти до кроку 5;

5) проінвертувати команду EnCh_tek[number_ch]=1 і зберегти значення команди до почтаку діагностики EnCh_do_dinamic[number_ch]=0;

6) встановити ознаку динамічної діагностики каналу DINAMIC_FLAG[number_ch]=SET_FLAG, встановити час тайм-ауту на контроль каналу dinamic_timer[number_ch]=4 (400 μs);

7) збільшити значення номеру каналу number_ch++ і перейти до кроку 1;

8) вихід із підпрограми.

Виклик підпрограми відбувається за вектором переривання USART1.

Вхідними даними є поточні значення команд управління, отримані в запиті EnCh_tek[number_ch], попередні значення команд EnCh_pred[number_ch] і значення команд динамічної діагностики Diag_Ch[number_ch].

Вихідними даними є поточні значення команд управління EnCh_tek[number_ch], отримані в результаті аналізу команд діагностики Diag_Ch[number_ch].

4.2.19 Підпрограма управління транзисторними ключами за командою, прийнятою в запиті Enable_Disable_Transistor

Підпрограма управління транзисторними ключами Enable_Disable_Transistor керує вихідними транзисторними каналами згідно даних, отриманих в запиті від ЦП і даним, отриманим в результаті виконання підпрограми Read_Transistor_Command. Алгоритм складає наступну послідовність кроків:

1) якщо номер каналу number_ch менше чотирьох, то перейти до кроку 2, інакше перейти до кроку 6;

2) якщо змінилась поточна команда управління каналом (EnCh_tek[number_ch] не рівно EnCh_pred[number_ch]), то перейти до кроку 3, інакше перейти до кроку 5;

3) якщо встановлена команда ввімкнення транзисторного ключа, то ввімкнути транзисторний ключ, встановити вихідний час тайм-ауту на перемикання контрольного оптрона Time_delay_optron_transistor[number_ch] і перейти до кроку 5, інакше перейти до кроку 4;

4) якщо встановлена команда вимкнення транзисторного ключа, вимкнути транзисторний ключ, встановити вихідний час тайм-ауту на перемикання контрольного оптрона Time_delay_optron_transistor[number_ch], скинути ознаку в каналі KZ[number_ch]=0, скинути прапор повторного ввімкнення NUMBER_OF_ENTER_FLAG[number_ch]= RESET_FLAG і перейти до кроку 5;

5) збільшити значення номеру каналу number_ch++ і перейти до кроку 1;

6) вихід із підпрограми.

Виклик підпрограми відбувається за вектором переривання USART1.

Вхідними даними є поточні значення команд управління, отримані в запиті EnCh_tek[number_ch], і попередні значення команд EnCh_pred[number_ch].

Вихідними даними є стан портів управління транзистором (PB5–PB8), час тайм-ауту на перемикання контрольного оптрона Time_delay_optron_transistor[number_ch], стан прапору повторного ввімкнення NUMBER_OF_ENTER_FLAG[number_ch].

4.2.20 Підпрограма контролю справності транзисторного ключа в режимі поточної експлуатації Control_Transistor

Підпрограма контролю транзисторних каналів в порядку поточної експлуатації Control_Transistor контролює стан транзисторного ключа по виходу діагностичного оптрону і формує ознаку дієздатності каналу в режимі поточної експлуатації. Алгоритм складає наступну послідовність кроків:

1) якщо номер каналу number_ch менше чотирьох, то перейти до кроку 2, інакше перейти до кроку 11;

2) переписати результати діагностики, отримані в двох суміжних циклах контролю (RlErr[2][number_ch]=RlErr[1][number_ch], RlErr[1][number_ch]=RlErr[0][number_ch]), і перейти до кроку 3;

3) якщо час на перемикання контрольного оптрона збіг Time_delay_optron_transistor[number_ch]=0, то перейти до кроку 4, інакше перейти до кроку10;

4) зчитати поточний стан контрольного оптрона Diag_transistor[number_ch] и перейти к шагу 5;

5) если Diag_transistor[number_ch]= 0 (ввімкнений стан транзистора), то перейти до кроку 6,інакше перейти до кроку 7;

6) ChSt[number_ch]=1 (встановити ввімкнений стан каналу number_ch в відповіді). Якщо встановлена команда на ввімкнення каналу EnCh_tek[number_ch]=1, то результат поточної діагностики каналу RlErr[0][number_ch]=0 (канал справний), інакше встановити ознаку відмови каналу RlErr[0][number_ch]=1 і перейти до кроку 9;

7) якщо Diag_transistor[number_ch]=1 (вимкнений стан транзистора), то перейти до кроку 8, інакше перейти до кроку 9;

8) ChSt[number_ch]=0 (встановити вимкнений стан каналу number_ch в відповіді). Якщо встановлена команда на вимкнення транзистора EnCh_tek[number_ch]=0, то результмат поточної діагностики каналу

RlErr[0][number_ch]=0 (канал справний), інакше встановити ознаку відмови каналу RlErr[0][number_ch]=1;

9) виконати мажорування результату діагностики за принципом «2 из 3» (результат діагностики каналу – RlErr[number_ch]);

10) збільшити значення номеру каналу number_ch++ и перейти до кроку 1;

11) вихід із підпрограми.

Виклик підпрограми відбувається за вектором переривання TIM2 (1 ms).

Вхідними даними є результати діагностики, отримані в суміжних циклах контролю RlErr[1][number_ch], RlErr[0][number_ch].

Вихідними даними є стан вихідних дискретних ChSt[number_ch] і результат діагностики каналу RlErr[number_ch] (по «2 из 3»).

4.2.21 Підпрограма контролю справності транзисторного ключа при динамічному контролі за командою, прийнятою в запиті Dinamic_Control_Transistor

Підпрограма Dinamic_Control_Transistor контролю транзисторних каналів за командою, прийнятою в запиті, контролює стан транзисторного ключа за виходом діагностичного оптрону і формує ознаку дієздатності каналу за запитом від ЦП. Алгоритм складає наступну послідовність кроків:

1) якщо номер каналу number_ch меншк чотирьох, то перейти до кроку 2, інакше перейти до кроку 8;

2) зчитати поточний стан контрольного оптрона Dinamic_Diag_transistor[number_ch] і перейти до кроку 3;

3) якщо Dinamic_Diag_transistor[number_ch]=0 (ввімкнений стан транзистора), то перейти до кроку 4, інакше перейти до кроку 5;

4) якщо встановлена команда на ввімкнення каналу EnCh_tek[number_ch]=1, то результат поточної діагностики каналу RlErr [number_ch]=0 (канал справний), інакше встановити ознаку відмови каналу RlErr[number_ch]=1 и перейти к шагу 7;

5) якщо Dinamic_Diag_transistor[number_ch]=1 (вимкнений стан транзистора), то перейти до кроку 6, інакше перейти до кроку 7;

6) якщо встановлена команда на вимкнення каналу EnCh_tek[number_ch]=0, то результат поточної діагностики каналу

RlErr[number_ch]=0 (канал справний), інакше встановити ознаку відмову каналу RlErr[number_ch]=1;

7) збільшити значення номера каналу number_ch++ и перейти к шагу 1;

8) вихід із підпрограми.

Виклик підпрограми відбувається за вектором переривання TIM3 (100 μs).

Вхідними даними є результати діагностики оптрона Dinamic_Diag_transistor[number_ch].

Вихідними даними є результати діагностики каналів RlErr[number_ch] (без мажорування за «2 із 3»).

4.2.22 Підпрограма виміру струму в транзисторних каналах ADC_current_measurement

Підпрограма виміру струму в транзисторних каналах ADC_current_measurement виконує вимір струму в транзисторних каналах за допомогою ADC2 і передає значення струму у відповідному повідомленні від пристрою. Алгоритм складає наступну послідовність кроків:

1) якщо номер каналу number_ch менее четырёх, то перейти до кроку, інакше перейти до кроку 7;

2) переписати значення коду АЦП отриманні в результаті дев’яти останніх вимірів і перейти до кроку 3;

3) зчитати поточне значення коду, що зберігається для кожного каналу в окремому регістрі ADC2, і перейти до кроку 4;

4) вирахувати середнє арифметичне значення коду за 10 останніх вимірів і перейти до кроку 5;

5) виконати розрахунок значення струму в каналі. Округлити результат і сформувати значення для відповідного повідомлення;

6) збільшити значення номера каналу number_ch++ і перейти до кроку 1;

7) вихід із підпрограми.

Виклик підпрограми відбувається в головній функції main кожну мілісекунду.

Вхідними даними є результати значення струму, що зберігаються в регістрах ADC2 (JDR1–JDR4).

Вихідними даними є розраховані значення в каналах current_result[number_ch].

4.2.23 Підпрограма скидання помилок прийому запиту Подпрограмма ERRS_ASK_RESET

Підпрограма ERRS_ASK_RESET виконує скидання помилок, знайдених при прийомі запиту від ЦП (змінна ERRS_ASK) після видачі їх в відповіді до ЦП.

Виклик підпрограми відбувається в підпрограмі USART1_IRQHandler.

Вхідними даними є ERRS_ASK.

Вихідними даними є ERRS_ASK.

4.2.24 Підпрограма формування результатів самодіагностики ERRS_MODULE_RESET_SET

Підпрограма ERRS_MODULE_RESET_SET виконує формування байта відповіді до ЦП ERRS_MODULE за результатами самодіагностики.

Алгоритм складає наступну послідовність кроків:

1) якщо немає помилок по напрузі живлення (ERR_Napr=0 – всі ознаки Err_3V1, Err_3V2, Err_5V1, Err_5V2 рівні нулю), то в байті відповіді ERRS_MODULE біт ERU=0, інакше – ERU=1;

2) якщо самодіагностика ЗСД пройшла успішно (ErrROM=0), то в байті відповіді ERRS_MODULE біт ERROM=0, інакше – ERROM=1;

3) якщо немає ознаки помилки (ErrType=0), то в байті відповіді ERRS_MODULE біт ERX=0, інакше – ERX=1;

4) якщо в результаті самодіагностики присутні помилки по напрухі живлення (ERU=0), немає помилки тестування ЗСД (ERROM=0), то в байті відповіді ERRS_MODULE біт RAB=0 (результат самодіагностики позитивний), інакше – RAB=1.

Підпрограма викликається в кожному циклі головної функції main.

Вхідними даними є ERR_Napr, ErrROM, ErrType.

Вихідними даними є ERRS_MODULE.

ВИСНОВОК

В ході проекту було проаналізовано можливі рішення задачі, обрано найбільш зручний під наші потреби шлях проектування і, власне, розробка пристрою.

Розроблений пристрій, а саме модуль вводу і формування дискретних сигналів, отримав вісім каналів вводу дискретних сигналів, вісім каналів їх подальшого формування з елементами контролю та діагностики реле, чотири з’єднувачі для підключення зовнішніх ланцюгів та загальне керування всієї системи у вигляді мікроконтролеру на базі сімейства STM32 та можливість з’єднання з інтерфейсом RS-485.

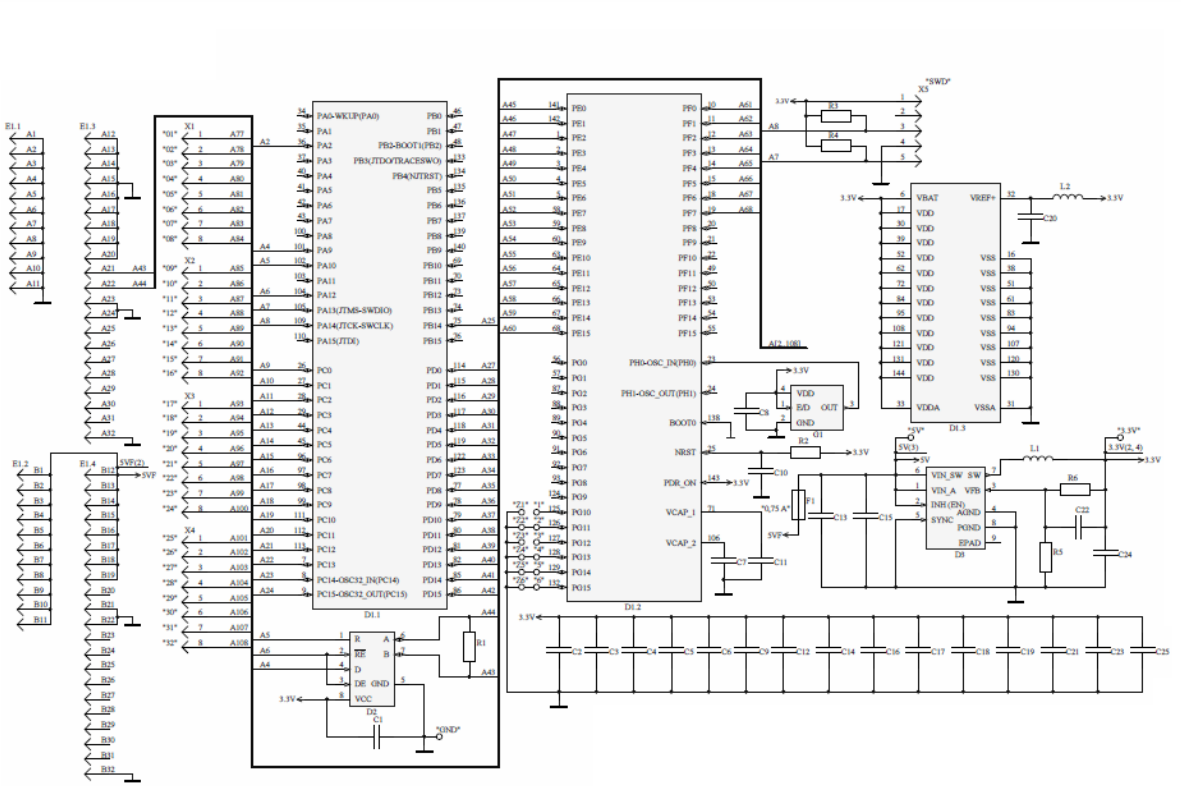
Основним принципом роботи пристрою є здійснення вхідними каналами ввід сигналів та формування імпульсу режекції струму, та забезпечення каналами дискретного виводу комутації виконуючих пристроїв для подальшої обробки та передачі сформованих сигналів.

Окрім основних функцій по опрацюванню та формуванню дискретних сигналів, пристрій також має функції діагностування програмних засобів мікроконтролера та стану вихідних каналів формування сигналів, контролю наявності та відповідності електроживлення, контроль за температурою пристрою для запобігання його недієздатності в разі її істотного перевищення існуючих вимог.

Проведена розробка безпечного та функціонального програмного забезпечення.

ДОДАТОК А

Схема електрична



ДОДАТОК Б

Перелік елементів

Познач.	Найменування	Кількість
C1, C4…C6, C8…C10, C12, C14…C21, C23, C25, C26, C29…C31	Конденсатор VJ0805 X7R-50B-0,1 X7R-50B-0,1мкФ Vishay	20
C2, C3, C13, C24, C27, C28	Конденсатор 1210 X7R-16B-22мкФ Murata	6
C7, C11	Конденсатор 0603 X7R-6,3B-2,2мкФ Murata	2
C22	Конденсатор VJ1206 COG-50B-4700пФ Vishay	1
D1	Мікросхема STM32F745ZGT6 ST Microelectronics	1
D2	Мікросхема SN65HVD10D Texas Instruments	1
D3	Мікросхема ST1S10PUR ST Microelectronics	1
D4…D6	Мікросхема MC74HC14ADG ON Semiconductor	3
F1	Запобіжник miniSMDC075 Raychem	1
F2	Запобіжник SMD100F/33-2 Tyco Electronics	1
G1	Генератор 0-25.0-VX3MH-T1 Jauch	1
L1	Дросель B82472-3 A-2,2мкГн Epcos	1
L2	Дросель BLM21AG102SN1D Murata	1
R1	Резистор RC1206 J R F 120R Yageo	1
R2…R4, R8	Резистор RC1206 J R F 10K Yageo	4
R5	Резистор RC1206 F R F 2K4 Yageo	1
R6	Резистор RC1206 F R F 7K5 Yageo	1

R7, R9	Резистор RC1206 F R F 10K Yageo	2
V1	Діод PMEG6020ER NXP	1
V2	Транзистор IRLML9301TRPBF International Rectifier	1
X1…X4	Вилка MSTB 2,5/8-GF-5,08 Phoenix Contact	4
X5	Вилка 90120-0765 Molex	1
U1…U8	Канал виводу дискретного сигналу	
C2	Конденсатор VJ0805 X7R-50B-0,1 X7R-50B-0,1мкФ Vishay	1
K1	Реле RM84-2012-35-1005 Relpol	1
R1	Резистор RC1206 J R F 10K Yageo	1
R2	Резистор RC1206 J R F 200R Yageo	1
R4	Резистор RC1206 J R F 470R Yageo	1
R5	Резистор RC0402 F R 10K Yageo	1
V1	Транзистор IRLML6346TRPBF International Rectifier	1
V2	Діод PMEG6020ER NXP	1
V3	Обмежувач напруги ESD5Z3.3T1G ON Semiconductor	1
Z1	Фільтр BLM31PG601SN1 Murata	1
U9…U16	Канал вводу дискретного сигналу 110В DC	
C1…C3	Конденсатор VJ0805 X7R-50B-0,1 X7R-50B-0,1мкФ Vishay	3
D1, D2	Мікросхема TLV431BIDBZT Texas Instruments	2
R1	Термістор PTC-600B-400 Ом Vishay	1

R2	Резистор RC1206 J R F 330R Yageo	1
R3	Резистор PO590-0,1K5 Vitrohm	1
R4, R5, R7, R8	Резистор RC1206 F R F 191K Yageo	4
R6	Резистор RC1206 F R F 6K34 Yageo	1
R9	Резистор RC1206 F R F 7K5 Yageo	1
R10	Резистор PO595-0,56K Vitrohm	1
R11, R12	Резистор RC1206 J R F 3K0 Yageo	2
R13, R14, R17, R18	Резистор RC1206 J R F 5K1 Yageo	4
R15, R16	Резистор RC1206 J R F 10K Yageo	2
V1	Обмежувач напруги 1V5KE350A Fairchild	1
V2	Твердотільне реле CPC1394GR IXYS	1
V3	Стабілітрон BZX84-C4V7 NXP	1
V4, V5	Оптопара SFH6186-5X001T Vishay	1

ДОДАТОК В

Код програми

main

```
/* Includes ------------------------------------------------------------------*/
#include "main.h"
/* USER CODE BEGIN Includes */
#include "DIDOM.h"
//#include "crcTable.h"
/* VerPO ---------------------------------------------------------------------*/
#pragma location = ".verPO"
__root const uint32_t __verPO = 0x00000000;

#pragma location = ".date_comp"
__root const char __date_comp[] = __DATE__;
#pragma location = ".time_comp"
__root const char __time_comp[] = __TIME__;
/* USER CODE END Includes */
/* Private variables ---------------------------------------------------------*/

/* USER CODE BEGIN PV */
/* Private variables ---------------------------------------------------------*/
/*прапори*/
uint8_t NEW_DATA_FLAG;//прапор нових прикладних даних (1-є нові дані, 0-немає)
uint8_t FULL_OTVET_FLAG;//прапор видачі відповіді (1 – видача з прикладними даними, 0- без прикладних)
uint8_t RX_START_FLAG;//прапор налаштування на прийом і початок прийому
uint8_t ASK_ER_FLAG;//прапор помилкової довжини запиту (не входить в діапазон 12-1084 байта)
```

uint8_t CRC32_ER_FLAG;//прапор невірного запиту

uint8_t CONTROL_DISKRETE_FLAG;//прапор контролю стану дискретного входу (1-є контроль, 0-немає)

uint8_t CONTROL_CONTACT_FLAG;//прапор контролю стану контактів реле (1-є контроль, 0-немає)

uint8_t CONTROL_RELAY_FLAG;//прааор контролю стану реле (1-є контроль, 0-немає)

uint8_t CONTROL_TRANSISTOR_FLAG;//прапор контролю стану транзистора (1 - є контроль, 0 - немає)

uint8_t CONTROL_TEMPERATURE_FLAG;//прапор контролю температури (1-є контроль, 0-немає)

uint8_t FLAG_REG[16];// наявність імпульсу режекції (0-пройшла, 1- йде)

uint8_t FLAG_TIME_OUT_REG[16];//тайм-аут після імпульсу режекції (0-збіг, 1-триває)

uint8_t FLAG_REG_OFF[16];//необхідність перевірки вузла режекції струму (0-проконтрольована, 1- не пройшла контроль)

uint16_t counter_opto_flag;//лічильник для прапора CONTROL_OPTO_FLAG

uint16_t counter_contact_flag;//лічильник для прапора CONTROL_CONTACT_FLAG

/*прапори для транзисторних каналів*/

uint8_t DOBLE_ENABLE_FLAG[4];//дозвіл повторного ввімкнення транзистора

uint8_t NUMBER_OF_ENTER_FLAG[4];//повторне ввімкнення по каналам (0-не було вмикання після КЗ, 1- було вмикання після КЗ)

/*змінні для вимірів*/

uint16_t voltage_resault_V1[10];//результат виміру напруги на шині V1

uint16_t voltage_resault_V2[10];//результат виміру напруги на V2

float temperature;////результат виміру температури

/*змінні для вимірів струму*/

uint32_t current_resault[4];//результат виміру струму за 4 каналам в А

int32_t current[4]; //струм в каналах 0-3

int32_t Result[4][10];//змінна для підрахунку струму каналу (показання АЦП)

```
int32_t Result_sum[4];//сума останніх 10 значень АЦП для кожного з 4 каналов
int32_t Result_maj[4];//середнє арифметичне для кожного з 4 каналів за 10 вимірів

uint8_t timer_off_transistor[4];//час паузи на повторне ввімкненян транзистора після КЗ
/*
float current1; //струм 1 канала
float current2; // струм 2 канала
float current3; // струм 3 канала
float current4; // струм 4 канала
*/

uint32_t Timer_Number_KZ[4];//лічильник входу в переривання за КЗ від каналу

/*буфери запиту, відповіді і зберігання даних */
uint8_t RxBuf[401];//буфер запиту для DMA
uint8_t	TxBuf[32];//буфер відповіді
uint8_t	DataBuf[50]; //буфер даних прийнятих в запиті
uint8_t	OutBuf[32]; //буфер даних для відповіді
uint32_t CRC32_Otv; //CRC32 відповіді
uint32_t CntLength = 0; //довжина прийнятого запиту (байт)
uint16_t calc_byte;

/*змінні в запиті*/
uint16_t Length=0; //довжина прикладних даних повідомлення
uint8_t Adr=0;//адреса абонента (від 0 до 255)
uint8_t Type_abonent=0;//тип абоненту
uint8_t Type_message=0;//тип повідомлення: 1-запит, 0-відповідь
```

uint16_t Cnt_message=0; //номер запиту в межах 1s

uint16_t Cnt_message_old=65535; // номер запиту в межах 1s (попереднє значення)

uint8_t Service_data=0; //резерв=0

uint8_t Tr[16];//час режекції каналу i (i=0 до i=15)

uint8_t Td[16];//час затримки каналу i (i=0 до i=15)

uint8_t EnCh[16];//команди управління дискретними каналами

uint8_t Diag_Ch[4];//команди динмічної діагностики транзистора (0- не виконувати діагностику, 1 -виконувати)

//uint8_t number_ch=0;//номер каналу від 0 до 15

/*змінні в відповіді */

uint8_t ChSt[16];//стан вихідних дискретних каналів

uint8_t RlEr[16];//результат діагностики реле вихідних дискретних каналів

uint8_t SD[16];//стан дисрктених входів

uint8_t Doff[16];//стан дискртених входів відносно порогу відпускання

uint8_t U5V1;//напруга 5V1

uint8_t U5V2;// напруга 5V2

uint8_t U3V1;// напруга 3V1

uint8_t U3V2;// напруга 3V2

uint8_t ERRS_MODULE;//змінна ERRS_MODULE формулюється в процесі виконання самодіагностики

uint8_t ERRS_ASK;//змінна ERRS_ASK формулюється за результатами помилок

uint32_t Length_Otv;//довжина прикладних даних відповіді

/*змінні для дискретних виходів*/

uint8_t EnCh_tek[16];//поточне значення команди управління реле (транзистором)

uint8_t EnCh_pred[16];//попереднє значення команди управління реле (транзистором)

uint8_t EnCh_do_dinamic[4];// попереднє значення команди управління транзистором (до виконання команди динамічної діагностики)

uint16_t Time_delay_relay_ON[16];//час на ввімкнення реле

uint16_t Time_delay_relay_OFF[16];//час на вимкнення реле

uint8_t Control_relay_counter;//лічильник для вілдіку інтервалу контролю стану реле

uint8_t voltage_number_conversion_V1;//лічильник вмірів напруги для шини V1(після переходу з 3V на 5V або навпаки)

uint8_t voltage_number_conversion_V2;//лічильник виміру напруги для шини V2(після переходу з 3V на 5V або навпаки)

uint8_t NC_state[16];//стан NC післс контролю

uint8_t NO_state[16];//стан NO після контролю

uint8_t RlErr [3][16];//помилка стану реле (відмова реле) (3 значення для мажорування)

uint8_t Relay_Contr_number;// змінна для підрахунку номера каналу реле, що перевіряється (глобальна) змінюється в TIM3

uint8_t State_Relay_Control_port[16];// стан порту, що відповідає за опитування реле GPIOF_X

uint8_t RlErr_Counter[16];// лічильник помилкових станів контактів реле

uint8_t COUNTER_ASK_ER_CRC32_ER=2;//декремент лічильника помилкових (підбитих запитів) (початкове значення 2 встановлюється при видачі запиту)

/* змінні для формування біта ERRS_MODULE*/

uint8_t ErrROM=0;//помилка ЗСД

uint8_t ERR_Napr=0;// помилки напруги по шинам 5V1,5V2,3V1,3V2

/* змінні для дискретних входів */

uint8_t Doff_tek[16];// поточне значення порога відпускання

uint8_t Doff_pred[16];// попереднє значення порога відпускання

uint8_t Don_tek[16];// поточне значення порога спрацьовування

uint8_t Don_pred[16];// попереднє значення порога спрацьовування

uint16_t Trej[16];// час режекції взятий до виконання

uint16_t Time_delay_diskret[16];// час затримки дискретних каналів прийнятий до виконання

uint8_t Timeout_diskret[16];// тайм-аут після режекции на повернення каналу в норму

uint8_t Cnt_OFF_AC[16];// лічильник 100-мікросекундних включень каналу "поріг відпускання"

uint8_t Cnt_ON_AC[16];// лічильник 100-мікросекундних включень каналу "поріг спрацьовування"

uint8_t Number_RlErrDiskret[16];// лічильник помилок

uint16_t TrejOFF[16];// час протягом якого оптрон порогу відпускання був погашений під час проходження імпульсу режекції

uint16_t TrejOFF_TIME[16];// час протягом якого проходить імпульс режекции (половина цього часу порівнюється з тривалістю згаслого оптрона, для контролю справності каналу)

uint8_t RlErrDiskret [3][16];//(відмова дискретного входу) (3 значення для мажорірованія)

uint8_t RlErrDiskret_maj [16];// (Відмова дискретного входу)

uint8_t RlErrDiskret_optoreleV2 [16];// (Відмова оптореле дискретного входу)

uint16_t Time_opros_diskret=1;// період опитування дискретних каналів (1 * 100 = 100 мкс)

/* змінні для транзисторних виходів */

uint16_t Time_delay_optron_transistor[4];/ / Час на перемикання оптрона контролю транзистора (близько 500 мкс)

uint8_t Diag_transistor[4];// стан порту (оптрона) контролю транзистора (1-транзистор вимкнений, 0-транзистор ввімкнений)

uint8_t Dinamic_Diag_transistor[4];// стан порту (оптрона) контролю транзистора (1-транзистор вимкнений, 0-транзистор ввімкнений) під час динамічної діагностики

uint8_t DINAMIC_FLAG[4];// ознака того, що для даного каналу виконується динамічна діагностика

uint32_t dinamic_timer[4];// час на включення транзистора для динамічної діагностики (поканально)

uint8_t RlEr_dinamic[4];// результат контролю при динамічної діагностиці

uint8_t number_ch_dinamic;// номер каналу для якого необхідна динамічна діагностика

uint16_t Timer_Transistor_Control=200;// таймер для відліку часу контролю транзистора поканально

uint16_t RlErr_Counter_Tansistor[4];// лічильник помилкових станів контактів реле

uint8_t KZ1_interupt;// ознака КЗ в каналі 1 який встановлюється в перериванні і переписується для буфера відповіді кожні 100 мкс (щоб при перериванні від АЦП видачі пакета не підміняти значення КЗ)

uint8_t KZ2_interupt;// ознака КЗ в каналі 1 який встановлюється в перериванні і переписується для буфера відповіді кожні 100 мкс (щоб при перериванні від АЦП видачі пакета не підміняти значення КЗ)

uint8_t KZ3_interupt;// ознака КЗ в каналі 1 який встановлюється в перериванні і переписується для буфера відповіді кожні 100 мкс (щоб при перериванні від АЦП видачі пакета не підміняти значення КЗ)

uint8_t KZ4_interupt;// ознака КЗ в каналі 1 який встановлюється в перериванні і переписується для буфера відповіді кожні 100 мкс (щоб при перериванні від АЦП видачі пакета не підміняти значення КЗ)

uint32_t timer_off_NUMBER_OF_ENTER_FLAG[4];

uint32_t timer_off_NUMBER_OF_ENTER_FLAG_DISABLEKZ[4];// таймер на відключення КЗ при вимкненому каналі

uint32_t FLAG_OTKAZ_DT[4];// прапор того, що відмовив датчик струму або є КЗ в каналі але немає команди на включення каналу

uint8_t Otkaz_tranz_max[4];

uint8_t Otkaz_tranz[4];

float temp;// Напруга в вольтах на датчику (використовується для налагодження, для більш наочного сприйняття)

```
/* для перевірки IWDT*/
volatile uint16_t OtlWDT;// для перевірки WDT

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);

/* USER CODE BEGIN PFP */
/* Private function prototypes -----------------------------------------------*/
void USART1_Begin_Init(void);              // початкова ініціалізація USART1
void Initialization_Private_variables (void); // підпрограма початкової ініціалізації змінних
void TIM2_Start(void);                     //запуск TIM2 (відлік 1 ms)
void TIM3_Start(void);                     //запуск TIM3 (відлік 100 mks)
void TIM4_Start(void);                     //запуск TIM4 (відлік 10 ms)
void ObrZapros(void);                      // підпрограма обробки (аналізу) буфера запиту (викликається в перериванні по прийому від USART1)
void Write_Data_Buf (void);                // підпрограма занесення даних з буфера запиту в буфер даних
void FormBufOtvet(void);                   // підпрограма формування буфера відповіді (викликається в перериванні по прийому від USART1)
void ERRS_ASK_RESET(void);                 // підпрограма скидання помилок ERRS_ASK після видачі відповіді *
void ERRS_MODULE_RESET_SET(void);          // підпрограма скидання / установки помилок ERRS_MODULE (за результатами самодіагностики)
void RxDMA(void);                          //прийо по DMA
void TxDMA(void);                          //передача по DMA
void Init_GPIO_MODULE(void);    // ініціалізація портів введення виведення
void InitGPIO_DIDOM(void);      // ініціалізація портів введення виведення
```

void Read_Transistor_Command(void); // формування команди управління транзистором за результатами даних прийнятих в запиті

void Enable_Disable_Relay(void); // підпрограма управління реле (вкл-викл)

void Enable_Disable_Transistor(void); // підпрограма управління транзистором (вкл-викл)

void Control_Contact(void); // підпрограма опитування контактів релevoid Control_Relay(void); // підпрограма контролю стану реле

void Control_Transistor(void); // підпрограма опитування стану транзистора (в порядку поточної експлуатації)

void Dinamic_Control_Transistor(void); // підпрограма опитування стану транзистора (динамічний контроль за командою в запиті Diag_Chj = 1)

//void Delay_time(unsigned int delay); // підпрограма формування тимчасових затримок

void Control_Diskret(void); // підпрограма опитування дискретних каналів

static void CRC_Config(void); // підпрограма ініціалізації модуля CRC

uint32_t calc_CRC32(uint32_t* pAdr, uint32_t nmbr);// підпрограма підрахунку CRC апаратним методом

uint32_t test_full_ROM(void); // підпрограма підрахунку CRC ЗСД (всього ЗСД) апаратним методом

uint32_t test_part_ROM(void); // підпрограма підрахунку CRC ПЗУ (ПЗУ частинами) апаратним методом

//uint32_t crc32r(uint32_t* p, uint32_t len);// підпрограма підрахунку CRC по таблиці

//void Delay(unsigned int Val); // підпрограма формування часу затримки

void ADC_voltage_measurement (void); // підпрограма вимірювання напруги живлення

void ADC_temperature_measurement (void); // підпрограма вимірювання температури

void ADC_current_measurement (void); // підпрограма вимірювання струму в каналі

/* USER CODE END PFP */

```
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

 int main(void)
{

  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
```

```
  MX_GPIO_Init();
  MX_DMA_Init();
  MX_USART1_UART_Init();
  MX_ADC1_Init();
  MX_TIM3_Init();
  MX_TIM2_Init();
  MX_TIM4_Init();
  MX_IWDG_Init();
 // MX_ADC2_Init();

  /* USER CODE BEGIN 2 */
 /*
 if(MODULE==ONLY){
  MX_ADC2_Init();
 }

 */


  /* конфігуруємо CRC модуль */
  CRC_Config();//конфігуруємо CRC модуль для підрахунку CRC



  / * Перевіряємо контрольну суму всього ЗСД * /
  if (test_full_ROM()){//якщо результат контролю не 0 (підрахунок CRC32 (дані
+ КР) Не 0)
    ErrROM=1;}//помилка ЗСД
  else {
    ErrROM=0;}//немає помилки ЗСД
```

```
  Identification_module_type();
  Init_GPIO_MODULE();

  if(MODULE==ONLY
 MX_ADC2_Init();
}

 / * Виконуємо початкову ініціалізацію змінних * /
 Initialization_Private_variables ();
 / * Виконуємо початкову ініціалізацію USART1 * /
 USART1_Begin_Init();
/ * Запускаємо TIM2 для відліку 1 ms * /
 TIM2_Start();
 / * Запускаємо TIM3 для відліку 100 mks * /
 TIM3_Start();
 / * Запускаємо TIM4 для відліку 5 ms * /
 TIM4_Start();
/ * Налаштовуємо USART1 на прийом по DMA2 і дозволяємо прийом * /
 RxDMA();


 //11/07/19

 / * Запускаємо безперервне перетворення ADC2 для 4 транзисторних каналів
* /
 if(MODULE==ONLY
 ADC2_start_cont_mode ();
  }
```

```
//   ADC2_start_cont_mode ();//11/07/19

 /* USER CODE END 2 */

 /* Infinite loop */
 /* USER CODE BEGIN WHILE */

 // LL_GPIO_ResetOutputPin(GPIOG, LL_GPIO_PIN_11);//!!!!!!!

 while (1)
 {
  / * Визначення тривалості циклу * /
 //LL_GPIO_SetOutputPin(GPIOG, LL_GPIO_PIN_11);
  //LL_GPIO_TogglePin(GPIOG, LL_GPIO_PIN_11);

  /*перескидання IWDG*/

   IWDG->KR=0x0000AAAA;

/ * Формуємо байт ERRS_MODULE за результатами самодіагностики * /
  ERRS_MODULE_RESET_SET();

  / * Якщо встановлений прапор RX_START_FLAG налаштовуємо USART1 на прийом по DMA2 і дозволяємо прийом * /
 if (RX_START_FLAG==SET_FLAG){// якщо встановлений прапор перемикання на прийом
    RX_START_FLAG=RESET_FLAG;//сбросить флаг скинути прапор
    RxDMA();//налаштуватися на прийом запустити прийом по DMA
    }
 / * Опитування дискретних каналів модуля * /
```

```
  Control_Diskret (); // опитування дискретних каналів модуля
   /*
  if(CONTROL_DISKRETE_FLAG==SET_FLAG){// перевірка стану дискретних входів
  Control_Diskret();
  Time_opros_diskret=1;// задати період опитування дискретних каналів (1 * 100 = 100 мкс)
  CONTROL_DISKRETE_FLAG=RESET_FLAG;}// закінчення перевірки стану дискретних входів
  */
  ADC1->SQR3=0x00000002;// вибрати 2й канал АЦП
  ADC_voltage_measurement ();//вимір напруги живлення
 if ( (voltage_resault_V1[0] < U5Vmin || voltage_resault_V1[0] > U5Vmax) && (voltage_resault_V1[1] < U5Vmin || voltage_resault_V1[1] > U5Vmax) && (voltage_resault_V1[2] < U5Vmin || voltage_resault_V1[2] > U5Vmax) ){// напруга на шині 5V (для трьох поспіль вимірювань) не входить в допустимий діапазон
  ERR_Napr|=Err5V1;// помилка вимірювання напруги на шині 5V1
    }// помилка вимірювання 5V
   else{// напруга на шині 5V не входить в допустимий діапазон
     ERR_Napr&=~Err5V1;// скидання помилки вимірювання 5V на шині 5V1
     }// скидання помилки вимірювання 5V
    if (voltage_resault_V1[0]<U5Vmax) {// якщо значення напруги не досягло максимуму то потрібно порахувати
      U5V1=(voltage_resault_V1[0]/22);}// значення у відповіді (одиниця молодшого розряду = 22 mV)
    else // якщо більше або дорівнює 5610 mV то видати у відповіді просто 255
      U5V1=0xFF;// вся шкала 255 значень
 }
 / * Управління та контроль вихідних дискретних (релейних) каналів
/ * Управління реле по команді прийнятої в запиті * /
```

```
/ * всі реле включаються в перериванні по USART1 щоб забезпечити час 200 мкс від отримання команди

 // Enable_Disable_Relay(); // включається в перериванні по USART1 щоб забезпечити час 200 мкс від отримання команди до видачі упр. впливу на реле

/ * Контроль реле кожну 1ms * /

  if (CONTROL_RELAY_FLAG==SET_FLAG){// перевірка стану реле кожну 1 ms

    //LL_GPIO_TogglePin(GPIOG, LL_GPIO_PIN_11);

    Control_Relay();

    CONTROL_RELAY_FLAG=RESET_FLAG;}// закінчення перевірки стану реле

  } //end of if(MODULE==ONLY)

    if(MODULE==ONLY)

  / * Опитування дискретних каналів модуля * /

  if (CONTROL_DISKRETE_FLAG==SET_FLAG){// перевірка стану дискретних входів кожні 200 mks

   // LL_GPIO_TogglePin(GPIOG, LL_GPIO_PIN_11);//

    Control_Diskret();

    Time_opros_diskret=1;// задати період опитування дискретних каналів (2 * 100 = 200 мкс)

    CONTROL_DISKRETE_FLAG=RESET_FLAG;}// закінчення перевірки стану дискретних входів

  } //end of if(MODULE==ONLY)


  / * Контроль температури * /

  if (CONTROL_TEMPERATURE_FLAG==SET_FLAG){// перевірка температури кожну 1 ms

   ADC_temperature_measurement (); // вимір температури

   CONTROL_TEMPERATURE_FLAG=RESET_FLAG;}// закінчення перевірки температури  /*

  /* USER CODE END WHILE */
```

```
  /* USER CODE BEGIN 3 */

  }
  /* USER CODE END 3 */

}

/** System Clock Configuration
*/
void SystemClock_Config(void)
{

  LL_FLASH_SetLatency(LL_FLASH_LATENCY_5);

  if(LL_FLASH_GetLatency() != LL_FLASH_LATENCY_5)
  {
  Error_Handler();
  }
  LL_PWR_SetRegulVoltageScaling(LL_PWR_REGU_VOLTAGE_SCALE2);

  LL_PWR_DisableOverDriveMode();

  LL_RCC_HSE_EnableBypass();

  LL_RCC_HSE_Enable();

   /* Wait till HSE is ready */
  while(LL_RCC_HSE_IsReady() != 1)
  {
```

```
  }
  LL_RCC_LSI_Enable();

   /* Wait till LSI is ready */
  while(LL_RCC_LSI_IsReady() != 1)
  {

  }
  LL_RCC_PLL_ConfigDomain_SYS(LL_RCC_PLLSOURCE_HSE,
LL_RCC_PLLM_DIV_19, 249, LL_RCC_PLLP_DIV_2);

  LL_RCC_PLL_Enable();

   /* Wait till PLL is ready */
  while(LL_RCC_PLL_IsReady() != 1)
  {

  }
  LL_RCC_SetAHBPrescaler(LL_RCC_SYSCLK_DIV_1);

  LL_RCC_SetAPB1Prescaler(LL_RCC_APB1_DIV_4);

  LL_RCC_SetAPB2Prescaler(LL_RCC_APB2_DIV_16);

  LL_RCC_SetSysClkSource(LL_RCC_SYS_CLKSOURCE_PLL);

   /* Wait till System clock is ready */
  while(LL_RCC_GetSysClkSource()                                    !=
LL_RCC_SYS_CLKSOURCE_STATUS_PLL)
```

```
  {
  }
  LL_Init1msTick(163815789);

  LL_SYSTICK_SetClkSource(LL_SYSTICK_CLKSOURCE_HCLK);

  LL_SetSystemCoreClock(163815789);

LL_RCC_SetUSARTClockSource(LL_RCC_USART1_CLKSOURCE_SYSCLK)
;

  /* SysTick_IRQn interrupt configuration */
  NVIC_SetPriority(SysTick_IRQn,
NVIC_EncodePriority(NVIC_GetPriorityGrouping(),0, 0));
}

/* USER CODE BEGIN 4 */

/**********************************************************************
************/
/********** Підпрограма запуску безперервного перетворення по ADC2
************
/**********************************************************************
************/
void ADC2_start_cont_mode (void)
{

  ADC2->CR2 = ADC_SOFTWARE_START;// Перетворення регулярної групи
запуститься установкою біта SWSTART (рівносильно ADC2-> CR2 =
ADC_CR2_EXTSEL; в регістрі CR2 параметр EXTSEL = 0xF)
```

ADC2->CR1 |=ADC_CR1_JAUTO;// Дозволити перетворення інжектовані групи після регулярної. Перетворення інжектування групи може відбуватися після перетворення хоча б одного регулярного каналу (так працює МК)

ADC2->CR1 |= ADC_CR1_SCAN;// Ввімкнтути режим сканування. Тобто, виконується опитування певного числа каналів в заданій послідовності. записаних в ADC_JSQR register (for injected channels) або ADC_SQR register (for regular channels)

ADC2->CR2 |= ADC_CR2_CONT;// Перетворення запускаються одне за іншим (continuous) без зупинки

ADC2->CR2 |= ADC_CR2_ADON;// Тепер включаємо АЦП (переклад АЦП з режиму енергозбереження)

ADC2->CR2 |= ADC_CR2_SWSTART;// Запуск перетворень (програмний)

// ADC2->CR2 |= ADC_CR2_JSWSTART;// Запуск перетворень (програмний)

}

/********** Підпрограма обробки прийнятого запиту **********

//**********/

void ObrZapros(void)

{

// LL_GPIO_ResetOutputPin(GPIOG, LL_GPIO_PIN_11);

// uint8_t i;

// uint8_t k=0;

// uint16_t calc_byte;

//CntLength=13;

calc_byte=DMA2_Stream2-> NDTR;

//CntLength=RxCnt;

CntLength=401-calc_byte;

//End_zapros=RxCnt=0;

```
 if (CntLength >=12 && CntLength  <= 400 && CntLength% 4 == 0){// довжина запиту поза діапазону 12-400 байт

    ASK_ER_FLAG=RESET_FLAG;// скинути прапор невірної довжини запиту
//    if(crc32r((uint32_t*)RxBuf, CntLength)==0){  // підрахунок і перевірка контрольної суми

     if(calc_CRC32((uint32_t*)RxBuf, CntLength)==0){ // підрахунок і перевірка контрольної суми апаратно

       CRC32_ER_FLAG=RESET_FLAG; // скинути прапор невірної КС запиту

       Adr=RxBuf[2];

       Cnt_message=(RxBuf[4]|RxBuf[5]<<8);//читаємо CNT

    //if (calc_CRC32((uint32_t*)RxBuf[0], CntLength)==0){

       Type_message=(RxBuf[3]>>7&0x01);// читаємо тип повідомлення (1-запит, 0-відповідь)

     if(Type_message==0x01){//прийнято запит (запит 1 відповідь 0)

       Type_abonent=RxBuf[3]&0x7F;// читаємо тип абонента

       if (Type_abonent==MODULE){// Тип абонента відповідає типу модуля

       //Cnt_message=(RxBuf[4]<<8|RxBuf[5]);// читаємо CNT

          Cnt_message=(RxBuf[4]|RxBuf[5]<<8);// читаємо CNT

       if (Cnt_message!=Cnt_message_old){// якщо немає помилки номера запиту в межах 1 s

       Cnt_message_old=Cnt_message;

       ERRS_ASK&=~CNT_ER;// скинути біт CNT_ER

       //Length=(RxBuf[0]<<8|RxBuf[1]);// читаємо довжину прикладних даних повідомлення

       Length=(RxBuf[0]|RxBuf[1]<<8);// читаємо довжину прикладних даних повідомлення

      // if ((Length==Length_Zapros && (CntLength-12) >=Length_Zapros) || Length==0){// якщо довжина прикладних даних запиту вірна ((значення L прийняте в запиті дорівнює 32 байтам (довжина запиту) і прийнято не менше 32 байт прикладних даних) або довжина прикладних даних нульова)

       if ((Length==Length_Zapros && (CntLength-12)==Length_Zapros) || Length==0){// якщо довжина прикладних даних запиту вірна ((значення L
```

прийняте в запиті дорівнює 32 байтам (довжина запиту) і прийнято 32 байта прикладних даних) або довжина прикладних даних нульова)

FULL_OTVET_FLAG=SET_FLAG; // встановити прапор видачі відсвіту з прикладними даними

Length_Otv=16;// встановити довжину прикладних даних у відповіді рівну 16

TIME_OUT_5ms=50;// оновити лічильник відліку тайм-ауту 5 ms

if (Length!=0){// якщо довжина прикладних даних не нульова

NEW_DATA_FLAG=SET_FLAG; // встановити прапор нових прикладних даних

}//end if Length!=0

/*

else {// довжина прикладних даних не нульова

FULL_OTVET_FLAG=SET_FLAG; // встановити прапор видачі відповіді з прикладними даними

Length_Otv=16;// встановити довжину прикладних даних у відповіді рівну 16

TIME_OUT_5ms=50;// оновити лічильник відліку тайм-ауту 5 ms

}

*/

}//end if Length

else {

ERRS_ASK|=L_ER;

FULL_OTVET_FLAG=RESET_FLAG; // скинути прапор видачі відповіді з прикладними даними (видати відповідь без прикладних даних)

Length_Otv=0;// встановити довжину прикладних даних у відповіді рівну 0

}// встановити біт L_ER

```
        }//end if Cnt_message
        else{
          ERRS_ASK|=CNT_ER;// встановити біт CNT_ER
        Cnt_message_old=Cnt_message; скинути прапор видачі відсвіту з прикладними даними (видати відповідь без прикладних даних)
        FULL_OTVET_FLAG=RESET_FLAG; // скинути прапор видачі відповіді з прикладними даними (видати відповідь без прикладних даних)
        Length_Otv=0;// встановити довжину прикладних даних у відповіді рівну 0
        }
       }//end if Type_abonent
       else{// невірний тип абонента (відповідь без прикладних даних)
          ERRS_ASK|=TYPE_ER;
        FULL_OTVET_FLAG=RESET_FLAG; // скинути прапор видачі відсвіту з прикладними даними (видати відповідь без прикладних даних)
        Length_Otv=0;// встановити довжину прикладних даних у відповіді рівну 0
          }
     }// end if Type_message
     else {
       ERRS_ASK|=Z_ER;// встановити біт Z_ER (прийнятий не запит)
      FULL_OTVET_FLAG=RESET_FLAG; // скинути прапор видачі відсвіту з прикладними даними (видати відповідь без прикладних даних)
      Length_Otv=0;// встановити довжину прикладних даних у відповіді рівну 0
      }//встановити прапор формування відповіді без прикладних даних
   }//end if crc32r
   else {//CRC32_ER
     if(COUNTER_ASK_ER_CRC32_ER!=0)// якщо не 0
      {
```

--COUNTER_ASK_ER_CRC32_ER;// декремент лічильника помилкових (підбитих запитів) (початкове значення 2 встановлюється при видачі запиту)

}

else// тільки якщо два рази підбили довжина або контролю сума встановлюємо ознака CRC32_ER

{

ERRS_ASK|=CRC32_ER;

}

CRC32_ER_FLAG=SET_FLAG; // встановити прапор невірної КС запиту

RX_START_FLAG=SET_FLAG;// встановити прапор перемикання на прийом

//RxDMA();

}//установить бит CRC32_ER (Помилка КС)

} //end if CntLength

/********** Підпрограма вимірювання струму в каналах ********** /

void ADC_current_measurement (void)

{

uint32_t i; //індекс

uint32_t j; //індекс

uint32_t voltage_IC[4]; // напруга на виході датчика струму

/* int32_t Result[4][10];// змінна для обчислень струму каналу 1

int32_t Result_sum[4];// сума останніх 10 значень для кожного з 4 каналів

int32_t Result_maj[4];// середнє арифметичне для кожного з 4 каналів

* /

// переписуємо попередні значення АЦП

for (i=0; i<4; i++){//канали 0-3 (1-4)

for (j=9; j>0; j--){// переписуємо останні 10 значень АЦП

Result[i][j]=Result[i][j-1];// 9-му привласнити 8, 8-му привласнити 7, і т д. 1-му привласнити 0, а нульовий переписується нижче

```
   }// end  for (j=9; j>0; j--)
  }// end for (i=0; i<4; i++)



 //читаем текущие значения АЦП
       Result[0][0] = ADC2->JDR1;// значення струму каналу 1 зчитані з АЦП
       Result[1][0] = ADC2->JDR3;// значення струму каналу 2 зчитані з АЦП
       Result[2][0] = ADC2->JDR2;// значення струму каналу 3 зчитані з АЦП
       Result[3][0] = ADC2->JDR4;// значення струму каналу 4 зчитані з АЦП


 // вважаємо середнє значення АЦП за останні 10 вимірювань
  for (i=0; i<4; i++){//канали 0-3  (1-4)
    for (j=0; j<10; j++){// останні 10 значень АЦП
      Result_sum[i]+=Result[i][j];// сума останніх 10 значень
      }
     Result_maj[i]=Result_sum[i]/10;// середнє арифметичне за останні 10 вимірювань
     Result_sum[i]=0;// обнулити суму
  }
  // зчитаємо значення струму в каналі
  for (i=0; i<4; i++){//канали 0-3  (1-4)
    if (current_resault[i]!=0x0F){


   //current[i] = (float) (((Result[i]*3300)/4096)-330)/66;    // значення струму в каналі
   // current[i] = ((((Result[i]*3300)/4096)-330)*1000)/66;    // значення струму в каналі в мА
   voltage_IC[i] = ((Result_maj[i]*3300)/4096);    // напруга на виході датчика в MV
   if (voltage_IC[i] <330){
```

```
      voltage_IC[i]=330;// захист від негативних чисел (в разі плавання нуля датчика струму)
     }
   current[i] = ((voltage_IC[i]-330)*1000)/66;   // значення струму в каналі в мА
   current_resault[i]=((current[i]+500)/1000)/2; // округлене значення струму в А тобто до результату додається 500 мА (тим самим округляючи його в більшу сторону, потім розподіл на 1000 для отримання результату в А
     } // end of if (current_resault[i]!=0xFF){
  }// end of for (i=0; i<4; i++)
}// end of void ADC_current_measurement (void)
/*****************************************************************************/
/************ Підпрограма формування затримки ****************************** /
/*****************************************************************************/
/*
  void Delay(unsigned int Val)
     {
      for( ; Val != 0; Val--)
      // if(Val != 0; Val--)

      __NOP();//1mks это Val=50
     }

*/
/********** Підпрограма формування тимчасових затримок ************************ /
/*void Delay_time(unsigned int delay)
{
  uint32_t counter;
```

```
   // LL_GPIO_ResetOutputPin(GPIOG, LL_GPIO_PIN_11);

    counter = ( delay * (SystemCoreClock / 1000000));
    while(counter != 0)
    {
      counter--;
    }

  //  LL_GPIO_SetOutputPin(GPIOG, LL_GPIO_PIN_11);
  }//end of Delay(uint_8t delay)
*/
/* USER CODE END 4 */
/**
  * @brief  This function is executed in case of error occurrence.
  * @param  None
  * @retval None
  */
void _Error_Handler(char * file, int line)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state */
  while(1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}

#ifdef USE_FULL_ASSERT
```

```
/**
  * @brief Reports the name of the source file and the source line number
  * where the assert_param error has occurred.
  * @param file: pointer to the source file name
  * @param line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t* file, uint32_t line)
{
 /* USER CODE BEGIN 6 */
 /* User can add his own implementation to report the file name and line number,
   ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
 /* USER CODE END 6 */

}
#endif
```

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Баскаков С.И. Радиотехнические цепи и сигналы. М.: Высшая школа, 1983. - 536 с

2. Дианов В.Н. Диагностика и надежность автоматических систем. - М.: МГИУ, 2005. - 160 с

3. Корнеев В.В., Киселев А.В. Современные микропроцессоры. - 3-е изд. - СПб.: БХВ-Петербург, 2003. - 448 с

4. Сергиенко А.Б. Цифровая обработка сигналов. СПб.: Питер. - 608 с.

5. Фрайден Дж. Современные датчики. Справочник. - М.: Техносфера, 2005. - 592 с.