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

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

ДИПЛОМНИЙ ПРОЕКТ (РОБОТА) БАКАЛАВРА

ПОЯСНЮВАЛЬНА ЗАПИСКА

НА ТЕМУ:

Програмні засоби забезпечення взаємодії мікропроцесорного субкомплекса контролю та управління АКНП з інтерфейсним модулем

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

Науковий керівник роботи:	_____ (підпис)	Лифар О. К. (ініціали, прізвище)
Консультант з охорони праці:	_____ (підпис)	ст. викл. Критська Я. О. (ініціали, прізвище)
Студент:	_____ (підпис)	Білов В. В. (ініціали, прізвище)
Група:		КІ-14бд

Сєвєродонецьк 2018

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

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

ЗАТВЕРДЖУЮ:
Завідувач кафедри
І.С. Скарга-Бандурова
«_____» _____20____р.

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

Білову Владиславу Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Програмні засоби забезпечення взаємодії мікропроцесорного субкомплекса контролю та управління АКНП з інтерфейсним модулем
керівник проекту (роботи) Лифар О. К.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "14" 05 2018 р. № 117/48
2. Термін подання студентом роботи
3. Вихідні дані до роботи матеріали переддипломної практики

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Використання проекту у комплекті пристроїв АКНП.
2. Огляд мікроконтролерної системи.
3 Прийом та передача інформації по інтерфейсам Ethernet та RS-422.

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

6. Консультанти розділів проекту (роботи)

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

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

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

Студент _____ **Білов В. В.**
(підпис) (прізвище та ініціали)

Науковий керівник _____ **Лифар О. К.**
(підпис) (прізвище та ініціали)

Р Е Ф Е Р А Т

Пояснювальна записка до дипломного проекту (роботи) бакалавра: 152с., 4 рис., 10 табл., 19 бібліографічних джерел посилань, 3 додатки.

Об’єкт розробки: Розробка програмного забезпечення для прийому, передачі інформації по інтерфейсам Ethernet та RS-422.

Мета роботи: розробка програмного забезпечення для друкованої плати, яка отримує інформацію, перетворює її та відправляє на пристрій реєстрації та відображення УРО-1.

В проекті виконано:

1. У вступі розкривається актуальність дослідження вибраної роботи, ставиться проблема, ціль та задачі дослідження, обирається об’єкт, вивчається методологічна база дослідження, її теоретична та практична значимість.
2. У першій частині виконується аналіз структурної та функціональної організації системи, методів вирішення поставленої задачі, аналіз апаратних та програмних засобів, використовуваних при її вирішенні, запропоновано процес розробки апаратної частини, а саме обґрунтування та вибір платформи для вирішення задачі.
3. У другій частині розглянуто мікропроцесор та мікропроцесорну систему. Виконано аналіз середовища QtCrearot. Описані його можливості та функції.
4. У третій частині описано запропоновані рішення, розроблені алгоритми та їх реалізація, процес розробки програмного забезпечення, його структура, можливості та вимоги до наявних ресурсів.
5. Заключення посвячено загальним висновкам щодо включення цього проекту у виробничий процес для більшої продуктивності.

Практичне значення, галузь застосування роботи: Результати дослідженнь та розробки можуть бути використані на атомних електростанціях.

Ключові слова: ОБРОБКА ІНФОРМАЦІЇ, ПЕРЕТВОРЕННЯ ПАКЕТІВ ДАНИХ, QT CREATOR.

Умови одержання дипломного проекту: СНУ ім. В. Даля, пр. Центральний 59-А,. м. Сєвєродонецьк, 93400.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

МІф-1 - модуль інтерфейсний

АКНП - апаратура контролю нейтронного потоку

АСУ ТП - автоматизована система управління технологічним процесом

АЕС - атомна електростанція

ПТО – пристрій технічного обслуговування

МЕК - Міжнародна електротехнічна комісія

МАГАТЕ - Міжнародне агентство по атомній енергії

ПТК - програмно-технічний комплекс

МСКУ - мікропроцесорні субкомплекси контролю і управління

ІЗІ – інструментальні засоби імітації

ПЛК - програмовані логічні контролери

СНЕ - системи нормальної експлуатації

СБ – управляючі системи безпеки

РВ - реакторне відділення

ТЗА - технічні засоби автоматизації

КМп - контролери мікропроцесорні

КВП – контрольно-вимірювальні прилади

ВСТУП

Основа діяльності ПрАТ "СНВО" Імпульс "- комплектування, поставка і здача програмно-технічних комплексів. Основні напрямки - розробка, виробництво і впровадження програмно-технічних засобів в різних галузях промисловості: тепловій і атомній енергетиці, хімії, нафтохімії, металургійній, газовій та інших. Продукція, що випускається сертифікована, відповідає вимогам стандартів, норм і рекомендацій МЕК і МАГАТЕ до інформаційних і керуючих систем, важливих для безпеки АЕС.

В даний час ПрАТ «СНВО «Імпульс» є провідним виробником програмно-технічних комплексів (ПТК) для АКНП. Для організації нижнього рівня ПТК призначені мікропроцесорні Субкомплекси контролю і управління (АКНП), які здійснюють прийом і необхідну обробку вимірювальної інформації від об'єкта, управління технологічним обладнанням, регулювання параметрів технологічного процесу, захист обладнання та блокування.

АКНП використовується для збору, обробки інформації та управління технологічними установками як автономно, так і в складі ПТК.

АКНП застосовуються в якості:

- підсистем нижнього рівня АСУ ТП;
- інтелектуальних автономних систем контролю і управління;
- промислових контролерів відмовостійких систем автоматизації об'єктів атомної енергетики класу безпеки 2 і нижче.

АКНП має всі функції і засоби, необхідні для створення сучасних систем управління технологічними процесами: реєстрацію і обробку параметрів процесу, регулювання, управління, захисту і блокування, сигналізації, обчислювальні операції, оптимізацію, експертні системи, візуалізацію процесу на екранах моніторів.

Метою даної дипломної роботи є розробка програмного забезпечення для

прийому інформації, перетворення і видачі по інтерфейсах Ethernet і RS-422 для забезпечення взаємодії АКНП з модулем інтерфейсним МІф-1.

1 АНАЛІЗ ЗАДАЧІ

1.1 Аналіз задачі

Ядерна енергетика є базовою галуззю з надзвичайно складним наукомістким технологічним комплексом, що забезпечує виробництво і видачу споживачам електричної та теплової енергії. В останні роки галузь стає одним з визначальних чинників у вирішенні енергетичних проблем України, виробляючи понад 50% електроенергії. Атомна енергетика залишається стабільним і надійним джерелом електроенергії.

Атомна електростанція (АЕС) - ядерна установка для виробництва енергії в заданих режимах та умовах застосування, що розташовується в межах передбаченої проектом території, на якій для здійснення цієї мети використовуються ядерний реактор (реактори) і комплекс необхідних систем, пристроїв, обладнання та споруд з необхідними працівниками (персоналом).

Енергоблок - майже автономна частина теплової або атомної електричної станції, що представляє собою технологічний комплекс для виробництва електроенергії, що включає різне обладнання, наприклад паровий котел або ядерний реактор, турбіну, турбогенератор, трансформатор, допоміжне тепломеханічне та електричне обладнання, паропроводи і трубопроводи живильної води і інше.

Комплект АКНП призначений для контролю нейтронного потоку, управління основними і допоміжними технологічними процесами АЕС. Комплект АКНП повинен забезпечувати працездатність, надійність і безпеку на АЕС.

В даний час існують наступні основні типи АСУ ТП:

1) керуюча обчислювальна система (КОС)

2) керуючі системи безпеки (КСБ).

Керуюча обчислювальна система (КОС) - призначена для управління, контролю, реєстрації інформації, обчислення показників і параметрів, в тому числі параметрів безпеки, а також для виконання допоміжних функцій з контролю та управління функціонуванням.

Керуючі системи нормальної експлуатації реакторного та турбінного відділень, призначені для забезпечення функцій технологічних захистів, блокувань, сигналізації та дистанційного керування виконавчими механізмами.

1.2 Опис тестування

1.2.1 Вимоги до модуля інтерфейсного МІф-1

1. Модуль інтерфейсний МІф-1 призначений для прийому інформації, перетворення і видачі по інтерфейсах Ethernet і RS-422.
2. МІф-1 повинен встановлюватися в монтажний каркас (ширина 19 ", висота 6U) пристрої технічного обслуговування ПТО-1. Розмір печатної плати – Е3; тип печатної плати – МПП (кількість слоїв – 4).
3. Передня панель блоку повинна мати ширину 4HP.
4. На передній панелі МІф-1 повинні розташовуватися п'ять світлодіодних індикаторів і ручки для зручності вилучення з каркаса і установки в каркас. Маркування та розміщення елементів виконати згідно малюнку 1.
5. Розташування елементів на ПП.

5.1. Елементи Z1, C1, C2, C4, C5 повинні розташовуватися поруч з відповідними виходами D1.

5.2. Елементи L1, C6, C7, X4, X5 повинні розташовуватися поруч з відповідними виходами D2. Елементи L2, C9, R5 повинні розташовуватися поруч з відповідними виходами G1.

5.3.Генератор G1 повинен розташовуватися поруч з D3. Елементи C11 - C21, L3 - L10 повинні розташовуватися поруч з відповідними виходами D3.

5.4. Елементи L11 - L20, C23, C24 повинні розташовуватися поруч з відповідними виходами D4, D5.

5.5. Елементи C41, C47, L38, 41 повинні розташовуватися поруч з відповідними виходами D10.

5.6. Елементи C45, C46, C48, C49, L40, L42 повинні розташовуватися поруч з відповідними виходами D11.

5.7. Конденсатори C45, C46 повинні розташовуватися поруч з відповідними виходами D11.

5.8. Контрольні точки Q1-Q15 виконати у вигляді металізованих отворів діаметром 0.8 mm.

5.9. Класи ланцюгів EthTX і EthRX необхідно вирівняти по довжині з похибкою не більше 1 mm.

5.10. Ланцюги TD_P, FTD_P і TD_N, FTD_N є диференційною парою з конденсаторами C43, C44. Дані ланцюги необхідно вирівняти по довжині з похибкою не більше 0,5 mm.

5.11. Ланцюги RD_P і RD_N є виходами парою. Дані ланцюги необхідно вирівняти по довжині з похибкою не більше 0,5 mm.

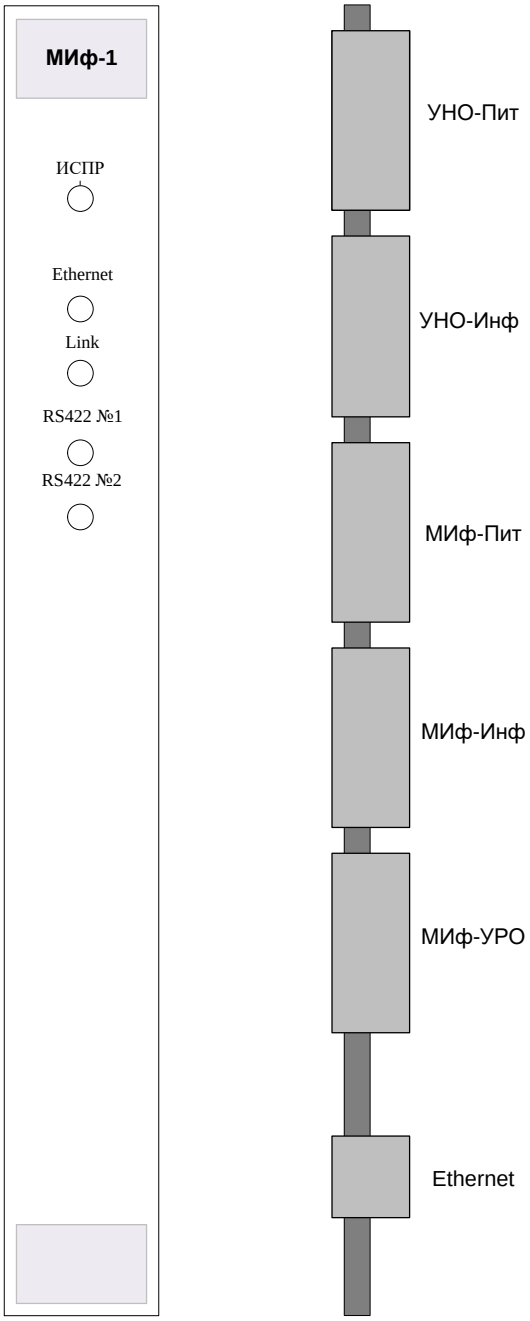


Рисунок 1.1 - Ескіз розміщення елементів на передній панелі і з заднього боку МІф-1

6. На платі повинні бути виконані маркування елементів.
7. Програма функціонування блоку повинна заноситися при його налаштуванні в мікросхему процесора D3.

8. Комплектність документації:

- Модуль інтерфейсний МІф-1. Схема електрична принципова;

- Модуль інтерфейсний МІф-1. Перелік елементів;

- Модуль інтерфейсний МІф-1. Карти електричних режимів роботи електрорадіопристроїв;

- Модуль інтерфейсний МІф-1. Інструкція з перевірки та приймання;

- Модуль інтерфейсний МІф-1. Дані записи інформації;

1.2.2 Загальні вимоги

Програмне забезпечення модуля интерфейсного МІф-1 (далі - ПЗ) призначено для виконання МІф-1 в складі АКНП функцій прийому і перетворення інформації від УФО-1 з подальшим відправленням її в пристрої реєстрації і відображення УРО-1 (зі складу АКНП) забезпечуючи підключення УРО-1 до УНО-1 зі складу АКНП.

ПЗ повинно бути розроблено як закрита для програмування користувачем мікропрограма функціонування мікроконтролера STM32F407VGT6 (далі - МК). ПЗ має інсталюватися в МК при виготовленні МІф-1.

ПЗ має забезпечувати такі функції:

- початкове тестування і конфігурація;
- прийом від УФО-1 (через порт UART4) даних в форматі ПТО, їх перетворення у формат УРО 1 та видачу у порти USART1 і USART2 (в два УРО-1);
- управління світловою індикацією (світлодіоди H1-H5);
- контроль працездатності (перевірка цілісності програмного коду).

ПЗ повинно відноситися до класу безпеки 2.

1.2.3 Вимоги до початкової ініціалізації

Після включення живлення і формування скидання МП ПО має виконати початкове конфігурування МП. Початкове конфігурування МП має включати ініціалізацію контролера вкладених переривань, схем тактування, таймера, таймера Whatchdog, UART, портів введення / виводу.

При ініціалізації таймера Watchdog необхідно задати тайм-аут спрацьовування таймера 0,5 с.

Швидкість перемикання станів GPIO повинна бути мінімальною.
Ініціалізацію UART4, USART1 і USART2 виконати наступними параметрами:

- швидкість - 57600 б / с;
- стоп-біт - 1;
- кількість біт даних - 9 (включаючи біт паритету);
- паритет - контроль на непарність;
- low control - відключений.
- Ініціалізацію таймера виконати наступними параметрами:
- номер таймера - 3 (TIM3);
- значення предподільника - 72;
- період - 500 мкс;
- counter Mode - UP.

1.2.4 Загальні вимоги до формату повідомлень

Обмін інформацією між пристроями повинен проводитися з використанням правил процедури BSC фірми IBM (знак-орієнтована процедура загальноцільового призначення для управління обміном даними

між користувачами) в частині використання наступних спеціальних символів:

– DLE = 10h (Data Link Escape, ознака наступного за ним керуючого символу);

– STX = 02h (Start of Text, ознака початку кадру);

– ETX = 03h (End of Text, ознака кінця кадру). Таким чином, для визначення початку повідомлення повинно застосовуватися поєднання керуючих символів DLE та STX, для визначення кінця повідомлення – DLE та ETX.

Для надання можливості передавати дані, що збігаються за кодуванням з керуючим символом DLE, повинна застосовуватися операція "байт-стаффінга". Вона полягає в тому, що в разі, якщо код ідентифікатора або інформаційної частини кадру збігається з кодом символу DLE (10h), то він передається двічі (дублюється). При прийомі інформації необхідно виконувати фільтрацію, тобто перевірку на наявність символу DLE (10h) в ідентифікатор та інформаційної частини кадру і видалення вставлених при передачі символів DLE (10h).

Загальний формат повідомлень показаний в таблиці 1.1

Таблиця 1.1 - Загальний формат повідомлень

Номер байта	Опис	Примітка
0	DLE (10h)	Початок повідомлення
1	STX (02h)	
2…n + 1	Дані повідомлення (Включаючи MasterByte, StatusByte, BCC1, BCC2)	n – Кількість байтів повідомлення
n + 2	DLE (10h)	Кінець повідомлення
n + 3	ETX (03h)	

1.2.5 Вимоги до інтерфейсу взаємодії з УФО-1

Взаємодія МІф-1 з УФО-1 засноване на ІРПС («струмова петля») і здійснюється в односторонньому порядку від УФО-1 в МІф-1.

МК МІф-1 повинен приймати повідомлення в UART4.

ПО МІф-1 має забезпечити прийом повідомлення з кодом функції FC = 01 від УФО-1, повідомлення з іншим кодом функції повинні ігноруватися.

Кількість байтів в повідомленні - 890 + w, де w - кількість дублюючих символів.

Формат повідомлення вказано в таблиці 1.2

Таблиця 1.2 - Формат повідомлення від УФО-1 в МІф-1

Номер байта	Опис	Примітка
0	DLE (10h)	Початок повідомлення
1	STX (02h)	
2	MasterByte (01h)	Ідентифікатор повідомлення
3…887 + w	Інформаційна частина повідомлення	w – кількість дублюючих символів
	BCC1	Контрольна сума повідомлення (ADC)
	BCC2	Контрольна сума повідомлення (XOR)
888 + w	DLE (10h)	кінець повідомлення
889 + w	ETX (03h)	

Після прийому повідомлення від УФО-1 необхідно виконати перевірку на наявність символу зі значенням 10h в ідентифікатор, інформаційної частини і

контрольних сумах повідомлення і його видалення.

Розрахунок значень контрольних сум BCC1 і BCC2 повинен виконуватися без урахування значень DLE, STX, ETX і дублюючих символів.

1.2.6 Вимоги до інтерфейсу взаємодії з УРО-1

Взаємодія МІф-1 з УРО-1 (2 шт.) Здійснюється за допомогою двох інтерфейсів RS-422 в односторонньому порядку від МІф-1 в УРО-1.

МК МІф-1 повинен передавати повідомлення в USART1 і USART2 після завершення перетворення повідомлення в формат УРО-1.

Кількість байтів в повідомленні - 1026 + w, де w - кількість дублюючих символів.

Формат повідомлення вказано в таблиці 1.3

Таблиця 1.3 - Формат повідомлення від МІф-1 в УРО-1

Номер байта	Опис	Примітка
0	DLE (10h)	Початок повідомлення
1	STX (02h)	
2	MasterByte (01h)	Ідентифікатор повідомлення
3…1023 + w	Інформаційна частина повідомлення	w – кількість дублюючих символів
	BCC1	Контрольна сума повідомлення (ADC)
	BCC2	Контрольна сума повідомлення (XOR)
1024 + w	DLE (10h)	Кінець повідомлення
1025 + w	ETX (03h)	

1.2.7 Вимоги до перетворення повідомлення з формату УФО-1 в формат УРО-1

Повідомлення від МІф-1 в УРО-1 має бути сформовано з повідомлення від УФО-1 в МІф-1 згідно з правилами, наведеними в Додатку А.

Номер каналу для повідомлення в УРО-1 формується за даними StatusByte з повідомлення від УФО-1. Дані для формування номера каналу вказані в таблиці 1.4

Таблиця 1.4 - Дані для формування номера каналу

StatusByte			Номер канала
AC (bit 4)	NC (bit 5)	CC (bit 6 та 7)	
0	0	n	n-1
0	1		n+2
1	0 або 1		n+5

Для формування поля StatusByte для повідомлення в УРО-1 необхідно перетворення поля StatusByte з повідомлення від УФО-1 за даними з таблиці 1.5

Таблиця 1.5 - Дані для формування поля StatusByte для повідомлення в УРО-1

StatusByte (З повідомлення від УФО-1)		StatusByte (Для повідомлення в УРО-1)	
AC (bit 4)	NC (bit 5)	AC (bit 4)	NC (bit 5)
0	0	0	0
0	1	0	0
1	0 або 1	1	0

Для перетворення архівного значення реактивності з формату УФО-1 в формат УРО-1 необхідно виконати наступні дії:

1) Виконати розрахунок значення реактивності в форматі Float виходячи з даних таблиці 1.6 за формулою 1.

Таблиця 1.6 - Архівне значення реактивності в форматі УФО-1

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 1	f.7-0							
Byte 2	f.15-8							
Byte 3	e.0	f.22-16						
Byte 4	s	e.7-1						

$$\rho = (s)\ 1.f \times 2^{(e-127)}, \text{ где} \qquad (1)$$

s – знак числа: 0 = позитивний, 1 = негативний,

e – порядок зі зміщенням на +127,

f – нормалізована мантиса.

1) Отримане значення реактивності (ρ) перетворити в код значення реактивності за даними з таблиці 7. Для цього необхідно встановити діапазон, до якого належить отримане значення реактивності (ρ) і виконати обчислення по формулі з таблиці 7 для відповідного діапазону.

2) Сформувати архівне значення реактивності в форматі УРО-1 (таблиця 1.7).

Таблиця 1.7 - Архівне значення реактивності в форматі УРО-1

Діапазон	Bit 15	Bit 14-0 (код значення реактивності)
от -0.5 до +0.5	0	Rnd [32000 · (ρ + 0.5)]
от -31.0 до +1.0	1	Rnd [1000 · (ρ + 31.0)]
Примітка - Rnd - функція округлення до найближчого цілого		

До сформованого пакету повідомлення повинна бути застосована операція «байт-стаффінга», тобто дублювання в ідентифікатор, інформаційної частини і контрольних сумах повідомлення значень, які збігаються з 10h.

Потім сформований пакет повідомлення необхідно привести до спільного формату повідомлень - додати ознаки початку повідомлення (DLE, STX) і кінця повідомлення (DLE, ETX).

Розрахунок значень контрольних сум BCC1 і BCC2 повинен виконуватися без урахування значень DLE, STX, ETX і дублюючих символів.

1.3 Вимоги до підпрограми самотестування

1.3.1 Тест оперативної пам'яті МК

Перевірка оперативної пам'яті МК повинна виконуватися після включення живлення МІф-1 і завершення ініціалізації.

У тесті оперативної пам'яті необхідно послідовно виконувати операції запису в усі осередки, читання і порівняння значень 0000h, FFFFh, 5555h і AAAAh, значення адреси осередку.

1.3.2 Тест цілісності програмного коду

Перевірка цілісності програмного коду повинна виконуватися одноразово при старті, а потім періодично в основному циклі ПЗ.

У кожному циклі повинні бути оброблені не більше 100 осередків Flash-пам'яті, підрахована контрольна сума прочитаних осередків. По завершенню зчитування всіх осередків пам'яті програм повинна бути підрахована загальна контрольна сума, виконано порівняння з її еталонним значенням і за підсумками перевірки повинен бути сформований код завершення тесту.

1.3.3 Вимоги до підпрограми управління індікацією

ПЗ має забезпечувати режими роботи елементів індикації МІф-1, наведені в таблиці 1.8.

Таблиця 1.8 - Режими роботи елементів індикації МІф-1

Індикатор	Режим роботи	Опис
ІСПР	Світіння зеленим кольором	МІф-1 справний
	Світіння червоним кольором	МІф-1 несправний. Помилка самодіагностики МІф-1, або помилка цілісності програмного коду.
	Індикатор вимкнений	Відсутнє живлення МІф-1 або вузол індикації несправний
Ethernet	Індикатор вимкнений	–
Link	Індикатор вимкнений	–
RS422 №1	Світіння зеленим кольором	Включається на час передачі повідомлення в УРО-1 №1
	Світіння червоним кольором	При відсутності прийнятого повідомлення (коректного) від УФО-1 понад 1600 мс
	Короткочасне відключення індикатора	Відсутня передача повідомлення в УРО-1 №1
	Індикатор вимкнений	Відсутнє живлення МІф-1 або вузол індикації несправний

Продовження таблиці 1.8 - Режими роботи елементів індикації МІф-1

RS422 №2	Світіння зеленим кольором	Чи включається на час передачі повідомлення в УРО-1 №2
	Світіння червоним кольором	При відсутності прийнятого повідомлення (коректного) від УФО-1 понад 1600 мс
	Короткочасне відключення індикатора	Відсутня передача повідомлення в УРО-1 №2
	Індикатор вимкнений	Відсутнє живлення МІф-1 або вузол індикації несправний

1.4 Вимоги до структури та елементів ПЗ

ПЗ повинно бути достатнім для виконання всіх функцій, зазначених у розділах 1.1 - 1.4 технічних вимог.

ПЗ повинно мати модульну структуру. Текст одного модуля повинен містити обмежену кількість операторів, мати ясну структуру, мати можливість легкої модифікації та тестування.

1.5 Вимоги до забезпечення захисту від відмов, спотворень вхідної інформації

Захист від спотворень прийнятої і видаваної інформації здійснюється шляхом формування та перевірки контрольних сум повідомлень (ADC і XOR).

Розрахунок значень контрольних сум BCC1 і BCC2 повинен виконуватися без урахування значень DLE, STX, ETX і дублюючих символів.

Для прийнятих повідомлень ПЗ має виконувати перевірку правильності кількості прийнятих байт. У разі неправильної кількості байт, повідомлення не повинно піддаватися подальшій обробці.

1.6 Вимоги до процесу розробки ПО

При розробці ПЗ повинно бути забезпечено його відповідність встановленим критеріям якості (надійність, ефективність, коректність, зручність застосування і т.п.).

Всі стадії процесу розробки ПО повинні бути детально задокументовані. Експлуатаційна документація повинна містити всі необхідні відомості для їх використання.

Вся документація по розробці ПО повинна бути викладена в доступній формі, зрозумілою фахівцям, які не брали участі в проведенні розробки ПО.

При розробці ПО слід уникати застосування методів і прийомів програмування, які ускладнюють програмування (складних розгалужень і циклів в програмах, складних індексів в масивах і т.д.). Слід переважно використовувати:

- уніфіковані позначення змінних в програмах;
- постійні масиви (заздалегідь визначеної довжини);
- підпрограми з мінімальним числом параметрів;
- підпрограми з одним входом і одним виходом.

1.7 Вимоги до верифікації ПЗ

Верифікація ПЗ МІф-1 повинна проводитися після кожного етапу розробки з оцінкою відповідності розроблюваних програмних засобів вимогам до

структури і функцій, діагностування і самоконтролю, забезпечення захистів від відмов, спотворень, помилкових і несанкціонованих дій.

Верифікація ПЗ і оформлення результатів верифікації повинні проводитися відповідно до вимог, що поширюються на ПО класу безпеки 2.

Висновок до розділу 1

У цьому розділі розглянуті загальні вимоги до модуля інтерфейсного МІф-1. Були виконані вимоги до його початкової ініціалізації, вимоги до інтерфейсу взаємодії та перетворення з формату УФО-1 у формат УРО-1. Також були виконані та реалізовані вимоги до процесу розробки ПЗ.

У подальшому буде реалізовано тест цілісності програмного коду, тест оперативної пам`яті та забезпечення від відмов, спотворень вхідної інформації.

При впровадженні цієї друкованої плати в апаратуру нейтронного потоку (АКНП) буде забезпечена безпека передачі інформації, а також попереджена кількість збоїв апаратури.

2 ОГЛЯД МІКРОПРОЦЕСОРНОЇ СИСТЕМИ. QT CREATOR, ЙОГО МОЖЛИВОСТІ ТА ФУНКЦІЇ

2.1 Мікропроцесор

Мікропроце́сор (англ. microprocessor) — інтегральна схема, яка виконує функції центрального процесора (ЦП) або спеціалізованого процесора. Сьогодні слово мікропроцесор є практично повним синонімом слова процесор, оскільки функціональний блок, що на ранніх стадіях розвитку обчислювальної техніки займали цілу плату чи навіть шафу, тепер вміщається в одну невеличку інтегральну схему із сотнями мільйонів транзисторів всередині. З середини 1980-х мікропроцесори витіснили інші види ЦП. Проте загалом це не так: центральні процесорні пристрої деяких суперкомп'ютерів навіть сьогодні є складними комплексами великих (ВІС) і надвеликих (НВІС) інтегральних схем.

Перша мікросхема успішно запрацювала 12 вересня 1958 року в компанії Texas Instruments. У 2000 році Нобелівську премію з фізики присудили Джеку Кілбі — за винахід інтегральної мікросхеми. Ще одним творцем інтегральної мікросхеми вважається Роберт Нойс, померлий в 1990 році (за правилами, Нобелівська вручається тільки живим вченим). Фізики як такої при створенні мікросхеми було небагато, але Килбі і Нойс «всього лише» придумали технологію, яка зробила переворот в електронній промисловості.

Перші мікропроцесори з'явилися на початку 1970-х і використовувалися в електронних калькуляторах для обробки 4-бітних слів, що представляли десяткові цифри в двійковому представленні. Досить скоро з'явилися інші вбудовані реалізації, такі як термінали, принтери, автоматичні прилади тощо, що використовували 4-бітні і 8-бітні мікропроцесори. Поява 8-бітних

процесорів з 16-бітною адресацією в середині 1970-х забезпечила достатній простір можливостей для реалізації перших мікропроцесорів загального призначення в мікрокомп'ютерах.

Довгий час процесори складалися з малих і середніх інтегральних схем, що містили в собі еквівалент від кількох до кількасот транзисторів. Інтеграція цілого центрального процесора в один чип значно зменшила вартість процесорної потужності. Послідовне впровадження мікросхем з більшим ступенем інтеграції робило цілі класи комп'ютерів застарілими, мікропроцесори з'явилися в широкому класі пристроїв, від малих вбудованих систем і ручних комп'ютерів до найбільших мейнфреймів і суперкомп'ютерів.

Починаючи з 1970-х збільшення процесорної потужності розвивається за правилами так званого закону Мура, який стверджує, що складність інтегральних мікросхем подвоюється кожні 18 місяців, за ті ж мінімальні гроші. В кінці 1990-х основним стримуючим фактором розвитку стало розсіюване мікропроцесором тепло.

Першим загальнодоступним мікропроцесором був 4-розрядний Intel 4004. Його змінили 8-розрядні Intel 8080 і 16-розрядний 8086, що заклали основи архітектури всіх сучасних настільних процесорів. Але внаслідок поширеності 8-розрядних модулів пам'яті був випущений 8088, клон 8086 з 8-розрядною шиною пам'яті. Потім пройшла його модифікація 80186. У процесорі 80286 з'явився захищений режим з 24-бітовою адресацією, що дозволяв використовувати до 16 МБ пам'яті. Процесор Intel 80386 з'явився в 1985 році і привніс покращений захищений режим, 32-бітову адресацію, що дозволила використовувати до 4 ГБ оперативної пам'яті і підтримку механізму віртуальної пам'яті.

2.2 Мікропроцесорна система

Виконання того чи іншого алгоритма можливо при наявності мікропроцесора та пристроїв, в яких зберігається програма. Відомо, що програма — це сукупність команд (правил), що виконуються в послідовності, заданій алгоритмом. Команди вибираються з пам'яті в послідовності, що задається процесором. Процесор визначає адреси елементів пам'яті, в яких зберігаються необхідні данні. Дані передаються в процесор, де перетворюються згідно з командами, і результати операції передаються знову в пам'ять.

Будь-яка мікропроцесорна система працює разом з рядом зовнішних пристроїв, одержуючи від них необхідну інформацію та передаючи іншу. Для зв'язку з зовнішними пристроями існує інтерфейс (англ. interface). Цим терміном позначається весь комплекс пристроїв, правил та технічних засобів, що регламентують та забезпечують обмін інформацією між мікропроцесором (включаючи пам'ять) та зовнішними пристроями. Головними в інтерфейсі є шини, або, як їх ще часто називають, магістралі. Магістраль — це сукупність провідників, для яких строго нормовані логічні рівні «0» та «1». Потужність сигналів на шинах має бути достатньою для живлення необхідної кількості приєднаних до них пристроїв. Для забезпечення цієї потужності використовуються спеціальні мікросхеми — шинні підсилювачі (ШП).

За призначенням, шини поділяються на три типи:

- адресні;
- даних;
- керування.

Але реально як в мікропроцесорній техниці, так і в комп'ютерній часто дві шини суміщують шляхом мультіплексування, що дещо знижує їх швидкодію, але набагато зменшує кількість виводів мікросхем.

2.3 Система команд мікропроцесорів

Найнижчим рівнем, який дозволяє описувати роботу цифрових пристроїв — це рівні логічних станів їх входів та виходів — таблиці станів.

Наступним рівнем є спосіб описання — це мова значень вхідних та вихідних сигналів, що складають мову мікрокоманд. Сукупність адрес та керуючих сигналів називаються мікрокомандою.

Третій рівень формалізації описання роботи мікропроцесора — це мова команд — тобто строга послідовність мікрокоманд, що записується в пам'яті мікропроцесорів. Тобто, команда, це слово, або набір слів, які дешифруються в послідовність мікрокоманд. Звідси витікає, що будь-який процесор має строго фіксований і обмежений набір команд, який є характерним для данного процесора. Будь-яка мікрокоманда характеризується своїм форматом. Під форматом мікрокоманди розуміється її протяжність та призначення кожного біта або їх групи. Команди, також мають свій фіксований формат. (Протяжність мікрокоманди — це стандартна для данного процесора кількість біт в слові). В залежності від протяжності команди, вона може складатися з одного, двох, та трьох слів.

Формат пам'яті мікропоцесорної системи також тісно пов'язаний з довжиною слова. Тому при зберіганні таких команд відповідно використовується адресний простір та пам'ять. Якщо, наприклад, команда складається з трьох слів, а використовується з послідовною адресацією, то для зберігання такої команди використовуються три послідовні адреси. Для того, щоб таку команду вибрати з пам'яті, необхідно мати спеціальні засоби, щоб забезпечити її представлення як єдине ціле.

Структура команд повністю залежить від структури мікропроцесора, але незалежно від типу процесора прийнято вважати, що однослівні команди повністю складаютья з коду операції. Двослівні команди складаються з коду

операції та однослівного операнда. Трислівні команди також складаються з двох частин: перша частина — код операції, а друга — адреса, або двослівний операнд.

Типи команд, що використовуються, тісно пов'язані з внутрішньою організацією та алгоритмом функціювання мікропрограмного автомата процесора, та внутрішньою системою синхронізації. Мікропроцесорна система функціонує синхронно з частотою тактових сигналів зовнішнього генератора. В залежності від типу мікропроцесорів використовується одно- або двохфазна синхронізація. Незалежно від цього в мікропроцесорних системах використовуються триваліші інтервали часу, ніж тактовий інтервал зовнішнього генератора. Одним з таких інтервалів є машинний цикл — це інтервал, протягом якого мікропроцесор звертається до пам'яті або пристрою вводу-виводу. Машинний цикл (МЦ) складає тільки частину циклу команди. На початку кожного МЦ на одному з виходів мікропроцесора з'вляється сигнал синхронізації, він передається по лінії шини керування в пам'ять або пристрої вводу-виводу і «сповіщає» про початок нового МЦ, в результаті чого досягається узгодження в часі зовнішніх пристроїв з роботою мікропроцесора.

Цикл команди — це інтервал часу, необхідний для вибірки з пам'яті команди, та її виконання. Він складається з 1-5 машинних циклів. Їхнє конкретне число залежить від складності операції, яка виконується в данній команді і дорівнює числу звернень мікропроцесора до пам'яті. Тривалість виконання команди визначається кількістю тактів в циклі команди та тривалістю такту.

Протягом циклу команди, що ділиться на дві фази, робота мікропроцесора виконується в такій послідовності. Пристрій керування задає початок чергового циклу шляхом формування сигналу, по якому число, що знаходиться в лічильнику команд, відправляється в буферний регістр адреси і через нього направляється для дешифрації. Після приходу від мікропроцесору сигналу

керування 'готовий' з елемента пам'яті, що знаходиться по вказаній адресі, зчитується слово команди, яке подається по шині даних в буферний регістр данних, а потім в пристрій керування, де дешифрується з допомогою кода операції. Ця послідовність операцій називається фазою виборки. За нею слідує виконавча фаза, в якій пристрій керування формує послідовність сигналів, необхідних для виконання команди. За цей час число, що знаходиться в лічильнику команд, збільшується на 1 (якщо довжина команди є 1) і формується адреса команди, що стоїть слідом за виконуємою. Вона зберігається в лічильнику до приходу сигнала, що задає початок чергового циклу команди.

Окрім адреси елемента в якому зберігається необхідний байт від мікропроцесора до пам'ті поступає сигнал по шині керування, який визначає характер операції — запис, або зчитування. Виконання вказаних операцій проходить протягом інтервалу часу, що називається часом доступу. По закінченні цього інтервалу від пам'яті в мікропроцесор подається сигнал готовності, який є сигналом початку прийому, або, відповідно, передачі сигналів в пам'ять. До одержання сигналу готовності мікропроцесор перебуває в стані очікування. Інтервал часу між імпульсами звернення до зовнішніх пристроїв та одержання від них відповіді називається циклом очікування.

Якщо, наприклад, цикл команди розглядати відповідно до команди вводу данних, то перші два машинних цикли будуть відноситись до фази виборки, а третій — до фази виконання команди. В усіх машинних циклах передається адреса, але в кожному циклі адреса належить своєму адресату, в першому — це адреса елемента, де здерігається код операції, в другому — адреса порта, що здерігає байт данних, в третьому — адреса акумулятора мікропроцесора, куди повинен поступити байт данних з порта.

2.4 QT Creator, можливості та функції

Qt Creator це повністю інтегроване середовище розробки (IDE), яке надає вам інструменти проектування і розробки складних додатків для безлічі настільних і мобільних платформ.

Рисунок 2.1 – візуальне зображення можливостей QT Creator

Одним з найголовніших досягнень Qt Creator є те, що він дозволяє команді розробників працювати над проектом на різних платформах з використанням загальних інструментів для розробки і налагодження.

Створення проекту дозволить:

– Групувати файли разом

- Додати власні кроки збірки
- Включити форми та файли ресурсів
- Вказати налаштування для додатків які запускаються

Ви можете або створити проект з нуля, або імпортувати існуючий проект. Qt Creator генерує всі необхідні файли в залежності від типу створюваного проекту. Наприклад, якщо ви оберете створення додатка з графічним інтерфейсом користувача (GUI), Qt Creator створить порожній .ui файл, який ви можете змінити в інтегрованому Qt Designer.

Qt Creator інтегрований з кроссплатформеними системами автоматизації збирання: qmake і CMake. Також ви можете імпортувати існуючі проекти, які не використовують qmake або CMake, і вказати Qt Creator просто проігнорувати вашу систему збирання.

Qt Creator поставляється з редактором коду і Qt Designer для проектування і складання графічних інтерфейсів користувача (GUI) з віджетів Qt.

Qt Creator відрізняється від текстового редактора тим, що знає як збирати і запускати додатки. Він розуміє мови C ++ і QML як код, а не як простий текст, це дозволяє:

- Дати вам можливість писати добре форматований код
- Вгадувати що ви хочете написати і доповнювати код
- Відображати повідомлення про помилки і попередження
- Дати вам можливість переміщатися між класами, функціями і символами
- Надавати вам контекстно-залежну довідку по класах, функціях та символах
- Показувати вам місце в коді де функція була описана або викликана.

Ви можете використовувати Qt Designer щоб мати у своєму розпорядженні і налаштовувати ваші віджети або діалоги і тестувати їх використовуючи різні

стилі. Створені за допомогою Qt Designer віджети і форми легко інтегруються в програмний код з використанням механізму сигналів і слотів Qt, які дозволять вам легко визначити поведінку графічних елементів. Всі властивості, встановлені в Qt Designer, можуть бути динамічно змінені в коді. Більш того, такі особливості як просування віджетів і власні модулі дозволять вам використовувати власні віджети з Qt Designer.

QT Creator дозволяє використовувати редактор для написання коду на Qt C ++ або на мові декларативного програмування QML. Також є можливість використовувати QML для створення дуже гнучкого інтерфейсу користувача з великого набору елементів QML. QML допомагає розробникам і дизайнерам працювати разом над створенням гнучких, призначених для користувача, інтерфейсів, які поширяться на портативних пристроях, таких як мобільні телефони, медіаплеєри, неттопи і нетбуки.

QML це розширення JavaScript, яке надає механізм декларативної збірки дерева об'єктів з елементів QML. QML покращує інтеграцію між JavaScript і існуючою системою Qt, заснованої на QObject, додає підтримку автоматичного зв'язування властивостей і забезпечує мережеву прозорість на рівні мови.

У Qt Creator інтегрований з набором корисних інструментів, такі як системи управління версіями і емулятор Qt.

Qt Creator може допомогти з налагодженням додатків. Він надає інтерфейси GNU Symbolic Debugger (gdb) і Microsoft Console Debugger (CDB) для налагодження звичайних додатків на C ++ і внутрішні отладчики для Java Script. Це включає можливість підключити мобільні пристрої до свого комп'ютера і налагоджувати запущені на них програми.

Qt Creator відображає сиру інформацію, надану отладчиками, явним і лаконічним чином з метою спростити процес налагодження наскільки можливо без обмеження можливостей отладчиков.

Висновок до розділу 2

У цьому розділі був виконаний загальний огляд мікропроцесора. Розглянута, проаналізована та вибрана платформа QtCreator. Розглянуті її переваги, функції та відмічені деякі унікальні можливості.

Для нашого проекту ми обрали Qt Creator, бо ця платформа дозволяє у короткі сроки розробити потрібну нам задачу. Саме завдяки їй можливо наочно показати працездатність та актуальність задачі, бо вона дозволяє у короткі строки ознайомитись з обладнанням та закріпити навички.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Вибір мови програмування

3.1.2 Огляд сучасних мов програмування

За рівнем (особливостям побудови) мови діляться на:

1. Машинно-орієнтовані (асемблери). Кожен оператор мови є мнемонічним (умовним) позначенням машинної команди. Природно, що кожен тип процесора має свій набір команд, а значить, свій асемблер. Асемблери використовуються для створення драйверів, програмування різних пристроїв, а також для написання фрагментів програм, де дуже важливо час виконання (оскільки на асемблері можна написати максимально ефективну програму);

2. Універсальні. Іноді їх поділяють на процедурно-орієнтовані та об'єктно-орієнтовані, але в даний час межа між цими видами стерлась. Ці мови використовуються найчастіше для вирішення найрізноманітніших завдань. І хоча кожна з мов має свої особливості, що робить її найбільш ефективною для вирішення певного виду завдань, але в принципі для вирішення будь-якого завдання можна вибирати будь-яку мову програмування.

Серед універсальних мов програмування в даний час найбільш поширені наступні:

1) Сі та його різновиди (C++, C #, C. Net). Широко використовується для розробки системного ПО, вбудованого ПО (тобто програм, що використовуються в різних приладах), рішення розрахункових завдань.

2) Паскаль (Турбо-паскаль, Object Pascal) був створений спеціально для навчання студентів програмуванню, але в даний час широко використовується при вирішенні різних завдань.

3) Delphi є "спадкоємцем" мови Паскаль; основні оператори в цих мовах однакові. Але Delphi має засіб для роботи з різними графічними об'єктами

(створення форм, кнопок, меню), а також для обробки складних структур даних. Тому він дуже популярний при розробці різних Windows - додатків.

4) Проблемно-орієнтовані мови, призначені для вирішення певних класів завдань. Наприклад, мова Lisp використовується для створення експертних систем. Мова Java використовується для розробки мережевих (Web) - додатків.

3.1.3 Загальна характеристика основних мов програмування

1) Об'єктно-орієнтоване програмування

Ідея об'єктно-орієнтованого програмування вперше була висунута в мові Smalltalk. В об'єктно-орієнтованому програмуванні введено поняття об'єкта і реалізовані механізми обчислення, що дозволяють:

– описувати структуру об'єкта;

– описувати дії з об'єктами;

– використовувати спеціальні правила успадкування об'єктів;

– встановити різну ступінь захисту компонентів об'єктів і визначити різні права доступу до них.

Об'єктно-орієнтоване розширення мови Паскаль, реалізовано фірмою Borland, знайшло дуже багато прихильників і є не тільки засобом для вивчення об'єктно-орієнтованого програмування, а й хорошим інструментом для створення прикладних програм. Між Object Pascal і C++ є багато спільного, але програмування на Object Pascal менш складне, ніж на C++ за рахунок меншого використання покажчиків, але програми створені на C ++ виходять швидшими.

2) Асемблер

Мова Асемблера об'єднує в собі переваги мови машинних команд і деякі риси мов високого рівня. Асемблер забезпечує можливість застосування символічних імен у вихідній програмі і позбавляє програміста від стомлюючої праці за визначенням пам'яті комп'ютера для команд, змінних і констант.

Асемблер дозволяє також гнучко і повно використовувати технічні можливості комп'ютера, як і мова машинних команд. Транслятор вихідних програм в Асемблері простіше транслятора, що вимагає для мови програмування високого рівня. На Ассемблері можна написати так само ефективно за розміром і часу виконання програму, як і програму на мові машинних команд. Ця гідність відсутня у мов високого рівня. Ця мова часто застосовують для програмування систем реального часу, технологічними процесами і устаткуванням, забезпечення роботи інформаційно-вимірювальних комплексів. До таких систем зазвичай висувають високі вимоги за обсягом займаної машинної пам'яті. Часто мова Асемблера доповнюється засобами формування макрокоманд, кожна з яких еквівалента цілій групі машинних команд. Таку мову називають мовою макроассемблера. Застосування мак «будівельних» блоків і наближає мову Асемблера до мови високого рівня.

3) Мова С ++

Мова С++ з'явилась на початку 80-х років. Створено Бьерном Страуструпом з первинною метою позбавити себе від програмування на асемблері.

С++ - мова загального призначення, область додатків якого - програмування систем в самому широкому сенсі. Крім цього, С++ успішно використовувався в багатьох додатках, які не вкладаються в ці рамки. Реалізація С++ здійснена для машин в діапазоні від найсучасніших мікрокомп'ютерів до найпотужніших суперкомп'ютерів і майже для всіх операційних систем.

С++ - універсальна мова програмування, задумана так, щоб зробити програмування приємнішим для серйозного програміста. За винятком другорядних деталей, С++ підтримує всі оператори Сі та доповнен новими операторами та можливостями. Крім можливостей, які дає Сі, С++ надає гнучкі та ефективні засоби визначення нових типів. Використовуючи визначення нових типів, які точно відповідають концепціям прикладної області, програміст

може розділяти програму на частини які легко діляться. Такий метод побудови програм часто називають абстракцією даних. Інформація про типи міститься в деяких об'єктах типів, визначених користувачем. Такі об'єкти прості і надійні у використанні в тих ситуаціях, коли їх тип не можна встановити на стадії компіляції. Програмування з застосуванням таких об'єктів часто називають об'єктно-орієнтованим. При правильному використанні цей метод дає більш короткі, простіші та легше контрольовані програми.

C++ був створений на основі мови програмування Сі та за невеликим винятком зберігає його як підмножина. Базова мова Сі спроектована так, що нагадує відповідність між цими типами, операціями і операторами і об'єктами, з якими безпосередньо доводиться мати справу машині: числами, символами і адресами.

C++ та його стандартні бібліотеки спроектовані так, щоб забезпечувати переносимість. Наявна на поточний момент реалізація мови буде йти в більшості систем, що підтримують Сі. З C++ програм можна використовувати Сі бібліотеки, і в C++ можна використовувати більшу частину інструментальних засобів, що підтримують програмування на Сі.

В результаті було вирішено вибрати мову програмування C++, так як для C++ існує безліч бібліотек і компіляторів. Програми, які написані на мові C++, працюють швидко і займають дуже мало пам'яті.

3.1.4 Огляд існуючих графічних інтерфейсів

В даний час використовуються такі найпоширеніші графічні засоби для візуального інтерфейсу:

- бібліотека MFC;
- бібліотека Qt.

3.1.5 Бібліотека класів MFC

Бібліотека класів MFC (Microsoft Foundation Classes) розроблена фірмою Microsoft.

Графічний інтерфейс цієї бібліотеки дозволяє в повній мірі використовувати всі можливості операційних систем сімейства Windows.

Додатки графічного інтерфейсу, як правило, містять декілька вікон, широко використовують піктограми, різні види курсорів, смуги прокрутки, меню і панелі інструментів. Все це призначено для створення «дружнього користувачеві» інтерфейсу програми.

Однак бібліотека класів MFC дозволяє розробляти програми лише в середовищі Windows і що важливо мають високу вартість.

3.1.6 Бібліотека класів Qt

Наступним найбільш поширеним видом візуалізації є бібліотека Qt. Qt - це бібліотека класів C++ і набір інструментального програмного забезпечення, призначених для побудови багатоплатформених додатків з графічним інтерфейсом і сповідують принцип "написавши одного разу - компілює в будь-якому місці", яка дозволяє запускати написане з її допомогою ПЗ в більшості сучасних операційних систем шляхом простої компіляції програми для кожної ОС без зміни вихідного коду. Включає в себе всі основні класи, які можуть знадобитися при розробці прикладного програмного забезпечення, починаючи від елементів графічного інтерфейсу і закінчуючи класами для роботи з мережею, базами даних і XML. Qt є повністю об'єктно-орієнтованим, легко розширюваним і підтримує техніку компонентного програмування.

Бібліотека Qt здобула репутацію мультиплатформенного набору інструментальних засобів, проте, не дивлячись на це, найчастіше вона використовується для розробки додатків на якій-небудь одній платформі.

Віджетами (widgets) називаються всі візуальні компоненти, з яких будується графічний інтерфейс. Кнопки, меню, смуги прокрутки і різноманітні рамки - все це віджети.

Особливість бібліотеки Qt є те, що вона являє собою єдину платформу для додатків, які можуть працювати під управлінням Windows 95 / 98 / 2000 / XP, Mac OS X, Linux, Solaris і інших версій UNIX. А головне, що при використанні її у вигляді інструментарію (а не для комерційного використання) - ця бібліотека є безкоштовною.

3.1.7 Особливості використання графічних інтерфейсів при розробці додатків

При глибокому розгляді можливостей бібліотек класів MFC і QT, а також способів їх використання можуть бути зроблені наступні висновки:

- MFC призначена для роботи під управлінням операційних систем Windows 95 / 98 / Millennium та Windows NT / 2000;

- Qt являє собою єдину платформу для додатків, які можуть працювати під управлінням Windows 95 / 98 / Me / 2000 / XP, Mac OS X, Linux, Solaris, HP-UX та інших версій Unix.

Слідуючи з вище сказаного, за основу використання при розробці програмних засобів налагодження необхідно використовувати бібліотеку класів Qt.

3.2 Умови виконання програми
3.2.1 Опис логічної структури програми

Склад і призначення функцій та класів наведені в таблиці 3.1

Таблиця 3.1 – Склад и призначення функцій та класів

Ім’я файлу	Ім’я функції та класи	Виконання функції
Main.cpp	Main()	Головна програма
	Init()	Програма ініціалізації
Mainwindow.cpp	QObject	Базовий клас для всіх об’єктів Qt. Його наслідує будь-який клас використовуючий сигнали та слоти
	QApplicatoin	Потрібен для використання графічного інтерфейсу
	QString	Надає строку символів Unicode. Зберігає 16-бітний QChar, де кожному QChar відповідає один символ Unicode
	QTime	Клас забезпечує функції годиника
	QBoxLayout	Клас вирівнює віджети по вертикалі

Продовження таблиці 3.1 – Склад и призначення функцій та класів

Mainwindow.cpp	QFile	Клас QFile успадкован від класа QIODevice. У ньому міститяться методи для роботи з файлами: відкриття, закриття, читання та запису даних.
	QMainWindow	Клас надає головне вікно програми
	QList	Це шаблоний клас, надає списки
	QLCDNumber	Клас відображає номер з цифрами, подібні LCD
	QIcon	Клас надає масштабовані значки в різних режимах і станах
Mainwindow.h	private	Доступ відкрит самому класу(функціям-членам класа) та «друзям» даного класу, як функціям так і класам
	protected	Доступ відкрит класам, похідним від даного
	public	Доступ відкрит усім, хто бачить визначення даного класу

3.2.2 Опис програми для обміну даними

Структура програми показана на рисунку 3.1

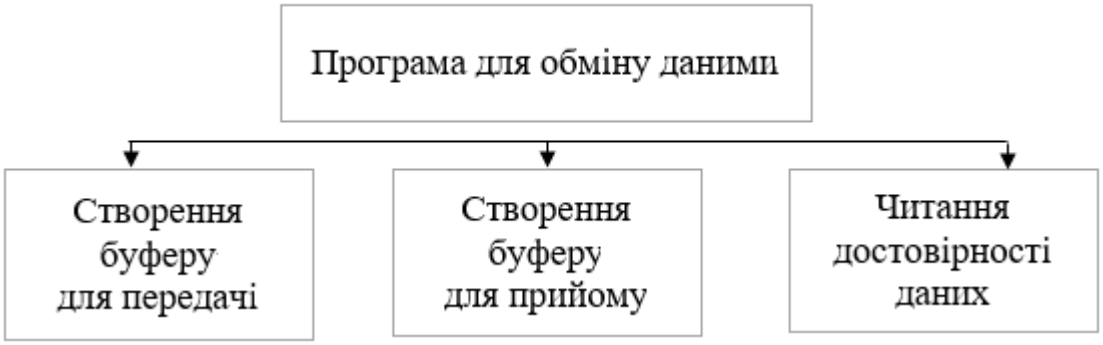


Рисунок 3.1 – Логічна структура програми

3.2.3 Інструкція для роботи с програмою ByteTest

Запуск програми виконуеться подвійним кліком ЛКМ по ярлику ByteTest.exe

При запуску ByteTest на екрані з’явиться головне вікно, показане на рисунку 3.2

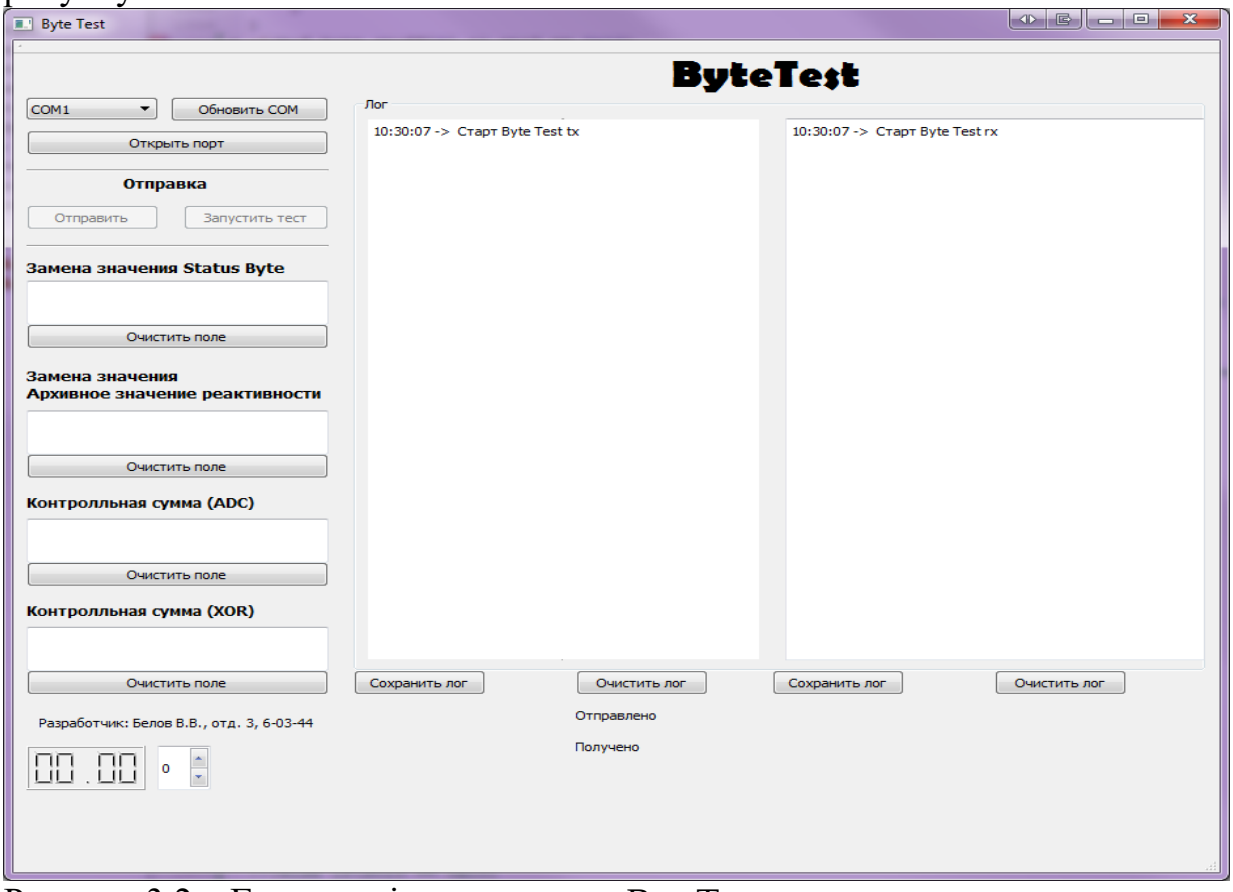


Рисунок 3.2 – Головне вікно програми ByteTest

Головне вікно містить наступні елементи:

- Випадаючий список;
- Кнопки: Оновити COM, Відкрити порт, Відправити, Запустити тест, Очистити поле, Зберегти лог;
- Поля для вводу інформації;
- Поля для виводу інформації;
- LCD дисплей для відліку часу роботи програми;
- Невидемі поля для виводу проміжного результату.

Кнопка «Оновити COM» дає можливість скинути поточний відкритий COM-порт.

Випадаючий список дає можливість вибрати один із підключених COM-портів.

Кнопка «Відкрити порт» - відкриває порт вибраний із випадаючого списку.

Кнопка «Відправити» - відправляє заздалегіть сформований пакет даних. При натискані на кнопку вона змінює назву на «Зупинити». Наступне натискання зупинить передачю пакетів даних.

Кнопка «Запустити тест» - виконує ті ж функції що й кнопка «Відправити», але відправляє інформацію кожні 0,5 сек. При натискані на кнопку вона змінює назву на «Зупинити тест». Наступне натискання зупинить передачю пакетів даних.

Кнопка «Очитити лог» - очищає поле з інформацією.

Кнопка «Зберегти лог» - сберігає у текстовому форматі прийняті та передані дані.

Для початку роботи програми необхідно вибрати із випадаючого списку COM-порт по якому буде відбуватися обмін інформацією. Наступним кроком потрібно натиснути на кнопку «Відкрити порт». Якщо зміняти значення «Status Byte, Архівне значення реактивності, ADC, XOR» не потрібно, натискаемо на кнопку «Відправити». Програма починає одиночну відправку інформації у

МІф-1. Якщо потрібно відправляти інформацію по таймеру, як вказано у ТЗ, зупиняємо программу, натискаючи на кнопку «Зупинини». Після натискаємо на кнопку «Запустити тест», наступне натисканя на цю кнопку зупинить передачю пакетів даних. При бажанні деякі значення можна змінити, наприклад для перевірки контрольної суми. Для цого вводимо за допомогю клавіатури значення у текстові поля. При натисканні на кнопку «Відправити», або «Запустити тест» программа замінить значення у коді.

У текстові поля «Лог» будуть записуватися прийнята та відправлена інформація. Цю інформацію можна зберігти за допомогю кнопки «Зберігти лог». Файл зберігаеться у кореневій папці з назвою log.tx або log.rx Інформація зберігаеться у форматі .txt У програмі є можливість очистити поля «Лог» за допомогою кнопки «Очистити лог». У невидемі поля які знаходяться нижче полів «Лог» буде виводитися кількість прийнятих та відправлених пакетів.

Для зупинки програми та виходу із неї потрібно натиснути на кнопку «Зупинити» або «Зупинити тест» та натиснути на хрестик у верхньому правому кутку програми.

Висновок до розділу 3

У цьому розділі був виконаний огляд сучасних мов програмування та їх характеристики. Були оглянуті існуючі графічні інтерфейси та умови виконання програми. Також була описана інструкція для роботи з програмою ByteTest

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

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

4.1 Вимоги до приміщення

Геометричні розміри приміщення зазначені у таблиці 4.1. Для зручності спільної роботи з іншими працівниками (обговорення ідей, з'ясування проблем і т.д.) в кімнаті є диван і журнальний стіл. Задля дотримання визначеного рівня мікроклімату в будівлі встановлено систему опалення та кондиціонування.

Для забезпечення потрібного рівного освітленості кімната має вікно та систему загального рівномірного освітлення, що встановлена на стелі. Для дотримання вимог пожежної безпеки встановлено порошковий вогнегасник та систему автоматичної пожежної сигналізації.

Таблиця 4.1 – Розміри робочого місця

Параметр	Значення
Довжина, м	6
Ширина, м	4
Висота, м	2,5
Площа, $м^2$	24
Об'єм, $м^3$	60

Згідно до санітарних норм мікроклімату виробничих приміщень [16] розмір площі для одного робочого місця оператора персонального комп'ютера має бути не менше 6 кв. м, а об'єм – не менше 20 куб. м. Отже, дане приміщення цілком відповідає зазначеним нормам.

4.1.1 Навантаження та напруженість процесу праці

За фізичним навантаженням робота відноситься до категорії легкі роботи (Іа), її виконують сидячи з періодичним ходінням. Щодо характеру організування виконання дипломної роботи, то він підпадає під нав’язаний режим, оскільки певні розділи роботи необхідно виконати у встановлені конкретні терміни. За ступенем нервово-психічної напруги виконання роботи можна віднести до ІІ – ІІІ ступеня і кваліфікувати як помірно напружений – напружений за умови успішного виконання поставлених завдань.

Під час виконання робіт використовують ПК та периферійні пристрої (лазерні та струменеві), що призводить до навантаження на окремі системи організму. Такі перекоси у напруженні різних систем організму, що трапляються під час роботи з ПК, зокрема, значна напруженість зорового аналізатора і довготривале малорухоме положення перед екраном, не тільки не зменшують загального напруження, а навпаки, призводять до його посилення і появи стресових реакцій.

Найбільшому ризику виникнення різноманітних порушень піддаються: органи зору, м’язово скелетна система, нервово-психічна діяльність, репродуктивна функція у жінок.

Тобто наявне психофізіологічні небезпечні та шкідливі фактори:

а) фізичного перевантаження:

- статичного;
- динамічного;

б) нервово-психічного перевантаження:

- розумового перенапруження;
- монотонності праці;
- перенапруження аналізаторів;
- емоційних перевантажень.

Рекомендовано застосування екранних фільтрів, локальних світлофільтрів (засобів індивідуального захисту очей) та інших засобів захисту, а також інші профілактичні заходи [17].

Роботу за дипломним проектом визнано, таку, що займає 50% часу робочого дня та за восьмигодинної робочої зміни рекомендовано встановити

додаткові регламентовані перерви тривалістю 15 хв через кожну годину роботи;

4.2 Пожежна безпека

Небезпека розвитку пожежі на обчислювальному центрі обумовлюється застосуванням розгалужених систем електроживлення ЕОМ, вентиляції і кондиціювання. Небезпека загоряння пов'язана з особливістю комп'ютерів – із значною кількістю щільно розташованих на монтажній платі і блоках електронних вузлів і схем, електричних і комутаційних кабелів, резисторів, конденсаторів, напівпровідникових діодів і транзисторів. Надійна робота окремих елементів і мікросхем в цілому забезпечується тільки в певних інтервалах температури, вологості і при заданих електричних параметрах. При відхиленні реальних умов експлуатації від розрахункових можуть виникнути пожежонебезпечні ситуації.

Висока щільність елементів в електронних схемах призводить до значного підвищення температури окремих вузлів (80…100°С). При проходженні електричного струму по провідниках і деталей виділяється тепло, що в умовах їх високої щільності може привести до перегріву, і може служити причиною запалювання ізоляційних матеріалів. Слабкий опір ізоляційних матеріалів дії температури може викликати порушення ізоляції і привести до короткого замикання між струмоведучими частинами обладнання (шини, електроди). Також ймовірна небезпека внаслідок перевантаження напруги, розрядки зарядів статичної електрики, пошкодження обладнання та електропроводки. Електростатичний розряд виникає під час тертя двох ізольованих матеріалів. Розряд статичної електрики може виникнути під час роботи вентилятора або комп'ютера. Кабельні лінії є найбільш пожежонебезпечними місцем. Наявність пального ізоляційного матеріалу, ймовірних джерел запалювання у вигляді електричних іскор і дуг, розгалуженість і недоступність роблять кабельні лінії місцем найбільш ймовірного виникнення і розвитку пожежі. Для зниження займистості і здатності поширювати полум'я кабелі покривають вогнезахисними покриттями.

Для гасіння пожеж в офісному приміщенні пропонується використовувати порошкові або вуглекислотні вогнегасники, так як вони є універсальними.

Виникнення пожежі можливе, якщо на об'єкті є горючі речовини, окиснювач і джерела запалювання. Вірогідність пожежної небезпеки приймається значною, якщо ймовірна взаємодія цих трьох чинників. Горючими компонентами є: будівельні матеріали для акустичної і естетичної обробки приміщень, перегородки, підлоги, двері, ізоляція силових, сигнальних кабелів і т.д.

Горючими матеріалами в приміщенні, де розташовані ЕОМ, є:

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

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

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

4) пластикат кабельний – матеріал ізоляції кабелів, горючий матеріал, показник горючості більше 2.1,

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

Для відводу теплоти від ЕОМ діє потужна система кондиціонування. Тому кисень, як окиснювач процесів горіння, є в будь-якій точці приміщень обчислювального центру.

Простори усередині приміщень в межах, яких можуть утворюватися або знаходиться пожежонебезпечні речовини і матеріали відповідно до [19] відносяться до пожежонебезпечної зони класу П-IIа. Це обумовлено тим, що в приміщенні знаходяться тверді горючі та важкозаймисті речовини та матеріали. Приміщенню, у якому розташоване робоче місце, присвоюється II ступень вогнестійкості.

Потенційними джерелами запалювання можуть бути:

- іскри і дуги короткого замикання;
- електрична іскра при замиканні і розмиканні ланцюгів;
- перегріви від тривалого перевантаження,
- відкритий вогонь і продукти горіння,
- наявність речовин, нагрітих вище за температуру самозаймання,

– розрядна статична електрика.

Причинами можливого загоряння і пожежі можуть бути:

– несправність електроустановки;

– конструктивні недоліки устаткування;

– коротке замикання в електричних мережах;

– запалювання горючих матеріалів, що знаходяться в безпосередній близькості від електроустановки.

Продуктами згорання, що виділяються на пожежі, є: окис вуглецю; сірчистий газ; окис азоту; синильна кислота; акромін; фосген; хлор і ін. При горінні пластмас, окрім звичних продуктів згорання, виділяються різні продукти термічного розкладання: хлорангідридні кислоти, формальдегіди, хлористий водень, синильна кислота, аміак, ацетон та ін.

4.3 Електробезпека

На робочому місці виконуються наступні вимоги електробезпеки [14]: ПК, периферійні пристрої та устаткування для обслуговування, електропроводи і кабелі за виконанням та ступенем захисту відповідають класу зони за ПУЕ (правила улаштування електроустановок), мають апаратуру захисту від струму короткого замикання та інших аварійних режимів. Лінія електромережі для живлення ПК, периферійних пристроїв і устаткування для обслуговування, виконана як окрема групова три- провідна мережа, шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів. Штепсельні з'єднання та електророзетки крім контактів фазового та нульового робочого провідників мають спеціальні контакти для підключення нульового захисного провідника. Електромережа штепсельних розеток для живлення персональних ПК, укладено по підлозі поруч зі стінами відповідно до затвердженого плану розміщення обладнання та технічних характеристик обладнання. Металеві труби та гнучкі металеві рукави заземлені. Захисне заземлення включає в себе заземлюючих пристроїв і провідник, який з'єднує заземлюючий пристрій з обладнанням, яке заземлюється - заземлюючий провідник.

4.4 Розрахунки

4.4.1 Розрахунок освітлення

Згідно з [18] для виробничих та адміністративних приміщень світловий коефіцієнт приймається не менше -1/8, в побутових – 1/10:

$$S_b = \left(\frac{1}{5} \div \frac{1}{10}\right) \times S_n, \tag{4.1}$$

де S_b – площа віконних прорізів, м^2;

S_n – площа підлоги, м^2.

$S_n = a \cdot b = 6 \cdot 4 = 24$ м^2,

$S = 1/10 \cdot 24 = 2{,}4$ м^2.

Приймаємо 1 вікно площею S=2,4 м^2.

Світильники загального освітлення розташовуються над робочими поверхнями в рівномірно-прямокутному порядку. Для організації освітлення в темний час доби передбачається обладнати приміщення, довжина якого складає 6 м, ширина 4 м, світильниками ЛПО2П, оснащеними лампами типа ЛБ (дві по 80 Вт) з світловим потоком 5200 лм кожна.

Розрахунок штучного освітлення виробляється по коефіцієнтах використання світлового потоку, яким визначається потік, необхідний для створення заданої освітленості при загальному рівномірному освітленні. Розрахунок кількості світильників n виробляється по формулі (4.2):

$$n = \frac{E \times S \times Z \times K}{F \times U \times M}, \tag{4.2}$$

де E – нормована освітленість робочої поверхні, визначається нормами – 300 лк;

S – освітлювана площа, м^2; S = 24 м^2;

Z – поправочний коефіцієнт світильника (1,1 для люмінесцентних ламп);

K – коефіцієнт запасу, що враховує зниження освітленості в процесі

експлуатації – 1,5;

U – коефіцієнт використання, залежний від типу світильника, показника індексу приміщення і т.п. – 0,575

M – число люмінесцентних ламп в світильнику – 2;

F – світловий потік лампи – 5200лм.

Підставивши числові значення у формулу (4.2), отримуємо:

$$n=\frac{300\times24\times1.1\times1.5}{5200\times0.575\times2}=1,98$$

Приймаємо освітлювальну установку, яка складається з 3-х світильників, які складаються з 2-х люмінесцентних ламп загальною потужністю 80 Вт, напругою – 220 В.

4.4.2 Розрахунок захисного заземлення

Згідно з класифікацією приміщень за ступенем небезпеки ураження електричним струмом [15], приміщення в якому проводяться всі роботи відноситься до першого класу (без підвищеної небезпеки). Під час роботи використовуються електроустановки з напругою живлення 36 В, 220 В, та 360 В. Опір контуру заземлення повинен мати не більше 4 Ом.

Розрахунок проводять за допомогою методу коефіцієнта використання (екранування) електродів. Коефіцієнт використання групового заземлювача η – це відношення діючої провідності цього заземлювача до найбільш можливої його провідності за нескінченно великих відстаней між його електродами. Коефіцієнт використання вертикальних заземлювачів $\eta_в$ в залежності від розміщення заземлювачів та їх кількості знаходиться в межах 0,4…0,99. Взаємну екрануючу дію горизонтального заземлювача (з'єднувальної смуги) враховують за допомогою коефіцієнта використання горизонтального заземлювача $\eta_с$.

Послідовність розрахунку:

1) Визначається необхідний опір штучних заземлювачів $R_{шт.з.}$:

$$R_{шт.з.} = \frac{R_д \cdot R_{пр.з.}}{R_{пр.з.} - R_д}, \quad (4.3)$$

де $R_{пр.з.}$ – опір природних заземлювачів; $R_д$ – допустимий опір заземлення. Якщо природні заземлювачі відсутні, то $R_{шт.з.} = R_д$.

Підставивши числові значення у формулу (4.3), отримуємо:

$$R_{шт.з.} = \frac{4 \cdot 40}{40\text{-}4} \approx 4 \text{ Ом}$$

2) Опір заземлення в значній мірі залежить від питомого опору ґрунту ρ, Ом·м. Приблизне значення питомого опору глини приймаємо $\rho = 40$ Ом·м (табличне значення).

3) Розрахунковий питомий опір ґрунту, $\rho_{розр}$, Ом·м, визначається відповідно для вертикальних заземлювачів $\rho_{розр.в}$, і горизонтальних $\rho_{розр.г}$, Ом·м за формулою:

$$\rho_{розр.} = \Psi \cdot \rho \quad (4.4)$$

де ψ – коефіцієнт сезонності для вертикальних заземлювачів і кліматичної зони з нормальною вологістю землі, приймається для вертикальних заземлювачів $\rho_{розр.в} = 1{,}7$ і горизонтальних $\rho_{розр.г} = 5{,}5$ Ом·м.

$$\rho_{розр.в} = 1{,}7 \cdot 40 = 68 \text{ Ом·м}$$
$$\rho_{розр.в} = 5{,}5 \cdot 40 = 220 \text{ Ом·м}$$

4) Розраховується опір розтікання струму вертикального заземлювача $R_в$, Ом, за (4.5).

$$R_В = \frac{\rho_{розр.в}}{2 \cdot \pi \cdot l_В} \cdot \left(\ln \frac{2 \cdot l_В}{d_{СТ}} + \frac{1}{2} \cdot \ln \frac{4 \cdot t + l_В}{4 \cdot t - l_В} \right), \quad (4.5)$$

де $l_в$ – довжина вертикального заземлювача (для труб – 2 − 3 м; $l_в = 3$ м);

$d_{ст}$ – діаметр стержня (для труб – 0,03 − 0,05 м; $d_{ст}$ = 0,05 м);

t – відстань від поверхні землі до середини заземлювача, яка визначається за ф. (4.6):

$$t=h_E+\frac{l_E}{2}, \tag{4.6}$$

де $h_в$ – глибина закладання вертикальних заземлювачів (0,8 м); тоді

$$t=0,8+\frac{3}{2}=2,3\text{ м};$$

$$R_в=\frac{68}{2\cdot\pi\cdot3}\cdot\left(\ln\frac{2\cdot3}{0,05}+\frac{1}{2}\cdot\ln\frac{4\cdot2,3+3}{4\cdot2,3-3}\right)=18,5\text{ Ом}$$

1) Визначається теоретична кількість вертикальних заземлювачів n штук, без урахування коефіцієнта використання $\eta_в$:

$$n=\frac{2R_E}{R_Д}=\frac{2\times18,5}{4}=9,28 \tag{4.7}$$

І визначається коефіцієнт використання вертикальних електродів групового заземлювача без врахування впливу з'єднувальної стрічки $\eta_в$ = 0,57 (табличне значення).

2) Визначається необхідна кількість вертикальних заземлювачів з урахуванням коефіцієнта використання $\eta_в$, шт:

$$n=\frac{2\cdot R_E}{R_Д\cdot\eta_E}=\frac{2\cdot18,5}{4\cdot0,57}\approx16 \tag{4.8}$$

3) Визначається довжина з’єднувальної стрічки горизонтального заземлювача $l_с$, м:

$$l_C=1,05\cdot L_B\cdot(n_B-1), \quad (4.9)$$

де $L_в$ – відстань між вертикальними заземлювачами, (прийняти за $L_в$ = 3м);
$n_в$ – необхідна кількість вертикальних заземлювачів.

$$l_с = 1,05\cdot 3\cdot(16-1)\approx 48 \text{ м}.$$

Визначається опір розтіканню струму горизонтального заземлювача (з’єднувальної стрічки) $R_г$, Ом:

$$R_г=\frac{\rho_{розр.г}}{2\cdot\pi\cdot l_с}\cdot\ln\frac{2\cdot l_с^2}{d_{см}\cdot h_г}, \quad (4.10)$$

де $d_{см}$ – еквівалентний діаметр смуги шириною b, $d_{см}$ = 0,95b, b = 0,15 м;

$h_г$ – глибина закладання горизантальних заземлювачів (0,5 м);

$l_с$ – довжина з’єднувальної стрічки горизонтального заземлювача $l_с$, м

$$R_г=\frac{220}{2\cdot\pi\cdot 48}\cdot\ln\frac{2\cdot 48^2}{0,95\cdot 0,15\cdot 0,5}=8,1 \text{ Ом}$$

4) Визначається коефіцієнт використання горизонтального заземлювача $\eta_{с.}$ відповідно до необхідної кількості вертикальних заземлювачів $n_в$.

Коефіцієнт використання з'єднувальної смуги $\eta_с$ = 0,3.

Розраховується результуючий опір заземлювального електроду з урахуванням з’єднувальної смуги:

$$R_{заг.} = \frac{R_E \cdot R_г}{R_E \cdot \eta_c + R_г \cdot n_E \cdot \eta_E} \leq R_д, \quad (4.11)$$

Висновок: дане захисне заземлення буде забезпечувати електробезпеку будівлі, так як виконується умова: $R_{заг} < 4$ Ом, а саме:

$$R_{заг} = \frac{18,5 \cdot 8,1}{18,5 \cdot 0,3 + 8,1 \cdot 16 \cdot 0,57} = 1,9 \leq R_д$$

При виникненню пожеж при роботі на ПЕОМ від таких можливими джерел запалювання як:

– іскри і дуги коротких замикань;

– перегрів провідників, резисторів та інших радіодеталей ПЕОМ, від тривалої перевантаження та наявність перехідного опору;

– іскри при розмиканні і розмиканні ланцюгів;

– розряди статичної електрики;

– необережному поводженню з вогнем, а також вибухи газо-повітряних і паро-повітряних сумішей.

Важливу увагу слід звернути на пожежну безпеку підприємства в цілому і окремих його приміщень. В приміщеннях не повинно накопичуватися сміття, непотрібний папір, мотлох та ін. речі, які не використовуються у виробничому процесі. Наявний вільний аварійний вихід за межі приміщення в разі пожежі, бути передбачені вогнегасники. Вони повинні бути в робочому стані і перевірятися згідно з нормами. У приміщеннях повинна бути пожежна сигналізація, вогнегасник. У разі виникнення пожежі необхідно повідомити в найближчу пожежну частину, убезпечити інших працівників і по можливості прийняти кроки по запобіганню можливих наслідків та усуненню пожежі.

ЗАКЛЮЧЕННЯ

В даному дипломному проекті була розроблена програма для контролю та обміну пакетами даних.

В ході роботи проведений аналіз існуючих інструментальних засобів емуляції вхідних сигналів.

У розділі «Аналіз задачі» були описані вимоги до модуля інтерфейсного Міф-1, а також загальні вимоги.

У розділі «Програмна реалізація» виконаний огляд сучасних мов програмування та їх характеристика. Були оглянуті існуючі графічні інтерфейси. Також були розглянуті умови виконання програми.

У розділі «Охорона праці» виконаний аналіз потенційно небезпечних і шкідливих виробничих факторів, розроблені заходи з техніки безпеки, заходи, що забезпечують виробничу санітарію та гігієну праці, виконаний аналіз і проведено розрахунок штучного освітлення, розроблені рекомендації та розрахунки по пожежної безпеки.

Список використаної літератури

1. Автоматизація в проектуванні і виробництві друкованих плат радіоелектронної апаратури Бахтін Б.І. тисяча дев'ятсот сімдесят дев'ять.

2. Короткий довідник радиомонтажника. Градиль А.В. 1974 р Градиль А.В. 1974

3. Основи виробництва радіоелектронної апаратури. Навчальний посібник Валетов В. А. 2007

4. Мова програмування С ++ Бьерн Страуструп 2004

5. Архітектура і програмування арифметичного співпроцесора. Григор'єв В.Л. (1991)

6. Введення в мікропроцесори. Програмне забезпечення, апаратні засоби, програмування. Льовенталь Л. (1983)

7. Коффрон Дж. Технічні засоби мікропроцесорних систем. Практичний курс (1983)

8. Мікропроцесори в питаннях і відповідях. Вуд А. (1985)

9. https://uk.wikipedia.org/wiki/%D0%9C%D1%96%D0%BA%D1%80%D0%BE%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D0%BE%D1%80%D0%BD%D0%B0_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0

10. http://irlib.vntu.edu.ua/bitstream/handle/123456789/2697/OMT.pdf?sequence1

11. http://elib.lutsk-ntu.com.ua/books/fepes/fizyka_ta_elektrotehnika/2011/11-83/

12. Антошина И. В. , Котов Ю. Т. Микропроцессоры и микропроцессорные системы

13. Шаповалов Ю. И. Програмирование микропроцессорных систем

14. НПАОП 0.00.-1.28-10 «Правил охорони праці під час експлуатації електронно-обчислювальних машин»;

15. НПАОП 0.00-4.15-98 «Положення про розробку інструкцій з охорони праці»;

16. ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень» ;

17. ДСанПіН 3.3.2.007-98 «Правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин»;

18. ДБН В.2.5-28:2015 «Державні Будівельні Норми України. Природне і штучне освітлення»;

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

Додаток А

Таблиця А.1 – Співвідношення прийнятих байт до відправлених

Співвідношення байт		Опис
МІф – УРО	УФО – МІф	
1	1	MasterByte (01h)
2	2	StatusByte (из таблицы 3)
3...8	3...8	Дата та час; t0
9...10	9...10	Номер кадру
11	-	Номер каналу (із таблиці 1.4)
12	-	Версія протоколу (80h)
13...18	11...16	Архівне значення дискретних сигналів, потужності та періода активного діапазону №01; t1=t0+5ms
19...24	17...22	Архівне значеннния дискретних сигналів, потужності та періода активного діапазона №02; t2=t0+10ms
25...606	23...604	Архівне значення дискретних сигналів, потужності та періода активного діапазона №03…99
607...612	605...610	Архівне значеннния дискретних сигналів, потужності та періода активного діапазона №100; t100=t0+500ms
613...616	-	Архівне значення реактивності №20.1 и №20.2; t20=t0+100ms (00000000h)

Продовження таблиці А.1 – Співвідношення прийнятих байт до відправлених

Співвідношення байт		Опис
МІф – УРО	УФО – МІф	
617...620	-	Архівне значення реактивності №20.3 (617, 618) = №20.4 (619, 620); t40=t0+200ms (із таблиці 1.7)
621...624	-	Архівне значення реактивності №40.1 та №40.2; t40=t0+200ms (00000000h)
625...628	-	Архівне значення реактивності №40.3 (625, 626) = №40.4 (627, 628); t40=t0+200ms (із таблиці 1.7)
629...632	-	Архівне значення реактивності №60.1 та №60.2; t60=t0+300ms (00000000h)
633...636	-	Архівне значення реактивності №60.3 (633, 634) = №60.4 (635, 636); t60=t0+300ms (із таблиці 1.7)
637...640	-	Архівне значення реактивності №80.1 и №80.2; t80=t0+400ms (00000000h)
641...644	-	Архівне значення реактивності №80.3 (641, 642) = №80.4 (643, 644); t80=t0+400ms (із таблиці 1.7)
645...648	-	Архівные значення реактивності №100.1 та №100.2; t100=t0+500ms (00000000h)
649...652	-	Архівне значення реактивності №100.3 (649, 650) = №100.4 (651, 652); t100=t0+500ms (із таблиці 1.7)

Продовження таблиці А.1 – Співвідношення прийнятих байт до відправлених

Співвідношення байтів		Опис
МІф – УРО	УФО – МІф	
653...656	651...654	Частота сигналу 1 робочого діапазону (від УД ДР)
657...658	655...656	Коефіцієнт тарировки сигналу 1 ДР
659...662	657...660	Частота сигналу 2 робочого діапазону (від УД ДП)
663...664	661...662	Коефіцієнт тарировки сигналу 2 ДР
665...668	-	Частота сигналу 1 робочого діапазону ДР1 (від УД ДР) (00000000h)
669...670	-	Коефіцієнт тарировки сигналу 1 ДР1 (01F4h)
671...674	-	Частота сигналу 2 робочого діапазону ДР1 (від УД ДП) (00000000h)
675...676	-	Коефіцієнт тарировки сигналу 2 ДР1 (01F4h)
677...680	663...666	Частота сигналу пускового діапазону (від УД ДП)
681...682	667...668	Коефіцієнт тарировки сигналу 1 ДП
683...686	663…666	Частота сигналу 2 пускового діапазону ДП (от УД ДП)
687...688	667...668	Коефіцієнт тарировки сигналу 2 ДП
689...692	669...672	Частота сигналу діапазону перегрузки (від УД СКП)
693...694	673...674	Коефіцієнт тарировки сигналу СКП
695...698	675...678	Значення потужності у робочому діапазоні ДР
699...700	679...680	Значення періода у робочому діапазоні ДР

Продовження таблиці А.1 – Співвідношення прийнятих байт до відправлених

Відповідність байт		Опис
МІф – УРО	УФО – МІф	
701...704	675...678	Значення потужність у робочому діапазоні ДР
705...706	679...680	Значення періода у робочому діапазоні ДР
707...710	681...684	Значення потужності у пусковому діапазоні ДП
711...712	685...686	Значення періода у пусковому діапазоні ДП
713...716	687...690	Значення потужності у діапазоні перегрузки СКП
717...718	691...692	Значення періода у діапазоні перегрузки СКП
719	693	Уставка потужності ДПР
720	694	Уставка періоду ДПР
721	695	Уставка потужності СКП
722	696	Уставка періоду СКП
723...726	697...700	Повний набір дискретних сигналів УНО
727...734	701...708	Конфігурація та працездатність пристроїв та блоків канала АКНП
735...736	797...798	Стан пристроїв детектування
737...1020	-	Діагностичні данні блоків та пристроїв каналу АКНП (0h)
1021	-	Контрольна сума (ADC)
1022	-	Контрольна сума (XOR)

Додаток Б

Код програми:

main.cpp

```
  #include "mainwindow.h"
#include <QApplication>
//#include "digitalclock.h"

int main(int argc, char *argv[])
{
    QStringList paths = QCoreApplication::libraryPaths();
    paths.append(".");
    paths.append("imageformats");
    paths.append("platforms");
    paths.append("sqldrivers");
    QCoreApplication::setLibraryPaths(paths);

    //setWindowIcon(QIcon(":123.ico"));
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    return a.exec();
}
```

mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include <QMainWindow>
```

```
#include <QMessageBox>
#include <QFileDialog>
#include <QTextStream>
#include <QTextEdit>
#include <QDateTime>
#include <QTime>
#include <QDate>
#include <QTextStream>
#include <QFile>
#include <QKeyEvent>
#include <QMouseEvent>
#include <QDebug>
#include <QTimer>
#include <QShortcut>
#include <QComboBox>
#include <QCheckBox>
#include <QThread>
#include <QLCDNumber>
//#include <QIcon>
//#include <abstractserial.h>

#include <qstring.h>

#include <QtSerialPort/QSerialPort>
#include <QtSerialPort/QSerialPortInfo>
#include <QObject>
#include <QLCDNumber>
```

```
namespace Ui {
class MainWindow;
}
   const char ProgramTitle [] = "Byte Test";
   enum Operations { NOTHING, ALLREAD, FIXREAD, INITADSP2191, SENDLDR, TEST_R, TEST_RT };

   #define   SetBit(reg, bit)         reg |= (1<<bit)
   #define   ClearBit(reg, bit)       reg &= (~(1<<bit))
   #define   BitIsSet(reg, bit)       ((reg & (1<<bit)) != 0)
   #define   BitIsClear(reg, bit)     ((reg & (1<<bit)) == 0)

class MainWindow : public QMainWindow
{
   Q_OBJECT

public:
   DigitalClock(QWidget *parent = 0);
   explicit MainWindow(QWidget *parent = 0);
   ~MainWindow();
signals:
   void savesettings (QString name, int baudrate, int DataBits, int Parity, int StopBits, int FlowControl);
   void writeData(QByteArray data);

private slots:
   void on_pbtnReloadCOM_clicked();
   void on_pbtnOpenSerial_clicked();
```

```
    void serialInputSlot();

//    void on_pbtnSend_clicked();

//    void on_pbtnSaveText_clicked();
//    void on_pbtnSaveText_2_clicked();

//    void on_pbtnClearLog_clicked();
//    void on_pbtnClearLog_2_clicked();

protected:

    void mouseDoubleClickEvent(QMouseEvent *event);

private:
    Ui::MainWindow *ui;
    QSerialPort *serial;
    Operations  serialOperation;
    QDate       *date;
    QTime       *time;
    QComboBox   *cbCommand;

//    char        *original_buf_from_UFO;          /*1*/
//    char        *cod_buf_from_UFO_1;
//    char        *cod_buf_from_UFO_2;
//    char        *cod_buf_from_UFO_3;
//    char        *cod_buf_from_UFO_4;
```

```
//    char       *cod_buf_from_UFO_5;

//    char       *cod_buf_from_UFO;              /*2*/
      char       *transformed_buf_from_UFO;      /*3*/
      char       *transformedcod_buf_from_UFO;   /*4*/
      char       *transformedcod_buf_from_URO;   /*5*/
//    char       *buf_byte_stuff;                /*6*/

      //char       *buf_tx_URO;
      //char       *buf_tx_UFO;


      char       numPortionAD;
      char       spinPortion;
      char       *str;
      quint64     fcnt;
      qint64      bytesToWrite;
      qint64      bytesToRead;
      QLCDNumber  *lcdNumber;
      QTimer      *timer1;
      QTimer      *timer2;
      qint64      t1cnt;
      quint64     t2cnt;
      QTimer      *timer_PC_connect;
      QTimer      *timer2_PC_connect;
      qint64      timer_PC_connect_cnt;
      quint64     timer2_PC_connect_cnt;
      QShortcut   *logLock;
```

```
bool        toogle_lock;


quint32 errorConnectIRDA;
quint32 errorConnectCOM;
quint32 error24;
quint32 error33;
quint32 errorT;
quint32 lblCntTx;
quint32 lblCntRx;
quint8 mesTransmitted;

quint8      *str1;
quint8      *str2;
quint8      *str3;
quint8      *str4;
char        mode;
quint16     sum_adc1;
quint16     sum_xor1;
quint16     sum_adc2;
quint16     sum_xor2;
quint16     sum_adc3;
quint16     sum_xor3;
quint16     sum_adc4;
quint16     sum_xor4;
quint16     sum_adc5;
quint16     sum_xor5;
quint16     sum_adc6;
```

```
    quint16    sum_xor6;
    quint16    sum_adc7;
    quint16    sum_xor7;

    quint16    asum1;
    quint16    xsum1;
    quint16    asum2;
    quint16    xsum2;
    quint16    asum3;
    quint16    xsum3;
    quint16    asum4;
    quint16    xsum4;
    quint16    asum5;
    quint16    xsum5;
    quint16    asum6;
    quint16    xsum6;
    quint16    asum7;
    quint16    xsum7;
private slots:

    void       tickTimer1();

    void       tick_timer_PC_connect();
    void       slotShortcutLock();

    void on_lineEditPass_editingFinished();
    quint16 ADC1(char *Data);
    quint16 XOR1(char *original_buf_from_UFO);
```

```
quint16 ADC2(char *cod_buf_from_UFO);
quint16 XOR2(char *cod_buf_from_UFO);

quint16 ADC3(char *cod_buf_from_UFO);
quint16 XOR3(char *cod_buf_from_UFO);

quint16 ADC4(char *cod_buf_from_UFO);
quint16 XOR4(char *cod_buf_from_UFO);

quint16 ADC5(char *cod_buf_from_UFO);
quint16 XOR5(char *cod_buf_from_UFO);

quint16 ADC6(char *cod_buf_from_UFO);
quint16 XOR6(char *cod_buf_from_UFO);

quint16 ADC7(char *cod_buf_from_UFO);
quint16 XOR7(char *cod_buf_from_UFO);

void on_pbtnTest_clicked();
void on_pbtnSend_clicked();
void on_pbtnClearLog_clicked();
void on_pbtnSaveText_clicked();
void on_pbtnSaveText_2_clicked();
void on_pbtnClearLog_2_clicked();
void on_pbtnDelete_clicked();
void on_pbtnDelete_2_clicked();
void on_pbtnDelete_3_clicked();
```

```
    void on_pbtnDelete_4_clicked();

};

#endif // MAINWINDOW_H
```

mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QThread>
#include <QLCDNumber>
#include <QtGui>

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    setWindowTitle(QString(ProgramTitle));
    QStringList serialsList;
    QList <QSerialPortInfo> sil = QSerialPortInfo::availablePorts();
    QSerialPortInfo info;
    foreach (info, sil) {
        if (info.isValid()) {
            serialsList.append(info.portName());
        }
    }
```

```
    ui->cbSerials->clear();
    ui->cbSerials->addItems(serialsList);
    ui->pbtnOpenSerial->setText("Открыть порт");
    ui->pbtnSend->setText("Отправить");
    ui->pbtnSend->setEnabled(false);
    ui->lineEditPass->hide();
    ui->lineEditPass_2->hide();
    ui->pbtnTest->setEnabled(false);
    ui->pbtnReloadCOM->setText("Обновить COM");
    ui->pbtnSaveText->setText("Сохранить лог");

    mesTransmitted = 0;
    lblCntTx = 0;
    lblCntRx = 0;

//    ui->spinPortion->setMaximum(0x7FFFFFFF);
//    ui->spinPortion->hide();

//                    connect(ui->cbCommand,      SIGNAL(activated(int)),this,
SLOT(slotChangeCommandList(int)));
//    ui->cbCommand->addItems(commandList);

    logLock = new QShortcut(this); // Инициализирую объект
    logLock->setKey(Qt::CTRL + Qt::SHIFT + Qt::Key_H);
    connect(logLock, SIGNAL(activated()), this, SLOT(slotShortcutLock()));
    toogle_lock = true;
    timer1 = new QTimer (this);
    timer2 = new QTimer (this);
```

```
    timer1->setInterval(500);
    timer1->stop();
    t1cnt = 0;
    t2cnt = 0;
    connect(timer1, SIGNAL(timeout()), this, SLOT(tickTimer1()));
    //connect(timer2, SIGNAL(timeout()), this, SLOT(tickTimer2()));

    timer_PC_connect = new QTimer (this);
    timer2_PC_connect = new QTimer (this);
    timer_PC_connect->setInterval(100);
    timer2_PC_connect->setInterval(100);
    timer_PC_connect->stop();
    timer2_PC_connect->stop();
    timer_PC_connect_cnt = 0;
    timer2_PC_connect_cnt = 0;
    connect(timer_PC_connect, SIGNAL(timeout()), this,
SLOT(tick_timer_PC_connect()));

    ui->label->setText("ByteTest");

    serial  = new QSerialPort (this);
    serialOperation = NOTHING; //prevOperation = NOTHING;
    connect(serial, SIGNAL(readyRead()), this, SLOT(serialInputSlot()));
    date = new QDate;
    time = new QTime;

    ui->textEdit->append(time->currentTime().toString("hh:mm:ss") + " -> " + " Старт
Byte Test tx");
```

```
    ui->textEdit_2->append(time->currentTime().toString("hh:mm:ss") + " -> " + "
Старт Byte Test rx");



    //cod_buf_from_UFO = new char[891];
    //buf_from_URO = new char [1027];

    QLCDNumber* plcd = new QLCDNumber;
    QSpinBox* pspb = new QSpinBox;
    //QWidget* wgt = new QWidget;
    plcd->setSegmentStyle(QLCDNumber::Flat);
    plcd->setMode(QLCDNumber::Dec);
    plcd->setDigitCount(6);
    pspb->setMaximum(10000000);
    QObject::connect(pspb, SIGNAL(valueChanged(int)),
                     plcd, SLOT(display(int)));
    QVBoxLayout* pvbxLayout = new QVBoxLayout;
    pvbxLayout->addWidget(plcd);
    pvbxLayout->addWidget(pspb);
//    wgt->setLayout(pvbxLayout);
//    wgt->resize(250, 150);
//    wgt->show();
    ui->lcdNumber->display(QTime::fromMSecsSinceStartOfDay(timer1->interval() -
timer1->remainingTime()).toString("hh.mm.ss"));
}

MainWindow::~MainWindow()
{
```

```
    delete serial;
    //delete data;
    delete time;
//    delete [] original_buf_from_UFO;
//    delete [] cod_buf_from_UFO;

//    delete [] cod_buf_from_UFO_1;
//    delete [] cod_buf_from_UFO_2;
//    delete [] cod_buf_from_UFO_3;
//    delete [] cod_buf_from_UFO_4;
//    delete [] cod_buf_from_UFO_5;

//    delete [] transformed_buf_from_UFO;
//    delete [] transformedcod_buf_from_UFO;
//    delete [] transformedkod_buf_from_URO;
//    delete [] buf_byte_stuff;
    delete ui;
}

void MainWindow::on_pbtnReloadCOM_clicked()
{
    QStringList serialsList;
    QList <QSerialPortInfo> sil = QSerialPortInfo::availablePorts();
    QSerialPortInfo info;
    foreach (info, sil) {
        if (info.isValid()) {
            serialsList.append(info.portName());
        }
```

```
    }
    ui->cbSerials->clear();
    ui->cbSerials->addItems(serialsList);

}
void MainWindow::slotShortcutLock()
{

    errorConnectCOM = 0;
    errorConnectIRDA = 0;

     error24 = 0;
     error33 = 0;
     errorT = 0;
}

void MainWindow::on_pbtnOpenSerial_clicked()
{
    ui->label->setStyleSheet("color: rgb(0, 0, 0)");

    //QMessageBox:: warning (this,  QString(ProgramTitle) + " Ошибка", "Не открыт последовательный порт");

    if (serial->isOpen()) {
        serial->close();
        ui->cbSerials->setEnabled(true);
       // ui->textEdit->setEnabled(true);
        ui->pbtnSend->setEnabled(false);
```

```
        ui->pbtnTest->setEnabled(false);
        ui->pbtnReloadCOM->setEnabled(true);
        ui->pbtnOpenSerial->setText("Открыть порт");
        ui->textEdit->append(time->currentTime().toString()+ " -> " + " Закрыт " + serial->portName());
        ui->textEdit_2->append(time->currentTime().toString()+ " -> " + " Закрыт " + serial->portName());
       // if (loging) ui->tbConsole->append("Последовательный интерфейс закрыт");
    } else {serial->setPortName(ui->cbSerials->currentText());
        if (!serial->open(QIODevice::ReadWrite)) {
            ui->pbtnOpenSerial->setStyleSheet("color: rgb(0, 0, 0);");
            ui->textEdit->append(time->currentTime().toString()+ " -> " + " <font color='red'> Ошибка </font>" + serial->portName() + ": " + serial->errorString());
            ui->textEdit_2->append(time->currentTime().toString()+ " -> " + " <font color='red'> Ошибка </font>" + serial->portName() + ": " + serial->errorString());
           // ui->textEdit->setStyleSheet("color: rgb(0, 0, 0)");
            ui->pbtnOpenSerial->setChecked(false);
            QMessageBox::critical(this, QString(ProgramTitle) + " - Ошибка", serial->portName() + " -> " + serial->errorString());
        } else {
         //   if (loging) ui->tbConsole->append("Открыт последовательный интерфейс (" + serial->portName() + ")");

            serial->setPortName(ui->cbSerials->currentText());
            serial->setBaudRate(57600);
            serial->setDataBits(QSerialPort::Data8);
            serial->setStopBits(QSerialPort::OneStop);
            serial->setParity(QSerialPort::OddParity);
```

```
            serial->setReadBufferSize(2042);

            ui->pbtnOpenSerial->setText("Закрыть порт");
            ui->textEdit->append(time->currentTime().toString() + " -> " + " Открыт " +
serial->portName());
            ui->textEdit_2->append(time->currentTime().toString() + " -> " + " Открыт "
+ serial->portName());
            ui->cbSerials->setEnabled(false);
            ui->pbtnSend->setEnabled(true);
            ui->pbtnTest->setEnabled(true);
            ui->pbtnReloadCOM->setEnabled(false);

        }
    }
}
void MainWindow::tickTimer1()
{
    on_pbtnSend_clicked();
    timer1->start();
}

void MainWindow::on_pbtnSend_clicked()
{
    ui->label->setStyleSheet("color: rgb(34, 216, 168)");
/*original buffer*/
char original_buf_from_UFO[886];
    original_buf_from_UFO[0] = 253; //MB
    original_buf_from_UFO[1] = 250; //SB
```

```
    for(quint16 i = 2; i < 8; i++) //DataTime
    {
        original_buf_from_UFO[i] = 0x11;
    }
    original_buf_from_UFO[8] = 0x10;
    original_buf_from_UFO[9] = 0x10;
    for(quint16 i = 10; i < 610; i++) //Архивные значения дискретных сигналов...
    {
        original_buf_from_UFO[i] = 0x04;
    }
    for(quint16 i = 610; i < 630; i++) //Архивные значения реактивности канальной
    {
        original_buf_from_UFO[i] = 0x01;
    }
    for(quint16 i = 630; i < 650; i++) // Архивные значения реактивности (среднее по комплекту)
    {
        original_buf_from_UFO[i] = -0x64;
    }
    for(quint16 i = 650; i < 654; i++)
    {
        original_buf_from_UFO[i] = 0x04;
    }
    original_buf_from_UFO[654] = 0xfa;
    original_buf_from_UFO[655] = 0xfa;
    for(quint16 i = 656; i < 660; i++)
    {
        original_buf_from_UFO[i] = 0x04;
```

```
}
original_buf_from_UFO[660] = 0xfa;
original_buf_from_UFO[661] = 0xfa;
for(quint16 i = 662; i < 666; i++)
{
   original_buf_from_UFO[i] = 0x04;
}
original_buf_from_UFO[666] = 0xfa;
original_buf_from_UFO[667] = 0xfa;
for(quint16 i = 668; i < 672; i++)
{
   original_buf_from_UFO[i] = 0x04;
}
original_buf_from_UFO[672] = 0xfa;
original_buf_from_UFO[673] = 0xfa;
for(quint16 i = 674; i < 678; i++)
{
   original_buf_from_UFO[i] = 0x96;
}
original_buf_from_UFO[678] = 0x02;
original_buf_from_UFO[679] = 0x02;
for(quint16 i = 680; i < 684; i++)
{
   original_buf_from_UFO[i] = 0x01;
}
original_buf_from_UFO[684] = 0x02;
original_buf_from_UFO[685] = 0x02;
for(quint16 i = 686; i < 690; i++)
```

```
{
   original_buf_from_UFO[i] = 0x00;
}
original_buf_from_UFO[690] = 0x02;
original_buf_from_UFO[691] = 0x02;

original_buf_from_UFO[692] = 0x02;
original_buf_from_UFO[693] = 0x02;
original_buf_from_UFO[694] = 0x02;
original_buf_from_UFO[695] = 0x02;
for(quint16 i = 696; i < 700; i++)
{
   original_buf_from_UFO[i] = 0x06;
}
for(quint16 i = 700; i < 708; i++)
{
   original_buf_from_UFO[i] = 0x08;
}
for(quint16 i = 708; i < 796; i++)
{
   original_buf_from_UFO[i] = 0x57;
}
original_buf_from_UFO[796] = 0x02;
original_buf_from_UFO[797] = 0x02;
for(quint16 i = 798; i < 814; i++)
{
   original_buf_from_UFO[i] = 0x10;
}
```

```
    for(quint16 i = 814; i < 856; i++)
    {
        original_buf_from_UFO[i] = 0x29;
    }
    for(quint16 i = 856; i < 868; i++)
    {
        original_buf_from_UFO[i] = 0xc;
    }
    for(quint16 i = 868; i < 884; i++)
    {
        original_buf_from_UFO[i] = 0x10;
    }
    original_buf_from_UFO[884] = 0x01; //ADC
    original_buf_from_UFO[885] = 0x01; //XOR

quint16 asum1 = ADC1(original_buf_from_UFO);
original_buf_from_UFO[884] = asum1;

quint16 xsum1 = XOR1(original_buf_from_UFO);
original_buf_from_UFO[885] = xsum1;

serial->write(original_buf_from_UFO,sizeof(original_buf_from_UFO));
lblCntTx++;
ui->lblCntTx->setText("Отправленно " +QString::number(lblCntTx));
mesTransmitted = 1;
/*end original buffer*/

/*coding buffer*/
```

```
char cod_buf_from_UFO[891];
    quint8 str1, str2, str3, str4;
str1 = ui->lineEdit->text().toInt();
    if(str1)
    {
        cod_buf_from_UFO[0] = 0x10; //DLE
        cod_buf_from_UFO[1] = 0x02; //STX
        cod_buf_from_UFO[2] = 0x01; //MB
        cod_buf_from_UFO[3] = str1; //SB
        for(quint16 i = 4; i < 10; i++) //DataTime
        {
            cod_buf_from_UFO[i] = 0x11;
        }
        cod_buf_from_UFO[10] = 0x10;
        cod_buf_from_UFO[11] = 0x10;
        for(quint16 i = 12; i < 612; i++) //Архивные значения дискретных сигналов...
        {
            cod_buf_from_UFO[i] = 0x04;
        }
        for(quint16 i = 612; i < 632; i++) //Архивные значения реактивности канальной
        {
            cod_buf_from_UFO[i] = 0x01;
        }
        for(quint16 i = 632; i < 652; i++) // Архивные значения реактивности (среднее по комплекту)
        {
            cod_buf_from_UFO[i] = -0x64;
```

```
}
for(quint16 i = 652; i < 656; i++)
{
   cod_buf_from_UFO[i] = 0x04;
}
cod_buf_from_UFO[656] = 0xfa;
cod_buf_from_UFO[657] = 0xfa;
for(quint16 i = 658; i < 662; i++)
{
   cod_buf_from_UFO[i] = 0x04;
}
cod_buf_from_UFO[662] = 0xfa;
cod_buf_from_UFO[663] = 0xfa;
for(quint16 i = 664; i < 668; i++)
{
   cod_buf_from_UFO[i] = 0x04;
}
cod_buf_from_UFO[668] = 0xfa;
cod_buf_from_UFO[669] = 0xfa;
for(quint16 i = 670; i < 674; i++)
{
   cod_buf_from_UFO[i] = 0x04;
}
cod_buf_from_UFO[674] = 0xfa;
cod_buf_from_UFO[675] = 0xfa;
for(quint16 i = 676; i < 680; i++)
{
   cod_buf_from_UFO[i] = 0x96;
```

```
}
cod_buf_from_UFO[680] = 0x02;
cod_buf_from_UFO[681] = 0x02;
for(quint16 i = 682; i < 686; i++)
{
   cod_buf_from_UFO[i] = 0x01;
}
cod_buf_from_UFO[686] = 0x02;
cod_buf_from_UFO[687] = 0x02;
for(quint16 i = 688; i < 692; i++)
{
   cod_buf_from_UFO[i] = 0x00;
}
cod_buf_from_UFO[692] = 0x02;
cod_buf_from_UFO[693] = 0x02;

cod_buf_from_UFO[694] = 0x02;
cod_buf_from_UFO[695] = 0x02;
cod_buf_from_UFO[696] = 0x02;
cod_buf_from_UFO[697] = 0x02;
for(quint16 i = 698; i < 702; i++)
{
   cod_buf_from_UFO[i] = 0x06;
}
for(quint16 i = 702; i < 710; i++)
{
   cod_buf_from_UFO[i] = 0x08;
}
```

```
        for(quint16 i = 710; i < 798; i++)
        {
           cod_buf_from_UFO[i] = 0x57;
        }
        cod_buf_from_UFO[798] = 0x02;
        cod_buf_from_UFO[799] = 0x02;
        for(quint16 i = 800; i < 816; i++)
        {
           cod_buf_from_UFO[i] = 0x10;
        }
        for(quint16 i = 816; i < 858; i++)
        {
           cod_buf_from_UFO[i] = 0x29;
        }
        for(quint16 i = 858; i < 870; i++)
        {
           cod_buf_from_UFO[i] = 0xc;
        }
        for(quint16 i = 870; i < 886; i++)
        {
           cod_buf_from_UFO[i] = 0x10;
        }
        cod_buf_from_UFO[886] = 0x01; //ADC
        cod_buf_from_UFO[887] = 0x01; //XOR
       cod_buf_from_UFO[889] = 0x10; //DLE
       cod_buf_from_UFO[890] = 0x03; //ETX
quint16 asum2 = ADC2(cod_buf_from_UFO);
cod_buf_from_UFO[884] = asum2;
```

```
quint16 xsum2 = XOR2(cod_buf_from_UFO);
cod_buf_from_UFO[885] = xsum2;
   }
//char cod_buf_from_UFO_2[891];
str2 = ui->lineEdit_2->text().toInt();
   if(str2)
   {
      cod_buf_from_UFO[0] = 0x10; //DLE
      cod_buf_from_UFO[1] = 0x02; //STX
      cod_buf_from_UFO[2] = 0x01; //MB
      cod_buf_from_UFO[3] = 0x01; //SB
      for(quint16 i = 4; i < 10; i++) //DataTime
      {
         cod_buf_from_UFO[i] = 0x11;
      }
      cod_buf_from_UFO[11] = 0x10;
      for(quint16 i = 12; i < 612; i++) //Архивные значения дискретных сигналов...
      {
         cod_buf_from_UFO[i] = 0x04;
      }
      for(quint16 i = 612; i < 632; i++) //Архивные значения реактивности канальной
      {
         cod_buf_from_UFO[i] = str2;
      }
      for(quint16 i = 632; i < 652; i++) // Архивные значения реактивности (среднее по комплекту)
```

```
{
   cod_buf_from_UFO[i] = -0x64;
}
for(quint16 i = 652; i < 656; i++)
{
   cod_buf_from_UFO[i] = 0x04;
}
cod_buf_from_UFO[656] = 0xfa;
cod_buf_from_UFO[657] = 0xfa;
for(quint16 i = 658; i < 662; i++)
{
   cod_buf_from_UFO[i] = 0x04;
}
cod_buf_from_UFO[662] = 0xfa;
cod_buf_from_UFO[663] = 0xfa;
for(quint16 i = 664; i < 668; i++)
{
   cod_buf_from_UFO[i] = 0x04;
}
cod_buf_from_UFO[668] = 0xfa;
cod_buf_from_UFO[669] = 0xfa;
for(quint16 i = 670; i < 674; i++)
{
   cod_buf_from_UFO[i] = 0x04;
}
cod_buf_from_UFO[674] = 0xfa;
cod_buf_from_UFO[675] = 0xfa;
for(quint16 i = 676; i < 680; i++)
```

```
{
   cod_buf_from_UFO[i] = 0x96;
}
cod_buf_from_UFO[680] = 0x02;
cod_buf_from_UFO[681] = 0x02;
for(quint16 i = 682; i < 686; i++)
{
   cod_buf_from_UFO[i] = 0x01;
}
cod_buf_from_UFO[686] = 0x02;
cod_buf_from_UFO[687] = 0x02;
for(quint16 i = 688; i < 692; i++)
{
   cod_buf_from_UFO[i] = 0x00;
}
cod_buf_from_UFO[692] = 0x02;
cod_buf_from_UFO[693] = 0x02;

cod_buf_from_UFO[694] = 0x02;
cod_buf_from_UFO[695] = 0x02;
cod_buf_from_UFO[696] = 0x02;
cod_buf_from_UFO[697] = 0x02;
for(quint16 i = 698; i < 702; i++)
{
   cod_buf_from_UFO[i] = 0x06;
}
for(quint16 i = 702; i < 710; i++)
{
```

```
    cod_buf_from_UFO[i] = 0x08;
}
for(quint16 i = 710; i < 798; i++)
{
    cod_buf_from_UFO[i] = 0x57;
}
cod_buf_from_UFO[798] = 0x02;
cod_buf_from_UFO[799] = 0x02;
for(quint16 i = 800; i < 816; i++)
{
    cod_buf_from_UFO[i] = 0x10;
}
for(quint16 i = 816; i < 858; i++)
{
    cod_buf_from_UFO[i] = 0x29;
}
for(quint16 i = 858; i < 870; i++)
{
    cod_buf_from_UFO[i] = 0xc;
}
for(quint16 i = 870; i < 886; i++)
{
    cod_buf_from_UFO[i] = 0x10;
}
cod_buf_from_UFO[886] = 0x01; //ADC
cod_buf_from_UFO[887] = 0x01; //XOR
```

```
        //buffer_UFO_MIF_BS[887]          =          adc_sum(buffer_UFO_MIF_BS,
sizeof(buffer_UFO_MIF_BS));
        //buffer_UFO_MIF_BS[888]          =          xor_sum(buffer_UFO_MIF_BS,
sizeof(buffer_UFO_MIF_BS));
        cod_buf_from_UFO[889] = 0x10; //DLE
        cod_buf_from_UFO[890] = 0x03; //ETX
quint16 asum3 = ADC3(cod_buf_from_UFO);
cod_buf_from_UFO[884] = asum3;

quint16 xsum3 = XOR3(cod_buf_from_UFO);
cod_buf_from_UFO[885] = xsum3;
    }
//char cod_buf_from_UFO_3[891];
str3 = ui->lineEdit_3->text().toInt();
    if(str3)
    {
        cod_buf_from_UFO[0] = 0x10; //DLE
        cod_buf_from_UFO[1] = 0x02; //STX
        cod_buf_from_UFO[2] = 0x01; //MB
        cod_buf_from_UFO[3] = 0x01; //SB
        for(quint16 i = 4; i < 10; i++) //DataTime
        {
            cod_buf_from_UFO[i] = 0x11;
        }
        cod_buf_from_UFO[10] = 0x10;
        cod_buf_from_UFO[11] = 0x10;
        for(quint16 i = 12; i < 612; i++) //Архивные значения дискретных сигналов...
        {
```

```
    cod_buf_from_UFO[i] = 0x04;
  }
  for(quint16 i = 612; i < 632; i++) //Архивные значения реактивности канальной
  {
    cod_buf_from_UFO[i] = 0x01;
  }
  for(quint16 i = 632; i < 652; i++) // Архивные значения реактивности (среднее по комплекту)
  {
    cod_buf_from_UFO[i] = -0x64;
  }
  for(quint16 i = 652; i < 656; i++)
  {
    cod_buf_from_UFO[i] = 0x04;
  }
  cod_buf_from_UFO[656] = 0xfa;
  cod_buf_from_UFO[657] = 0xfa;
  for(quint16 i = 658; i < 662; i++)
  {
    cod_buf_from_UFO[i] = 0x04;
  }
  cod_buf_from_UFO[662] = 0xfa;
  cod_buf_from_UFO[663] = 0xfa;
  for(quint16 i = 664; i < 668; i++)
  {
    cod_buf_from_UFO[i] = 0x04;
  }
```

```
cod_buf_from_UFO[668] = 0xfa;
cod_buf_from_UFO[669] = 0xfa;
for(quint16 i = 670; i < 674; i++)
{
   cod_buf_from_UFO[i] = 0x04;
}
cod_buf_from_UFO[674] = 0xfa;
cod_buf_from_UFO[675] = 0xfa;
for(quint16 i = 676; i < 680; i++)
{
   cod_buf_from_UFO[i] = 0x96;
}
cod_buf_from_UFO[680] = 0x02;
cod_buf_from_UFO[681] = 0x02;
for(quint16 i = 682; i < 686; i++)
{
   cod_buf_from_UFO[i] = 0x01;
}
cod_buf_from_UFO[686] = 0x02;
cod_buf_from_UFO[687] = 0x02;
for(quint16 i = 688; i < 692; i++)
{
   cod_buf_from_UFO[i] = 0x00;
}
cod_buf_from_UFO[692] = 0x02;
cod_buf_from_UFO[693] = 0x02;

cod_buf_from_UFO[694] = 0x02;
```

```
cod_buf_from_UFO[695] = 0x02;
cod_buf_from_UFO[696] = 0x02;
cod_buf_from_UFO[697] = 0x02;
for(quint16 i = 698; i < 702; i++)
{
   cod_buf_from_UFO[i] = 0x06;
}
for(quint16 i = 702; i < 710; i++)
{
   cod_buf_from_UFO[i] = 0x08;
}
for(quint16 i = 710; i < 798; i++)
{
   cod_buf_from_UFO[i] = 0x57;
}
cod_buf_from_UFO[798] = 0x02;
cod_buf_from_UFO[799] = 0x02;
for(quint16 i = 800; i < 816; i++)
{
   cod_buf_from_UFO[i] = 0x10;
}
for(quint16 i = 816; i < 858; i++)
{
   cod_buf_from_UFO[i] = 0x29;
}
for(quint16 i = 858; i < 870; i++)
{
   cod_buf_from_UFO[i] = 0xc;
```

```
        }
        for(quint16 i = 870; i < 886; i++)
        {
           cod_buf_from_UFO[i] = 0x10;
        }
        cod_buf_from_UFO[886] = str3; //ADC
        cod_buf_from_UFO[887] = 0x01; //XOR

        //buffer_UFO_MIF_BS[887]          =          adc_sum(buffer_UFO_MIF_BS,
sizeof(buffer_UFO_MIF_BS));
        //buffer_UFO_MIF_BS[888]          =          xor_sum(buffer_UFO_MIF_BS,
sizeof(buffer_UFO_MIF_BS));
        cod_buf_from_UFO[889] = 0x10; //DLE
        cod_buf_from_UFO[890] = 0x03; //ETX
quint16 asum4 = ADC4(cod_buf_from_UFO);
cod_buf_from_UFO[884] = asum4;

quint16 xsum4 = XOR4(cod_buf_from_UFO);
cod_buf_from_UFO[885] = xsum4;
   }
//char cod_buf_from_UFO_4[891];
str4 = ui->lineEdit_4->text().toInt();
if(str4)
{
   cod_buf_from_UFO[0] = 0x10; //DLE
   cod_buf_from_UFO[1] = 0x02; //STX
   cod_buf_from_UFO[2] = 0x01; //MB
   cod_buf_from_UFO[3] = 0x001; //SB
```

```
    for(quint16 i = 4; i < 10; i++) //DataTime
    {
        cod_buf_from_UFO[i] = 0x11;
    }
   cod_buf_from_UFO[10] = 0x10;
   cod_buf_from_UFO[11] = 0x10;
    for(quint16 i = 12; i < 612; i++) //Архивные значения дискретных сигналов...
    {
        cod_buf_from_UFO[i] = 0x04;
    }
    for(quint16 i = 612; i < 632; i++) //Архивные значения реактивности канальной
    {
        cod_buf_from_UFO[i] = 0x01;
    }
    for(quint16 i = 632; i < 652; i++) // Архивные значения реактивности (среднее по комплекту)
    {
        cod_buf_from_UFO[i] = -0x64;
    }
    for(quint16 i = 652; i < 656; i++)
    {
        cod_buf_from_UFO[i] = 0x04;
    }
    cod_buf_from_UFO[656] = 0xfa;
    cod_buf_from_UFO[657] = 0xfa;
    for(quint16 i = 658; i < 662; i++)
    {
        cod_buf_from_UFO[i] = 0x04;
```

```
}
cod_buf_from_UFO[662] = 0xfa;
cod_buf_from_UFO[663] = 0xfa;
for(quint16 i = 664; i < 668; i++)
{
   cod_buf_from_UFO[i] = 0x04;
}
cod_buf_from_UFO[668] = 0xfa;
cod_buf_from_UFO[669] = 0xfa;
for(quint16 i = 670; i < 674; i++)
{
   cod_buf_from_UFO[i] = 0x04;
}
cod_buf_from_UFO[674] = 0xfa;
cod_buf_from_UFO[675] = 0xfa;
for(quint16 i = 676; i < 680; i++)
{
   cod_buf_from_UFO[i] = 0x96;
}
cod_buf_from_UFO[680] = 0x02;
cod_buf_from_UFO[681] = 0x02;
for(quint16 i = 682; i < 686; i++)
{
   cod_buf_from_UFO[i] = 0x01;
}
cod_buf_from_UFO[686] = 0x02;
cod_buf_from_UFO[687] = 0x02;
for(quint16 i = 688; i < 692; i++)
```

```
{
    cod_buf_from_UFO[i] = 0x00;
}
cod_buf_from_UFO[692] = 0x02;
cod_buf_from_UFO[693] = 0x02;

cod_buf_from_UFO[694] = 0x02;
cod_buf_from_UFO[695] = 0x02;
cod_buf_from_UFO[696] = 0x02;
cod_buf_from_UFO[697] = 0x02;
for(quint16 i = 698; i < 702; i++)
{
    cod_buf_from_UFO[i] = 0x06;
}
for(quint16 i = 702; i < 710; i++)
{
    cod_buf_from_UFO[i] = 0x08;
}
for(quint16 i = 710; i < 798; i++)
{
    cod_buf_from_UFO[i] = 0x57;
}
cod_buf_from_UFO[798] = 0x02;
cod_buf_from_UFO[799] = 0x02;
for(quint16 i = 800; i < 816; i++)
{
    cod_buf_from_UFO[i] = 0x10;
}
```

```
    for(quint16 i = 816; i < 858; i++)
    {
        cod_buf_from_UFO[i] = 0x29;
    }
    for(quint16 i = 858; i < 870; i++)
    {
        cod_buf_from_UFO[i] = 0xc;
    }
    for(quint16 i = 870; i < 886; i++)
    {
        cod_buf_from_UFO[i] = 0x10;
    }
    cod_buf_from_UFO[886] = 0x01; //ADC
    cod_buf_from_UFO[887] = str4; //XOR
    cod_buf_from_UFO[889] = 0x10; //DLE
    cod_buf_from_UFO[890] = 0x03; //ETX
quint16 asum5 = ADC5(cod_buf_from_UFO);
cod_buf_from_UFO[884] = asum5;

quint16 xsum5 = XOR5(cod_buf_from_UFO);
cod_buf_from_UFO[885] = xsum5;
}
//str1 = ui->lineEdit->text().toInt() &&
//       str2 = ui->lineEdit_2->text().toInt() &&
//                  str3 = ui->lineEdit_3->text().toInt() &&
//                             str4 = ui->lineEdit_4->text().toInt();
//char cod_buf_from_UFO_5[891];
    if(str1 && str2 && str3 && str4)
```

```
{
  cod_buf_from_UFO[0] = 0x10; //DLE
  cod_buf_from_UFO[1] = 0x02; //STX
  cod_buf_from_UFO[2] = 0x01; //MB
  cod_buf_from_UFO[3] = str1; //SB
   for(quint16 i = 4; i < 10; i++) //DataTime
   {
      cod_buf_from_UFO[i] = 0x11;
   }
   cod_buf_from_UFO[10] = 0x10;
   cod_buf_from_UFO[11] = 0x10;
   for(quint16 i = 12; i < 612; i++) //Архивные значения дискретных сигналов...
   {
      cod_buf_from_UFO[i] = 0x04;
   }
   for(quint16 i = 612; i < 632; i++) //Архивные значения реактивности канальной
   {
      cod_buf_from_UFO[i] = str2;
   }
   for(quint16 i = 632; i < 652; i++) // Архивные значения реактивности (среднее по комплекту)
   {
      cod_buf_from_UFO[i] = -0x64;
   }
   for(quint16 i = 652; i < 656; i++)
   {
      cod_buf_from_UFO[i] = 0x04;
```

```
}
cod_buf_from_UFO[656] = 0xfa;
cod_buf_from_UFO[657] = 0xfa;
for(quint16 i = 658; i < 662; i++)
{
   cod_buf_from_UFO[i] = 0x04;
}
cod_buf_from_UFO[662] = 0xfa;
cod_buf_from_UFO[663] = 0xfa;
for(quint16 i = 664; i < 668; i++)
{
   cod_buf_from_UFO[i] = 0x04;
}
cod_buf_from_UFO[668] = 0xfa;
cod_buf_from_UFO[669] = 0xfa;
for(quint16 i = 670; i < 674; i++)
{
   cod_buf_from_UFO[i] = 0x04;
}
cod_buf_from_UFO[674] = 0xfa;
cod_buf_from_UFO[675] = 0xfa;
for(quint16 i = 676; i < 680; i++)
{
   cod_buf_from_UFO[i] = 0x96;
}
cod_buf_from_UFO[680] = 0x02;
cod_buf_from_UFO[681] = 0x02;
for(quint16 i = 682; i < 686; i++)
```

```
{
   cod_buf_from_UFO[i] = 0x01;
}
cod_buf_from_UFO[686] = 0x02;
cod_buf_from_UFO[687] = 0x02;
for(quint16 i = 688; i < 692; i++)
{
   cod_buf_from_UFO[i] = 0x00;
}
cod_buf_from_UFO[692] = 0x02;
cod_buf_from_UFO[693] = 0x02;

cod_buf_from_UFO[694] = 0x02;
cod_buf_from_UFO[695] = 0x02;
cod_buf_from_UFO[696] = 0x02;
cod_buf_from_UFO[697] = 0x02;
for(quint16 i = 698; i < 702; i++)
{
   cod_buf_from_UFO[i] = 0x06;
}
for(quint16 i = 702; i < 710; i++)
{
   cod_buf_from_UFO[i] = 0x08;
}
for(quint16 i = 710; i < 798; i++)
{
   cod_buf_from_UFO[i] = 0x57;
}
```

```
cod_buf_from_UFO[798] = 0x02;
cod_buf_from_UFO[799] = 0x02;
for(quint16 i = 800; i < 816; i++)
{
   cod_buf_from_UFO[i] = 0x10;
}
for(quint16 i = 816; i < 858; i++)
{
   cod_buf_from_UFO[i] = 0x29;
}
for(quint16 i = 858; i < 870; i++)
{
   cod_buf_from_UFO[i] = 0xc;
}
for(quint16 i = 870; i < 886; i++)
{
   cod_buf_from_UFO[i] = 0x10;
}
cod_buf_from_UFO[886] = str3; //ADC
cod_buf_from_UFO[887] = str4; //XOR

//buffer_UFO_MIF_BS[887] = adc_sum(buffer_UFO_MIF_BS,
sizeof(buffer_UFO_MIF_BS));
//buffer_UFO_MIF_BS[888] = xor_sum(buffer_UFO_MIF_BS,
sizeof(buffer_UFO_MIF_BS));
cod_buf_from_UFO[889] = 0x10; //DLE
cod_buf_from_UFO[890] = 0x03; //ETX
quint16 asum6 = ADC6(cod_buf_from_UFO);
```

```
cod_buf_from_UFO[884] = asum6;

quint16 xsum6 = XOR6(cod_buf_from_UFO);
cod_buf_from_UFO[885] = xsum6;
   }
   else
   {
      cod_buf_from_UFO[0] = 0x10; //DLE
      cod_buf_from_UFO[1] = 0x02; //STX
      cod_buf_from_UFO[2] = 0x01; //MB
      cod_buf_from_UFO[3] = 0x01; //SB
      for(quint16 i = 4; i < 10; i++) //DataTime
      {
         cod_buf_from_UFO[i] = 0x11;
      }
      cod_buf_from_UFO[10] = 0x10;
      cod_buf_from_UFO[11] = 0x10;
      for(quint16 i = 12; i < 612; i++) //Архивные значения дискретных сигналов...
      {
         cod_buf_from_UFO[i] = 0x04;
      }
      for(quint16 i = 612; i < 632; i++) //Архивные значения реактивности канальной
      {
         cod_buf_from_UFO[i] = 0x01;
      }
      for(quint16 i = 632; i < 652; i++) // Архивные значения реактивности (среднее по комплекту)
```

```
{
   cod_buf_from_UFO[i] = -0x64;
}
for(quint16 i = 652; i < 656; i++)
{
   cod_buf_from_UFO[i] = 0x04;
}
cod_buf_from_UFO[656] = 0xfa;
cod_buf_from_UFO[657] = 0xfa;
for(quint16 i = 658; i < 662; i++)
{
   cod_buf_from_UFO[i] = 0x04;
}
cod_buf_from_UFO[662] = 0xfa;
cod_buf_from_UFO[663] = 0xfa;
for(quint16 i = 664; i < 668; i++)
{
   cod_buf_from_UFO[i] = 0x04;
}
cod_buf_from_UFO[668] = 0xfa;
cod_buf_from_UFO[669] = 0xfa;
for(quint16 i = 670; i < 674; i++)
{
   cod_buf_from_UFO[i] = 0x04;
}
cod_buf_from_UFO[674] = 0xfa;
cod_buf_from_UFO[675] = 0xfa;
for(quint16 i = 676; i < 680; i++)
```

```
{
   cod_buf_from_UFO[i] = 0x96;
}
cod_buf_from_UFO[680] = 0x02;
cod_buf_from_UFO[681] = 0x02;
for(quint16 i = 682; i < 686; i++)
{
   cod_buf_from_UFO[i] = 0x01;
}
cod_buf_from_UFO[686] = 0x02;
cod_buf_from_UFO[687] = 0x02;
for(quint16 i = 688; i < 692; i++)
{
   cod_buf_from_UFO[i] = 0x00;
}
cod_buf_from_UFO[692] = 0x02;
cod_buf_from_UFO[693] = 0x02;

cod_buf_from_UFO[694] = 0x02;
cod_buf_from_UFO[695] = 0x02;
cod_buf_from_UFO[696] = 0x02;
cod_buf_from_UFO[697] = 0x02;
for(quint16 i = 698; i < 702; i++)
{
   cod_buf_from_UFO[i] = 0x06;
}
for(quint16 i = 702; i < 710; i++)
{
```

```
    cod_buf_from_UFO[i] = 0x08;
}
for(quint16 i = 710; i < 798; i++)
{
    cod_buf_from_UFO[i] = 0x57;
}
cod_buf_from_UFO[798] = 0x02;
cod_buf_from_UFO[799] = 0x02;
for(quint16 i = 800; i < 816; i++)
{
    cod_buf_from_UFO[i] = 0x10;
}
for(quint16 i = 816; i < 858; i++)
{
    cod_buf_from_UFO[i] = 0x29;
}
for(quint16 i = 858; i < 870; i++)
{
    cod_buf_from_UFO[i] = 0xc;
}
for(quint16 i = 870; i < 886; i++)
{
    cod_buf_from_UFO[i] = 0x10;
}
cod_buf_from_UFO[886] = 0x01; //ADC
cod_buf_from_UFO[887] = 0x01; //XOR
cod_buf_from_UFO[889] = 0x10; //DLE
cod_buf_from_UFO[890] = 0x03; //ETX
```

```
quint16 asum7 = ADC7(cod_buf_from_UFO);
cod_buf_from_UFO[884] = asum7;

quint16 xsum7 = XOR7(cod_buf_from_UFO);
cod_buf_from_UFO[885] = xsum7;
   }
/*buffer for Byte Staffing*/
char buf_byte_stuff[895];
buf_byte_stuff[0] = cod_buf_from_UFO[0]; //DLE
buf_byte_stuff[1] = cod_buf_from_UFO[1]; //STX
buf_byte_stuff[2] = cod_buf_from_UFO[2]; //MB
if(str1 == 0x10) //SB
{
  buf_byte_stuff[3] = 0x10;
  buf_byte_stuff[4] = 0x10;
}else
{
   buf_byte_stuff[3] = cod_buf_from_UFO[3];
}
if(buf_byte_stuff[4] == 0x10)
{
   for(quint16 i = 5; i < 11; i++) //DataTime
   {
      buf_byte_stuff[i] = cod_buf_from_UFO[i-1];
   }
}else
{
   for(quint16 i = 4; i < 10; i++) //DataTime
```

```
    {
        buf_byte_stuff[i] = cod_buf_from_UFO[i];
    }
}
if(buf_byte_stuff[4] == 0x10)
{
    buf_byte_stuff[11] = 0x10;
    buf_byte_stuff[12] = 0x10;
}else
{
    buf_byte_stuff[10] = 0x10;
    buf_byte_stuff[11] = 0x10;
}
if(buf_byte_stuff[4] == 0x10)
{
    buf_byte_stuff[13] = 0x10;
    buf_byte_stuff[14] = 0x10;
}else
{
    buf_byte_stuff[12] = 0x10;
    buf_byte_stuff[13] = 0x10;
}
if(buf_byte_stuff[4] == 0x10)
{
    for(quint16 i = 15; i < 615; i++) //Архивные значения дискретных сигналов...
    {
        buf_byte_stuff[i] = cod_buf_from_UFO[i-3];
    }
```

```
}else
{
   for(quint16 i = 14; i < 614; i++) //Архивные значения дискретных сигналов...
   {
      buf_byte_stuff[i] = cod_buf_from_UFO[i-2];
   }
}
if(buf_byte_stuff[4] == 0x10)
{
   for(quint16 i = 615; i < 635; i++) //Архивные значения реактивности канальной
   {
      buf_byte_stuff[i] = cod_buf_from_UFO[i-3];
   }
}else
{
   for(quint16 i = 614; i < 634; i++) //Архивные значения реактивности канальной
   {
      buf_byte_stuff[i] = cod_buf_from_UFO[i-2];
   }
}
if(buf_byte_stuff[4] == 0x10)
{
   for(quint16 i = 635; i < 655; i++) // Архивные значения реактивности (среднее по комплекту)
   {
      buf_byte_stuff[i] = cod_buf_from_UFO[i-3];
   }
}else
```

```
{
   for(quint16 i = 634; i < 654; i++) // Архивные значения реактивности (среднее по комплекту)
   {
      buf_byte_stuff[i] = cod_buf_from_UFO[i-2];
   }
}
if(buf_byte_stuff[4] == 0x10)
{
   for(quint16 i = 655; i < 659; i++)
   {
      buf_byte_stuff[i] = cod_buf_from_UFO[i-3];
   }
}else
{
   for(quint16 i = 654; i < 658; i++)
   {
      buf_byte_stuff[i] = cod_buf_from_UFO[i-2];
   }
}
if(buf_byte_stuff[4] == 0x10)
{
  buf_byte_stuff[659] = cod_buf_from_UFO[656];
  buf_byte_stuff[660] = cod_buf_from_UFO[657];
}else
{
   buf_byte_stuff[658] = cod_buf_from_UFO[656];
   buf_byte_stuff[659] = cod_buf_from_UFO[657];
```

```
}
if(buf_byte_stuff[4] == 0x10)
{
   for(quint16 i = 661; i < 665; i++)
   {
      buf_byte_stuff[i] = cod_buf_from_UFO[i-3];
   }
}else
{
   for(quint16 i = 660; i < 664; i++)
   {
      buf_byte_stuff[i] = cod_buf_from_UFO[i-2];
   }
}
if(buf_byte_stuff[4] == 0x10)
{
   buf_byte_stuff[665] = cod_buf_from_UFO[662];
   buf_byte_stuff[666] = cod_buf_from_UFO[663];
}else
{
   buf_byte_stuff[664] = cod_buf_from_UFO[662];
   buf_byte_stuff[665] = cod_buf_from_UFO[663];
}
if(buf_byte_stuff[4] == 0x10)
{
   for(quint16 i = 667; i < 671; i++)
   {
      buf_byte_stuff[i] = cod_buf_from_UFO[i-3];
```

```
    }
}else
{
    for(quint16 i = 666; i < 670; i++)
    {
        buf_byte_stuff[i] = cod_buf_from_UFO[i-2];
    }
}
if(buf_byte_stuff[4] == 0x10)
{
    buf_byte_stuff[671] = cod_buf_from_UFO[668];
    buf_byte_stuff[672] = cod_buf_from_UFO[669];
}else
{
    buf_byte_stuff[670] = cod_buf_from_UFO[668];
    buf_byte_stuff[671] = cod_buf_from_UFO[669];
}
if(buf_byte_stuff[4] == 0x10)
{
    for(quint16 i = 673; i < 677; i++)
    {
        buf_byte_stuff[i] = cod_buf_from_UFO[i-3];
    }
}else
{
    for(quint16 i = 672; i < 676; i++)
    {
        buf_byte_stuff[i] = cod_buf_from_UFO[i-2];
```

```
    }
}
if(buf_byte_stuff[4] == 0x10)
{
    buf_byte_stuff[677] = cod_buf_from_UFO[674];
    buf_byte_stuff[678] = cod_buf_from_UFO[675];
}else
{
    buf_byte_stuff[676] = cod_buf_from_UFO[674];
    buf_byte_stuff[677] = cod_buf_from_UFO[675];
}
if(buf_byte_stuff[4] == 0x10)
{
    for(quint16 i = 679; i < 683; i++)
    {
        buf_byte_stuff[i] = cod_buf_from_UFO[i-3];
    }
}else
{
    for(quint16 i = 678; i < 682; i++)
    {
        buf_byte_stuff[i] = cod_buf_from_UFO[i-2];
    }
}
if(buf_byte_stuff[4] == 0x10)
{
    buf_byte_stuff[683] = cod_buf_from_UFO[680];
    buf_byte_stuff[684] = cod_buf_from_UFO[681];
```

```
}else
{
   buf_byte_stuff[682] = cod_buf_from_UFO[680];
   buf_byte_stuff[683] = cod_buf_from_UFO[681];
}
if(buf_byte_stuff[4] == 0x10)
{
   for(quint16 i = 685; i < 689; i++)
   {
      buf_byte_stuff[i] = cod_buf_from_UFO[i-3];
   }
}else
{
   for(quint16 i = 684; i < 688; i++)
   {
      buf_byte_stuff[i] = cod_buf_from_UFO[i-2];
   }
}
if(buf_byte_stuff[4] == 0x10)
{
   buf_byte_stuff[689] = cod_buf_from_UFO[686];
   buf_byte_stuff[690] = cod_buf_from_UFO[687];
}else
{
   buf_byte_stuff[688] = cod_buf_from_UFO[686];
   buf_byte_stuff[689] = cod_buf_from_UFO[687];
}
if(buf_byte_stuff[4] == 0x10)
```

```
{
   for(quint16 i = 691; i < 695; i++)
   {
      buf_byte_stuff[i] = cod_buf_from_UFO[i-3];
   }
}else
{
   for(quint16 i = 690; i < 694; i++)
   {
      buf_byte_stuff[i] = cod_buf_from_UFO[i-2];
   }
}
if(buf_byte_stuff[4] == 0x10)
{
   buf_byte_stuff[695] = cod_buf_from_UFO[692];
   buf_byte_stuff[696] = cod_buf_from_UFO[693];
}else
{
   buf_byte_stuff[694] = cod_buf_from_UFO[692];
   buf_byte_stuff[695] = cod_buf_from_UFO[693];
}
if(buf_byte_stuff[4] == 0x10)
{
   buf_byte_stuff[697] = cod_buf_from_UFO[694];
   buf_byte_stuff[968] = cod_buf_from_UFO[695];
}else
{
   buf_byte_stuff[696] = cod_buf_from_UFO[694];
```

```
    buf_byte_stuff[697] = cod_buf_from_UFO[695];
}
if(buf_byte_stuff[4] == 0x10)
{
    buf_byte_stuff[699] = cod_buf_from_UFO[696];
    buf_byte_stuff[700] = cod_buf_from_UFO[697];
}else
{
    buf_byte_stuff[698] = cod_buf_from_UFO[696];
    buf_byte_stuff[699] = cod_buf_from_UFO[697];
}
if(buf_byte_stuff[4] == 0x10)
{
    for(quint16 i = 701; i < 705; i++)
    {
        buf_byte_stuff[i] = cod_buf_from_UFO[i-3];
    }
}else
{
    for(quint16 i = 700; i < 704; i++)
    {
        buf_byte_stuff[i] = cod_buf_from_UFO[i-2];
    }
}
if(buf_byte_stuff[4] == 0x10)
{
    for(quint16 i = 705; i < 713; i++)
    {
```

```
        buf_byte_stuff[i] = cod_buf_from_UFO[i-3];
    }
}else
{
    for(quint16 i = 704; i < 712; i++)
    {
        buf_byte_stuff[i] = cod_buf_from_UFO[i-2];
    }
}
if(buf_byte_stuff[4] == 0x10)
{
    for(quint16 i = 713; i < 801; i++)
    {
        buf_byte_stuff[i] = cod_buf_from_UFO[i-3];
    }
}else
{
    for(quint16 i = 712; i < 800; i++)
    {
        buf_byte_stuff[i] = cod_buf_from_UFO[i-2];
    }
}
if(buf_byte_stuff[4] == 0x10)
{
    buf_byte_stuff[801] = cod_buf_from_UFO[798];
    buf_byte_stuff[802] = cod_buf_from_UFO[799];
}else
{
```

```
    buf_byte_stuff[800] = cod_buf_from_UFO[798];
    buf_byte_stuff[801] = cod_buf_from_UFO[799];
}
if(buf_byte_stuff[4] == 0x10)
{
    for(quint16 i = 803; i < 819; i++)
    {
        buf_byte_stuff[i] = cod_buf_from_UFO[i-3];
    }
}else
{
    for(quint16 i = 802; i < 818; i++)
    {
        buf_byte_stuff[i] = cod_buf_from_UFO[i-2];
    }
}
if(buf_byte_stuff[4] == 0x10)
{
    for(quint16 i = 819; i < 861; i++)
    {
        buf_byte_stuff[i] = cod_buf_from_UFO[i-3];
    }
}else
{
    for(quint16 i = 818; i < 860; i++)
    {
        buf_byte_stuff[i] = cod_buf_from_UFO[i-2];
    }
```

```
}
if(buf_byte_stuff[4] == 0x10)
{
   for(quint16 i = 861; i < 873; i++)
   {
      buf_byte_stuff[i] = cod_buf_from_UFO[i-3];
   }
}else
{
   for(quint16 i = 860; i < 872; i++)
   {
      buf_byte_stuff[i] = cod_buf_from_UFO[i-2];
   }
}
if(buf_byte_stuff[4] == 0x10)
{
   for(quint16 i = 873; i < 889; i++)
   {
      buf_byte_stuff[i] = cod_buf_from_UFO[i-3];
   }
}else
{
   for(quint16 i = 872; i < 888; i++)
   {
      buf_byte_stuff[i] = cod_buf_from_UFO[i-2];
   }
}
if(buf_byte_stuff[4] && str3 == 0x10)
```

```
{
   buf_byte_stuff[889] = 0x10; //ADC
   buf_byte_stuff[890] = 0x10; //ADC
}else if(str3 == 0x10)
{
   buf_byte_stuff[888] = 0x10; //ADC
   buf_byte_stuff[889] = 0x10; //ADC
}else if(buf_byte_stuff[4] == 0x10)
{
   buf_byte_stuff[889] = cod_buf_from_UFO[886]; //ADC
}else
{
   buf_byte_stuff[888] = cod_buf_from_UFO[886]; //ADC
}
if(buf_byte_stuff[890] && str4 == 0x10) //если в ячейке [4] есть знач. 0x10 и установ. str3 и установ. str4
{
   buf_byte_stuff[891] = 0x10; //XOR
   buf_byte_stuff[892] = 0x10; //XOR
}else if(buf_byte_stuff[889] && str4 == 0x10) // если установ str3 без знач. в яч. №[4] и установ. str4
{
   buf_byte_stuff[890] = 0x10; //XOR
   buf_byte_stuff[891] = 0x10; //XOR
}else if(buf_byte_stuff[4] && str4 == 0x10)
{
   buf_byte_stuff[890] = 0x10;
   buf_byte_stuff[891] = 0x10;
```

```
}else if(buf_byte_stuff[4] && str3 == 0x10)
{
   buf_byte_stuff[891] = cod_buf_from_UFO[887];
}else if(str4 == 0x10)
{
   buf_byte_stuff[890] = 0x10; //XOR
   buf_byte_stuff[891] = 0x10; //XOR
}else if(str3 == 0x10)
{
   buf_byte_stuff[890] = cod_buf_from_UFO[887];
}else if(buf_byte_stuff[4] == 0x10)
{
   buf_byte_stuff[890] = cod_buf_from_UFO[887]; //XOR
}else
{
   buf_byte_stuff[889] = cod_buf_from_UFO[887]; //XOR
}
//if(buf_byte_stuff[4] == 0x10)
//{
//    buf_byte_stuff[892] = cod_buf_from_UFO[889]; //DLE
//    buf_byte_stuff[893] = cod_buf_from_UFO[890]; //ETX
//}else if(str3 == 0x10)
//{
//    buf_byte_stuff[891] = cod_buf_from_UFO[889]; //DLE
//    buf_byte_stuff[892] = cod_buf_from_UFO[890]; //ETX
//}else if(buf_byte_stuff[4] && str3 == 0x10)
//{
//    buf_byte_stuff[892] = cod_buf_from_UFO[889]; //DLE
```

```
//    buf_byte_stuff[893] = cod_buf_from_UFO[890]; //ETX
//}else if(str4 == 0x10)
//{
//    buf_byte_stuff[891] = cod_buf_from_UFO[889]; //DLE
//    buf_byte_stuff[892] = cod_buf_from_UFO[890]; //ETX
//}else if(buf_byte_stuff[4] && str4 == 0x10)
//{
//    buf_byte_stuff[893] = cod_buf_from_UFO[889]; //DLE
//    buf_byte_stuff[894] = cod_buf_from_UFO[890]; //ETX
//}else if(buf_byte_stuff[889] && str4 == 0x10)
//{
//    buf_byte_stuff[892] = cod_buf_from_UFO[889]; //DLE
//    buf_byte_stuff[893] = cod_buf_from_UFO[890]; //ETX
//}else if(buf_byte_stuff[890] && str4 == 0x10)
//{
//    buf_byte_stuff[892] = cod_buf_from_UFO[889]; //DLE
//    buf_byte_stuff[893] = cod_buf_from_UFO[890]; //ETX
//}else
//{
//    buf_byte_stuff[891] = cod_buf_from_UFO[889]; //DLE
//    buf_byte_stuff[892] = cod_buf_from_UFO[890]; //ETX
//}

//for(quint16 i = 2; i < 889; i++)
//{
//    buf_byte_stuff[0] = cod_buf_from_UFO[0];
//    buf_byte_stuff[1] = cod_buf_from_UFO[1];
```

```
//    if(cod_buf_from_UFO[i] != 0x10) //если значение не равно 0x10
//    {
//              buf_byte_stuff[i] = cod_buf_from_UFO[i]; //переноситься содержимое ячейки кодированного буфера в аналогичную ячейку буффера buf_byte_stuff
//    } else if(cod_buf_from_UFO[i] == 0x10) //если значение равно 0x10
//    {
//        buf_byte_stuff[i] = 0x10;
//        buf_byte_stuff[i+1] = 0x10;
//    }
//}
/*end buf_byte_stuff*/
    serial->write(cod_buf_from_UFO, sizeof(cod_buf_from_UFO));
       lblCntTx++;
      ui->lblCntTx->setText("Отправлено " + QString::number(lblCntTx));
      mesTransmitted = 1;

      ui->textEdit->append(time->currentTime().toString("hh:mm:ss") + " -> " + " Отправлено ");
/*end coding buffer*/
}

/*Контрольная сумма для оригинального буфера*/
quint16 MainWindow::ADC1(char *aData)
{
    //quint8 pData = atoi(Data);
    //quint8 pData = static_cast<quint8>(*Data);
    static quint16 sum_adc1 = 0;
    for(quint16 ix = 0; ix < 884; ix++)
```

```
    {
        sum_adc1 += (quint8)*aData;//сумма
        if(sum_adc1 > 0xff)
        {
            sum_adc1 &= ~(1 << 8); //установка 8-ого бит в 0
            sum_adc1 += 1; //добавление 1-го разряда
        }
        *aData++;
    }

    return sum_adc1;
}
quint16 MainWindow::XOR1(char *xData)
{
  static quint16 sum_xor1 = 0;
  for(quint16 i = 0; i < 884; i++)
  {
      sum_xor1 ^= (quint8)*xData;
      *xData++;
  }

  return sum_xor1;
}
/*end*/
/*Контрольная сумма для кодированого буфера str1*/
quint16 MainWindow::ADC2(char *aData)
{
    static quint16 sum_adc2;
```

```
    for(quint16 ix = 3; ix < 888; ix++)
    {
        sum_adc2 += (quint8)*aData;//сумма
        if(sum_adc2 > 0xff)
        {
            sum_adc2 &= ~(1 << 8); //установка 8-ого бит в 0
            sum_adc2 += 1; //добавление 1-го разряда
        }
        *aData++;
    }
    return sum_adc2;
}
quint16 MainWindow::XOR2(char *xData)
{
  static quint16 sum_xor2;
  for(quint16 i = 3; i < 888; i++)
  {
      sum_xor2 ^= (quint8)*xData;
      *xData++;
  }
  return sum_xor2;
}
/*end*/
/*Контрольная сумма для кодированого буфера str2*/
quint16 MainWindow::ADC3(char *aData)
{
    static quint16 sum_adc3;
    for(quint16 ix = 3; ix < 888; ix++)
```

```
    {
        sum_adc3 += (quint8)*aData;//сумма
        if(sum_adc3 > 0xff)
        {
            sum_adc3 &= ~(1 << 8); //установка 8-ого бит в 0
            sum_adc3 += 1; //добавление 1-го разряда
        }
        *aData++;
    }
    return sum_adc3;
}
quint16 MainWindow::XOR3(char *xData)
{
  static quint16 sum_xor3;
  for(quint16 i = 3; i < 888; i++)
  {
      sum_xor3 ^= (quint8)*xData;
      *xData++;
  }
  return sum_xor3;
}
/*end*/
/*Контрольная сумма для кодированого буфера str3*/
quint16 MainWindow::ADC4(char *aData)
{
    static quint16 sum_adc4;
    for(quint16 ix = 3; ix < 888; ix++)
    {
```

```
        sum_adc4 += (quint8)*aData;//сумма
        if(sum_adc4 > 0xff)
        {
           sum_adc4 &= ~(1 << 8); //установка 8-ого бит в 0
           sum_adc4 += 1; //добавление 1-го разряда
        }
        *aData++;
    }
    return sum_adc4;
}
quint16 MainWindow::XOR4(char *xData)
{
  static quint16 sum_xor4;
  for(quint16 i = 3; i < 888; i++)
  {
     sum_xor4 ^= (quint8)*xData;
     *xData++;
  }
  return sum_xor4;
}
/*end*/
/*Контрольная сумма для кодированого буфера str4*/
quint16 MainWindow::ADC5(char *aData)
{
    static quint16 sum_adc5;
    for(quint16 ix = 3; ix < 888; ix++)
    {
        sum_adc5 += (quint8)*aData;//сумма
```

```
        if(sum_adc5 > 0xff)
        {
           sum_adc5 &= ~(1 << 8); //установка 8-ого бит в 0
           sum_adc5 += 1; //добавление 1-го разряда
        }
        *aData++;
    }
    return sum_adc5;
}
quint16 MainWindow::XOR5(char *xData)
{
  static quint16 sum_xor5;
  for(quint16 i = 3; i < 888; i++)
  {
     sum_xor5 ^= (quint8)*xData;
     *xData++;
  }
  return sum_xor5;
}
/*end*/
/*Контрольная сумма для кодированого буфера str1 str2 str3 str4*/
quint16 MainWindow::ADC6(char *aData)
{
    static quint16 sum_adc6;
    for(quint16 ix = 3; ix < 888; ix++)
    {
        sum_adc6 += (quint8)*aData;//сумма
        if(sum_adc6 > 0xff)
```

```
        {
            sum_adc6 &= ~(1 << 8); //установка 8-ого бит в 0
            sum_adc6 += 1; //добавление 1-го разряда
        }
        *aData++;
    }
    return sum_adc6;
}
quint16 MainWindow::XOR6(char *xData)
{
  static quint16 sum_xor6;
  for(quint16 i = 3; i < 888; i++)
  {
      sum_xor6 ^= (quint8)*xData;
      *xData++;
  }
  return sum_xor6;
}
/*end*/
/*Контрольная сумма для кодированого буфера else*/
quint16 MainWindow::ADC7(char *aData)
{
    static quint16 sum_adc7;
    for(quint16 ix = 3; ix < 888; ix++)
    {
        sum_adc7 += (quint8)*aData;//сумма
        if(sum_adc7 > 0xff)
        {
```

```
            sum_adc7 &= ~(1 << 8); //установка 8-ого бит в 0
            sum_adc7 += 1; //добавление 1-го разряда
        }
        *aData++;
    }
    return sum_adc7;
}
quint16 MainWindow::XOR7(char *xData)
{
  static quint16 sum_xor7;
  for(quint16 i = 3; i < 888; i++)
  {
     sum_xor7 ^= (quint8)*xData;
     *xData++;
  }
  return sum_xor7;
}
/*end*/

void MainWindow::on_pbtnTest_clicked()
{
    ui->label->setStyleSheet("color: rgb(25, 0, 0)");
   if(ui->pbtnTest->text() == "Запустить тест") {
      ui->pbtnOpenSerial->setEnabled(false);
      ui->pbtnSend->setEnabled(false);
      ui->pbtnTest->setText("Остановить тест");
      timer1->start();
   }
```

```
  else {
      timer1->stop();
      timer_PC_connect->stop();
      ui->pbtnTest->setText("Запустить тест");
      ui->pbtnSend->setEnabled(true);
      ui->pbtnOpenSerial->setEnabled(true);
  }
}

void MainWindow:: tick_timer_PC_connect()
{
   timer_PC_connect_cnt++;
   if(timer_PC_connect_cnt > 4 && mesTransmitted == 1){
       mesTransmitted = 0;
       errorConnectCOM++;
       timer_PC_connect_cnt = 0;
       ui->label->setStyleSheet("color: rgb(255, 0, 0)");
   }
}

void MainWindow::on_lineEditPass_editingFinished()
{
   if(ui->lineEditPass->text() == "0000"){
      ui->lineEditPass->setText("");
      ui->textEdit->setEnabled(true);
      ui->textEdit_2->setEnabled(true);
      ui->lineEdit->setEnabled(true);
      ui->lineEdit_2->setEnabled(true);
```

```
        ui->lineEdit_3->setEnabled(true);
        ui->pbtnClearLog->show();
    } else {
        ui->lineEditPass->setText("");
    }
}

void MainWindow::on_pbtnClearLog_clicked()
{
    ui->textEdit->clear();
    ui->label->setStyleSheet("color: rgb(0, 0, 0)");
}

void MainWindow::on_pbtnSaveText_clicked()
{
    QFile log_txFile("log_tx.txt");

    if(!log_txFile.open(QIODevice::Append)){
        qDebug() << "Ошибка открытия файла 'log_tx.txt'";
        QMessageBox::critical(this,QString(ProgramTitle),"Ошибка открытия файла
'log_tx.txt'" );
    }
    QTextStream log_tx(&log_txFile);
    QString text = ui->textEdit->toPlainText();
    log_tx<<text<<endl;
    log_txFile.close();
    ui->textEdit->append(time->currentTime().toString() + " -> " + " Лог сохранен в
log_tx.txt");
```

```
}


void MainWindow::serialInputSlot()
{
   timer_PC_connect->stop();
   quint64  cnt = serial->bytesAvailable();
   ui->textEdit_2->append(time->currentTime().toString("hh:mm:ss")  +  "  ->  "  +  "
Принято " + QString::number(cnt) + " Byte");
   /**/
   transformedcod_buf_from_URO[0] = 0x01; //MB;
   transformedcod_buf_from_URO[1] = 0x01; //SB; /*из таблицы 3*/
   transformedcod_buf_from_URO[2] = transformedcod_buf_from_UFO[2];
   transformedcod_buf_from_URO[3] = transformedcod_buf_from_UFO[3];
   transformedcod_buf_from_URO[4] = transformedcod_buf_from_UFO[4];
   transformedcod_buf_from_URO[5] = transformedcod_buf_from_UFO[5];
   transformedcod_buf_from_URO[6] = transformedcod_buf_from_UFO[6];
   transformedcod_buf_from_URO[7] = transformedcod_buf_from_UFO[7];
   transformedcod_buf_from_URO[8] = transformedcod_buf_from_UFO[8];
   transformedcod_buf_from_URO[9] = transformedcod_buf_from_UFO[9];
   transformedcod_buf_from_URO[10] = 0x11; /*из таблицы 4*/
   transformedcod_buf_from_URO[11] = 0x80;
   transformedcod_buf_from_URO[12] = transformedcod_buf_from_UFO[10];
   transformedcod_buf_from_URO[13] = transformedcod_buf_from_UFO[11];
   transformedcod_buf_from_URO[14] = transformedcod_buf_from_UFO[12];
   transformedcod_buf_from_URO[15] = transformedcod_buf_from_UFO[13];
   transformedcod_buf_from_URO[16] = transformedcod_buf_from_UFO[14];
   transformedcod_buf_from_URO[17] = transformedcod_buf_from_UFO[15];
```

```
    transformedcod_buf_from_URO[18] = transformedcod_buf_from_UFO[16];
    transformedcod_buf_from_URO[19] = transformedcod_buf_from_UFO[17];
    transformedcod_buf_from_URO[20] = transformedcod_buf_from_UFO[18];
    transformedcod_buf_from_URO[21] = transformedcod_buf_from_UFO[19];
    transformedcod_buf_from_URO[22] = transformedcod_buf_from_UFO[20];
    transformedcod_buf_from_URO[23] = transformedcod_buf_from_UFO[21];

    for(uint16_t i=24; i<606; i++)
    {
      transformedcod_buf_from_URO[i]=transformedcod_buf_from_UFO[i-2];
/*25..606___23..604*/
    }

    transformedcod_buf_from_URO[606] = transformedcod_buf_from_UFO[604];
    transformedcod_buf_from_URO[607] = transformedcod_buf_from_UFO[605];
    transformedcod_buf_from_URO[608] = transformedcod_buf_from_UFO[606];
    transformedcod_buf_from_URO[609] = transformedcod_buf_from_UFO[607];
    transformedcod_buf_from_URO[610] = transformedcod_buf_from_UFO[608];
    transformedcod_buf_from_URO[611] = transformedcod_buf_from_UFO[609];
    transformedcod_buf_from_URO[612] = 0x00;
    transformedcod_buf_from_URO[613] = 0x00;
    transformedcod_buf_from_URO[614] = 0x00;
    transformedcod_buf_from_URO[615] = 0x00;
    transformedcod_buf_from_URO[616] = 0x11; /*из таблицы 7*/
    transformedcod_buf_from_URO[617] = 0x11; /*из таблицы 7*/
    transformedcod_buf_from_URO[618] = 0x11; /*из таблицы 7*/
    transformedcod_buf_from_URO[619] = 0x11; /*из таблицы 7*/
    transformedcod_buf_from_URO[620] = 0x00;
```

```
transformedcod_buf_from_URO[621] = 0x00;
transformedcod_buf_from_URO[622] = 0x00;
transformedcod_buf_from_URO[623] = 0x00;
transformedcod_buf_from_URO[624] = 0x11; /*из таблицы 7*/
transformedcod_buf_from_URO[625] = 0x11; /*из таблицы 7*/
transformedcod_buf_from_URO[626] = 0x11; /*из таблицы 7*/
transformedcod_buf_from_URO[627] = 0x11; /*из таблицы 7*/
transformedcod_buf_from_URO[628] = 0x00;
transformedcod_buf_from_URO[629] = 0x00;
transformedcod_buf_from_URO[630] = 0x00;
transformedcod_buf_from_URO[631] = 0x00;
transformedcod_buf_from_URO[632] = 0x11; /*из таблицы 7*/
transformedcod_buf_from_URO[633] = 0x11; /*из таблицы 7*/
transformedcod_buf_from_URO[634] = 0x11; /*из таблицы 7*/
transformedcod_buf_from_URO[635] = 0x11; /*из таблицы 7*/
transformedcod_buf_from_URO[636] = 0x00;
transformedcod_buf_from_URO[637] = 0x00;
transformedcod_buf_from_URO[638] = 0x00;
transformedcod_buf_from_URO[639] = 0x00;
transformedcod_buf_from_URO[640] = 0x11; /*из таблицы 7*/
transformedcod_buf_from_URO[641] = 0x11; /*из таблицы 7*/
transformedcod_buf_from_URO[642] = 0x11; /*из таблицы 7*/
transformedcod_buf_from_URO[643] = 0x11; /*из таблицы 7*/
transformedcod_buf_from_URO[644] = 0x00;
transformedcod_buf_from_URO[645] = 0x00;
transformedcod_buf_from_URO[646] = 0x00;
transformedcod_buf_from_URO[647] = 0x00;
transformedcod_buf_from_URO[648] = 0x11; /*из таблицы 7*/
```

```
transformedcod_buf_from_URO[649] = 0x11; /*из таблицы 7*/
transformedcod_buf_from_URO[650] = 0x11; /*из таблицы 7*/
transformedcod_buf_from_URO[651] = 0x11; /*из таблицы 7*/
transformedcod_buf_from_URO[652] = transformedcod_buf_from_UFO[650];
transformedcod_buf_from_URO[653] = transformedcod_buf_from_UFO[651];
transformedcod_buf_from_URO[654] = transformedcod_buf_from_UFO[652];
transformedcod_buf_from_URO[655] = transformedcod_buf_from_UFO[653];
transformedcod_buf_from_URO[656] = transformedcod_buf_from_UFO[654];
transformedcod_buf_from_URO[657] = transformedcod_buf_from_UFO[655];
transformedcod_buf_from_URO[658] = transformedcod_buf_from_UFO[656];
transformedcod_buf_from_URO[659] = transformedcod_buf_from_UFO[657];
transformedcod_buf_from_URO[660] = transformedcod_buf_from_UFO[658];
transformedcod_buf_from_URO[661] = transformedcod_buf_from_UFO[659];
transformedcod_buf_from_URO[662] = transformedcod_buf_from_UFO[660];
transformedcod_buf_from_URO[663] = transformedcod_buf_from_UFO[661];
transformedcod_buf_from_URO[664] = 0x00;
transformedcod_buf_from_URO[665] = 0x00;
transformedcod_buf_from_URO[666] = 0x00;
transformedcod_buf_from_URO[667] = 0x00;
transformedcod_buf_from_URO[668] = 0x01;
transformedcod_buf_from_URO[669] = 0xf4;
transformedcod_buf_from_URO[670] = 0x00;
transformedcod_buf_from_URO[671] = 0x00;
transformedcod_buf_from_URO[672] = 0x00;
transformedcod_buf_from_URO[673] = 0x00;
transformedcod_buf_from_URO[674] = 0x01;
transformedcod_buf_from_URO[675] = 0xf4;
transformedcod_buf_from_URO[676] = transformedcod_buf_from_UFO[662];
```

```
transformedcod_buf_from_URO[677] = transformedcod_buf_from_UFO[663];
transformedcod_buf_from_URO[678] = transformedcod_buf_from_UFO[664];
transformedcod_buf_from_URO[679] = transformedcod_buf_from_UFO[665];
transformedcod_buf_from_URO[680] = transformedcod_buf_from_UFO[666];
transformedcod_buf_from_URO[681] = transformedcod_buf_from_UFO[667];
transformedcod_buf_from_URO[682] = transformedcod_buf_from_UFO[662];
transformedcod_buf_from_URO[683] = transformedcod_buf_from_UFO[663];
transformedcod_buf_from_URO[684] = transformedcod_buf_from_UFO[664];
transformedcod_buf_from_URO[685] = transformedcod_buf_from_UFO[665];
transformedcod_buf_from_URO[686] = transformedcod_buf_from_UFO[666];
transformedcod_buf_from_URO[687] = transformedcod_buf_from_UFO[667];
transformedcod_buf_from_URO[688] = transformedcod_buf_from_UFO[668];
transformedcod_buf_from_URO[689] = transformedcod_buf_from_UFO[669];
transformedcod_buf_from_URO[690] = transformedcod_buf_from_UFO[670];
transformedcod_buf_from_URO[691] = transformedcod_buf_from_UFO[671];
transformedcod_buf_from_URO[692] = transformedcod_buf_from_UFO[672];
transformedcod_buf_from_URO[693] = transformedcod_buf_from_UFO[673];
transformedcod_buf_from_URO[694] = transformedcod_buf_from_UFO[674];
transformedcod_buf_from_URO[695] = transformedcod_buf_from_UFO[675];
transformedcod_buf_from_URO[696] = transformedcod_buf_from_UFO[676];
transformedcod_buf_from_URO[697] = transformedcod_buf_from_UFO[677];
transformedcod_buf_from_URO[698] = transformedcod_buf_from_UFO[678];
transformedcod_buf_from_URO[699] = transformedcod_buf_from_UFO[679];
transformedcod_buf_from_URO[700] = transformedcod_buf_from_UFO[674];
transformedcod_buf_from_URO[701] = transformedcod_buf_from_UFO[675];
transformedcod_buf_from_URO[702] = transformedcod_buf_from_UFO[676];
transformedcod_buf_from_URO[703] = transformedcod_buf_from_UFO[677];
transformedcod_buf_from_URO[704] = transformedcod_buf_from_UFO[678];
```

```
transformedcod_buf_from_URO[705] = transformedcod_buf_from_UFO[679];
transformedcod_buf_from_URO[706] = transformedcod_buf_from_UFO[680];
transformedcod_buf_from_URO[707] = transformedcod_buf_from_UFO[681];
transformedcod_buf_from_URO[708] = transformedcod_buf_from_UFO[682];
transformedcod_buf_from_URO[709] = transformedcod_buf_from_UFO[683];
transformedcod_buf_from_URO[710] = transformedcod_buf_from_UFO[684];
transformedcod_buf_from_URO[711] = transformedcod_buf_from_UFO[685];
transformedcod_buf_from_URO[712] = transformedcod_buf_from_UFO[686];
transformedcod_buf_from_URO[713] = transformedcod_buf_from_UFO[687];
transformedcod_buf_from_URO[714] = transformedcod_buf_from_UFO[688];
transformedcod_buf_from_URO[715] = transformedcod_buf_from_UFO[689];
transformedcod_buf_from_URO[716] = transformedcod_buf_from_UFO[690];
transformedcod_buf_from_URO[717] = transformedcod_buf_from_UFO[691];
transformedcod_buf_from_URO[718] = transformedcod_buf_from_UFO[692];
transformedcod_buf_from_URO[719] = transformedcod_buf_from_UFO[963];
transformedcod_buf_from_URO[720] = transformedcod_buf_from_UFO[964];
transformedcod_buf_from_URO[721] = transformedcod_buf_from_UFO[695];
transformedcod_buf_from_URO[722] = transformedcod_buf_from_UFO[696];
transformedcod_buf_from_URO[723] = transformedcod_buf_from_UFO[697];
transformedcod_buf_from_URO[724] = transformedcod_buf_from_UFO[698];
transformedcod_buf_from_URO[725] = transformedcod_buf_from_UFO[699];
transformedcod_buf_from_URO[726] = transformedcod_buf_from_UFO[700];
transformedcod_buf_from_URO[727] = transformedcod_buf_from_UFO[701];
transformedcod_buf_from_URO[728] = transformedcod_buf_from_UFO[702];
transformedcod_buf_from_URO[729] = transformedcod_buf_from_UFO[703];
transformedcod_buf_from_URO[730] = transformedcod_buf_from_UFO[704];
transformedcod_buf_from_URO[731] = transformedcod_buf_from_UFO[705];
transformedcod_buf_from_URO[732] = transformedcod_buf_from_UFO[706];
```

```
    transformedcod_buf_from_URO[733] = transformedcod_buf_from_UFO[707];
    transformedcod_buf_from_URO[734] = transformedcod_buf_from_UFO[796];
    transformedcod_buf_from_URO[735] = transformedcod_buf_from_UFO[797];

    for(uint16_t i=736; i<1020; i++)
    {
      transformedcod_buf_from_URO[i] = 0x00;
    }

    transformedcod_buf_from_URO[1020] = 0x01; /*контрольная сумма (ADC)*/
    transformedcod_buf_from_URO[1021] = 0x01; /*контрольная сумма (XOR)*/
    /**/
    quint64  rdCnt = serial->read(transformedcod_buf_from_UFO, cnt);
    ui->lblCntRx->setText("Принято " + QString::number(lblCntRx));
    QString strMes;
    for (quint64 i = 0; i < rdCnt; i++) {
        strMes += QString::number((quint8)transformedcod_buf_from_UFO[i], 16) + "
";
    }
    //ui->textEdit->append(strMes);
    ui->textEdit_2->append(strMes);
    lblCntRx++;
    ui->lblCntRx->setText("Принято " + QString::number(lblCntRx));
    ui->label->setStyleSheet("color: rgb(0, 204, 0)");
}

void MainWindow::mouseDoubleClickEvent(QMouseEvent *){
    //cleanErrors();
```

```
}

void MainWindow::on_pbtnSaveText_2_clicked()
{
   QFile log_rxFile("log_rx.txt");
   if(!log_rxFile.open(QIODevice::Append)){
      qDebug() << "Ошибка открытия файла 'log_rx.txt'";
      QMessageBox::critical(this,QString(ProgramTitle),"Ошибка открытия файла 'log_rx.txt'" );
   }

   QTextStream log_rx(&log_rxFile);
   QString text = ui->textEdit_2->toPlainText();
   log_rx<<text<<endl;
   log_rxFile.close();
   ui->textEdit_2->append(time->currentTime().toString() + " -> " + " Лог сохранен в log_rx.txt");
}

void MainWindow::on_pbtnClearLog_2_clicked()
{
   ui->textEdit_2->clear();
   ui->label->setStyleSheet("color: rgb(0, 0, 0)");
}

void MainWindow::on_pbtnDelete_clicked()
{
   ui->lineEdit->clear();
```

```
    ui->label->setStyleSheet("color: rgb(0, 0, 0)");
}

void MainWindow::on_pbtnDelete_2_clicked()
{
    ui->lineEdit_2->clear();
    ui->label->setStyleSheet("color: rgb(0, 0, 0)");
}

void MainWindow::on_pbtnDelete_3_clicked()
{
    ui->lineEdit_3->clear();
    ui->label->setStyleSheet("color: rgb(0, 0, 0)");
}

void MainWindow::on_pbtnDelete_4_clicked()
{
    ui->lineEdit_4->clear();
    ui->label->setStyleSheet("color: rgb(0, 0, 0)");
}
```

Додаток В

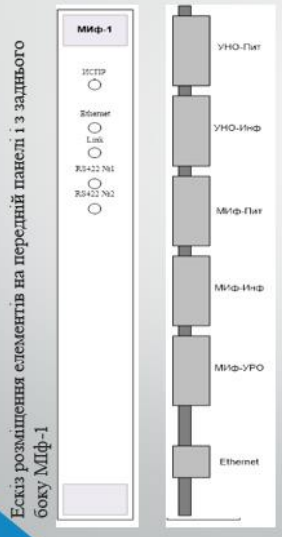
Міністерство освіти і науки, молоді та спорту України
Технологічний інститут СНУ ім. В. Даля (м. Сєвєродонецьк)

Програмні засоби забезпечення взаємодії мікропроцесорного субкомплекса контролю та управління АКНП з інтерфейсним модулем

Студент гр. КІ-14бд **Белов В. В.**

Керівник проекту **Лифар О. К.**

Вимоги до модуля інтерфейсного МІф-1



Ескіз розміщення елементів на передній панелі і з заднього боку МІф-1

Програмне забезпечення модуля интерфейсного МІф-1 призначено для виконання МІф-1 в складі АКНП функцій прийому і перетворення інформації від УФО-1 з подальшим відправленням її в пристрої реєстрації і відображення УРО-1 забезпечуючи підключення УРО-1 до УНО-1 зі складу АКНП.

ПЗ повинно бути розроблено як закрита для програмування користувачем мікропрограма функціонування мікроконтролера STM32F407VGT6. ПЗ має інсталюватися в МК при виготовленні МІф-1.

ПЗ має забезпечувати такі функції:

- початкове тестування і конфігурація;
- прийом від УФО-1 (через порт UART4) даних в форматі ПТО, їх перетворення у формат УРО 1 та видачу у порти USART1 і USART2 (в два УРО-1);
- управління світловою індикацією (світлодіоди H1-H5);
- контроль працездатності (перевірка цілісності програмного коду).

Вимоги до початкової ініціалізації

Після включення живлення і формування скидання МП ПО має виконати початкове конфігурування МП. Початкове конфігурування МП має включати ініціалізацію контролера вкладених переривань, схем тактування, таймера, таймера Whatchdog, UART, портів введення / виводу.

При ініціалізації таймера Watchdog необхідно задати тайм-аут спрацьовування таймера 0,5 с.

Швидкість перемикання станів GPIO повинна бути мінімальною.

Ініціалізацію UART4, USART1 і USART2 виконати наступними параметрами:

- швидкість - 57600 б / с;
- стоп-біт - 1;
- кількість біт даних - 9 (включаючи біт паритету);
- паритет - контроль на непарність;
- low control - відключений.
- Ініціалізацію таймера виконати наступними параметрами:
- номер таймера - 3 (TIM3);
- значення предподільника - 72;
- період - 500 мкс;

Загальні вимоги до формату повідомлень

Обмін інформацією між пристроями повинен проводитися з використанням правил процедури BSC фірми IBM (знак-орієнтована процедура загальноцільового призначення для управління обміном даними між користувачами) в частині використання наступних спеціальних символів:

- DLE = 10h (Data Link Escape, ознака наступного за ним керуючого символу);
- STX = 02h (Start of Text, ознака початку кадру);
- ETX = 03h (End of Text, ознака кінця кадру). Таким чином, для визначення початку повідомлення повинно застосовуватися поєднання керуючих символів DLE та STX, для визначення кінця повідомлення – DLE та ETX.

Для надання можливості передавати дані, що збігаються за кодуванням з керуючим символом DLE, повинна застосовуватися операція "байт-стаффінга". Вона полягає в тому, що в разі, якщо код ідентифікатора або інформаційної частини кадру збігається з кодом символу DLE (10h), то він передається двічі (дублюється). При прийомі інформації необхідно виконувати фільтрацію, тобто перевірку на наявність символу DLE (10h) в ідентифікатор та інформаційної частини кадру і видалення вставлених при передачі символів DLE (10h).

Загальні вимоги до формату повідомлень

Загальний формат повідомлень

Номер байта	Опис	Примітка
0	DLE (10h)	Початок повідомлення
1	STX(02h)	
2…n+1	Дані повідомлення (Включаючи MasterByte, StatusByte, BCC1, BCC2)	n – Кількість байтів повідомлення
n+2	DLE (10h)	Кінець повідомлення
n+3	ETX (03h)	

QT Creator, можливості та функції

Одним з найголовніших досягнень Qt Creator є те, що він дозволяє команді розробників працювати над проектом на різних платформах з використанням загальних інструментів для розробки і налагодження.

Створення проекту дозволить:

- Групувати файли разом
- Додати власні кроки збірки
- Включити форми та файли ресурсів
- Вказати налаштування для додатків які запускаються

Ви можете або створити проект з нуля, або імпортувати існуючий проект. Qt Creator генерує всі необхідні файли в залежності від типу створюваного проекту.

Огляд існуючих графічних інтерфейсів

Бібліотека класів MFC

Бібліотека класів MFC (Microsoft Foundation Classes) розроблена фірмою Microsoft.

Графічний інтерфейс цієї бібліотеки дозволяє в повній мірі використовувати всі можливості операційних систем сімейства Windows.

Додатки графічного інтерфейсу, як правило, містять декілька вікон, широко використовують піктограми, різні види курсорів, смуги прокрутки, меню і панелі інструментів. Все це призначено для створення «дружнього користувачеві» інтерфейсу програми.

Однак бібліотека класів MFC дозволяє розробляти програми лише в середовищі Windows і що важливо мають високу вартість.

Огляд існуючих графічних інтерфейсів

Бібліотека класів Qt

Наступним найбільш поширеним видом візуалізації є бібліотека Qt. Qt - це бібліотека класів C++ і набір інструментального програмного забезпечення, призначених для побудови багатоплатформених додатків з графічним інтерфейсом і сповідують принцип "написавши одного разу - компілює в будь-якому місці", яка дозволяє запускати написане з її допомогою ПЗ в більшості сучасних операційних систем шляхом простої компіляції програми для кожної ОС без зміни вихідного коду. Включає в себе всі основні класи, які можуть знадобитися при розробці прикладного програмного забезпечення, починаючи від елементів графічного інтерфейсу і закінчуючи класами для роботи з мережею, базами даних і XML. Qt є повністю об'єктно-орієнтованим, легко розширюваним і підтримує техніку компонентного програмування.

Інструкція для роботи с програмою ByteTest

Запуск програми виконується подвійним кліком ЛКМ по ярлику ByteTest.exe

При запуску ByteTest на екрані з'явиться головне вікно, показане на рисунку

Для початку роботи програми необхідно вибрати із випадаючого списку COM-порт по якому буде відбуватися обмін інформацією. Наступним кроком потрібно натиснути на кнопку «Відкрити порт». Якщо зміняти значення «Status Byte, Архівне значення реактивності, ADC, XOR» не потрібно, натискаємо на кнопку «Відправити». Програма починає одиночну відправку інформації у МІф-1. Якщо потрібно відправляти інформацію по таймеру, як вказано у ТЗ, зупиняємо программу, натискаючи на кнопку «Зупинини». Після натискаємо на кнопку «Запустити тест», наступне натисканя на цю кнопку зупинить передачю пакетів даних. При бажанні деякі значення можна змінити, наприклад для перевірки контрольної суми. Для цого вводимо за допомогю клавіатури значення у текстові поля. При натисканні на кнопку «Відправити», або «Запустити тест» программа замінить значення у коді.

У текстові поля «Лог» будуть записуватися прийнята та відправлена інформація. Цю інформацію можна зберігти за допомогю кнопки «Зберігти лог». Файл зберігається у кореневій папці з назвою log.tx або log.rx Інформація зберігається у форматі .txt У програмі є можливість очистити поля «Лог» за допомогою кнопки «Очистити лог». У невидемі поля які знаходяться нижче полів «Лог» буде виводитися кількість прийнятих та відправлених пакетів.

Для зупинки програми та виходу із неї потрібно натиснути на кнопку «Зупинити» або «Зупинити тест» та натиснути на хрестик у верхньому правому кутку програми.