МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет

Факультет електроніки та інформаційних технологій

Кафедра електроніки і комп'ютерної техніки

«До захисту допущено»
Завідувач кафедри ЕКТ

_____ Анатолій ОПАНАСЮК
(підпис) (Ім’я та ПРІЗВИЩЕ)
____ _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня «магістр»

зі спеціальності 171 «Електроніка»

освітньо-професійної програми «Електронні системи та компоненти»

на тему:

ЕЛЕКТРОННА СИСТЕМА РОЗПОДІЛЕНОГО ЗБОРУ ТА ЗАХИСТУ ІНФОРМАЦІЇ НА ОСНОВІ ПОСЛІДОВНИХ КАНАЛІВ ЗВ’ЯЗКУ

Здобувача групи ЕС.м-21 _____ Лобанова Андрія Олександровича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ _____
(підпис) (Ім’я та ПРІЗВИЩЕ)

Керівник, доцент, к.т.н., доцент Ігор КУЛИК _____
(підпис)

Консультант з техніко-економічної частини,
доцент, к.е.н., доцент Олександр МАЦЕНКО _____
(підпис)

Суми – 2023

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Факультет електроніки та інформаційних технологій

Кафедра електроніки і комп'ютерної техніки

Напрям підготовки 171 «Електроніка»

Освітня програма Електронні системи та компоненти

ЗАТВЕРДЖУЮ

Зав. кафедрою Опанасюк А. С.

"___"_____ 2023 р.

З А В Д А Н Н Я

на кваліфікаційну роботу магістра

1. Тема роботи _____

затверджена наказом по університету "06" листопада 2023 р. № 1233-VI.

2. Термін здачі студентом завершеної роботи _____

3. Вихідні дані до роботи _____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що належить розробити) 1) Огляд літератури та поставлення задачі роботи. 2) Науково-дослідна частина. 3) Розробка електронної системи з використанням отриманих результатів дослідження. 4) Техніко-економічна частина.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) 1) Схема електрична структурна. 2) Схема алгоритму. 3) Схема електрична функціональна. 4) Схема електрична принципова.

6. Консультанти з кваліфікаційної роботи

Розділи	Консультанти	Завдання видав	Завдання прийняв
Техніко-економічна частина	Маценко О. М.		

7. Дата видачі завдання _____

8. Керівник роботи _____

9. Завдання прийняв до виконання _____

КАЛЕНДАРНИЙ ПЛАН

п/п	Назва етапів дипломного проекту	Термін виконання етапів роботи	Примітки
1	Огляд літератури та постановка завдання проектування	06.11.23 – 13.11.23	
2	Науково-дослідна частина	14.11.23 – 21.11.23	
3	Розробка алгоритму функціонування та структурної схеми електронної системи	22.11.23 – 29.11.23	
4	Розробка функціональної схеми електронної системи	30.11.23 – 04.12.23	
5	Розробка схеми електричної принципової електронної системи	05.12.23 – 12.12.23	
6	Техніко-економічна частина	13.12.23 – 14.12.23	
8	Оформлення пояснювальної записки	15.12.23 – 16.12.23	
9	Оформлення графічного матеріалу	17.12.23 – 18.12.23	
10	Представлення роботи керівнику і отримання відгуку	19.12.23	
11	Представлення роботи кафедрі для отримання рецензії	19.12.23	

Студент _____

Керівник роботи _____

«___» _____ 2023 р.

РЕФЕРАТ

Пояснювальна записка: 94 аркуші; 37 малюнків; 16 джерел.

Графічні матеріали: 1 схема алгоритм, 1 схема електрична структурна, 1 схема електрична функціональна, 1 схема електрична принципова.

У першому описується актуальність роботи, наводиться результати науково-дослідної роботи та на основі отриманих даних проектується пристрій.

У другому розділі проводиться розрахунок собівартості проектованого устрою та економічний афект.

Ключові слова: передача даних, послідовний інтерфейс, протокол передачі Modbus, циклічна сума, мікроконтролер, контрольна сума CRC-16.

ЗМІСТ

СПИСОК УМОВНИХ СКОРОЧЕНЬ

АСУТП – автоматизована система управління технологічним процесом
БВІ – блок відображення інформації
БПІ – блок підрахунку імпульсів
БСК – блок статичних конденсаторів
БУД – блок управління двигуном
БУК – блок управління конденсаторами
ВІС – велика інтегральна схема
ГПП – головні понижуючі підстанції
ГТІ – генератор тактових імпульсів
ДРП – джерело реактивної потужності
ІС – інтегральна схема
МК – мікроконтролер
МП – мікропроцесор
МПС – мікропроцесорна система
ОЗП – оперативне запам'ятувальний пристрій
ПІП – програмований інтерфейсній пристрій
рПЗП – репрограмований постійний запам'ятувальний пристрій
СД – синхроний двигун
СПД – система передачі даних
СПД – система передачі даних
ТН – трансформатор напруги
ТТ – трансформатор струму
ЦП – центральний процесор
ША, ШД, ШУ – шини адреси, даних, управління відповідно

ВВЕДЕННЯ

Широке використання систем автоматичного управління, підкріплене розвитком архітектури та елементної бази мікропроцесорів та мікроЕОМ, веде до організації комплексних систем автоматики, що мають розподілену структуру. До класу таких систем можуть бути віднесені великі робототехнічні комплекси, автоматизовані технологічні лінії та ділянки, цехи-автомати, гнучкі автоматизовані виробництва. Усі названі класи САУ характеризуються жорсткими вимогами щодо надійності та високої інтенсивності міжмодульного обміну інформацією. Як модулі при цьому можуть виступати мікроЕОМ, їх різні блоки (обробки, відображення, зберігання даних), виконавчі механізми, системи збирання інформації тощо.

При проектуванні систем передачі даних з початкових завдань є вибір інтерфейсу передачі даних і протоколу передачі.

Простота та низькі апаратні вимоги (порівняно з паралельними інтерфейсом) роблять інтерфейс RS-485 привабливим вибором для побудови систем обміну інформацією.

Практично всі комп'ютери у промисловому виконанні оснащені засобами організації інформаційного обміну з використанням даних інтерфейсів. Сучасні інтелектуальні датчики елементи управління поряд з традиційним інтерфейсом RS-232C також можуть мати у своєму складі підсистему послідовного введення-виведення інформації на базі інтерфейсу RS-485. Програмовані логічні контролери багатьох виробників як засоби організації територіально-розподілених систем збору даних та управління містять ту чи іншу реалізацію інтерфейсів RS-422/RS-485.

Одним з найпоширеніших протоколів передачі даних, що застосовуються в промислових мережах, є промисловий протокол Modbas та його модифікації. Причина популярності Modbas полягає у відкритості системи, гнучкій довжині пакету та різні можливості реалізації.

При виборі протоколу передачі у промислових мережах основні параметри вибору – швидкість передачі і переданих даних.

При написанні кваліфікаційної роботи основною метою науково-дослідної частини є аналіз та порівняння сімейства протоколів Modbas, а саме Modbas RTU

та Modbas ASCII а також аналіз споживаної обчислювальної потужності та час реалізації промислових протоколів передачі даних.

Результати науково-дослідної частини реалізуємо при проектуванні розподіленого модуля збору та передачі даних з обраним протоколом передачі даних.

У техніко-економічній частині розраховуються собівартість пристрою та економ ефект, проводиться порівняльний аналіз проектованого пристрою та пристрою аналога.

1 ОСНОВНА ЧАСТИНА

1.1 Огляд літератури та постановка завдання дослідження

Вітчизняний і зарубіжний досвід переконливо доводить, що розвиток підприємств металургійного, енергетичного та хімічного комплексу, вирішення проблем якості та конкурентоспроможності продукції на світовому ринку вимагають докорінного вдосконалення систем збирання, зберігання, обробки, передачі та використання інформації.

Сучасні тенденції розвитку промисловості характеризуються розробкою, впровадженням та широким використанням інформаційних систем та технологій. Основою інформаційних технологій та систем є комп'ютери та комп'ютерні мережі, що мають найбагатше програмне забезпечення, а також системи управління базами даних, комп'ютерні системи підтримки прийняття рішень, методологічною основою яких є теорія систем та системний аналіз, теорія моделювання даних, технологічних процесів та знань.

1.1.1 Загальні відомості.

Інтерфейс RS-485 - широко поширений високошвидкісний і завадостійкий промисловий послідовний інтерфейс передачі даних. Практично всі сучасні комп'ютери в промисловому виконанні, більшість інтелектуальних датчиків і виконавчих пристроїв, програмовані логічні контролери поряд з традиційним інтерфейсом RS-232 містять у своєму складі ту чи іншу реалізацію інтерфейсу RS-485. Інтерфейс RS-485 базується на стандарті EIA RS-422/RS-485. Головна відмінність RS-485 від також широко поширеного RS-232 - можливість об'єднання декількох пристроїв. Інтерфейс RS-485 забезпечує обмін даними між кількома пристроями по одній двопровідній лінії зв'язку в напівдуплексному режимі. Широко використовується у промисловості під час створення АСУ ТП.

Інтерфейс RS485 найбільше часто використовується при створенні сучасних локальних мереж різного призначення, як у промислових виробах, так і в аматорській практиці. Основними перевагами інтерфейсу є:

- щодо низька собівартість мікросхем драйверів, що знижує вартість апаратної реалізації мережевих диспетчерів, тобто. вузлів зв'язку між

мережевим середовищем (лініями зв'язку) та ядром станції (вузла) мережі, тобто. мікроконтролерною або мікропроцесорною системою;

- Використання в мережах на базі інтерфейсу RS485 всього трьох проводів (третій, загальний, не завжди є обов'язковим), що значно знижує собівартість всієї системи, оскільки відомо, що собівартість мережного середовища сучасних локальних мереж практично завжди становить більше 60 % від вартості всієї системи;

- Мікросхеми драйверів мають малі габаритні розміри. Найбільш часто використовуються мікросхеми, виконані в корпусі DIP8 зі стандартним розташуванням висновків, що стали de facto, промисловим стандартом. Мікросхеми драйверів використовують лише декілька дискретних елементів для ланцюгів захисту, використання яких не є обов'язковим. Малі габаритні розміри мікросхем драйверів та мінімальна кількість обв'язки заощаджує площу друкованої плати, що також позитивно позначається на вартості системи;

- сучасні мікросхеми мають досить низьке енергоспоживання, багато хто з них за відсутності активності в мережі автоматично переходить у режим економії, що знижує енергоспоживання системи;

- сучасні мікросхеми драйверів мають підвищену здатність навантаження. Якщо рання більшість мікросхем було нараховано працювати з 32 станціями, то сучасні моделі забезпечують нормальне функціонування до 256 станцій;

- В даний час випускаються мікросхеми у високій граничній швидкістю передачі. Це дозволяє створювати високошвидкісні мережі, і знижує кількість помилок в мережі за рахунок поліпшення форми сигналу, що передається;

- драйвери інтерфейсу RS485 мають досить просте керування. Особливості організації мереж, їх схемотехніка, способи управління доступом до каналу та приклади програмування досить описані.

- мікросхеми інтерфейсу RS485 випускають багато фірм світу. Однак безперечним лідером у розробці та випуску нових мікросхем драйверів є відома фірма MAXIM. В даний час фірма випускає більше 80 типів мікросхем драйверів інтерфейсу RS485/422.

Мережа, побудована на інтерфейсі RS-485, є прийомопередавачі, з'єднані за допомогою крученої пари - двох скручених проводів. В основі інтерфейсу RS-485 лежить принцип диференціальної (балансної) передачі даних. Суть його полягає у передачі одного сигналу по двох дротах.

Причому з одного дроту (умовно А) йде оригінальний сигнал, а з іншого (умовно В) - його інверсна копія. Інакше кажучи, якщо одному проводі " 1 " , то іншому " 0 " і навпаки. Таким чином, між двома проводами крученої пари завжди є різниця потенціалів: при "1" вона позитивна, при "0" - від'ємна.

Саме цією різницею потенціалів і передається сигнал. Такий спосіб передачі забезпечує високу стійкість до синфазної перешкоди. Синфазною називають перешкоду, що діє на обидва дроти лінії однаково. Наприклад, електромагнітна хвиля, проходячи через ділянку лінії зв'язку, наводить в обох проводах потенціал. Якщо сигнал передається потенціалом в одному дроті щодо загального, як в RS-232, то наведення на цей провід може спотворити сигнал щодо добре поглинає загального наведення ("землі"). Крім того, на опорі довгого загального дроту падатиме різниця потенціалів зе-мель - додаткове джерело спотворень. А при диференціальній передачі спотворення не відбувається. Справді, якщо два дроти пролягають близько один до одного, та ще перевиті, то наведення на обидва дроти однакове. Потенціал в обох однаково навантажених проводах змінюється однаково, при цьому інформаційна різниця потенціалів залишається без змін.

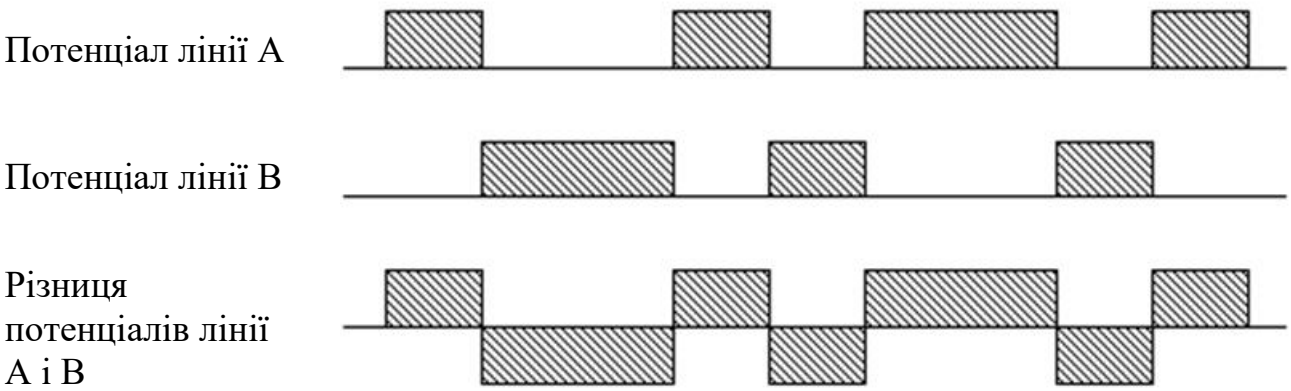

Рисунок 1.1 – Принцип диференціальної передачі даних

Пристрої, що підключаються до інтерфейсу RS-485, характеризуються важливим параметром входу приймача: “одиниця навантаження” (“Unit Load” - UL). За стандартом у мережі допускається використання до 32 одиниць навантаження, тобто. до 32 пристроїв, кожен з яких навантажує лінію 1 UL. В даний час існують мікросхеми приймачів з характеристикою менше 1 UL, наприклад - 0,25 UL. У цьому випадку кількість фізично підключених до лінії

пристрої можна збільшити, але сумарна кількість UL в одній лінії не повинна перевищувати 32.

При великих відстанях між пристроями, пов'язаними по кручений парі і високих швидкостях передачі починають проявлятися так звані ефекти довгих ліній. Наука Електротехніка пропонує вирішення цієї проблеми. Будь-яка лінія зв'язку має такий параметр, як хвильовий опір Zв. Воно залежить від показників використовуваного кабелю, але з довжини. Для зазвичай застосовуваних лініях зв'язку кручених пар Zв=120 Ом. Виявляється, що якщо на віддаленому кінці лінії, між провідниками крученої пари включити резистор з номіналом рівним хвильовому опору лінії, то електромагнітна хвиля дійшла до "глухого кута" поглинається на такому резистори. Звідси його назви - резистор, що узгоджує, або "термінатор". Великий мінус погодження на резисторах - підвищене споживання струму від передавача, адже до лінії включається низькоомне навантаження. Тому рекомендується включати передавач тільки на час надсилання посилки. Є способи зменшити споживання струму, включаючи послідовно з резистором, що узгоджує, конденсатор для розв'язки по постійному струму. Однак такий спосіб має свої недоліки. Для коротких ліній (кілька десятків метрів) та низьких швидкостей (менше 38400 бод) погодження можна взагалі не робити.

На малюнку 1.2 зображено принцип підключення приймачів і структура локальної мережі побудованої на основі інтерфейсу RS-485.

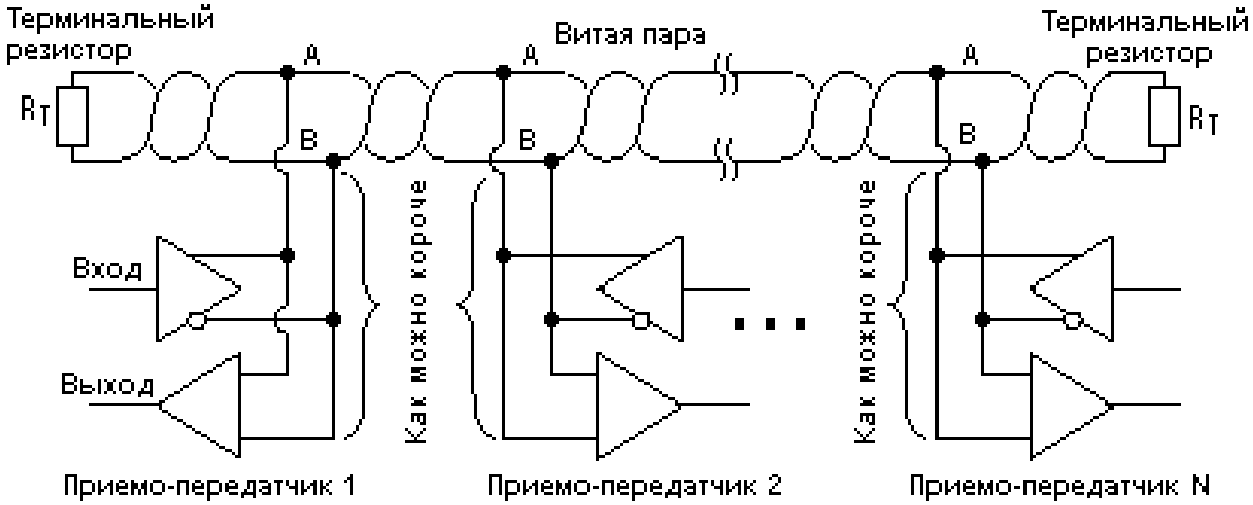

Рисунок 1.2 – Локальна мережа на основі інтерфейсу RS-485

Параметри інтерфейсу RS – 485 приведені в таблиці 1.1.

Таблиця 1.1 – Параметри інтерфейсу RS – 485

Параметр	Значення
Допустиме число передавачів / приймачів	32 / 32
Максимальна довжина кабелю	1200 м
Максимальна швидкість зв’язку	10 Мбіт/с
Діапазон навантаження "1" передавача	+1.5 ...+6 В
Діапазон навантаження "0" передавача	-1.5...-6 В
Діапазон синфазного навантаження передавача	-1...+3 В
Допустимий діапазон навантажень приймача	-7...+12 В
Пороговий діапазон чутливочті приймача	±200 мВ
Максимальний струм короткого замикання драйвера	250 мА
Допустимий опір навантаження передавача	54 Ом
Вхідний опір приймача	12 кОм
Максимальний час наростання сигналу передавача	30% біта

При підключенні слід правильно приєднати сигнальні ланцюги, звані A і B. Переполюсовка не страшна, але пристрій працювати не буде. Мережа має бути прокладена по топології шини, без відгалужень. Пристрої слід підключати до кабелю дроти мінімальної довжини.

Віта пара є оптимальним рішенням для прокладки мережі, оскільки має найменше паразитне випромінювання сигналу і добре захищене від наведень. В

умовах підвищених зовнішніх перешкод застосовують кабелі з екранованою крученою парою, при цьому екран кабелю з'єднують із захисною "землею" пристрою.

За природою інтерфейсу RS-485 пристрої не можуть передавати одночасно - буде конфлікт передавачів. Отже, потрібно розподілити право на передачу між пристроями. Звідси основний поділ: централізований (одномастерний) обмін і децентралізований (багатомайстерний).

У централізованій мережі один пристрій завжди провідний (майстер). Воно генерує запити і команди іншим (відомим) пристроям. Ведені пристрої можуть передавати лише за командою ведучого.

Як правило, обмін між веденими йде лише через ведучого, хоча для прискорення обміну можна організувати передачу даних від одного веденого до іншого за командою ведучого.

У децентралізованій мережі роль ведучого може передаватися від устрою до пристрою або за деяким алгоритмом черговості, або за командою поточного ведучого до наступного (передача маркера ведучого). При цьому ведомий пристрій може у своїй відповіді ведучому передати запит на перехід в режим ведучого і чекати дозволу або заборони.

Інтерфейс RS-485 не прив'язаний до конкретних протоколів передачі даних, на даному інтерфейсі можна реалізувати такі протоколи передачі даних як ProfiBus DP, ModBus, CAN, DMX512, CAN, CANopen, DCON.

1.1.2 Модулі розподіленої системи збору даних.

Все більше поширення на підприємствах Росії та України при використанні сучасної ідеології автоматизації різних технологічних процесів отримують моду ADAM фірми «Advantech».

Модулі серії ADAM 5000 – це модулі розподілених систем бору даних та управління. Вони призначені для побудови територіально розподілених систем збору даних та управління, забезпечують виконання наступних функцій:

- аналогове введення-виведення;
- дискретне введення-виведення;
- первинне перетворення інформації;
- ADAM 5000 здійснює прийом команд від віддаленої обчислювальної

системи та передачі на її адресу перетворених даних з використанням інтерфейсу RS-485.

Фірма «Advantech» виробляє моделі інтелектуальних УСО серії ADAM 5000 – моделі ADAM 5510, яка, крім усього іншого, є вільно програмованим IBM-сумісним мікроконтролером.

Програмований мікроконтролер ADAM 5510 призначений для використання в локальних і розподілених системах автоматизації в якості автономного контролера. Він забезпечує прийом і видачу аналогових і дискретних сигналів, первинне перетворення сигналів за запрограмованим користувачем алгоритмів та обмін інформацією по послідовним каналам зв'язку на базі інтерфейсу RS-485. Приклад комплектування системи на базі модулів серії ADAM 50x представлений на рис. 1.3.

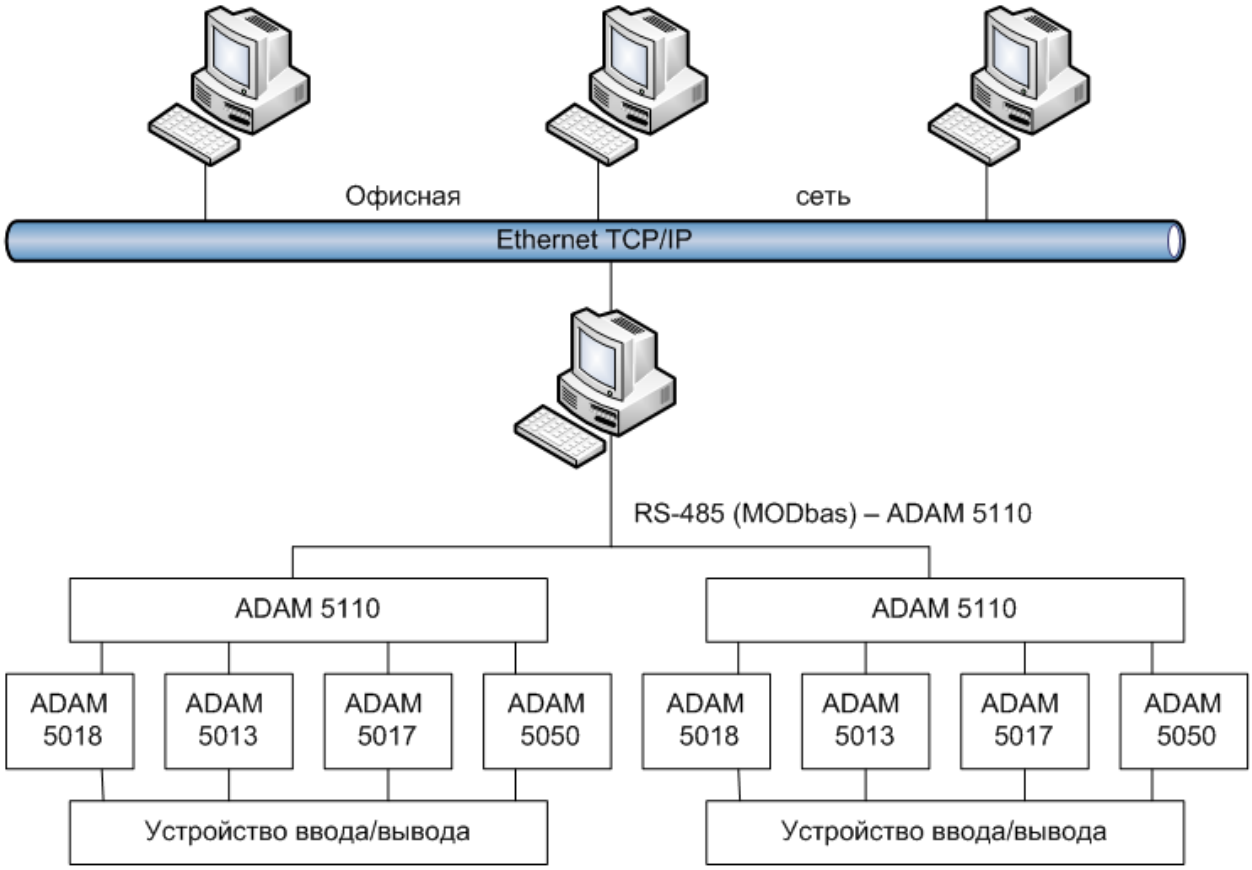

Рисунок 1.3 – Інформаційна система на базі модулів ADAM 50XX

Поява модуля ADAM 5510 дозволило розташувати керуючу програму користувача у пам'яті контролера, що скоротило час реакції у контурі управління віртуально до 5–10 мкс. Цей факт дозволив застосувати системи на базі ADAM 5510 для вирішення задач жорсткого реального часу.

Контролер має відкриту архітектуру і може програмуватися за допомогою традиційних мов програмування (С, асемблер), а також функціональних блокових діаграм. Для цього є можливість підтримки пакета Ultralogik (FBD - Function Block Diagram, мова функціональних блокових діаграм, стандарт МЕК 61131-3).

Ultralogik - це система розробки програмного забезпечення збору даних та управління для промислових контролерів. Програмуючий на Ultralogik може не знати нічого про внутрішній пристрій ADAM 5510. Для нього важливо розуміти які модулі вводу-виводу присутні в його контролері, і на які лінії в них приходять ті чи інші сигнали.

У Ultralogik входять бібліотеки для всіх модулів введення-виведення ADAM 5510, бібліотеки алгоритмів збору даних та управління (ПІ-закони, ПІД-закони і т.д.), передбачена можливість підключення функцій мовами С, С++, Паскаль, Асемблер.

ADAM 5018. 8-канальний модуль введення для підключення термопар;

ADAM 5013. 3-канальний модуль введення для підключення термометрів опору;

ADAM 5017. 8-канальний модуль аналогового введення;

ADAM 5050. 16-канальний універсальний модуль дискретного вводу-виводу.

1.1.3 Постановка завдання дослідження.

У цій дипломній роботі необхідно провести дослідження та порівняльний аналіз промислових протоколів збору та передачі даних, реалізованих на послідовному інтерфейсі. Провести аналіз обчислюваної потужності споживаної при реалізації промислових протоколів.

Спроектувати модуль розподіленого збору даних, зі швидкістю передачі даних 3 Мбіт по каналу зв'язку, реалізованому на інтерфейсі RS-485 і промисловим протоколом передачі даних ModBas для забезпечення обміну даних з відомими контролерами.

Даний модуль має бути оснащений двома COM-портами для зв'язку з персональним комп'ютером і модемом і двома Ethernet каналами з системою 10Base-T для клієнта серверного свята пристрою з сервером зі швидкістю

передачі даних рівною 10 Мбіт/С. Провести розрахунки та Ethernet каналу на ефективність використання за наявності в каналі 10 клієнтів.

1.2 Науково-дослідна частина

1.2.1 Протоколи передачі.

Найчастіше при асинхронної передачі файлів на ПК базова структура пакета (чи кадру) складається з груп полів. Тільки одне з цих полів містить фактичні дані. Інші поля, звані службовими, містять інформацію, необхідну приймачеві для підтвердження того, що пакет не містить помилок.

Протоколи автоматичного запиту повторної передачі (ARQ)

Найбільш загальним типом пакетного протоколу є протокол автоматичного запиту повторної передачі (ARQ), при використанні якого в пакетах визначається наявність помилки, і відсутність підтвердження правильності пакета автоматично призводить до його повторної передачі.

Принцип передачі і очікування ARQ полягає в тому, що приймач отримує пакет і після того, як він перевірить правильність його номера по відношенню до попереднього, обчислює приватну контрольну суму для даних цього пакета. У разі відповідності контрольній сумі, зазначеній у пакеті, приймач видає підтвердження у вигляді АСК (або він посилає NAK). Коли передавач приймає АСК, він передає наступний пакет.

Принцип безперервного ARQ (ковзне вікно) полягає в тому, що передавач посилає кілька пакетів поспіль без затримки між пакетами. Приймач разом з номером пакета посилає NAC або АСК (як і у разі передачі та очікування ARQ).

Передавач безперервно перевіряє потік підтверджень, що повертаються, і відстежує пакети з помилками. Наприкінці передачі пакети, у яких було виявлено помилки, передаються повторно.

Протокол передачі та очікування ARQ залишається найпоширенішим протоколом передачі файлів для ПК. Для передачі файлів використовуються два типи протоколів - XMODEM і Kermit.

Протокол XMODEM. XMODEM є простим протоколом передачі та очікування ARQ, який використовує поле даних фіксованої довжини. Контрольне число є однобайтною арифметичною контрольною сумою. Структура пакета зображена на рис. 1.4.

Початок заголовку	Номер поточного пакету	Доповнення до 1 номеру поточного пакету	Дані (128 байт)	Арифметична контрольна сума

Рисунок 1.4 – Структура кадру протоколу XMODEM

Значення початок заголовка – SOH у двійковому коді ASCII. Значення поля номер поточного пакета приймає значення 256 і є доповненням до першого пакета. Довжина даних є фіксованою і може утримувати як двійкові дані, так і слова двійкового коду ASCII. Арифметична контрольна сума обчислюється для поля даних методом розподілу за модулем 226 і має один байт.

Коротко алгоритм роботи протоколу XMODEM можна представити таким чином:

- приймач посилає передачу символ NAK, щоб ініціювати передачу;

- потім передавач посилає об'єднаний пакет 128-байтний блок даних. Символ АСК, прийнятий передавачем, починає передачу наступного пакета, а символ CAN скасовує передачу;

- коли всі дані будуть передані, відправник передає єдиний символ EOT, який приймач підтверджує символ АСК.

Інші версії протоколу XMODEM використовують замість однобайтової контрольної суми (XMODEM-CRC) однобайтовий контроль за допомогою циклічного надлишкового коду (CRC). Це незворотний алгоритм CRC з CCITT по-ліноміальним дільником наведено у формулі 1.1:

$$П = X^{16} + X^{12} + X^5 + 1, \qquad (1.1)$$

де П – CCIT поліном,

Х – константа одиниці.

Проблеми протоколу XMODEM:

- Протокол XMODEM призначений для каналів зв'язку з низькими частотами появи помилок. Якщо модем не має вбудованого алгоритму виправлення помилок, то XMODEM не працюватиме. Більш високі швидкості

передачі даних і погана якість каналу зв'язку (що пропонують дешевші послуги) роблять проблему ще гострішою;

- Лінійні перешкоди, що призводять до численних змін бітів, часто проходять через просту систему виявлення помилок протоколу XMODEM і залишаються непоміченими;

- прості символи управління (АСК, NAK, EOT) часто псуються, що призводить до передачі інших символів і, відповідно, до помилкових дій і більших тимчасових витрат. Якщо в результаті псування символу виходить символ Control-X, передача файлу припиняється;

- ковзні вікна, що є ефективнішими під час використання протоколу, у цій ситуації використовувати не можна, оскільки символи відповіді NAK чи АСК немає відповідного порядкового номера. Це сприяє неефективній роботі комунікаційної системи з великими часом затримки;

- Для створення пакета символів потрібні 8 бітів, що не дозволяє використовувати їх в комунікаційних системах, які відводять на кожен символ тільки 7 бітів.

Великою перевагою протоколу XMODEM є те, що він передбачений всіма поширеними комунікаційними програмами (такими, як Windows Terminal). Цей стандарт de facto корисний для передачі файлів між різними несумісними комп'ютерними системами, які мають лише протокол XMODEM.

Слід зазначити, що протокол XMODEM часто використовується просто для перенесення більш ефективного протоколу передачі файлів (такого, як ZMODEM) з однієї машини на іншу. Результатом може бути велика економія часу і, відповідно, оплата за час зв'язку.

Протокол YMODEM. Цей протокол був введений як поліпшення протоколу XMODEM і набув дуже широкого поширення, оскільки став загальним надбанням і був написаний мовою С.

Функціональні можливості протоколу YMODEM:

- протокол YMODEM має менше службових витрат, ніж XMODEM, оскільки його кадр містить 1024 байта, які можна передавати одним блоком. Протокол YMODEM досить гнучкий і дозволяє зменшити розмір блоку даних до 128 байт, якщо частота появи помилок дуже велика (що призводить до численних повторних передач);

- протокол YMODEM надійніший, ніж XMODEM. Щоб припинити передачі файлу, повинна бути прийнята послідовність символів: Control-X, Control-X (або два символи CAN). Це запобігає ненавмисним припинення передачі файлу внаслідок помилок на лінії;

- для забезпечення високого ступеня виявлення помилок використовується контроль за допомогою циклічного надлишкового коду;

- інформація, пов'язана з файлом (ім'я, час, дата та розмір файлу), також передається на комп'ютер, що приймає;

- можна передавати кілька файлів, використовуючи можливість протоколу здійснювати групову передачу;

- В основі передачі файлу лежать 8-розрядні дані. Це призводить до деяких проблем, обговорених раніше для протоколу XMODEM;

- Для передачі пакетів даних використовується принцип очікування ARQ. Неможливість використання адекватного ковзного вікна спричиняє неефективність системи з великими тимчасовими затримками.

Протокол ZMODEM. Цей протокол був розроблений для подолання обмежень, що є в попередніх протоколах передачі даних.

Характерні особливості протоколу ZMODEM:

- крім групового перенесення файлів, що є стандартною можливістю;

- Корисною особливістю є те, що ZMODEM може починати повторну передачу файлу з тієї точки, в якій стався обрив зв'язку (з точністю до байта, на якому була порушена зв'язок);

- протокол ZMODEM не буде робити передачу файлу, якщо такий файл вже існує на жорсткому диску комп'ютера, що приймає. Ця перевірка проводиться протоколом ZMODEM автоматично і дозволяє заощадити час;

- проводиться краще виявлення та корекція помилок, що реалізується використанням механізму CRC-32;

- Можна стискати дані - це збільшує швидкість передачі даних;

В основі протоколу Kermit лежить пакетний протокол передачі та очікування ARQ. Передавач передає пакет, а потім чекає від приймача підтвердження прийняття пакета. Приймач може запросити наступний пакет (символ АСК), або повторну передачу попереднього пакета (символ NAK).

Хоча протокол Kermit багато в чому схожий з протоколом XMODEM, але має важливі відмінності:

– Kermit може передавати кілька файлів за одну сесію передачі файлів;

- Пакети можуть мати різну довжину;

- канали введення/виводу повинні передавати лише друковані символи ASCII;

- Визначено кілька типів пакетів;

- відповіді приймача повинні складатися з пакетів;

- передавач та приймачі «домовляються» про важливі робочі параметри, такі як заповнююча інформація тощо;

- Назва файлу включається до протоколу;

- можна стискати дані - це збільшує швидкість передачі даних;

- Повідомляючі пакети роблять протокол відкритим.

Робоча послідовність протоколу Kermit починається аналогічно протоколу XMODEM, яким приймач передає повторні NAK пакети до тих пір, поки передавач не відповість відправленням повідомлення пакету, названого в протоколі Kermit ініціюючим пакетом. Приймач повідомляє про свої переваги шляхом їхнього включення в пакет АСК. Коли весь файл буде передано, відправник передає спеціальний пакет закінчення файлу. Якщо є ще файли, які потрібно передати, він передає заголовковий пакет наступного файла. Коли всі файли будуть передані, передавач надсилає пакет закінчення передачі, що сигналізує про закінчення сесії. Даля пакета, що передається, може змінюватися в залежності від вимог приймача і передавача при цьому довжина пакета даних вказана в полі - довжина пакета.

Пакет протоколу Kermit зображено на рис 1.5.

Мітка	Довжина пакету	Номер пакету	Тип	Дані	Перевірка 1	Перевірка 2	Перевірка 3

Рисунок 1.5 - Пакет протоколу Kermit

Порівняльні характеристики типів передачі файлів та атрибутів протоколів наведені в таблиці 1.2.

Таблиця 1.2 – Порівняння протоколів передачі файлів

Опис		**Xmodem**	**Xmodem**	**Xmodem**	**Kermit**
Типи передачі файлів	Одиночний файл	Так	Так	Так	Так
	Група файлів	Ні	Так	Так	Так
	Текст/дані/двійкова інформація	Так	Так	Так	Так
	Кількість бітів даних	8	8	8	7 или 8
	Стиснення даних	Ні	Ні	Так	Так
	Контроль за помилками	Контрольна сума	Контрольна сума/CRC-CCITT	CRC-CCITT/ CRC-32	Контрольна сума/CRC-CCITT
	Відповідь при контролі за помилками	Один символ	Один символ	Один символ	Пакетний контроль за помилками
	Розмір пакету даних(байти)	128	1024	1024	0-95
Передача атрибутів файлів	Розмір/час/дата файлу	Ні	Так	Так	Так
	Підтримка ковзаючого вікна	Один пакет	Один пакет	1 або декілька пакетів	1 або декілька пакетів
	Параметри згоди	Ні	Так	Так	Так
	Пакети в Windows	Ні	Ні	Так	Так
	Тип контролю за помилками	Ні	Так	Так	Так
	Стиснення даних	Ні	Ні	Так	Так
	Контроль XON/XOFF	Ні	Ні	Так	Так

1.2.2 Промисловий протокол Modbus.

Протокол Modbus набув широкого поширення в сучасних системах автоматики. Специфікація протоколу не визначає тип фізичного рівня мережі передачі даних, залишаючи вибір за розробником.

На практиці широкого поширення набула зв'язка, що складається з мережі RS-485 в якості фізичного середовища передачі даних і протоколу Modbus як логічний рівень мережі. Таке рішення дозволяє використовувати Modbus у найширшому спектрі мікроконтролерних пристроїв, починаючи з найпростіших, таких як різні інтелектуальні датчики, і закінчуючи складними розподіленими системами на базі програмованих логічних контролерів.

Протокол побудований за схемою «ведучий-відомий» (master-slave). У системі виділяється один провідний пристрій (майстер), який ініціалізує будь-яку транзакцію у мережі. Всі інші пристрої є відомими та виконують команди майстра або передають інформацію у відповідь на запит майстра. Протокол передбачає два типи адресації: індивідуальну, коли повідомлення адресується одному веденому пристрою, і широкомовну (broadcast messages), при якій повідомлення адресується всім пристроям мережі. При індивідуальній адресації ведений пристрій повертає майстру повідомлення у відповідь, при широкомовній адресації відповідні повідомлення не надсилаються.

Повідомлення, надіслані майстром, мають таку структуру: адресу відомого пристрою (або код широкомовного повідомлення), код, що визначає дії веденого пристрою, дані та контрольна сума. Повідомлення-повідомлення складається з поля, що підтверджує виконання дії, даних і контрольної суми. Якщо при прийомі повідомлення від майстра відбулася помилка, чи ведений пристрій з яких-небудь причин не може виконати запитуваний дію, ведений пристрій посилає майстру повідомлення про помилку.

Існує також розширена версія протоколу, яка називається Modbus Plus. Протокол Modbus Plus формує одну рангову мережу, в якій будь-який пристрій може ініціалізувати транзакцію і, таким чином, кожен пристрій може виступати в ролі, як ведучого, так і веденого в різних транзакціях.

Мережа Modbus може працювати в одному з двох режимів: RTU та ASCII. У режимі RTU інформація передається «як є», в двійковому коді. У режимі ASCII

інформація передається у текстовому вигляді як послідовність символів '0'–'9', 'A'–'F' в ASCII-кодуванні.

Вибір режиму RTU або ASCII, а також установки мережі, такі як швидкість передачі, біт парності тощо, вибираються користувачем при конфігуруванні контролерів і повинні бути однакові для всіх пристроїв в мережі.

Режим ASCII Головною перевагою даного режиму є те, що символи можуть передаватися з інтервалом аж до однієї секунди без виникнення помилки передачі даних. Недоліком даного режиму є його знижена (більш ніж удвічі) інформаційна пропускна здатність порівняно з режимом RTU за рівної швидкості фізичної лінії. Передача символу здійснюється в наступному форматі: 1 стартовий біт, 7 біт даних (молодший розряд передається першим), 1 біт контролю парності (або непарності; за відсутності контролю парності даний біт відсутній), 1 стоповий біт, якщо контроль парності присутній, 2 стопові біти за відсутності контролю парності.

Повідомлення в даному режимі починається з символу двокрапки (код 3Ah), закінчується послідовністю символів "повернення каретки", "переведення рядка" (CRLF, код 0D0Ah). Між передачею символів можливі інтервали часу до 1 секунди. При перевищенні тайм-ауту пристрій, що приймає, фіксує помилку передачі.

У деяких ранніх реалізаціях протоколу повідомлення закінчується контрольною сумою, без послідовності CRLF. Таким чином, пристрій, що приймає, повинен почекати як мінімум 1 секунду після прийому контрольної суми, і якщо послідовність CRLF не отримана, повідомлення вважається успішно прийнятим.

Режим RTU. Передача символу в режимі RTU складається з наступних етапів: 1 стартовий біт, 8 біт даних (молодший розряд передається першим), 1 біт контролю парності (непарності), за відсутності контролю парності даний біт відсутній, 1 стоповий біт, якщо контроль парності присутній,

стопова біта за відсутності контролю парності.

Повідомлення являє собою послідовність символів, що передаються безперервно, без пауз. При виникненні паузи тривалістю більше 1,5 t, де t-час передачі одного символу при заданій швидкості передачі, пристрій приймає фіксує помилку в прийомі повідомлення і починає прийом нового повідомлення

після паузи. Повідомлення випереджається і закінчується паузою не менше 3,5 t при типовому значенні 4 t.

Аналіз завадостійкості. У даному протоколі передачі даних використовується метод формування контрольної суми циклічними надлишковими кодами CRC16. Розмір контрольної суми 16 біт. Даний алгоритм дозволяє тільки визначити наївність помилки, а виправлення передачі запиту та повторного передачі повідомлення.

Імовірність, що кодова відстань між прийнятим і переданим повідомленнями довжиною 64 біта дорівнює d розраховується за формулою 1.2:

$$p(d) = p_l^d (1 - p_l)^{64-d}, \qquad (1.2)$$

де p_l – ймовірність спотворення одного біта.

Нехай $\Pr_{CRCerr}(d)$ - ймовірність збігу контрольної суми CRC-16 в інформаційному слові, що знаходиться на відстані d від інформаційного слова, що передається. Тоді повна ймовірність виявлення помилки контролем перевірочних символів CRC16 розраховується по формуле 1.3:

$$\Pr_{CRCerr}(p_l) = \sum_{d=2}^{p_l - n} p_l^d (1 - p_l)^{64-d} \Pr_{CRCerr}(d), \qquad (1.3)$$

де p_l - ймовірність спотворення одного біта,

$\Pr_{CRCerr}(d)$ - ймовірність помилки, що не виявляється.

Імовірність оцінена за допомогою моделювання та дорівнює $1{,}9 \cdot 10^{-5}$.

Порівняльний аналіз режимів. Відмінності описаної структури та характеристик режимів передачі RTU та ASCII протоколу Modbus у розділах 1.2 мають важливі наслідки:

Наслідок 1. Оскільки для передачі одного байта інформації в режимі ASCII потрібно передати по лінії зв'язку два символи (тобто два байти), режим ASCII приблизно вдвічі повільніший, ніж режим RTU (при однаковій швидкості UART).

Наслідок 2. Через те, що в режимі ASCII існують особливі «стартові» та «стопові» символи, програма може фіксувати початок і кінець передачі з набагато вищим ступенем достовірності, ніж у режимі RTU, де єдиною ознакою кінця передачі служить невелика пауза.

Наслідок 3. Оскільки в режимі RTU паузи між передачею окремих бай-тів у повідомленні неприпустимі, це може викликати деякі труднощі в програмній реалізації даного режиму. Наприклад, якщо передача повідомлення перервана будь-яким обробником переривання, пристрій приймач повідомлення визнає цю паузу кінцем передачі з усіма наслідками, що звідси випливають у вигляді розбіжності контрольної суми, тоді всю транзакцію доведеться починати заново. В результаті в програму доводиться вводити додаткові механізми, покликані забезпечити безперервність передачі повідомлення. У ряді випадків достатньо заблокувати переривання на час передачі повідомлення, якщо цього зробити не можна, можна відкласти передачу на тимчасовий проміжок, в якому гарантовано не може виникнути переривання з тривалою обробкою, здатної порушити процес передачі повідомлення.

Ми бачимо, що обидва режими мають свої переваги та недоліки, і неможливо однозначно рекомендувати застосування одного з них «на всі випадки життя». Найкращим рішенням буде підтримка в приладі, що розробляється, обох режимів протоколу Modbus.

1.2.3 Аналіз споживаної обчислювальної потужності.

При формуванні та обробці пакета даних згідно з різними промисловими протоколами передачі даних основний час і обчислювальна потужність, як приймача, так і передавача йде на обчислення та перевірку контрольної суми. Для протоколів, які під час трансакції формують довгих повідомлень, поширеним і дуже ефективним механізмом перевірки повідомлення на наявність помилки є контроль за допомогою надлишкового циклічного коду. CRC заснований на використанні понять в галузі математики, званої теорією алгебри. Механізм контролю за допомогою циклічного надлишкового коду, очевидно, дуже ефективний у виявленні помилок, особливо «пакетних помилок», коли зовнішні перешкоди тимчасово пригнічують сигнал і псують цілий рядок бітів. CRC контроль ефективний для повідомлень будь-якої довжини.

Існують два основних методи розрахунку CRC – побітовий та табличний. Їхні основні характеристики наведені в таблиці 1.3.

Контрольні суми різняться за надійністю контролю цілісності даних. Чим більша розрядність суми, що розраховується, тим вона надійніша, але і тим складніше її реалізація. При розрахунку можуть використовуватися різні поліноми, які зазвичай обумовлюються стандартом використовуваного інтерфейсу.

Таблиця 1.3 - Характеристики методів розрахунку

Метод розрахунку	Переваги	Недоліки
Побітовий	Невеликі затрати пам’яті	Низькашвидкість виконання
Табличний	Більш висока швидкість виконання	Більші затрати пам’яті

Реалізація бітового алгоритму. Схема алгоритму представлена на малюнку 1.6. Вихідний код, що відповідає даному алгоритму, наведено в додатку.

На малюнку 1.6 для кожного блоку проставлено час виконання операцій у тактах процесора. Підпрограма розрахунку контрольної суми приймає два параметри:

- довжину повідомлення, яка передається словом через два регістри LENL і LENH, де LENL - молодший, а LENH - старший байт слова;

- покажчик на повідомлення, який передається через пару адресних реєстрів XH та XL.

Розрахована контрольна сума зберігається у регістрах SUMH і SUML, де SUMH – старший, а SUML – молодший байт суми. У побітовому алгоритмі час розрахунку залежить від вхідних даних. Тому при аналізі часу розрахунку береться два випадки найгірший (виконується операція XOR з поліномом) та найкращий (операція XOR не виконується).

На практиці можна стверджувати, що дані будуть оброблені не повільніше, ніж у найгіршому випадку, і не швидше, ніж у найкращому разі. Виділимо постійну і змінну тимчасові складові виконання алгоритму. Постійна складова визначається накладними витратами при реалізації підпрограми - завантаження/вивантаження зі стека, ініціалізація початкових значень змінних, інверсія суми по закінченні розрахунку. Змінна складова визначається реалізацією алгоритму. Дані про тимчасові витрати в тактах процесорного часу наведені в таблиці 1.4.

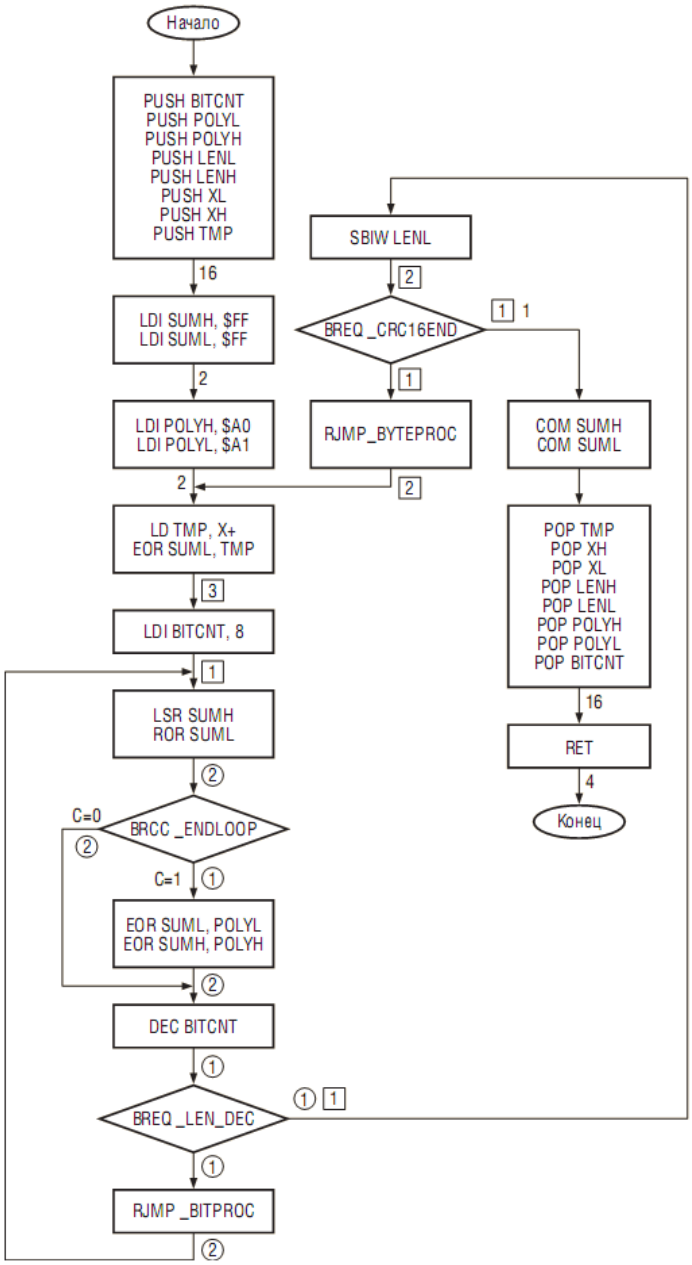

Рисунок 1.6 – Блок-схема алгоритму побітового розрахунку CRC-16

Таблиця 1.4 – Тимчасові витрати при реалізації побітового алгоритму

Розмір повідомлення	Найкращий випадок			Найгірший випадок		
	Тактів в циклі обробки байту	Всього тактів	Тактів на один байт	Тактів в циклі обрабки байту	Всього тактів	Тактів на один байт
10	64	117,0	117,0	72	125	125,0
20	128	191,0	95,5	144	207	103,5
40	256	339,0	84,8	288	371	92,8
80	512	635,0	79,4	576	699	87,4
160	1024	1227,0	76,7	1152	1355	84,7
320	2048	2411,0	75,3	2140	2667	83,3

Блок-схема алгоритму розрахунку контрольної суми CRC16 табличним методом представлена малюнку 1.7.

З блок-схеми алгоритму видно, що його виконання залежить від кількості байт оброблюваного повідомлення не залежить від їх значення. При розрахунку табличний спосіб використовуються теж параметри, що і в підпрограмі розрахунку послідовним методом. Дані про тимчасові витрати в тактах процесорного часу наведені в таблиці 1.5.

Таблиця 1.5 - Тимчасові витрати при реалізації табличного алгоритму

Розмір повідомлення	Тактів поза циклом	Такті в в циклі	Всього тактів	Тактів на один байт
1	29	18	47	47,0
2	29	36	65	32,5
4	29	72	101	25,3
8	29	144	173	21,6
18	29	288	317	19,8
32	29	576	605	18,9

Дані про використовувану Flash-пам'ять та SRAM-пам'ять при реалізації табличного та побитного алгоритму наведені в таблиці 5.

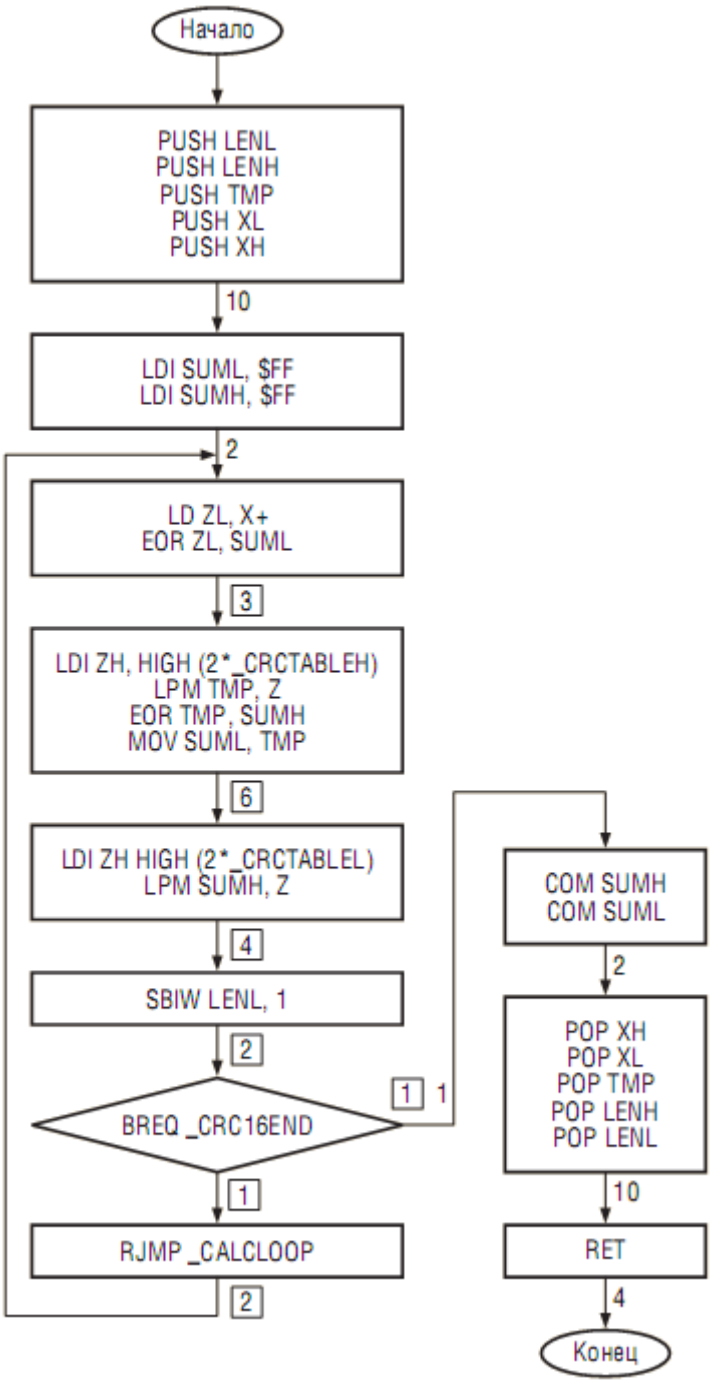

Рисунок 1.7 – Блок-схема табличного алгоритму

Порівняння швидкості виконання алгоритму. Для порівняння швидкості розрахунку побудуємо графіки, що відображають залежність часу виконання розрахунку в процесорних тактах (рис. 1.8) і мілісекунд при тактовій частоті 16 МГц (рис. 1.9) від кількості оброблюваних байт повідомлення.

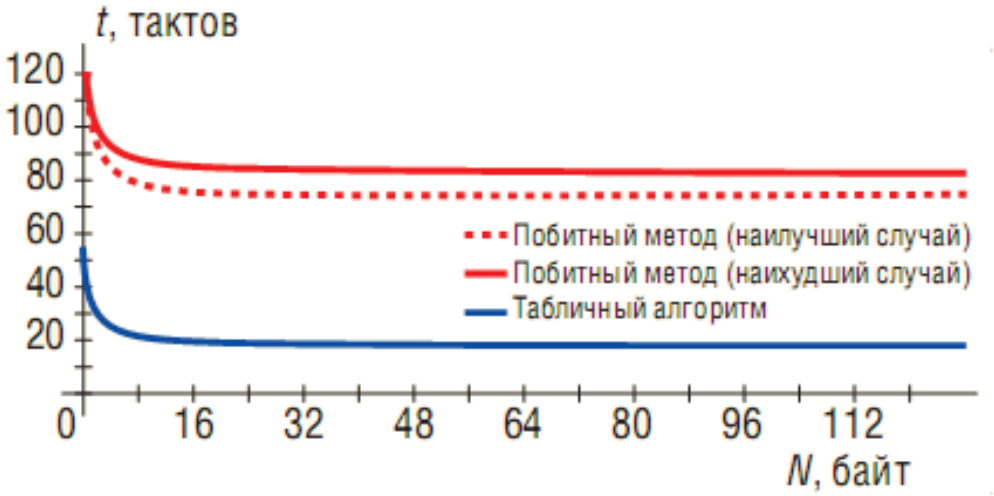

Рисунок 1.8 – Порівняння часу виконання алгоритмів (у тактах)

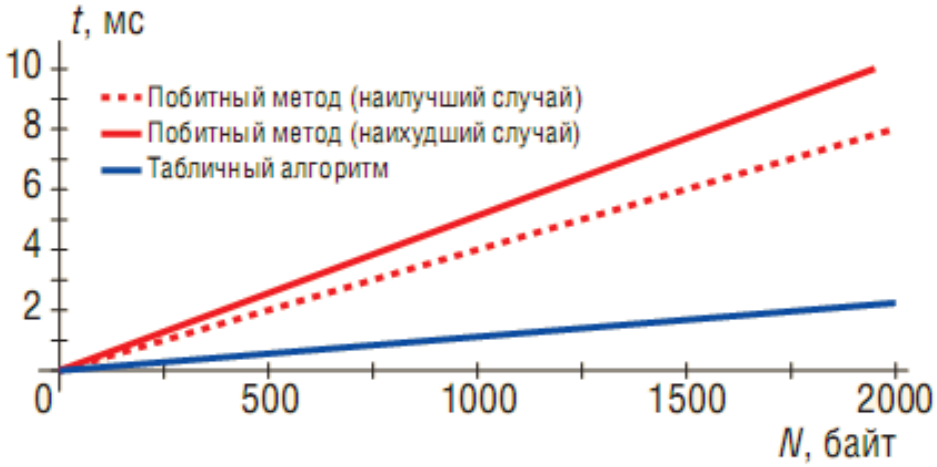

Рисунок 1.9 – Порівняння часу виконання алгоритмів (у мілісекундах)

Таблиця 1.7 - Порівняння обсягів пам'яті

Алгоритм	Flash-пам’ять, байт	SRAM, байт
Побітовий	74	0
Табличний	564	0

Проаналізувавши отримані дані, можна зробити такі висновки:

1. табличний алгоритм перевищує побітовий як мінімум в 2,5 рази за швидкістю виконання при розмірі повідомлення в 1 байт і в 4 рази при розмірі повідомлення більше 32 байт;

2. табличний алгоритм займає на 490 байт більше пам'яті програм, ніж побітовий;

3. зважаючи на детермінованість часу виконання підпрограми розрахунку контрольної суми табличним алгоритмом, її можна використовувати як затримку на задану величину.

Таким чином, використання табличного алгоритму при підрахунку контрольної суми CRC програмним способом є більш ефективним навіть при використанні молодших моделей мікроконтролерів серії AVR.

1.3 Розробка електронного пристрою з використанням отриманих результатів дослідження

1.3.1 Обґрунтування алгоритму функціонування і структурної схеми проекту, що проектується.

Значення полів і структура повідомлень передають по каналу зв'язку, а також алгоритми функціонування приймачів описується стандартами протоколів передачі даних. Вибір конкретного рішення залежить від архітектури системи, відстані, числа контрольованих точок, вимоги до пропускної спроможності і надійність каналу. Сьогодні є десятки різноманітних промислових протоколів передачі даних. Найбільш популярними є Industrial Ethernet, Profibus, Controller Area Net Work Bus (CANbus), MAP/TOP, Allen Bradley Data Highway, EthernetIP.

Протокол передачі Modbus був розроблений компанією Gould Modicon (тепер AEG) для систем управління процесами. Розробник може вибирати між інтерфейсами RS-422, RS-485, Ethernet або струмовою петлею 20 мА, кожен з яких підходить для швидкостей передачі даних, що визначаються протоколом. Хоча протокол Modbus є відносно повільним порівняно з іншими шинами, він має перевагу в тому, що широко поширений серед виробників та користувачів вимірювальних приладів. Устаткування з протоко-лом Modbus випускають від 20 до 30 виробників, а в промисловості вже давно використовуються багато подібних систем. На даний час протокол Modbus використовується більш ніж у 40% промислових комунікаційних систем.

Доступ до шини Modbus здійснюється на принципі головного/підлеглого пристрою, причому протокол передбачає для одного головного пристрою до 247 підлеглих пристроїв. Транзакції виконує лише головний пристрій.

Транзакції мають тип запитів/відповідей, коли звернення здійснюється лише до одного підпорядкованого пристрою або відразу до всіх підпорядкованих пристроїв (без відповіді).

Транзакція включає один кадр запиту та один кадр відповіді або один кадр звернення до всіх пристроїв.

Деякі параметри протоколу Modbus фіксовані, такі як формат кадру, послідовність кадрів, обробка комунікаційних помилок та виняткових ситуацій, а також виконувані функції. Інші параметри можна вибрати. До них відноситься середовище, характеристики та режим передачі - RTU або ASCII.

Характеристики встановлюються на кожному пристрої і не можуть бути змінені під час роботи системи.

Протокол Modbus формує кадри передачі повідомлень між головним і підлеглим пристроями. Інформацією повідомлень є адреса передбачуваного приймача, дія приймача, дані, необхідні для виконання цієї дії, та засоби контролю за помилками. Підлеглий пристрій зчитує повідомлення і, якщо помилок не виявлено, виконує дії і посилає відповідь головного пристрою. Інформація у відповідному повідомленні: адреса підлеглого пристрою, виконана дія, результат дії та контролю за помилками. Якщо початкове повідомлення призначалося всім устройств, то від підлеглих пристроїв ніякого підтвердження не требуется.

Формат повідомлення. Транзакція складається з одного запиту від головного пристрою до конкретного підпорядкованого пристрою та однієї відповіді від пристрою до головного пристрою. Обидва повідомлення мають формат кадрів Modbus. Кожен кадр повідомлення складається із серії байтів, згрупованих у чотири поля. Слід зазначити, що кожен із зазначених байтів записується у шістнадцятковому форматі (Hex), а не в кодах ASCII.

Перше поле кожного кадру повідомлення є адресним полем, що складається з одного байта інформації. У кадрах запитів цей байт ідентифікує контролер, до якого звернений запит. Результуючий кадр відповіді починається з адреси відповідного пристрою. Кожен підлеглий пристрій може мати адресне поле в межах від 1 до 247, хоча кількість підлеглих пристроїв практично менше. Типова конфігурація Modbus складається з одного головного пристрою та двох-трьох підлеглих пристроїв.

Адресне поле	Поле функції	DATA Поле даних	Поле контролю за помилками
1 байт	1 байт	Змінна довжина	2 байта

Рисунок 1.10 – Формат кадру повідомлення Modbus

Друге поле кожного повідомлення є полем функції, яке складається з одного байта інформації. У запиті головного пристрою цей байт ідентифікує функцію, яку PLC повинен виконати.

Якщо цільовий PLC здатний виконати потрібну функцію, то поле функції його відповіді буде луною оригінального запиту. В іншому випадку в полі функції відповіді старшому байту буде надано значення одиниці, що є сигналом негативної відповіді.

Третє поле кадру повідомлення є полем даних, і його довжина змінюється залежно від функції, визначеної полі функції. У запиті головного пристрою це поле містить інформацію, яка потрібна PLC для виконання функції, що запитується. У відповіді PLC це поле містить дані, які запитують головний пристрій.

Два останні байти повідомлення включають поле контролю за помилками. Числове значення цього поля обчислюється з використанням циклічного надлишкового коду (CRC-16) для кадру повідомлення. Контроль за помилками забезпечує, щоб пристрої не реагували на повідомлення, які були зіпсовані під час передачі.

Спосіб синхронізації. Для забезпечення надійного зв'язку прийом повідомлення має бути синхронізований із його передачею. Іншими словами, пристрій, що приймає, повинен бути здатним виявити початок нового кадру повідомлення.

Відповідно до протоколу Modbus RTU, синхронізація кадру здійснюється шляхом обмеження часу паузи між послідовними символами у кадрі

повідомлення. Якщо протягом часу, що відповідає трьом символам (приблизно три мілісекунди), приймач не виявляє новий символ, то затримане повідомлення буде виключено з розгляду. Наступний байт інтерпретуватиметься як адресне поле нового рядка повідомлення.

Контроль над помилками. Поширеним і дуже ефективним механізмом перевірки повідомлення на наявність помилки є контроль за допомогою циклічного надлишкового коду. CRC заснований на використанні понять з галузі математики, званої теорією алгебри, і є досить простим у реалізації. Використовуючи 16-бітове контрольне значення, CRC обіцяє визначення помилок, що передбачається таблицею 1.8. Механізм контролю за допомогою циклічного надлишкового коду очевидно, дуже ефективний у виявленні помилок, особливо «пакетних помилок», коли зовнішні перешкоди тимчасово пригнічують сигнал і псують цілий рядок бітів. CRC контроль ефективний для сполучень будь-якої довжини.

Таблиця 1.8 – Виявлення помилок із використанням CRC

Кратність помилки	Ймовірність виправлення
Помилки в одному біті	100%
Помилки в двох бітах	100%
Непарна кількість помилок	100%
Пакет ошибок короче 16 бит	100%
Пакет ошибок точно равный 16 битам	99,9969%
Все другие пакетные ошибки	99,9984%

CRC-16 є широко поширеним поліномом, особливо в промислових протоколів. Поліном наведено у формулі 1.4.

$$X^{16} + X^{12} + X^{2}+1, \qquad (1.4)$$

де X – константа одиниці.

Метод виявлення помилок CRC-16 використовує арифметику за модулем 2, де додавання і віднімання дають один і той же результат. Вихід еквівалентний логічної операції що виключає АБО (XOR).

Використовуючи цей алгоритм як основу, можна отримати наступне рівняння 1.5:

(Повідомлення х 2^{16}) / Дільник = Частка + Залишок, (1.5)

де: Повідомлення - Потік бітів, наприклад, послідовність ASCII;

2^{16} - при множенні ефективно додає 16 нулів з правого боку повідомлення;

Дільник - число, на яке проводиться розподіл числа, що вийшов, і є породжуючим поліномом;

Приватне - результат розподілу;

Залишок - значення, що залишилося в результаті поділу, яке є контрольною сумою CRC.

З рівняння 1.4 отримуємо рівні 1.5:

[(Повідомлення х 2^{16}) + Залишок] / Дільник = Частка, (1.6)

Передавач формує згідно формули 1.4 формує контрольну суму (Залишок) і передає приймачеві, приймач отримавши пакет даних згідно з формулою 1.5 проводить ділення повідомлення та контрольної суми на утворює поліном CRC-16. У разі правильної передачі результат обчислення (Приватне) дорівнює «0», інакше переданий пакет був спотворений.

Розглянуті схеми виявлення помилок дозволяють приймачеві виявити, що дані були зіпсовані. Вони не надають засоби для виправлення помилкового символу чи кадру. Ця корекція зазвичай проводиться приймачем, що інформує передавач, що була виявлена помилка, та запитом іншої копії надісланого повідомлення. Цей комбінований цикл виявлення помилки/коригування називається контролем над помилками.

Середня ефективна швидкість передачі при реалізації протоколу Modbus з передачею інформації розміром 5 байт розраховується за формулою для системи з вирішально зворотним зв'язком 1.6:

$$V_{\text{ñðóô}} = V\left(1-\frac{P_o n_{\text{á}}}{1-P_o n_{\text{á}}}\right)\left(1-\frac{r+n_{\text{ñë}}}{n_{\text{á}}}\right)=1{,}8\,M\text{áàéò / ñåê},$$

(1.6)

Де V = 3Мбіт/сек - швидкість передачі по каналу зв'язку;

$P_o = 10^{-6}$ - ймовірність спотворення пакета, що передається.

Узагальнений алгоритм обміну даними протоколу Modbus. Узагальнений алгоритм обміну даними між провідним і веденим пристроєм можна подати у вигляді, зображеному на рис. 1.11.

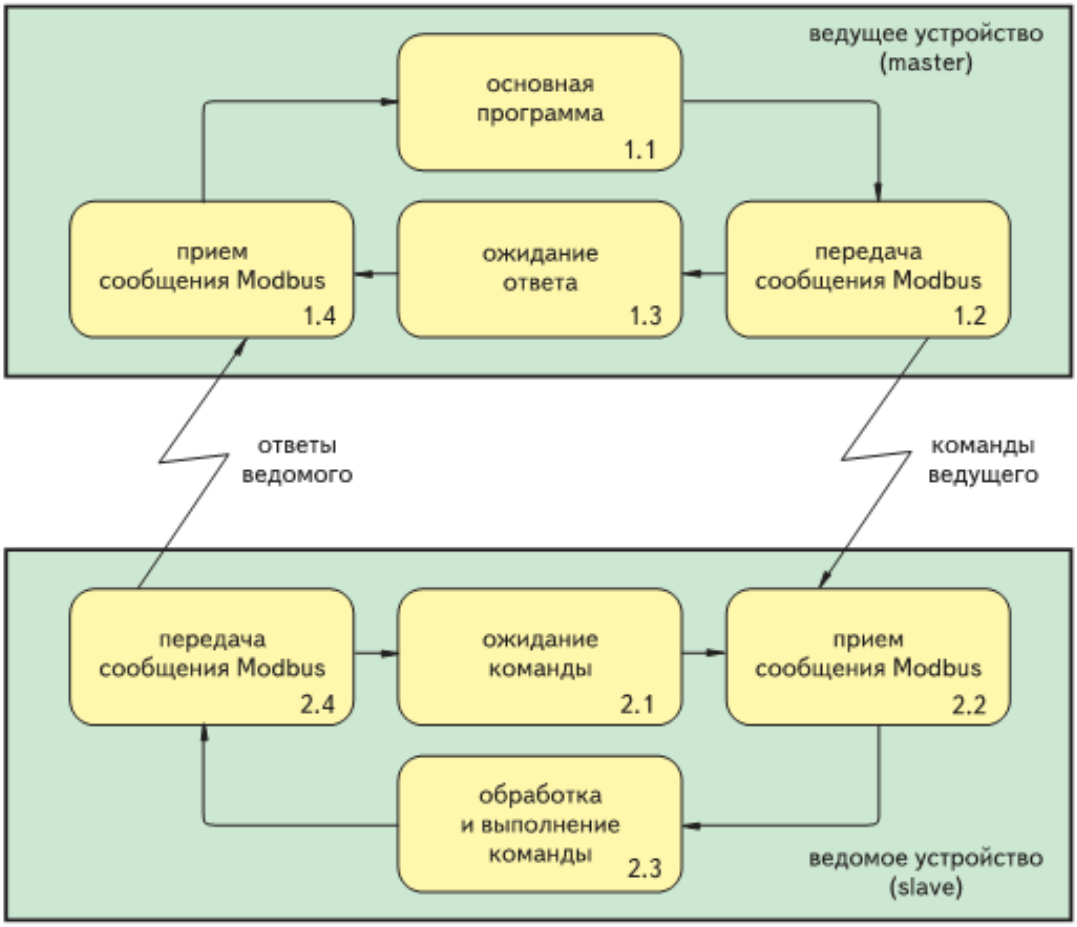

Рисунок 1.11 – Узагальнений алгоритм обміну даними між провідним та веденим пристроєм

Отже, провідного пристрою в процесі виконання цільової програми (стан 1.1 на рис.1.11) потрібно було запитати дані віддаленого модуля або встановити виходи віддаленого модуля у певний стан. Провідний пристрій (далі ведучий або майстер) формує повідомлення Modbus, яке в загальному випадку містить адресу веденого, код команди, додаткові дані та контрольну суму. Сформоване

повідомлення передається через UART мікроконтролера та фізичну лінію зв'язку веденому пристрою (стан 1.2). Після передачі повідомлення ведучий перетворюється на стан очікування відповіді відомого (стан 1.3). Відомий пристрій, що знаходиться в стані очікування команди ведучого (стан 2.1) при надходженні даних до буфера UART переходить у стан прийому (стан 2.2) і у разі успішного завершення прийому переходить до декодування та виконання команди (стан 2.3). Після виконання команди ведений пристрій формує повідомлення у відповідь, яке в загальному випадку містить адресу веденого пристрою (тобто власну адресу), код команди, додаткові дані та контрольну суму. Сформоване повідомлення передається провідному пристрою (стан 2.4), і після успішного прийому відповіді веденого (стан 1.4), провідний пристрій продовжує роботу за основною програмою, а ведене переходить до очікування наступної команди. У цьому транзакція Modbus завершується.

Передача та прийом повідомлення. Передача повідомлення в режимі RTU здійснюється найпростішим чином (рис.1.12): передавальний пристрій формує повідомлення (стан 1), чекає, у разі необхідності, звільнення буфера передавача UART (стан 2) і відправляє повідомлення (стан 3). Яких-небудь позаштатних ситуацій та помилок тут чекати не доводиться.

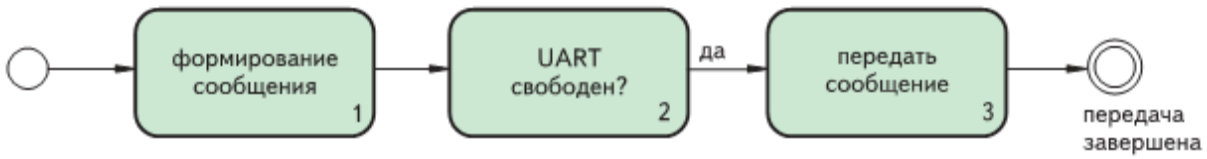

Рисунок 1.12 – Кінцевий автомат передачі повідомлення Modbus у режимі RTU

Кінцевий автомат, який виконує прийом повідомлення Modbus у режимі RTU представлений на рис.1.13. Його робота проста. У стані 1 обнулюється лічильник буфера I. У стані 2 кожен прийнятий символ записується в буфер Data. Перехід зі стану 2 (приміщення повідомлення в буфер) в стан 3 (перевірка поля адреси) здійснюється за тайм-аутом, величина якого повинна дорівнювати 3,5 тривалості передачі одиночного байта.

У стані відбувається перевірка поля адреси повідомлення. Для керованого пристрою це поле повинно співпадати з адресою пристрою. Якщо ж прийом здійснюється провідним пристроєм, тобто повідомлення є відповіддю веденого на

раніше надіслану команду, в полі адреси повинен утримуватися адреса відповідного веденого. Якщо це не так, то повідомлення адресоване іншому пристрою, і автомат знову переходить у стан 1 і чекає на початок наступного повідомлення. Якщо адреса збігається, автомат перетворюється на стан 4 «обчислення контрольної суми». Якщо контрольна сума збігається з переданою в повідомленні, прийом повідомлення вважається успішним і автомат припиняє роботу. Автомат також здійснює обробку низки позаштатних ситуацій, які можуть виникнути під час прийому повідомлення. До таких ситуацій належать такі:

- Розмір буфера перевищив максимально допустимий;
- Не збігається контрольна сума.

Виникнення помилок такого роду при роботі пристрою свідчить про погану якість зв'язку (сильні електромагнітні перешкоди, погане екранування занадто велика довжина лінії зв'язку, не дотримання правил монтажу і т.п.). Поява помилок може також свідчити про некоректну роботу передаючого пристрою.

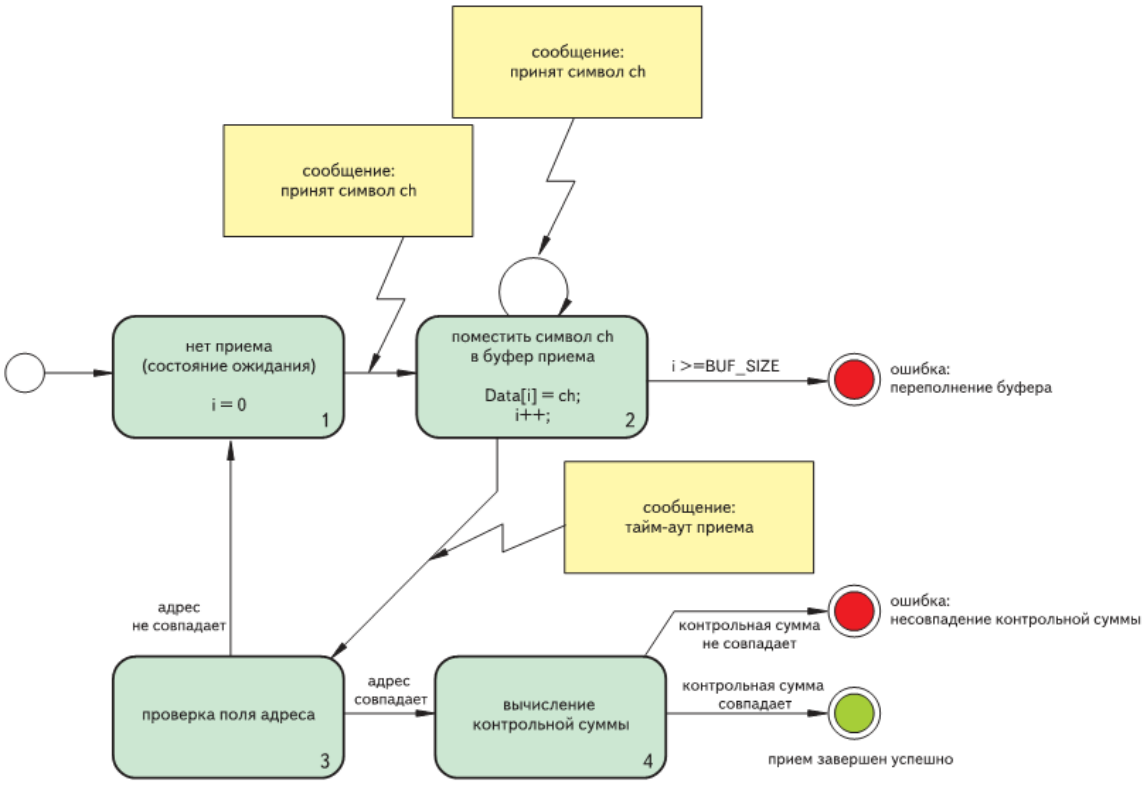

Рисунок 1.13 - Кінцевий автомат прийому повідомлення Modbus в режимі RTU

Узагальнений алгоритм роботи провідного устрою Modbus.

Узагальнений алгоритм роботи провідного пристрою Modbus зображено на рис.1.14. Робота автомата управляється двома повідомленнями «ззовні»: «почати транзакцію» - повідомлення від основної програми (передбачається, що посилка Modbus формується і розміщується в буфері передачі основною програмою) і «початок прийому» - повідомлення, яке сигналізує про надходження символу до приймального буфера UART і формується відповідним обробником переривання UART. Отже, потік після ініціалізації UART (стан 1) перетворюється на стан очікування 2. Як тільки основна програма віддає команду початку транзакції, автомат перетворюється на стан 3, у якому відбувається передача повідомлення.

Після передачі повідомлення контролер чекає на відповідь веденого пристрою (стан 4).

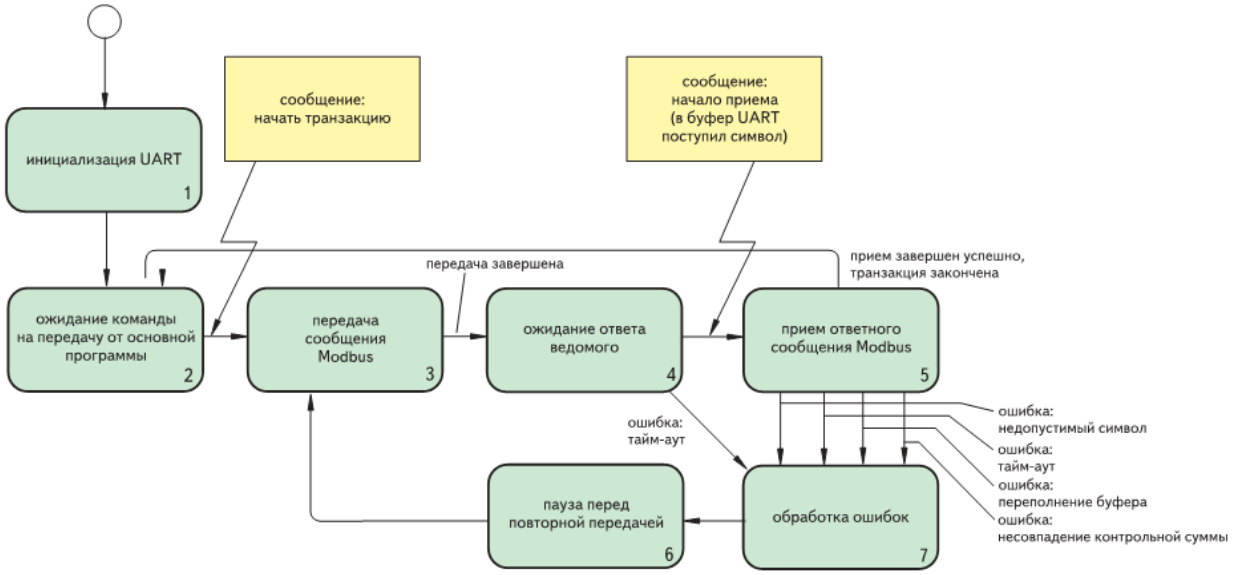

Рисунок 1.14 – Узагальнений алгоритм роботи провідного пристрою Modbus

Очікування відповіді обмежене за часом і зазвичай не перевищує кількох секунд.

При надходженні першого символу повідомлення у відповідь в буфер UART автомат переходить в стан 5 (в цьому стані розміщений один з розглянутих вище алгоритмів прийому повідомлення). При успішному прийомі повідомлення транзакція вважається завершеною, і автомат перетворюється на стан 2, очікуючи наступної команди основний програми. При виникненні помилки при прийомі автомат витримує паузу (стан 6) і повторює передачу повідомлення (стан 3). Можна звернути увагу, що між станами 5 і 6 на рис.5 зображено стан 7 «обробка

помилок». У принципі, протокол Modbus не передбачає будь-якої специфічної обробки помилок у прийнятих повідомленнях (у разі виникнення помилки при прийомі провідне пристрій має просто повторити передачу команди). Однак ми можемо ввести в програму пристрою, наприклад журнал помилок, причому з підрозділом причин виникнення тієї чи іншої помилки (несупадання контрольної суми тощо). Наявність такої інформації може бути дуже корисною при налагодженні системи.

Спостерігаючи за інтенсивністю виникнення помилок у каналі зв'язку, ми можемо підібрати оптимальну швидкість передачі у мережі Modbus.

Узагальнений алгоритм роботи керованого пристрою Modbus. Узагальнений алгоритм роботи веденого пристрою Modbus зображено на рис.1.15.

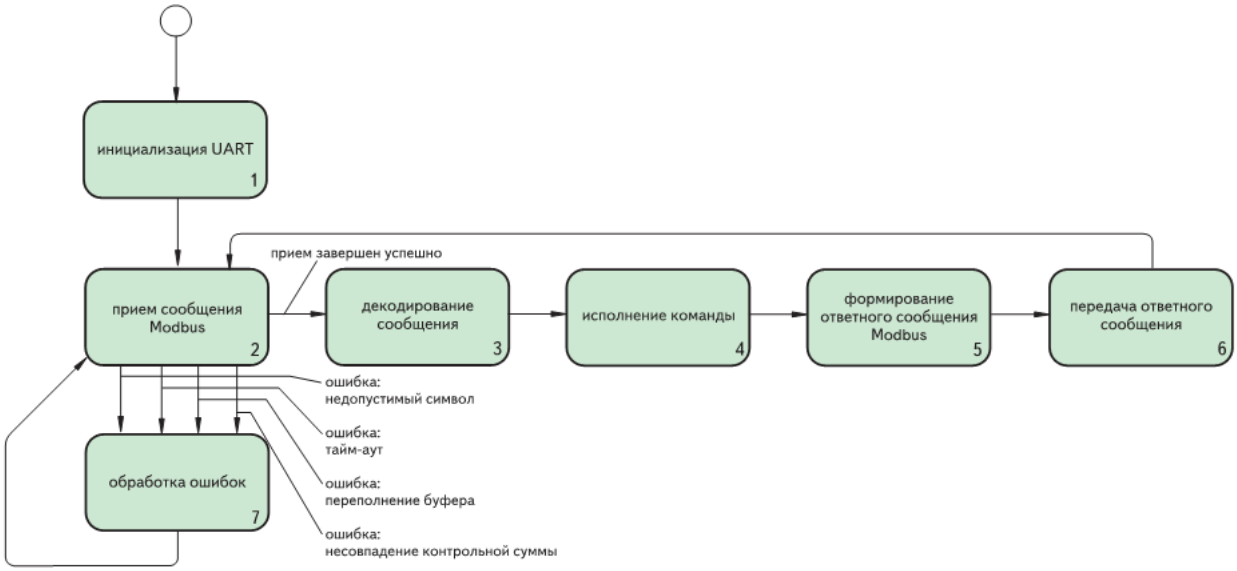

Рисунок 1.15 - Узагальнений алгоритм роботи керованого пристрою Modbus

Він схожий на відповідний алгоритм провідного пристрою, стій лише різницею, що операції відбуваються у зворотному порядку: спочатку приймається повідомлення (стан2), потім, після декодування команди, її виконання інформування повідомлення у відповідь (стану 3-5) відбувається передача відповіді провідного пристрою. Відмінність полягає і в тому, що при виникненні помилки прийому ведений пристрій не повинен посилати повідомлення у відповідь, воно просто повинно дочекатися, коли майстер передасть команду повторно. Стан 7, як і в попередньому випадку, не є обов'язковим, але дуже корисно при налагодженні.

Кадр даних - протокол роботи EterNet. Усі дані передаються у вигляді пакета чи кадру. Кадр складається з оболонки, що містить інформацію, що управляє (таку як байти синхронізації та адреси) і реальне повідомлення. Кожен вузол визначає адресу призначення та зчитує дані, якщо кадр призначений йому.

Кадр даних Ethernet має структуру, подану у табл. 1.7.

Таблиця 1.7 – Формат кадру IEEE 802.3

Преамбула	Початковий розмежувач	Адреса призначення	Адреса джерела	Індикатор даних	Дані	Контрольна CRC сума
7 байт	1 байт	2 або 6 байт	2 або 6 байт	2 байта	46–1500 байт	4 байта

Преамбула - поле складається з семи байтів, кожен з яких має двійкове значення 10101010. Метою поля є забезпечення синхронізації з кадром всіх приймаючих контролерів доступу до середовища.

Розмежувач початку кадру (SFD) SFD відзначає початок кадру та має двійкове значення 10101011.

Адреса призначення може складатися з 16 чи 48 бітів – це залежить від конфігурації системи. Насправді адреса призначення майже завжди складається з 48 бітів. Адреса повинна мати однакову довжину всім вузлів. Вузол з цим адресою прочитає повідомлення.

Адреса джерела може складатися з 16 або 48 бітів - це залежить від конфігурації системи. Насправді він майже завжди складається з 48 бітів. Це адреса вузла, який надіслав дані.

Індикатор довжини це двобайтовий індикатор довжини визначає, скільки байтів міститься у полі даних.

Дане поле містить реальні дані, які можуть мати довжину від 46 до 1500 байтів. Мінімальна довжина поля даних визначається необхідністю виявлення конфлікту, а максимальна довжина обмежує час доступу будь-якого вузла 1,2 мілісекундами. Якщо довжина реальних даних становить менше 46 байтів, поле даних має бути доповнено до 46 байтів.

Контрольна послідовність кадру це 32-розрядна величина, яка використовується для виявлення помилок за допомогою циклічного надлишкового коду (CRC).

Послідовність передачі:

1. Контролер доступу до середовища утворює кадр даних.

2. Контролер доступу до середовища контролює лінію на предмет відсутності несучої.

3. Приймач вузла передає кадр у вигляді потоку бітів.

4. Приймач контролює лінію, щоб виявити можливий конфлікт.

У разі виникнення конфлікту передавач посилає незначну послідовність, припиняє передачу і повторює спробу, через короткий проміжок часу, що визначається випадковим чином.

Послідовність прийому:

1. Контролер доступу до середовища виявляє сигнал, що надходить від приймача.

2. Включається сигнал виявлення несучої, щоб заборонити будь-яку передачу контролером доступу до середовища.

3. Контролер доступу до середовища використовує бамти преамбули для забезпечення синхронізації.

4. Перевіряється адреса призначення і якщо вона збігається з адресою вузла, провадиться зчитування даних.

5. Дані зчитані, перевіряється відповідність індикатора довжини з довжиною повідомлення. Обчислюється FCS та порівнюється зі значенням поля FCS.

Якщо обидва порівняння є позитивними, то повідомлення передається далі.

Побудова блоку схеми прийому даних провідним пристроєм. Максимальний розмір пакета даних переданих за протоколом передачі Modbus має не перевищувати 255 байт. При прийомі пакета даних більше встановленої довжини прийом даних припиняється і передається повідомлення про некоректну передачу повідомлення. Також повідомлення про некоректну передачу повідомлення надсилається у разі розбіжності контрольної суми та при неприпустимому значенні функції.

У разі некоректної передачі даних від ведучого до веденого пристрою, ведений пристрій передає пакет даних з повідомленням про тип помилки і запит на повторну передачу повідомлення. У разі некоректної передачі від веденого до

провідного пристрою, провідний пристрій повторює запит після закінчення часу очікування повідомлення у відповідь.

Загальний алгоритм функціонування модуля розподіленого збору даних наведено малюнку 1.16

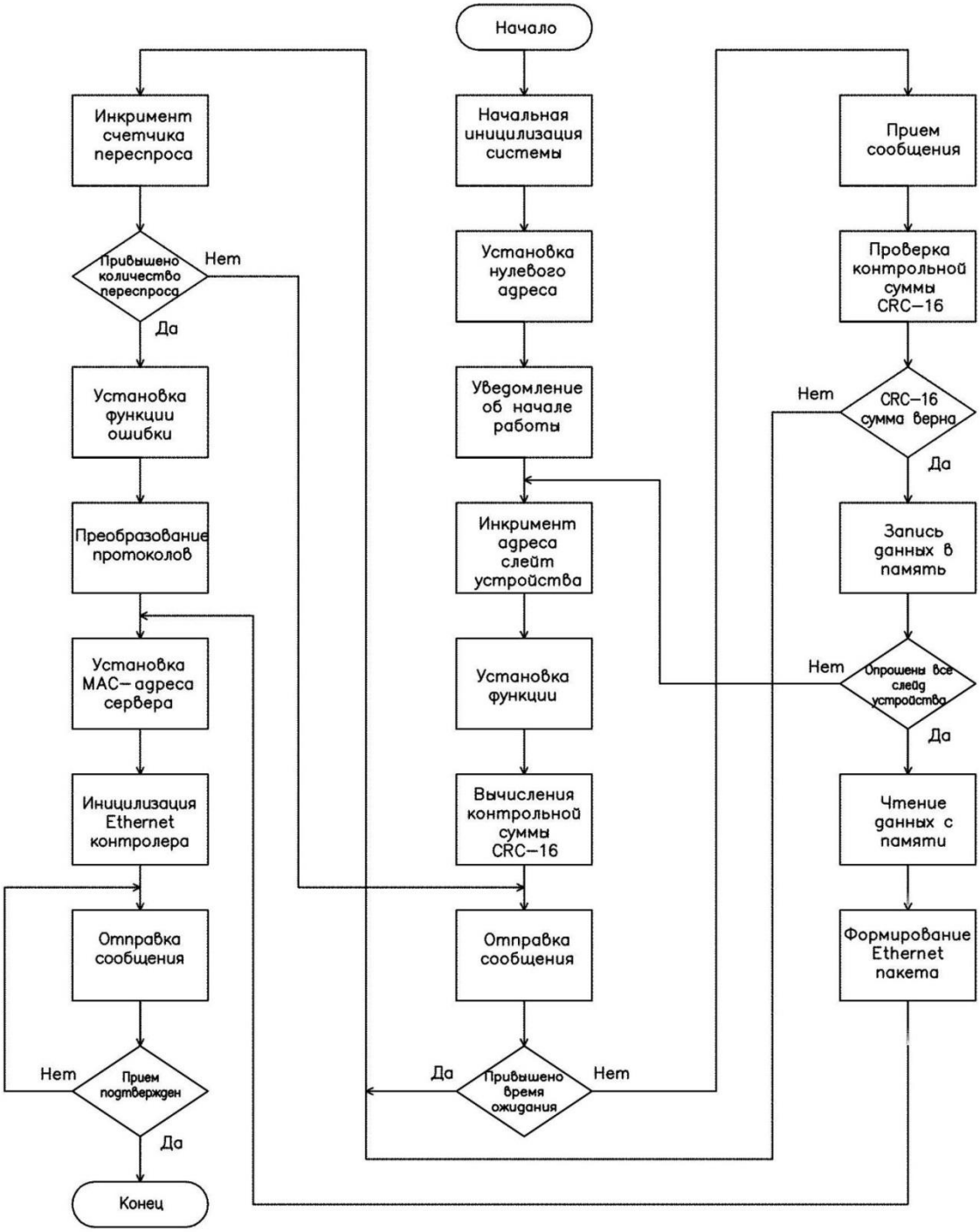

Рисунок 1.16 - Алгоритм функціонування модуля розподіленого збору даних

Структурна схема проектованого модуля розподіленого збору даних наведено на рисунку 1.17.

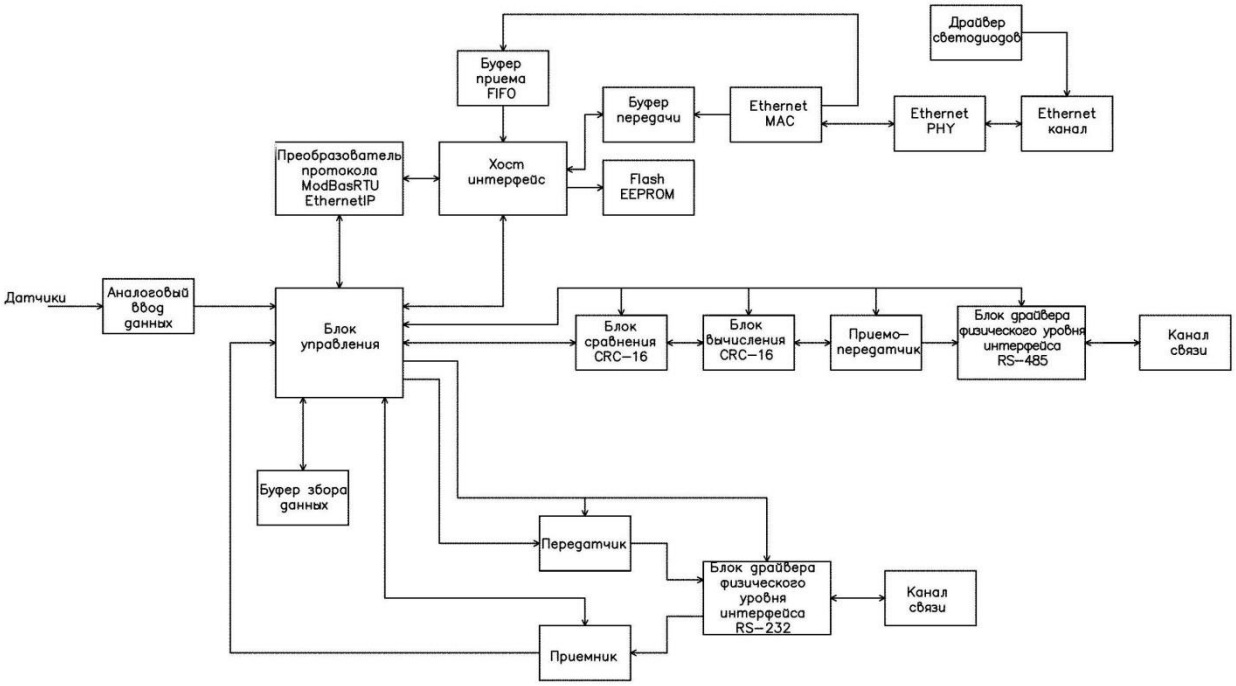

Рисунок 1.17 - Структурна схема проектованого пристрою

Блок керування – блок призначений для керування розподіленого модуля та виконання основної програми технічного процесу.

Буфер збору даних призначений для гранення інформації, зібраної з датчиків і переданих веденими пристроями по каналу зв'язку з інтерфейсом rs-485.

Блок драйвера фізичного рівня інтерфейсу rs-485 призначений для перетворення сигналу та формування необхідної амплітуди для передачі даних у канал зв'язку.

Блок обчислення CRC16 – блок, призначений для обчислення контрольної суми пакета, що відправляється і прийнятого за алгоритмом CRC-16 для забезпечення достовірності прийнятої та відправленої інформації.

Блок порівняння CRC16 – призначений для порівняння розрахованої контрольної суми та суми переданої по каналу зв'язку.

Блок перетворювач протоколів Modbus RTU/EthernetIP призначений для перетворення протоколу Modbus RTU в EthernetIP для передачі даних серверу по каналу Ethernet.

Хост інтерфейс - блок забезпечує зв'язок пристрою управління з буфером прийому і буфером передачі паралельної 16 розрядної шині, адреса шина і шина даних суміщена.

Буфер прийому призначений для зберігання даних, прийнятих з каналу даних, розмір буфера становить 4 кілобайти.

Буфер передачі призначений для зберігання даних перед відправкою до каналу зв'язку розмір буфера становить 4 кілобайти.

Драйвера світлодіодів - блок управління світлодіодами, які сигналізують наявність каналу і передачу даних.

Ethernet MAC – блок зберігання MAC – адресу та формування SPX протоколу.

Flash EEPROM - блок енергонезалежної пам'яті типу Flash призначений для зберігання алгоритмів створення Ethernet пакета і функціонування протоколів передачі даних по Ethernet каналу зв'язку, розмір Flash пам'яті становить 8 кілобайт.

1.3.2 Розробка схеми електричного функціонального проектованого пристрою.

Введення даних з модулів аналогових вводів даних представлений в 16-ти розрядною шиною, що з'єднується з двома 8-ми розрядними паралельними портами вводу/виводу мікроконтролера. Сигнал готовності від зовнішнього пристрою подається на вхід обробки зовнішнього переривання мікроконтролера, що дозволяє пристрою переключиться на зчитування даних від зовнішнього пристрою.

У разі потреби віддаленого перезавантаження пристрою на вхід $\overline{RESET}$ подається напруга логічного нуля.

Для організації послідовного інтерфейсу RS-485 використовується драйвера інтерфейсу RS-485. Як правило, драйвера інтерфейсу RS-485 мають послідовний вхід передавача і вихід приймача мікросхеми.

Обмін даних між мікроконтролером та драйвером інтерфейсу RS-485 здійснюється через послідовний периферійний інтерфейс SPI. Причини вибору периферійного інтерфейсу SPI полягає в гнучкому програмному виборі швидкості передачі даних і виборі формату передачі (від молодшого біта до старшого біта або навпаки).

При передачі даних пристроєм дані бітно видаються на виведення MOSI (Master Out Slave In) на вхід передавача DI драйвера RS-485, при цьому драйвер RS-485 повинен бути встановлений в режим передавача і вимкнений режим приймача для запобігання колізії. Для цього на входах DE та драйвера RS-485 подаються 1, інакше вхід передавача знаходиться у високо імпедансному стані. Формат передачі від мікроконтролера і швидкість передачі визначається програмно.

При прийомі даних пристроєм дані бітно приймаються на вхід MISO (Master In Slave Out) з виходу приймача RO драйвера RS-485, при цьому драйвер RS-485 повинен бути встановлений в режим приймача і відключений режим приймача. Для цього необхідно на входах DE та драйвера RS-485 подати напруги логічного 0.

Канал зв'язку інтерфейсу RS-485 представлений у вигляді двопровідної лінії зв'язку в напівдуплексному режимі.

Для реалізації послідовного інтерфейсу RS-232c використовується драйвера інтерфейсу RS-232c в інтегральному виконанні. Як правило, драйвера інтерфейсу RS-232c містять послідовний вхід передавача і вихід приймача, з'єднані з мікроконтролером. Вихід передавача та вхід приймача виводяться у канал зв'язку через COM-порт.

Обмін даних між мікроконтролером та драйвером інтерфейсу RS-232 здійснюється через універсальний синхронний/асинхронний приймач-передавач (USART). Даний модуль приймач-передавача забезпечує повно-дуплексний обмін по послідовному каналу, при цьому швидкість передачі даних може змінюватись у досить широких межах. Довжина посилки може складати від 5 до 9 бітів. У всіх модулях в обов'язковому порядку присутні схеми контролю та формування біта парності.

Модуль USART, реалізований у мікроконтролерах, можуть виявляти такі позаштатні ситуації:

- Переповнення;
- Помилка кадрування;
- Невірний старт-біт.

Для зменшення ймовірності збою в модулі реалізовано таку функцію, як фільтрація перешкод.

При передачі даних пристроєм дані побитно надходять через виведення TxDn на вхід T_IN драйвера інтерфейсу RS-232c. При прийомі даних дані з виходу приймача R_OUT надходять на вхід RxDn мікроконтролера.

Переданий кадр повідомлення починається зі старт-біта, за яким слідує молодший біт слова. Після старшого біта слова, дані слідує один або два стоп-біти. Якщо включено схему формування біта парності, він включається між старшим бітам слова даних і першим стоп бітом. Вид кадру представлений малюнку 1.18.

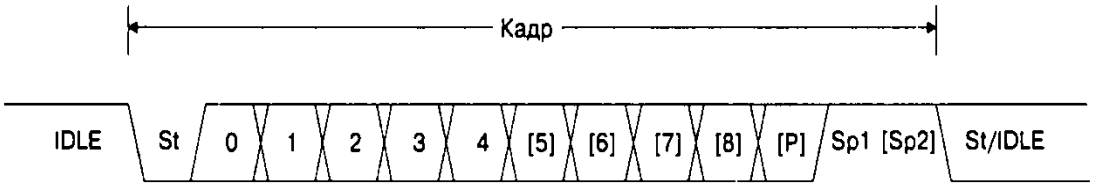

Рисунок 1.18 - Формат кадру

ST - старт біт, завжди дорівнює 0;

[n] - біті слова даних;

P – біт парності;

Sp1 - стоп біт, завжди дорівнює 1;

IDLE – немає обміну по лінії RxDn або TxDn, має бити 1.

Проектований пристрій з'єднується із зовнішнім пристроєм за інтерфейсом RS-232c через комутаційний порт (COM-порт). У цьому пристрої використовуємо два 9-штиркові роз'єми DB-9 включених в режимі мінімальний нуль-модемний кабель (рисунок 1.19) і повний нуль-модемний кабель (рисунок 1.20).

Мінімальний нуль-модемний кабель використовується для зв'язку пристроїв за інтерфейсом RS-232c, повний нуль-модемний кабель служить для з'єднання пристрою з модемом.

Сигнали 9-штекерного роз'єму та їх опис наведено в таблиці 1.9.

Зв'язок модуля розподіленого збору даних з віддаленим сервером через Ethernet канал системою 10Base-T. Система 10Base-T має меншу вартість установки порівняно з іншими системами та використовує кручену пару. Кожен вузол підключається до центрального хаба двома парами (одна пара прийому, а друга передачі). Хаб може знаходитись на відстані 100 м.

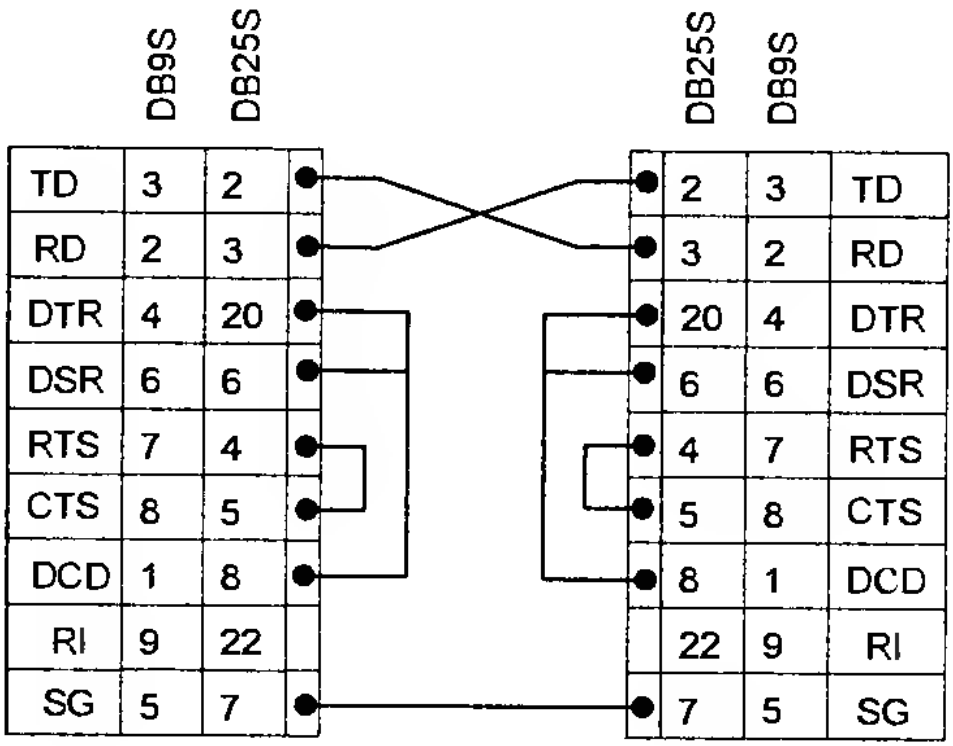

Рисунок 1.19 – Мінімальний нуль-модемний кабель

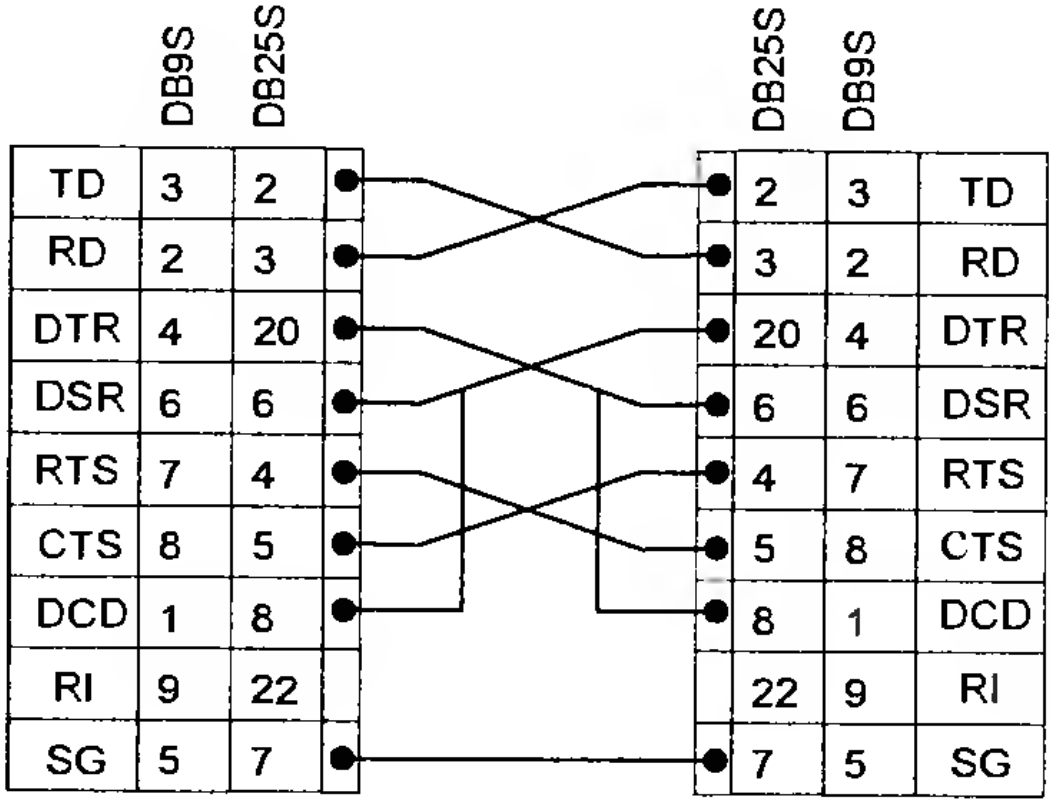

Рисунок 1.20 – Повний нуль-модемний кабель

Таблиця 1.9 – Сигнали 9-штекрного роз'єму та їх описи

Контакт DB-9	Позначення	Опис	Абревіатура	I/O
1	CF	Виявлення несучої	DCD	I
2	BB	Прийняті дані	RxD (RD)	I
3	BA	Передані дані	TxD (TD)	O
4	CD	Готовність	DTR	O
5	AB	Сигнальне заземлення	SG	I/O
6	CC	Готовність DCE	DSR	I/O
7	CACJ	Запит передачі готовність до прийому	RTS	O
8	CB	Готовність передачі	CTS	O
9	CE	Індикатор виклику	RI	I

Підключення до інтерфейсних плат здійснюється за допомогою модульних 8-контактних роз'ємів RJ-45, як показано на рис. 1.21.

Хаб забезпечує роботу шини між вузлами і може мати AUI роз'єми для товстого Ethernet, тонкого Ethernet або оптоволоконних приймачів.

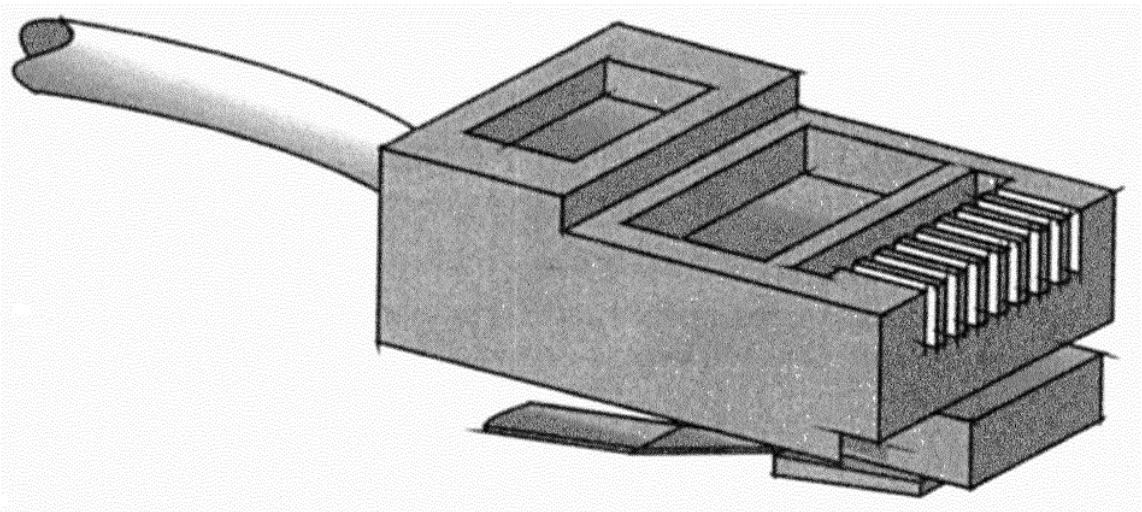

Рисунок 1.21 – Роз’єм RJ-45 для витих пар

1.3.3 Вибір елементної бази.

Мікроконтролери AVR фірми Atmel є 8-бітними мікроконтролерами, призначеними для використання в апаратурі зв'язку та контрольно-вимірювальної апаратури. Вони виготовлені за малоспоживною КМОП – технологією, яка в поєднанні з удосконаленою RISC-архітектурою дозволяє досягти найкращого співвідношення вартість/швидкості/енергоспоживання.

До особливостей мікроконтролерів AVR сімейства Mega можна віднести:

- FLASH-пам'ять програм об'ємом від 8 до 256 Кбайт (число циклів стирання/запису не менше 10 000);

- оперативна пам'ять (статична ОЗП) обсягом від 512 байт до 8 Кбайт;

- пам'ять даних на основі ЕСППЗУ (EEPROM) об'ємом від 256 байт до 4 Кбайт (кількість циклів стирання/запису не менше 100 000);

- можливість захисту від читання та модифікації пам'яті програм та даних;

- Можливість програмування безпосередньо в системі через послідовні інтерфейси SPI і JTAG;

- Можливість самопрограмування;

- Наявність декількох режимів зниженого енергоспоживання;

- Наявність детектора зниженої напруги живлення (Brown-Out Detector-BOD);

- Можливість програмного зниження частоти тактового генератора.

У цей пристрій доцільно використовувати мікроконтролер моделі ATmega1281V-12AI/I. Даний мікроконтролер містить послідовний периферійний інтерфейс SPI, універсальний синхронний/асинхронний приймач-передавач (USART), і має достатню обчислювальну потужність. Параметри даного мікроконтролера наведено у таблиці 1.10.

Таблиця 1.10 – Параметри мікроконтролера ATmega128

Модель	Пам’ять програм (FLASH) [Кбайт]	Пам’ять даних (ОЗУ) [байт]	Пам’ять даних (EEPROM) [байт]	Кількість контактів входів/ виходів	Навантаження живлення [В]	Тактова частота [МГц]
ATmega 128	128	4K	4K	53	4,5…5,5	0…16
ATmega 128L					2,7…5,5	0…8

Розташування висновків мікроконтролера ATmega128 у корпусі TQFP-64 та MLF-64 наведено на рис. 1.22

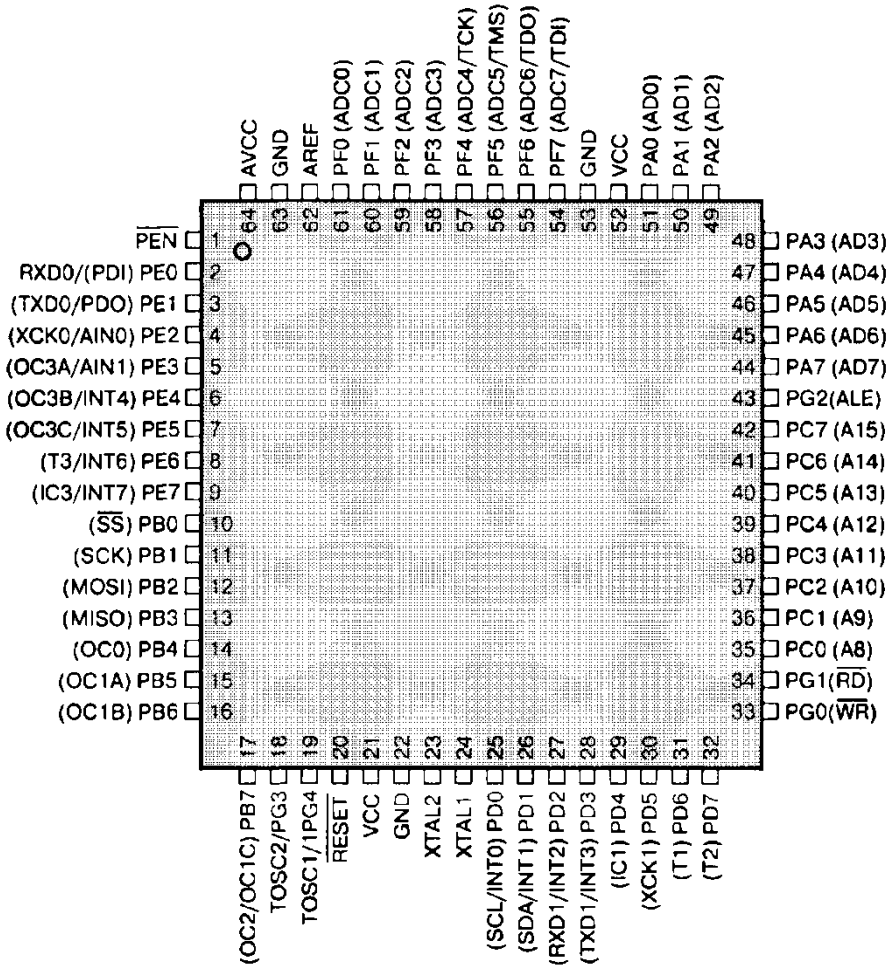

Рисунок 1.22 – Розташування моделі ATmega128

Призначення, тип та нумерація висновків мікроконтролера ATmega128 у корпусі TQFP-64 та MLF-64 наведені в таблиці 1.11.

У цьому мікроконтроллері реалізований послідовний периферійний інтерфейс SPI (Serial Peripheral Interface) призначений для обміну даними між мікроконтролером і різними периферійними пристроями, такими як цифрові потенціометри, ЦАП/АЦП, FLASH-ПЗУ та ін. За допомогою цього інтерфейсу також може здійснюватися обмін даними AVR.

Таблиця 1.11 – Опис висновків моделі ATmega128

Позначення	Номер виводу	Тип виводу	Опис
XTAL1	24	I	Вход тактового генератора
XTAL2	23	O	Выход тактового генератора
$\overline{RESET}$	20	I	Вход сброса
Порт А. 8-бітний двонаправлений порт вводу/виводу із внутрішніми підтягуючими резисторами			
PAO	51	I/O	0-й біт порта А
PAl	50	I/O	1-й біт порта А
PA2	49	I/O	2-й біт порта А
PA3	48	I/O	3-й біт порта А
PA4	47	I/O	4-й біт порта А
PA5	46	I/O	5-й біт порта А
PA6	45	I/O	6-й біт порта А
PA7	44	I/O	7-й біт порта А
Порт С. 8-бітний двонаправлений порт вводу/виводу із внутрішніми підтягуючими резисторами			
PCO	35	I/O	0-й біт порта С
PCl	36	I/O	1-й біт порта С
PC2	37	I/O	2-й біт порта С
PC3	38	I/O	3-й біт порта С
PC4	39	I/O	4-й біт порта С
PC5	40	I/O	5-й біт порта С
PC6	41	I/O	6-й біт порта С
PC7	42	I/O	7-й біт порта С

Виводи послідовного периферійного інтерфейсу SPI			
$\overline{SS}$	10	I/O	Вибір Slave-пристрою на шині
SCK	11	I/O	Вихід (Master) або вхід (Slave) тактового сигналу модуля SPI

Позначення	Номер виводу	Тип виводу	Опис
MOSI	12	I/O	Вихід (Master) або вхід (Slave) даних модуля SPI
MISO	13	I/O	Вихід (Master) або вхід (Slave) даних модуля SPI
Універсальний синхронний/асинхронний прийомопередавач (USART)			
RXD1	27	I/O	Вхід USART1
TXD1	28	I/O	Вихід USART1
XCK1	30	I/O	Вхід/вихід зовнішнього тактового сигнала USART1
RXD0	2	I/O	Вхід USART0
TXD0	3	I/O	Вихід USART0
XCK0	4	I/O	Вхід/вихід зовнішнього тактового сигнала USART0
INT7	9	I/O	Вхід зовнішнього переривання
Порт F. 8-бітний двонаправлений порт вводу/виводу із внутрішніми підтягуючими резисторами			
PFO	61	I/O	0-й біт порта F
PF1	60	I/O	1-й біт порта F
PF2	59	I/O	2-й біт порта F
PF3	58	I/O	3-й біт порта F
PF4	59	I/O	4-й біт порта F
PF5	60	I/O	5-й біт порта F
PF6	61	I/O	6-й біт порта F
PF7	62	I/O	7-й біт порта F

Структурна схема модуля SPI наведена малюнку 1.21.

Тимчасові діорами передачі даних за послідовним периферійним інтерфейсом наведені на малюнку 1.24.

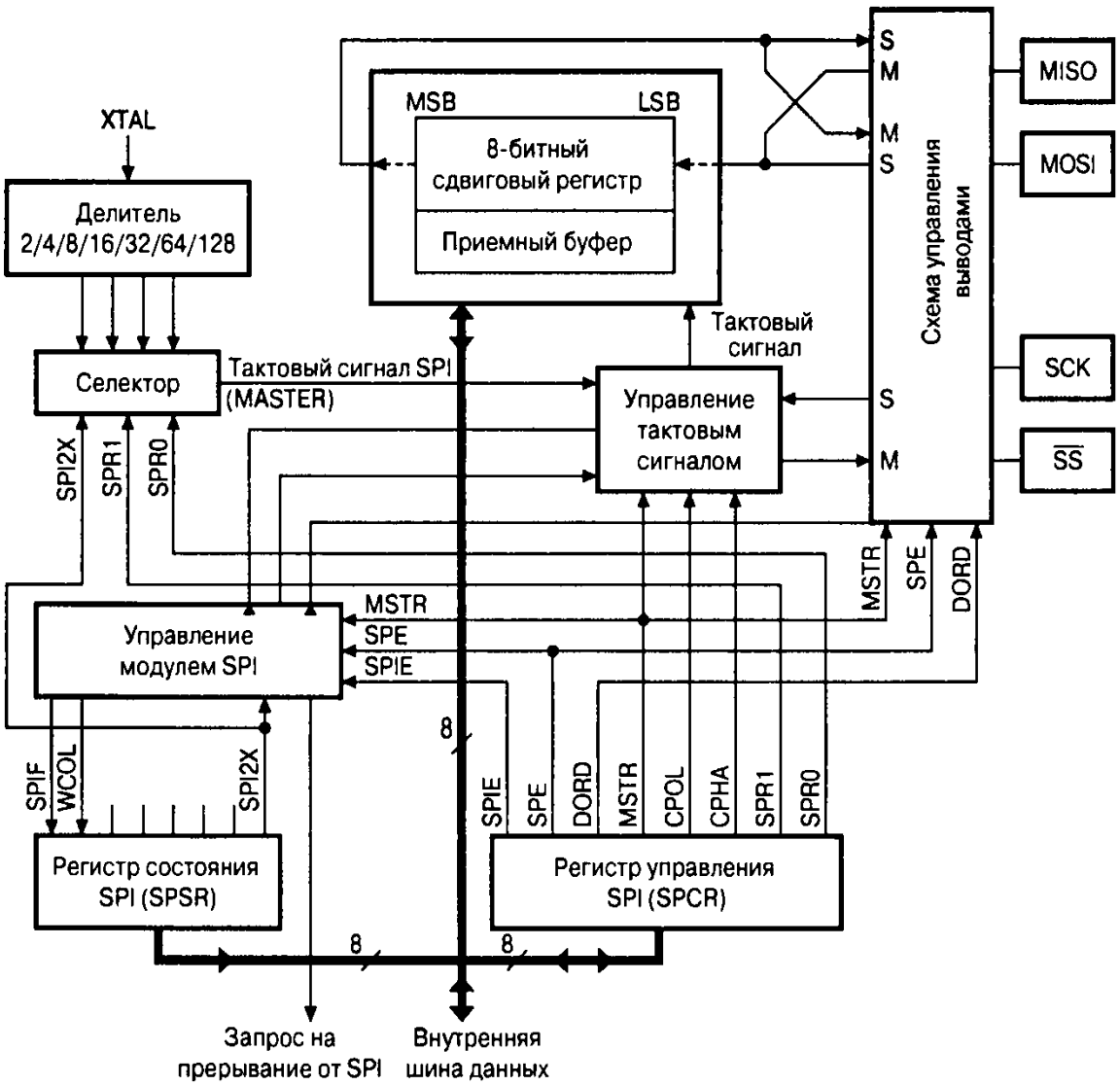

Рисунок 1.23 – Структурна схема модуля SPI

Частота тактового сигналу SCK і відповідно швидкість передачі даних за інтерфейсом визначаються програмно встановлюються біти SPR1, SPR0 реєстру SPCR і біт SPI2X регістра SPSR. Зрозуміло, йдеться про мікроконтролер, що працює в режимі Master, так як саме він є джерелом тактового сигналу. Завдання тактової частоти SCK наведено у таблиці 1.12.

Таким чином, частота тактового сигналу SCK послідовного інтерфейсу SPI при тактовій частоті мікроконтролера 12 МГц варіюється від 93 КГц до 6 МГц.

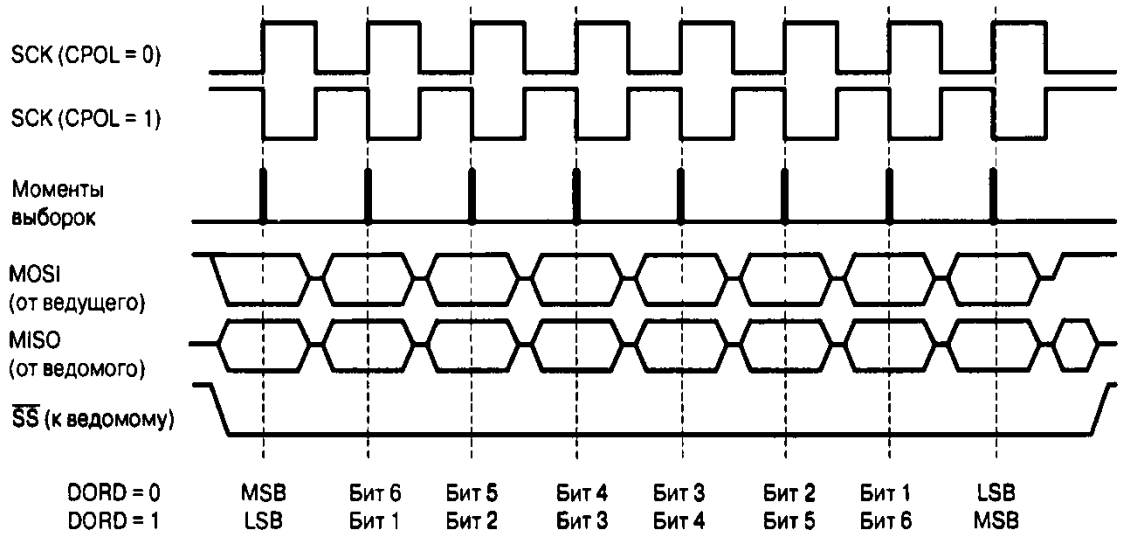

Рисунок 1.24 – Тимчасові діорами передачі даних SPI

Таблиця 1.12 - Завдання частоти тактового сигналу SCK

SPI2X	SPR1	SPR0	Частота сигналу SCK
0	0	0	$f_{CLK}/4$
0	0	1	$f_{CLK}/16$
0	1	0	$f_{CLK}/64$
0	1	1	$f_{CLK}/128$
1	0	0	$f_{CLK}/2$
1	0	1	$f_{CLK}/8$
1	1	0	$f_{CLK}/32$
1	1	1	$f_{CLK}/64$

де f_{CLK} – тактова частота мікроконтролера.

У всіх мікроконтролерах сімейства mega вбудовані модуль універсального синхронно/асинхронного приймач-передавача (USART). Модель ATmega1281V-12AI/I у своєму складі має два модулі універсального синхрон-но/асинхронного приймач-передавача, які забезпечують повнодуплексний обмін по послідовному каналу.

Структурна схема одного модуля USART наведена малюнку 1.25

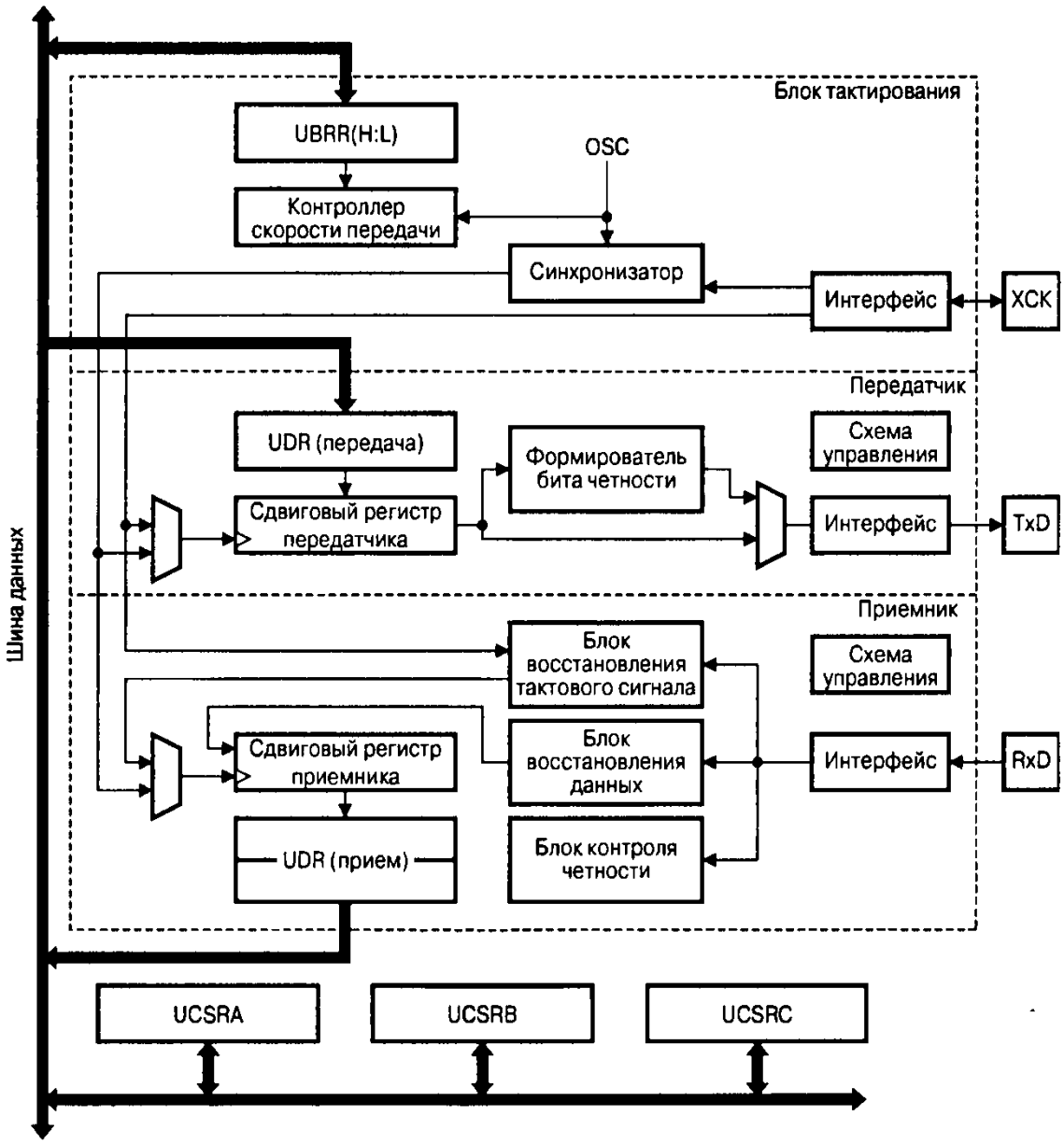

Рисунок 1.25 – Структурна схема модуля USART

Швидкість обміну даних визначається за формулою 1.7:

$$BAUD = \frac{fck}{16 \cdot (UBBR + 1)}, \qquad (1.7)$$

де BAUD – швидкість передачі біт/с;

fck – тактова частота мікроконтролера;

UBBR – вміст регістру контролю швидкості передачі.

Послідовний інтерфейс передачі RS-485 реалізується через драйвер інтерфейсу RS-485 марки MAX3443E фірми MAXIM. Схематичне зображення драйвера інтерфейсу RS-485 наведено малюнку 1.26.

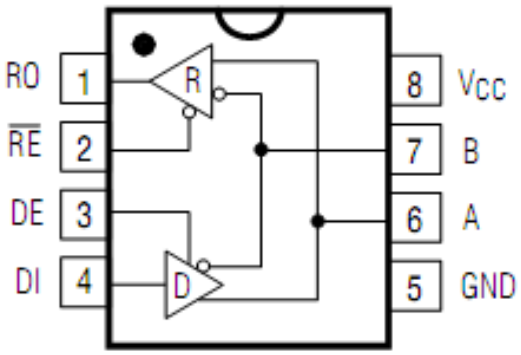

Рисунок 1.26 – Схематичне зображення MAX3443E

Драйвер інтерфейсу RS-485 марки MAX3443E має вбудований захист від перепадів напруги до 60 В і функція гарячого підключення приймача та передавача до каналу зв'язку Гарантовано забезпечує прийомо-передачу даних на швидкості до 10Мбіт/с. Драйвера MAX3443E можуть підключатися до загальної шини до 128 штук. Організацію локальної мережі за участю драйвера MAX3443E наведено на малюнку 1.27.

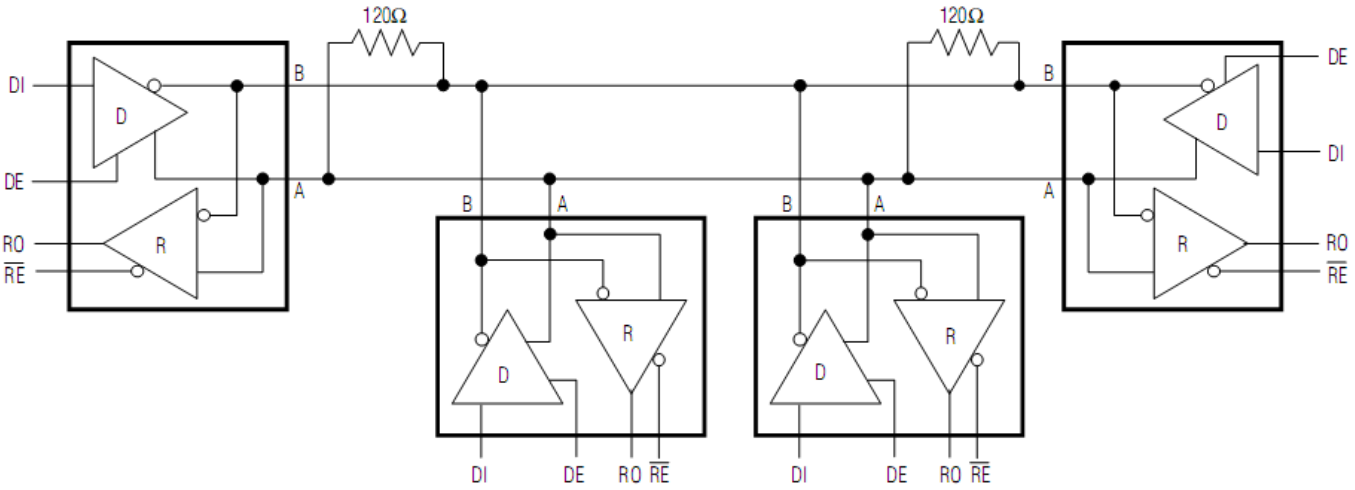

Рисунок 1.27 – Організація локальної мережі за участю драйвера MAX3443E

Паралельно драйверам MAX3443E підключаються два термінальні резистори опору 120 Ом.

Призначення, тип та нумерація висновків драйвер інтерфейсу RS-485 марки MAX3443E наведено у таблиці 1.13.

Таблиця 1.13 – Призначення, тип та нумерація висновків драйвера інтерфейсу RS-485 MAX3443E

Позначення	Номер виводу	Опис
RO	1	Вихід приймача
$\overline{RE}$	2	Включення приймача
DE	3	Включення передавача
DI	4	Вхід передавача
GND	5	Заземлення
A	6	Неінвертуючий вихід приймача
B	7	Інвертуючий вихід приймача
Vcc	8	Живлення

Драйвер інтерфейсу RS485c підключається за типовою схемою включення, наведеною в описі розробника [8].

Для реалізації послідовного інтерфейсу передачі даних RS-232c застосовуються драйвера інтерфейсу RS-232c марки MAX1336E фірми MAXIM.

Схематичне зображення драйвера інтерфейсу RS-485 наведено малюнку 1.28.

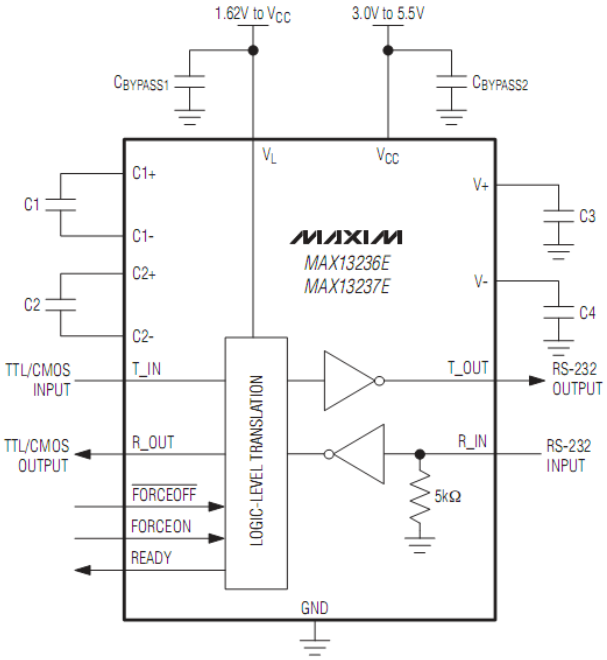

Рисунок 1.28 – Схематичне зображення MAX1326E

Призначення, тип та нумерація висновків драйвер інтерфейсу RS-232 з марки MAX3443E наведено в таблиці 1.14.

Таблиця 1.14 – Призначення, тип та нумерація висновків драйвера інтерфейсу RS-485 MAX13236E

Позначення	Номер виводу	Опис
READY	14	Готовність до передачі даних
C1+	16	Позитивний вивід для конденсатора
V+	15	Позитивне живлення плати
C1-	1	Негативний вивід для конденсатора
C2+	2	Позитивний вивід для конденсатора
C2-	3	Негативний вивід для конденсатора
V-	4	Негативне живлення плати
RIN	5	Вхід приймача RS-232c
ROUT	6	Вихід приймача
VL	7	Живлення яке визначає логіку
TIN	8	Вхід передавача
FORCEON	9	Включення захисту
TOUT	10	Вихід передавача RS-232c
GND	11	Заземлення
VCC	12	Живлення
$\overline{\text{FORCEOFF}}$	13	Виключення захисту

Драйвер інтерфейсу RS232c підключається за типовою схемою включення, наведеною в описі розробника [7].

Для реалізації каналу Ethernet використовується Etherne-контролер CP2200. CP220x є малогабаритні однокристальні IEEE802.3 10-BaseT Ethernet-контролери і призначені для інтеграції Ethernet-інтерфейсу в різні системи. Дані ІС реалізують функції MAC- і PHY-рівень стека мережевих протоколів і забезпечують доступ до Ethernet-мереж будь-яким мікроконтролерам або хост пристроям, що мають не менше 11 вільних ліній введення-виведення. CP2200 випускається в 48-вивідному корпусі типу TQFP розміром 9x9мм.

CP2201 відрізняється від нього типом корпусу (28-висновний QFN розміром 5x5мм), реалізацією драйвера світлодіодів та неможливістю підтримки не мультиплексованого режиму взаємодії із зовнішньою хост-системою. Структурна схема контролера CP2200 представлена на рис.1.29.

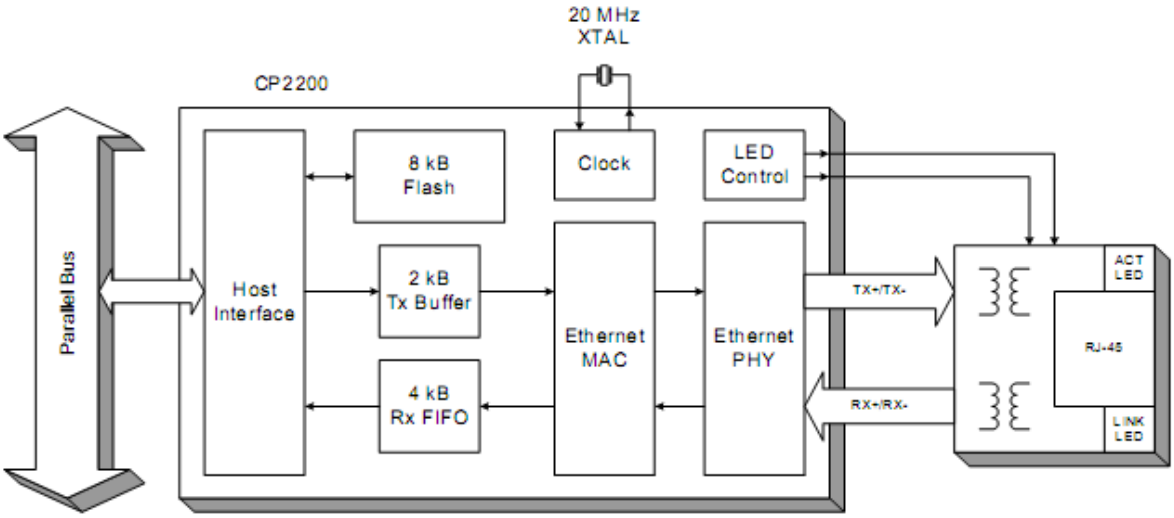

Рисунок 1.29 – Структурна схема CP2200

Як видно з рис.1,29 CP2200 містить паралельний хост-інтерфейс, енергонезалежну пам'ять типу Flash EEPROM об'ємом 8 кбайт, 2-кілобайтний буфер передачі, 4-кілобайтний буфер прийому типу FIFO, Ethernet-контролери MAC- та PHY-рівнів, а також тактовий генератор та драйвер світлодіодів. Модуль паралельного хост-інтерфейсу дозволяє передавати дані зі швидкістю до 30Мбіт/с, працює з шинами формату Intel (Motorola) та підтримує 8-розрядні (мультиплексований та не мультиплексований) режими взаємодії із зовнішньою хост-системою.

Даний модуль забезпечує доступ до регістрів спеціального призначення, за допомогою яких здійснюються прийом та передача даних, а також управління функціонуванням та контроль стану всіх модулів CP2200. Крім того, хост-інтерфейс здатний генерувати сигнал переривання, який може пробуджувати хост-процесор з режиму сну при отриманні пакетів або при підключенні CP2200 до мережі (Wake-on-LAN).

Загалом підтримується 14 подій, які можуть спричинити переривання. Якщо хост-процесор не має окремого висновку для підключення сигналу переривання, він може періодично опитувати регістри стану переривань CP2200 з метою виявлення відповідних подій.

Передача даних через мережу Ethernet здійснюється за допомогою буфера передачі обсягом 2 кбайта. CP2200 забезпечує дуже простий інтерфейс передачі Ethernet-пакетів, що вимагає від хост-системи завантажити в буфер передачі лише MAC-адреси відправника і одержувача пакета (по 6 байт кожен), довжину і тип пакета (2 байти) і дані (46-1500байт). Формування решти полів пакету відповідно до специфікації IEEE 802.3 (преамбула, ознака початкового кадру, контрольна сума CRC), а також доповнення поля даних нулями (при необхідності) здійснюється автоматично.

Інтерфейс прийому Ethernet-даних крім буфера прийому FIFO об'ємом 4 кбайта містить також буфер швидкого перетворення адрес (TBL), приймальний фільтр та хеш-таблицю. Буфер TBL складається з 8 записів, кожна з яких містить початкову адресу (у буфері прийому), довжину та іншу інформацію (тип, ознаки помилки CRC, неповного пакета та ін) для одного отриманого пакета, тобто в буфері прийому може знаходитися до 8 пакетів даних одночасно. Прийомний фільтр і хеш-таблиця дозволяють налаштувати CP2200 таким чином, щоб у процесі прийому небажані пакети відкидалися автоматично.

Приймальний фільтр відкидає пакети за їх типом, проте підтримуються не всі типи пакетів (тільки широкомовні, групові пакети, пакети з помилкою CRC та неповні довжиною менше 64 байт). Тому в CP2200 реалізована можливість довільного доступу до даних у буфері прийому, що дозволяє хост-процесору прочитати певні байти прийнятого пакета та вирішити, чи варто копіювати весь пакет даних на згадку про хост-систему.

За допомогою хеш-таблиці можна встановити діапазон MAC-адрес, які будуть дозволені для запису в буфер прийому; пакети з адресами, які не потрапляють у встановлений діапазон, відкидатимуться.

Серед інших функцій Ethernet-інтерфейсу, що реалізуються вбудованими контролерами PHY- і MAC-рівнів автоматично, слід виділити наступні:

- Розширення коротких пакетів до мінімальної довжини;
- обчислення та перевірка контрольної суми (CRC);
- Перевірка довжини пакета;
- повна сумісність із стандартами 100/1000 Base-T;
- Підтримка стандартів Plug-and-Play (auto-negotiations, auto-polarity);
- дуплексний та напівдуплексний режими роботи;

- Виявлення колізій з автоматичним повтором передачі;

- тестування та налагодження в режимі поворотної петлі (можливі два режими: із замиканням каналу передачі на PHY-рівні або на MAC-рівні);

- перевірка наявності зв'язку із мережею;

- Виявлення та придушення шумів на прийомі;

- відсікання некоректних пакетів під час передачі.

Слід також відзначити, що буфери прийому і передачі Ethernet-інтерфейсу відображені водному і тому ж діапазоні адресного простору і звернення до них здійснюється за допомогою одного всього покажчика адреси, проте реєстри даних (для операцій читання-запису) використовуються різні.

Крім буферів прийому та передачі в CP2200 є ще один блок пам'яті, вбудована Flash-пам'ять об'ємом 8 кбайт. Вона розміщена в окремому адресному просторі і доступ до неї здійснюється за допомогою власних покажчика адреси та регістру даних. Ця пам'ять може використовуватися для зберігання користувачів констант, контенту вебсервера, а також в якості енергонезалежної пам'яті загального призначення.

При виготовленні IC в останні шість осередків цієї пам'яті на заводі записується унікальна 48-розрядна MAC-адреса, що в більшості випадків дозволяє не присвоювати серійні номери розроблюваним пристроям.

Flash-пам'ять стирається 512-байтними секторами і програмується по-байтно. Управління операціями записування-стирання Flash-пам'яті здійснюється автоматично апаратними засобами і не перешкоджає роботі паралельного хост-інтерфейсу. Хост-система повідомляється про закінчення операцій доступу до Flash-пам'яті за допомогою переривань.

Для тактування CP2200 може використовуватися вбудований тактовий генератор (в цьому випадку необхідно підключити зовнішній кварцовий резонатор 20 МГц), або зовнішній сигнал тактування. При зникненні з будь-яких причин тактових імпульсів буде згенеровано сигнал скидання тривалістю близько 1 мс, у тому числі на зовнішньому виведенні /RST, який можна використовувати для скидання зовнішніх модулів.

Сигнал скидання генерується також при зникненні напруги живлення або зниження його граничну величину.

Типові схему включення Ethernet-контролерів CP2200, рекомендовані компанією Silicon Laboratories наведено на рис. 1.30.

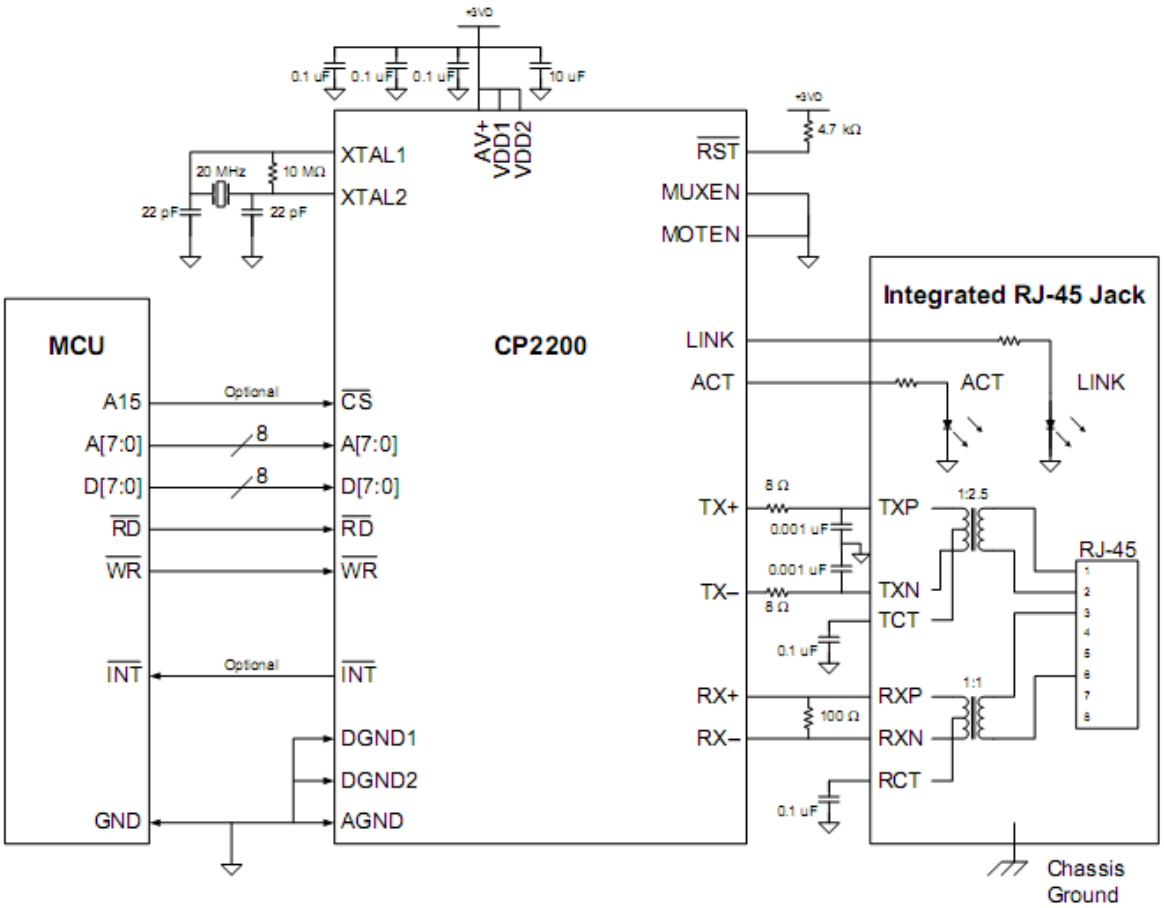

Рисунок 1.30 - Типова схема включення CP2200

Значення висновків мікросхеми CP2200 наведено у таб. 1.15.

Таблиця 1.15 - Значення висновків мікросхеми CP2200

Позначення	Номер виводу	Тип виводу	Опис
AV+	5	Живлення	Аналогове живлення +5В
AGND	4		Аналогова земля
VDD1	13	Живлення	Цифрове живлення +5В
DGND1	14		Цифрова земля
VDD2	30	Живлення	Цифрове живлення +5В
DGND2	31		Цифрова земля
RST	15	D I/O	Перезавантаження пристрою

Продовження таблиці 1.15

LINK	3	D Out	Сигнал з’єднання
ACT	2	D Out	Сигнал активності
XTAL1	46	A In	Вхід тактового генератора
XTAL2	45	A Out	Вихід тактового генератора
TX+	9	A Out	10Base-T передавач
TX–	10	A Out	10Base-T передавач
RX+	7	A In	10Base-T приймач
RX–	6	A In	10Base-T приймач
MOTEN	43	D In	Включення шини Моторола
MUXEN	44	D In	Включення шини Інтел
INT	42	D In	Сигнал переривання
CS	41	D In	Вибір пристрою
RD	39	D In	Читання
WR	40	D In	Запис
ALE	32	D In	Вибір шини
A0-A7	27,28,29,32, 33,34,37,38	D In	Шина адреси
D0-D7	16-23	D I/O	Шина даних

1.3.4 Розробка та розрахунок важливих електричних схем вузлів та блоків пристрою.

Розрахунок продуктивності Ethernet. Припустимо, що в черзі на передачу пакета знаходяться N вузлів. Число являє собою повне завантаження системи і передбачається, що воно більше 1.

У Ethernet-системі існує конфліктний інтервал, або час, протягом якого вузли намагаються виробляти передачу, перш ніж їм буде надана гарантія на управління шиною. Протягом цього часу, рівного тривалості 512 бітів, можливі конфлікти через затримку поширення сигналу по мережі.

Якщо прийняти, що вузол має можливість передачі у будь-який інтервал часу, рівну 1/N, то ймовірність затримки становить [1 - 1/N].

Імовірність (P) того, що точно один вузол спробує передавати в цьому тимчасовому інтервалі і досягне успіху в отриманні контролю за шиною, дорівнює:

$$P = \left[1 - \left\{\frac{1}{N}\right\}\right]^{N-1}, \qquad (1.8)$$

Де P - ймовірність, що шина буде вільна;

N – кількість вузлів.

Тобто ймовірність того, що очікування не буде, перш ніж буде отримано контроль за шиною, є P.

Імовірність очікування лише одного часового інтервалу становить, а ймовірність очікування N часових інтервалів - $P(1-P)^N$. Це геометрична послідовність із середнім значенням (1 – P)/P.

Отже, середня кількість тимчасових інтервалів, які вузол повинен очікувати, перш ніж він отримає доступ до шини, становить:

$$S = \frac{(1-P)}{P}, \qquad (1.9)$$

де S – середній час очікуваних інтервалів.

Для N=10 ймовірність вільної шини при зверненні пристрою становить:

$$P = \left[1 - \left\{\frac{1}{N}\right\}\right]^{N-1} = \left[1 - \left\{\frac{1}{10}\right\}\right]^{10-1} = 0,387, \qquad (1.10)$$

Тоді середня кількість очікуваних інтервалів:

$$S = \frac{(1-P)}{P} = \frac{(1-0,387)}{0,387} = 1,58, \qquad (1.11)$$

Ефективність Е визначається відсотковим співвідношенням часу, коли мережа передає корисні дані. Кожен пакет має максимум 192 службові біти (витрати), що складаються з:

7 байтів преамбули,

1 байт SFD,

6 байтів адреси призначення,

6 байтів адреси джерела,

2 байти індикатора довжини,

4 байти CRC.

Максимальний розмір пакету складає 1526 байтів, або 12208 бітів, а мінімальний розмір пакета - 72 байти, або 576 бітів. Між кожним кадром є проміжок, що дорівнює часу 96 бітам, що з швидкості 20 Мбіт/с становить 4,8 мікросекунд.

Ефективність Е для мінімального та максимального пакета розраховується за формулою 1.12:

$$Å = \frac{Dp}{(Dp + O + T)}, \qquad (1.12)$$

де Dp – корисні дані;

O – очікування;

T – час затримки.

Для кадру максимального розміру:

$$Å_{MAX} = \frac{6104}{(6104 + 256 + 48)} = 95,2\%, \qquad (1.13)$$

Для мінімального розміра:

$$Å_{MIN} = \frac{288}{(288 + 256 + 48)} = 48,6\%, \qquad (1.14)$$

Зменшення ефективності викликається збільшується частиною часу фіксованих витрат і проміжку між кадрами.

1.3.5 Розробка програмного забезпечення проектованого пристрою.

Розробка алгоритму обчислення контрольної суми. Для вирахування контрольної суми методом циклічного надлишкового коду передавач виконує алгоритм:

- взяти повідомлення, що складається із потоку бітів;

- Додати 16 нулів з правого боку повідомлення;

- розділити по модулю 2 на наступне число, дільник (або породжує поліном), наприклад, 11000000000000101 (CRC-16), а результуючий залишок буде називатися контрольною сумою CRC;

- додати залишок у вигляді 16-розрядного числа до потоку вихідного повідомлення (тобто замінити 16 нулів 16-розрядним залишком) і передати це на приймач.

Приймач після отримання пакета даних проводить такі операції:

- бере повне повідомлення плюс біти контрольної суми CRC і ділить те саме число, яке використовував передавач;

- якщо помилок немає, то результуючий залишок складатиметься з нулів (згідно з рівнянням 1.2 розділу 1);

- якщо помилки є, то залишок не дорівнюватиме нулю.

Загальний алгоритм обчислення контрольної суми CRC16 для передавача зображено рисунку 1.31.

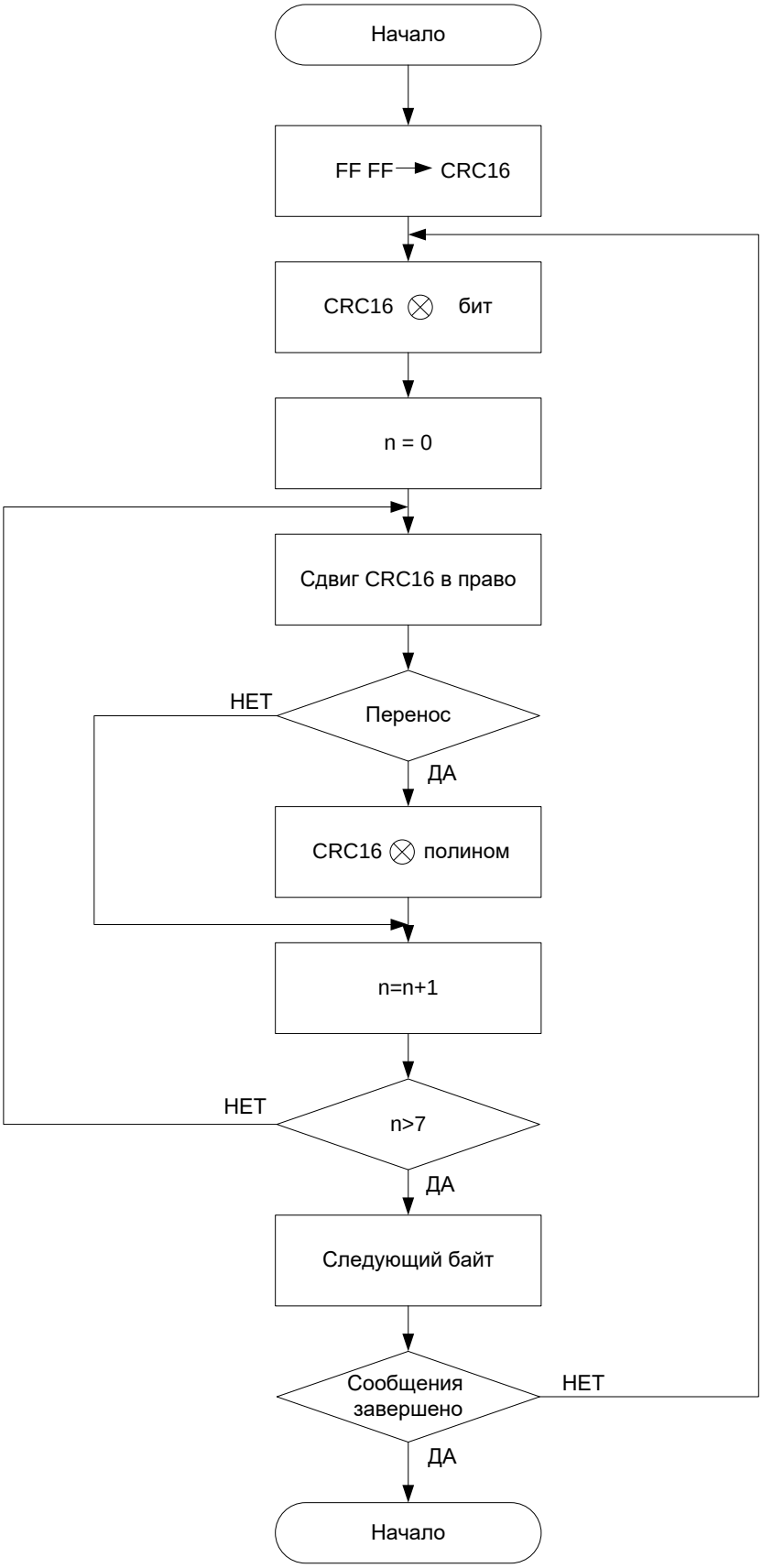

Рисунок 1.31 – Алгоритм обчислення контрольної суми

2 ТЕХНІКО-ЕКОНОМІЧНА ЧАСТИНА

2.1 Розрахунок собівартості проектованого пристрою

Собівартість пристрою (установки) - це виражені у грошовій формі поточні витрати підприємства на її виробництво та збут. Витрати виробництва пристрою (установки) формують виробничу собівартість, а витрати виробництва і збут - повну собівартість. Розрахунок собівартості пристрою (установки) за статтями витрат називається калькуляцією.

Витрати, пов'язані з виробництвом та збутом реалізацією пристрою (установки) групуються за такими статтями:

1. Матеріали та комплектуючі.
2. Основна вести.
3. Додаткова вести.
4. Відрахування на соціальні заходи.
5. Витрати на утримання та експлуатацію обладнання.
6. Загальновиробничі витрати.
7. Адміністративні витрати.
8. Витрати збут.

2.1.1 Матеріали та комплектуючі.

Витрати на матеріали та комплектуючі вироби визначаються виходячи з ціни за одиницю матеріалу/комплектуючого та їх необхідної кількості. Результати розрахунків за цією статтею наведено у таблиці 2.5.

Витраті матеріали складають 5 відсотків.

З урахуванням транспортно-заготівельних витрат (кт-с = 5 ÷ 15%) вартість комплектуючих та матеріалів розраховується за формулою 2.2.

Таблиця 2.5 - Розрахунок витрат за комплектуючі

№ з/п	Найменування комплектуючих	Кількість, шт	Ціна од., грн	Вартість, грн
<u>Мікросхеми</u>				
1	ATMEGA128-16AU	2	225	450
2	MAX222EPN	2	36,8	73,6

3	MAX490CPA	4	52,81	211,24
4	CP2200-GQR	1	141.48	141.48
5	M27C4001-10F1	1	82	82
6	74HC245M1R	1	15	15
Конденсатори				
7	GRM1885C1H220J	11	1,5	16,5
8	GRM188F51H103Z	8	1,5	12
9	GRM1885C1H561J	1	0,36	0,36
Резистори				
10	С2-23 імп. 0.25 Вт, 1%, 10 Ом	2	0,16	0,32
11	С2-23 імп. 0.25 Вт, 1%, 100 Ом	1	0,16	0,16
12	С2-23 імп. 0.25 Вт, 1%, 1.0 МОм	3	0,16	0,48
Інші елементи				
13	HCMOS 6МГц	2	47	94
14	VCHO 20МГц	1	70	70
15	DB- 9F	2	6	12
16	TP8P8C	1	50	50
17	FDC-16	3	4,5	13,5
18	6P4C/B	4	2,25	9
Всього:				**1102**

$$KM=(K+0,05K)\cdot(100+k_{т-з})/100 = 1272 \text{ грн.} \qquad (2.2)$$

2.1.2 Витрати на основну заробітну плату (Зо):

$$З_о = \sum_{i=1}^{n} ТС \cdot Ч = 24 \times 11,36 = 272 \text{ грн.} \qquad (2.3)$$

де $ТС$ – тарифна ставка робітника, що задіяний у виробництві устрою, грн./година;

$Ч$ – витрачене робітником час на виробництво й налагодження пристрою;

n – кількість працівників задіяних у виробництві.

Годинна тарифна ставка розраховується виходячи із величини місячного окладу спеціаліста:

$$С_г = \frac{Т_м}{В_ф \times 8} = \frac{2000}{22 \times 8} = 11{,}36 \text{ грн} \qquad (2.4)$$

Де $Т_м$ – місячний оклад (ставка) спеціаліста, грн;

$В_ф$ - фактично відпрацьований час за розрахунковий період (місяць), днів (змін).

Додаткова заробітна плата (10 ÷ 30% від Зо):

$$З_д = З_0 \times \frac{К_д}{100} = 272 \times 0{,}2 = 54{,}4 \text{ грн} \qquad (2.5)$$

де $К_д$ – відсоток додаткової заробітної плати.

Відрахування на соціальні заходи включають відрахування від суми основної та додаткової заробітної плати за встановленими ставками:

- на обов'язкове державне пенсійне страхування;
- на державне страхування від нещасних випадків;
- на обов'язкове державне соціальне страхування на випадок безробіття;
- У зв'язку з тимчасовою втратою працездатності та витратами, обумовленими народженням дитини та похованням

$$В_{св} = (З_0 + З_д) \times \frac{36{,}3}{100} = (272 + 54{,}4) \times 0{,}363 = 119{,}46 \text{ грн}, \quad (2.6)$$

Витрати на утримання та експлуатацію обладнання.

У разі, якщо обладнання знаходиться на балансі підприємства витрати на утримання та експлуатацію обладнання (ВУЕ) = основна зарплата *% ВУЕ, (приймаємо % ВУЕ = 120 ÷ 150%).

$$ВУЕ = З_0 \times \frac{ВУЕ\%}{100} = 272 \times 1{,}3 = 353{,}6 \text{ грн}, \quad (2.7)$$

Загальновиробничі витрати.

Є витрати, пов'язані з управлінням підрозділом, витрати на службові відрядження співробітників підрозділу (цеху), амортизаційні відрахування від вартості основних фондів загальноцехового призначення і т.д.

Визначаються у розмірі 130 ÷ 250% від основної зарплати.

$$В_з = З_0 \times 2 = 272 \times 2 = 544 \text{ грн}, \quad (2.8)$$

Сума статей 1.1-1.6 складає виробничу собівартість устрою (установки).

Адміністративні витрати.

Можуть включати:

- витрати, пов'язані з управлінням підприємства;
- витрати на службові відрядження адміністрації підприємства;
- витрати на пожежну та сторожову охорону;
- витрати, пов'язані з підготовкою (навчанням) та перепідготовкою кадрів;
- витрати перевезення працівників до місця роботи і назад;
- витрати на сплату процентів за фінансові кредити, а також процентів за товарні та комерційні кредити, витрати, пов'язані зі сплатою процентів за користування матеріальними цінностями, взятими в оренду (лізинг);
- витрати, пов'язані з оплатою послуг комерційних банків та інших кредитно-фінансових установ;
- податки, відрахування.

Визначаються у вигляді 140-200% від основний зарплати.

$$В_а = З_0 \times 1{,}5 = 272 \times 1.5 = 408 \text{ грн}, \quad (2.9)$$

Витрати збут.

Включають витрати на рекламу і перед реалізаційну підготовку пристрою (установки). Орієнтовно ці витрати визначаються у вигляді 5-10% від виробничої собівартості.

$$В_п = С_в \times 0{,}05 = 2605 \times 0{,}05 = 130 \text{ грн}, \qquad (3.10)$$

Сума статей 1-8 становить повну собівартість пристрою (установки). Виробнича собівартість розраховується як сума 1-6 статей калькуляції. Калькуляція собівартості пристрою (установки) зводиться до таблиці 2.6.

Таблиця 2.6 – Калькуляція собівартості пристрою (установки)

Найменування статей калькуляції	**Проектний варіант**
1. Матеріали і комплектуючі	1272
2. Розходи на основну заробітню плату	272
3. Додаткова заробітня плата	54,4
4. Відчислення на соціальні заходи	119,46
5. Розходи на зміст та експлуатацію обладнання	343,6
6. Загальновиробничі розходи	544
Виробнича собівартість	**2605**
7. Адміністративні розходи	408
8. Розхлди на збут	130
Повна собівартість пристрою (установки)	**3143**

2.2 Визначення ціни пристрою (налаштування)

У ринковій економіці існують різні методи ціноутворення: з/в та прибуток, забезпечення фіксованого обсягу прибутку, залежно від рівня попиту тощо. [3].

Розрахунок оптової ціни пристрою (установки) проведемо за схемою «собівартість плюс прибуток».

$$Ц_{опт} = С + П, \qquad (2.11)$$

де С - повна собівартість пристрою (установки),

П – величина прибутку.

Прибуток визначається виходячи з нормативу (показника) рентабельності виробництва продукції, що встановлюється підприємством:

$$R = \frac{П}{C} \cdot 100\% \qquad (2.12)$$

де R - рентабельність пристрою (установки), приймається у розмірі до 35% від його собівартості.

Тоді оптова ціна пристрою (установки) визначається:

$$Ц_{опт} = C + \frac{R \cdot C}{100} = 3143 + 1100 = 4243 грн. \qquad (2.13)$$

Відпускна ціна пристрою (установки) включає податок на додану вартість:

$$Ц_{розд} = Ц_{опт} \cdot 1,2 = 5091 грн, \qquad (2.14)$$

де 20% – ПДВ.

Позитивні сторони даної методики полягають у її простоті, комплексної очевидності такої функції ціни як відшкодування витрат на виробництво і забезпечення прибутковості від створення та реалізації пристрою (установки). Недолік цієї методики полягає в тому, що вона не враховує ринкові фактори ціноутворення і насамперед попит. Однак в умовах ринкової економіки існують ситуації, коли підприємствам доцільно її застосовувати: в умовах відсутності конкуренції (монополії), при обмеженні рентабельності продукції з боку держави, виконанні одноразових замовлень, виготовленні оригінальної продукції.

Необхідно відзначити, що для встановлення реальної ціни яка б відповідала умовам існуючого ринку пристрою (установки), необхідні відповідні маркетингові дослідження.

2.3 Визначення річних витрат користувача

Річні експлуатаційні витрати ПРЕ включають витрати на електроенергію (живлення пристрою), заробітну плату обслуговуючого персоналу, амортизаційні відрахування, витрати на поточний ремонт та інші витрати.

Витрати на електроенергію (живлення пристрою) визначаються за такою формулою:

$$Ве= W \cdot Ce \cdot t = 0,5 \cdot 0,57 \cdot 8760 = 2496 грн, \qquad (2.15)$$

де W – потужність пристрою, кВт;

С – вартість одного кВт•год, грн/кВт•год (див. тариф для промислових користувачів);

t - час роботи пристрою на рік, год.

Розрахунок витрат на заробітну плату персоналу, що обслуговує пристрій, проводиться за необхідністю (залежно від паспорта спеціальності робітників) за формулою 2.16:

$$З_р = \sum_{j=1}^{m} T_{mj} \times k_{зj} \times k_{дj} \times k_C \times Ч_j \times 12 = 1500 \times 0.005 \times 1,3 \times 1,36 \times 1 = 13,26 \text{ грн} \qquad (2.16)$$

де Тмj – місячний оклад (ставка) спеціаліста j-ї професії;

kзj - коефіцієнт зайнятості (залежить від часу обслуговування пристрою фахівцем j-ї професії);

kдj – коефіцієнт, що враховує додаткову зарплату

(приймається kдj = 1,1-1,3);

kС-коефіцієнт, що враховує нарахування на заробітну плату

(береться kС = 1,363)

Чj-кількість фахівців j-ї професії;

m – кількість професій.

Результати розрахунку заробітної плати обслуговуючого персоналу необхідно подати в таблиці 2,7:

Таблиця 2.7 - Розрахунок заробітної плати обслуговуючого персоналу

Вид обслуговування	Професія	Оклад, грн	Коефіцієнт зайнятості	Кількість персонала, чол		Річна заробітня платня, грн	
				по приборуаналога	по проекту прибора		
Ремонт	електронщик	1500	0,005	1	1	13,26	13,26

Річна сума амортизаційних відрахувань визначається за відповідними нормами амортизації від первісної вартості пристрою, що включає: ціну пристрою; витрати на транспортування та монтаж, що приймаються у розмірі 5-15% від ціни.

Амортизацію електронних пристроїв (установок) доцільно нараховувати за прямолінійним методом:

$$A = F \cdot a = 0{,}2 \cdot 5303 = 1069 \text{грн}, \qquad (2.17)$$

де F - первісна вартість пристрою;

а – річна норма амортизації.

$$F = Ц_{розд} + В_{тм} = 5091 + 254 = 5345 \text{грн}, \qquad (2.18)$$

де

Црозд – роздрібна (договорна) ціна пристрою;

Втм - витрати на транспортування та монтаж пристрою.

Річна норма амортизації обчислюється прямолінійним методом щодо терміну корисного використання пристрою:

$$a = 1/T_{вик}, \qquad (2.19)$$

де Твік - термін корисного використання пристрою років. Згідно з новим податковим кодексом України допускається не менше 2 років.

Витрати на поточний ремонт та інші витрати включають вартість електронних елементів, які виходять з ладу протягом року і вартість демонтажних і монтажних робіт.

Витрати на ремонт приймаються рівними 5% від капітальних витрат з урахуванням прогнозування відмов:

$$Впр=F\cdot k_{пр}/100=267\text{грн}, \qquad (2.20)$$

де Кпр – відсоток витрат на поточний ремонт.

Результати розрахунку необхідно звести до таблиці 2.8.

Таблиця 2.8 – Річні експлуатаційні витрати користувача

Статті експлуатаційних розходів	Річні експлуатаційні розходи, грн.	
	Пристрій-аналог	Проектований пристрій
Вартість електроенергії	4993	2496
Амортизаційні відчислення	1457	1069
Розходт на поточний ремонт	364	267
Всього (Вре)	6814	3832

2.4 Розрахунок повної ціни користувача

Ціна користувача Цкор включає капітальні вкладення F і сумарні річні витрати на експлуатацію пристроїв Протягом терміну використання пристрою:

$$Ц_{кор}= F + Вре \cdot Т_{вик}=5345+3832\cdot 5=24505\text{грн}, \qquad (2.21)$$

$$Ц_{кор\_а}= F + Вре \cdot Т_{вик}=(7289+0{,}2*7289)+5*6814=42816\text{грн}. \qquad (2.21)$$

2.5 Оцінка конкурентоспроможності проектного устрою

Для оцінки конкурентоспроможності застосовується комплексний показник конкурентоспроможності Qкон, що визначається за інтегральними показниками конкурентоспроможності пристроїв:

$$Q_{кон} = Q_{нп}Q_т/Q_е, \qquad (2.23)$$

де Qнп - інтегральний показник за нормативними параметрами (може приймати виключно два значення: нуль, якщо пристрій не відповідає обов'язковим для даного ринку нормам і стандартам (частота і напруга живлення, точність вимірювання, конструктивні розміри, і одиницю, якщо відповідає По проектному пристрої Qнп = 1);

Qт - порівняльна конкурентоспроможність пристрою з його технічним рівнем (наприклад, якщо швидкість передачі, або пропускна здатність пристрою збільшена в 2,5 рази, то Qт = 2,5);

Qе – інтегральний економічний показник.

Конкурентоспроможність за економічними показниками Qе визначаємо за ціною користувача виробів:

$$Q_э = Ц_{кор\_пр} / Ц_{кор\_ан},, \qquad (2.24)$$

де Цкор_пр - ціна використання проектованого пристрою;

Цкор_ан – ціна використання пристрою-аналогу.

Якщо коефіцієнт конкурентоспроможності Qкон більше одиниці, відповідно, проектована система є конкурентоспроможною.

Економічний показник і комплексний показник конкурентоспроможності рівні:

$$Q_э = Ц_{кор\_пр} / Ц_{кор\_ан} = 24505/42816 = 0{,}57,$$

$$Q_{кон} = Q_{нп} Q_т / Q_е = (1+1)/0{,}57 = 2{,}5.$$

Таблиця 2.9 - Порівняння техніко-економічних показників електронної системи-аналогу та проектної електронної системи

Показник	Одиниця виміру	Значення	
		пристрій-аналог	Проектований пристрій (система)
Технічні:			
Потужність пристрою	Вт	1	0,5

Середнє напрацювання навідмову	год	80000	100000
Ймовірність безвідказної роботи	%	0,985	0,98
Габаритні розміри	мм^2	231x110x75	122,5x115x12
Вага	кг	0,35	0,3
Швидкість обробки даних	Кбит/с	10000	6000
Економічні:			
Виробнича собівартість	грн		2605
Повна собівартість	грн		3143
Відпускна ціна	грн	7289	*5091*
Річні експлуатаційні розходи	грн	6814	3832
Повна ціна користувача	грн	*42816*	*24505*
Показник якості за технічним рівнем	–	x	1
Інтегральний економічний показник конкурентоспроможності	–	x	0,57
Комплексний показник конкурентоспроможності	–	x	2,5

2.6 Висновок економічної частини

Розроблений пристрій має інтегральний економічний показник конкурентоспроможності нижче одиниці і комплексний показник конкурентоспроможності дорівнює 2,5, що свідчить про високу

конкурентоспроможність, хоча має однакові технічні характеристики з існуючим аналогом на сучасному ринку електроніки та програмного забезпечення.

Для підвищення економічного ефекту можна зробити низку дій:

1. Використання хворіючи на дешеві комплектуючі;
2. Поліпшення технічних характеристик та надійності приладу;
3. Зменшення споживаної потужності пристроєм.

ВИСНОВКИ

У кваліфікаційній роботі було розглянуто актуальність застосування інтерфейсів передачі даних RS-485 та протоколів передачі даних у розподілених системах збору даних та передачі даних. Було проведено аналіз модифікацій протоколу передачі даних у розподілених системах збору даних, таких як Modbas RTU та Modbas ASCII. При аналізі споживаної обчислювальної потужності розраховано кількість машинних тактів для обчислення контрольної суми CRC-16 при реалізації промислових протоколів. Приведено приклад розрахунку часу реалізації протоколу на мікроконтролерах серії ATmega при тактовій частоті 6 МГц. В результаті кваліфікаційної роботи ґрунтуючись на результатах науково-дослідної роботи, розроблено розподілений модуль збору та передачі даних з вбудованим програмним перетворювачем протоколів ModbasRTU/EthernetIP, та з інтерфейсами послідовної передачі даних RS-485 та RS-232 і трьома шістнадцяти розрядними послідовними від зовнішнього пристрою. Модуль оснащений каналом Ethernet для передачі даних у віддалені місця для серверів та диспетчерів. Крім того, було розроблено алгоритми функціонування розподілених модулів та синтезовано структурні схеми, функціональні схеми та принципові схеми. Розроблений розподільчий модуль містить чотири порти інтерфейсу RS-485 і два порти інтерфейсу RS-232, мінімальний та повний нуль-модемні кабелі, що дозволяють підключати пристрої до комп'ютерів та модемів, а також два порти Ethernet. Пристрій використовується в мережах контролю та управління технічними процесами, в мережах зв'язку та передачі даних у важкій металургії та енергетиці.

СПИСОК ЛІТЕРАТУРИ

1. Передача даних у системах контролю та управління: практичне керівництво.

https://deps.ua/ua/knowegable-base/reference-information/7824.html

2. Застосування SWITCH-технології при розробці прикладного програмного забезпечення для мікроконтролерів. Реалізація протоколу Modbus https://ela.kpi.ua/bitstream/123456789/18920/1/POSIBNIK_MK51%2016%D1%80_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC.pdf

3. CRC16 – Опис алгоритму та приклад розрахунку мовою Сі. http://piclist.by.ru/S¬CRC16¬RUS/CRC16.html.

4. Методи розрахунку CRC16 на мікроконтролерах серії PIC18. http://2aplusa.ru/info/inf2.html.

5. Елементарний посібник з CRC ¬ алгоритмів виявлення помилок / Р.М. Вільямс. Rocksoft Pty. 1993.

6. Atmel AVR Atmega16(L) datasheet.

7. Modbus Communications Protocol Reference Guide, Satek Powerful Solution, 2018

8. Data Sheet. 3Mbps RS-232 Transceivers with Low-Voltage Interface. max13234e-max13237e.

9. Data Sheet. Fail-Safe RS-485/J1708 Transceivers. max3440e-max3444e.

10.Проектування мікропроцесорних систем: Проектування мікропроцесорних систем на базі мікроконтролерів сімейства MCS–51: Периферійні модулі мікроконтролерів сімейства MCS–51: Навчальний посібник, Київ НТУУ “КПІ” 2016

11. Data Sheet. Single-chip ethernet controller CP2200.

ДОДАТОК А

```
union crc_union{
        unsigned short integ;
        BYTE bytes[2];
};
        struct {
        BYTE _1addr;
        BYTE _2cmd;
        BYTE _3data_start_addr_hi;
        BYTE _4data_start_addr_lo;
        BYTE _5data_length_hi;
        BYTE _6data_length_lo;
        BYTE _7crc1;
        BYTE _8crc2;
} zapros;
struct otvet4{
        BYTE _1from_addr;
        BYTE _2cmd;
        BYTE _3byte_count;
        BYTE _4data[4];
        BYTE _8crc1;
        BYTE _9crc2;
} ;
struct otvet6{
        BYTE _1from_addr;
        BYTE _2cmd;
        BYTE _3byte_count;
        BYTE _4data[6];
        BYTE _Acrc1;
        BYTE _Bcrc2;
} ;
struct otvet8{
        BYTE _1from_addr;
        BYTE _2cmd;
```

```
        BYTE _3byte_count;
        BYTE _4data[8];
        BYTE _Ccrc1;
        BYTE _Dcrc2;
} ;

//псевдоніми функцій
unsigned short __fastcall CRC16(unsigned char *Buffer, int Buffer_Length);
BOOL OpenCommPort(int Baudrate, int ByteSize, int Parity, int StopBits);
BOOL __fastcall floatNotNumber(BYTE src[4]);
float __fastcall IEEE754float(BYTE src[4]);
long __fastcall IEEE754long(BYTE src[4]);
double __fastcall IEEE754double(BYTE src[8]);
double ReadDouble(BYTE mbadr,BYTE addr_hi,BYTE addr_lo);
float ReadFloat(BYTE mbadr,BYTE addr_hi,BYTE addr_lo);
long ReadLong(BYTE mbadr,BYTE addr_hi,BYTE addr_lo);

//опис змінних
HANDLE Comport=NULL;
DCB dcb1;
COMMTIMEOUTS comtimeouts;
COMSTAT statistic;
DWORD nNumberOfBytesToWrite; //запись номера бита
DWORD lpNumberOfBytesWritten;
int IntEditTimeout=500;
BOOL BoolInvertedBytes=0;
AnsiString ComPort;
int IntBaudRate,IntParity,IntStopBits,IntByteSize;
BYTE MBadr;
//---------------------------------------------------------------------------
// допоміжна функція обчислення контрольної суми CRC16, яка використовується в
ModbusRTU
unsigned short __fastcall CRC16(unsigned char *Buffer, int Buffer_Length)
{
//
```

```
// Algorithm - CRC16 Standard, Polynom - 8005 (X16 + X15 + X2 + 1)
// Init const = ffff
// Check: CRC16("123456789",9) = BB3D
const unsigned short CRC16_tbl[ 256 ] = {
0x0000, 0xC0C1, 0xC181, 0x0140, 0xC301, 0x03C0, 0x0280, 0xC241, 0xC601,
0x06C0, 0x0780, 0xC741, 0x0500, 0xC5C1, 0xC481, 0x0440, 0xCC01, 0x0CC0,
0x0D80, 0xCD41, 0x0F00, 0xCFC1, 0xCE81, 0x0E40, 0x0A00, 0xCAC1, 0xCB81,
0x0B40, 0xC901, 0x09C0, 0x0880, 0xC841, 0xD801, 0x18C0, 0x1980, 0xD941,
0x1B00, 0xDBC1, 0xDA81, 0x1A40, 0x1E00, 0xDEC1, 0xDF81, 0x1F40, 0xDD01,
0x1DC0, 0x1C80, 0xDC41, 0x1400, 0xD4C1, 0xD581, 0x1540, 0xD701, 0x17C0,
0x1680, 0xD641, 0xD201, 0x12C0, 0x1380, 0xD341, 0x1100, 0xD1C1, 0xD081,
0x1040, 0xF001, 0x30C0, 0x3180, 0xF141, 0x3300, 0xF3C1, 0xF281, 0x3240, 0x3600,
0xF6C1, 0xF781, 0x3740, 0xF501, 0x35C0, 0x3480, 0xF441, 0x3C00, 0xFCC1,
0xFD81, 0x3D40, 0xFF01, 0x3FC0, 0x3E80, 0xFE41,
0xFA01, 0x3AC0, 0x3B80, 0xFB41, 0x3900, 0xF9C1, 0xF881, 0x3840, 0x2800,
0xE8C1, 0xE981, 0x2940, 0xEB01, 0x2BC0, 0x2A80, 0xEA41, 0xEE01, 0x2EC0,
0x2F80, 0xEF41, 0x2D00, 0xEDC1, 0xEC81, 0x2C40, 0xE401, 0x24C0, 0x2580,
0xE541, 0x2700, 0xE7C1, 0xE681, 0x2640, 0x2200, 0xE2C1, 0xE381, 0x2340,
0xE101, 0x21C0, 0x2080, 0xE041, 0xA001, 0x60C0, 0x6180, 0xA141, 0x6300,
0xA3C1, 0xA281, 0x6240, 0x6600, 0xA6C1, 0xA781, 0x6740, 0xA501, 0x65C0,
0x6480, 0xA441, 0x6C00, 0xACC1, 0xAD81, 0x6D40, 0xAF01, 0x6FC0, 0x6E80,
0xAE41, 0xAA01, 0x6AC0, 0x6B80, 0xAB41, 0x6900, 0xA9C1, 0xA881, 0x6840,
0x7800, 0xB8C1, 0xB981, 0x7940, 0xBB01, 0x7BC0, 0x7A80, 0xBA41, 0xBE01,
0x7EC0, 0x7F80, 0xBF41, 0x7D00, 0xBDC1, 0xBC81, 0x7C40, 0xB401, 0x74C0,
0x7580, 0xB541, 0x7700, 0xB7C1, 0xB681, 0x7640,
0x7200, 0xB2C1, 0xB381, 0x7340, 0xB101, 0x71C0, 0x7080, 0xB041, 0x5000,
0x90C1, 0x9181, 0x5140, 0x9301, 0x53C0, 0x5280, 0x9241, 0x9601, 0x56C0, 0x5780,
0x9741, 0x5500, 0x95C1, 0x9481, 0x5440, 0x9C01, 0x5CC0, 0x5D80, 0x9D41,
0x5F00, 0x9FC1, 0x9E81, 0x5E40, 0x5A00, 0x9AC1, 0x9B81, 0x5B40, 0x9901,
0x59C0, 0x5880, 0x9841, 0x8801, 0x48C0, 0x4980, 0x8941, 0x4B00, 0x8BC1,
0x8A81, 0x4A40, 0x4E00, 0x8EC1, 0x8F81, 0x4F40, 0x8D01, 0x4DC0, 0x4C80,
0x8C41, 0x4400, 0x84C1, 0x8581, 0x4540, 0x8701, 0x47C0, 0x4680, 0x8641, 0x8201,
0x42C0, 0x4380, 0x8341, 0x4100, 0x81C1, 0x8081, 0x4040 };

unsigned short CRC16 = 0xffff;
```

```
for (int i=0; i < Buffer_Length; i++)
{
	CRC16 = ( CRC16 >> 8 ) ^ CRC16_tbl[ ( CRC16 & 0xFF ) ^ Buffer ]; }
return CRC16;
}
	BOOL OpenCommPort(int baudrate, int bytesize, int parity, int stopbits)
{
	Comport = CreateFile("COM1",
	GENERIC_READ | GENERIC_WRITE,
	0, NULL, OPEN_EXISTING, 0, NULL );
	if (Comport!=INVALID_HANDLE_VALUE)
	{
		GetCommState(Comport, &dcb1);
		dcb1.BaudRate=baudrate;
	dcb1.fBinary=1;
	dcb1.fParity=1;
		dcb1.fDtrControl=0;
		dcb1.Parity=parity;
		dcb1.ByteSize=bytesize;
	dcb1.StopBits = stopbits;
	dcb1.fOutxCtsFlow=0;
		dcb1.fOutxDsrFlow=0;
		dcb1.fDsrSensitivity=0;
		dcb1.fOutX=0;
		dcb1.fInX=0;

		SetCommState(Comport, &dcb1);

		comtimeouts.ReadIntervalTimeout=0;
		comtimeouts.ReadTotalTimeoutMultiplier=0;
		comtimeouts.WriteTotalTimeoutMultiplier=0;
		comtimeouts.ReadTotalTimeoutConstant=500;
		comtimeouts.WriteTotalTimeoutConstant=500;
		SetCommTimeouts(&Comport,&comtimeouts);
		return true;
```

```
        }
        else return false;
}

//прочитати значення типу float из modbus-slave пристрою з адресою mbadr, адреса регістрів - addr_hi, addr_lo

float ReadFloat(BYTE mbadr,BYTE addr_hi,BYTE addr_lo)
{
        crc_union crc_float, crc_otv_flt;
        otvet4 otvet4f;
        DWORD p;
        crc_float.integ=0;

        zapros._1addr=mbadr;
        zapros._2cmd=0x03;
        zapros._3data_start_addr_hi=addr_hi;
        zapros._4data_start_addr_lo=addr_lo;
        zapros._5data_length_hi=0x00;
        zapros._6data_length_lo=0x02;
        crc_float.integ=CRC16(&zapros._1addr,6);
        zapros._7crc1=crc_float.bytes[0];
        zapros._8crc2=crc_float.bytes[1];
        for(p=0;p<3;p++)
        {
                ZeroMemory(&otvet4f,sizeof(otvet4f));
                PurgeComm(Comport,PURGE_TXCLEAR);
                PurgeComm(Comport,PURGE_RXCLEAR);
                if (WriteFile(
                        Comport,
                        &zapros,
                        sizeof(zapros),
                        &lpNumberOfBytesWritten,
        {
        Sleep(IntEditTimeout);
```

```
        ClearCommError(Comport,NULL,&statistic);
        if (statistic.cbInQue>=sizeof(otvet4f))
                ReadFile( Comport, &otvet4f, sizeof(otvet4f),
                &lpNumberOfBytesWritten, NULL );
                crc_otv_flt.integ=CRC16(&otvet4f._1from_addr,7);
        if ((crc_otv_flt.bytes[0]==otvet4f._8crc1)&&
                (crc_otv_flt.bytes[1]==otvet4f._9crc2))
                {
                        p=4;
                }
        else
                {
                        ZeroMemory(&otvet4f,sizeof(otvet4f));
                }
        }
}
if (floatNotNumber(otvet4f._4data))
        return 0;
        else
        return IEEE754float(otvet4f._4data);
}
//---------------------------------------------------------------------------
//допоміжна функція - перетворення 4 байт в число float
float __fastcall IEEE754float(BYTE src[4])
{
        BYTE aaa;
        float ffloat;
        if (BoolInvertedBytes)
{
        aaa=src[0];
        src[0]=src[3];
        src[3]=aaa;
        aaa=src[1];
        src[1]=src[2];
        src[2]=aaa;
```

```
}
_wmemcpy(&ffloat,src,4);
return ffloat;
}
// допоміжна функція - перевірка згідно IEEE754
	BOOL __fastcall floatNotNumber(BYTE src[4])
{
	return (((src[0]==0x7F) || (src[0]==0xFF)) && (src[1]>0x7F)
	& ((src[1]>0x80) || (src[2]>0x00) || (src[3]>0x00)));
}
//---------------------------------------------------------------------------
//прочитати значення типу long из modbus-slave пристою з адресою mbadr, адреса
регистрів - addr_hi, addr_lo
long ReadLong(BYTE mbadr,BYTE addr_hi,BYTE addr_lo)
{
	crc_union crc_long, crc_otv_lng;
	otvet4 otvet4l;
	DWORD p;
	crc_long.integ=0;
	zapros._1addr=mbadr;
	zapros._2cmd=0x03;
	zapros._3data_start_addr_hi=addr_hi;
	zapros._4data_start_addr_lo=addr_lo;
	zapros._5data_length_hi=0x00;
	zapros._6data_length_lo=0x02;
	crc_long.integ=CRC16(&zapros._1addr,6);
	zapros._7crc1=crc_long.bytes[0];
	zapros._8crc2=crc_long.bytes[1];
	for(p=0;p<3;p++)
	{
		ZeroMemory(&otvet4l,sizeof(otvet4l));
		PurgeComm(Comport,PURGE_TXCLEAR);
		PurgeComm(Comport,PURGE_RXCLEAR);
		if (WriteFile( Comport, &zapros, sizeof(zapros),
			&lpNumberOfBytesWritten, NULL ))
```

```
                {
                        Sleep(IntEditTimeout);
                        ClearCommError(Comport,NULL,&statistic);
                        if (statistic.cbInQue>=sizeof(otvet4l))
                        ReadFile( Comport,&otvet4l,sizeof(otvet4l),
                        &lpNumberOfBytesWritten, NULL );

                        crc_otv_lng.integ=CRC16(&otvet4l._1from_addr,7);
                        if ((crc_otv_lng.bytes[0]==otvet4l._8crc1) &&
                        (crc_otv_lng.bytes[1]==otvet4l._9crc2))
                        {
                                p=4;
                        }
                }}
        return IEEE754long(otvet4l._4data);
}
//---------------------------------------------------------------------------
// допоміжна функція - перетворення 4 байт в число long (власне до стандарту
IEEE754 відношення не має, просто)
long __fastcall IEEE754long(BYTE src[4])
{
        BYTE aaa;
        long ilong;
        if (BoolInvertedBytes)
        {
                aaa=src[0];
                src[0]=src[3];
                src[3]=aaa;
                aaa=src[1];
                src[1]=src[2];
                src[2]=aaa;
        }
_wmemcpy(&ilong,src,4);
return ilong;
}
```

```
//---------------------------------------------------------------------------
// допоміжна функція - перетворення 8 байт в число double
double __fastcall IEEE754double(BYTE src[8])
{
        BYTE aaa;
        double ddouble;
        if (BoolInvertedBytes)
        {
        aaa=src[0];
        src[0]=src[7];
        src[7]=aaa;
        aaa=src[1];
        src[1]=src[6];
        src[6]=aaa;
        aaa=src[2];
        src[2]=src[5];
        src[5]=aaa;
        aaa=src[3];
        src[3]=src[4];
        src[4]=aaa;
        }

_wmemcpy(&ddouble,src,8);
return ddouble;
}
//---------------------------------------------------------------------------
//прочитати значення типу double из modbus-slave пристрою з адресою mbadr,
адреса регістрів - addr_hi, addr_lo
double ReadDouble(BYTE mbadr,BYTE addr_hi,BYTE addr_lo)
{
        crc_union crc_double, crc_otv_dbl;
        otvet8 otvet8d;
        DWORD p;
        crc_double.integ=0;
        zapros._1addr=mbadr;
```

```
        zapros._2cmd=0x03;
        zapros._3data_start_addr_hi=addr_hi;
        zapros._4data_start_addr_lo=addr_lo;
        zapros._5data_length_hi=0x00;
        zapros._6data_length_lo=0x04;
        crc_double.integ=CRC16(&zapros._1addr,6);
        zapros._7crc1=crc_double.bytes[0];
        zapros._8crc2=crc_double.bytes[1];

for(p=0;p<3;p++)
{
        ZeroMemory(&otvet8d,sizeof(otvet8d));
        PurgeComm(Comport,PURGE_TXCLEAR);
        PurgeComm(Comport,PURGE_RXCLEAR);
        if (WriteFile (Comport, &zapros,sizeof(zapros),
                &lpNumberOfBytesWritten, NULL ))
        {
                Sleep(IntEditTimeout);
                ClearCommError(Comport,NULL,&statistic);
                if (statistic.cbInQue>=sizeof(otvet8d))
                ReadFile( Comport,&otvet8d, sizeof(otvet8d),
                &lpNumberOfBytesWritten, NULL );
                crc_otv_dbl.integ=CRC16(&otvet8d._1from_addr,11);
                if ((crc_otv_dbl.bytes[0]==otvet8d._Ccrc1) &&
                (crc_otv_dbl.bytes[1]==otvet8d._Dcrc2))
                {
                        p=4;
                }
        }
}
return IEEE754double(otvet8d._4data);
}
```