МИНИСТЕРСТВО НАУКИ
И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

НГТУ
НЭТИ | Факультет прикладной
математики и информатики

Кафедра теоретической и прикладной информатики

Лабораторная работа № 1

по дисциплине «Проектирование систем реального времени»

РАБОТА С ПОСЛЕДОВАТЕЛЬНЫМ ПОРТОМ ПЕРСОНАЛЬНОГО КОМПЬЮТЕРА

Бригада 3 БОРТНИКОВ НИКИТА

Группа ПМИМ-21 ЕФИМЕНКО АЛЕКСАНДРА

КУТУЗОВА ИРИНА

Преподаватель КОБЫЛЯНСКИЙ ВАЛЕРИЙ ГРИГОРЬЕВИЧ

Новосибирск, 2023

1. Цель работы

Целью работы является изучение принципов работы и методов программирования COM портов и изучение различных алгоритмов вычисления контрольной суммы при передаче данных через COM порт

2. Ход работы

1. Написать и отладить программу, определяющую базовые адреса последовательных портов.
2. Написать программу для передачи и приема данных через COM-порт. Основные требования:
    - возможность конфигурирования порта (скорость передачи, контроль четности, паритет четности);
    - возможность передачи пакета по строковому (ASCII или UTF-8) и по бинарному протоколам в зависимости от выбора пользователя, тип протокола включить в передаваемый пакет данных (например, 0 – строковый, 1 – бинарный);
    - возможность передачи символов кириллицы;
    - возможность вывода переданных и полученных данных, а также количества переданных и принятых байтов;
3. Соединить COM порты компьютера нуль – модемным кабелем. Если в компьютерном классе установлены компьютеры с одним COM портом, то соединить два соседних компьютера. Если COM-порты отсутствуют, то используйте виртуальные порты.
4. Запустить 2 копии программы, одну подключить к порту COM1, вторую – к COM2. Провести обмен данными.
5. В окне настройки порта COM1 включить контроль четности в состояние «четное», провести обмен данными, пояснить результаты.
6. В окне настройки порта COM1 отключить контроль четности и включить скорость передачи данных 9600 бит/с, провести обмен данными и пояснить результаты.
7. Написать и отладить функции вычисления простой контрольной суммы, LRC, CRC16 и CRC32. Добавить эти функции к программе, разработанной в п. 2.2, вычисляя контрольные суммы при передаче и при приеме данных, включить контрольную сумму в передаваемый пакет данных.
8. Провести тестирование функций для передачи символьной строки согласно Вашему варианту задания.

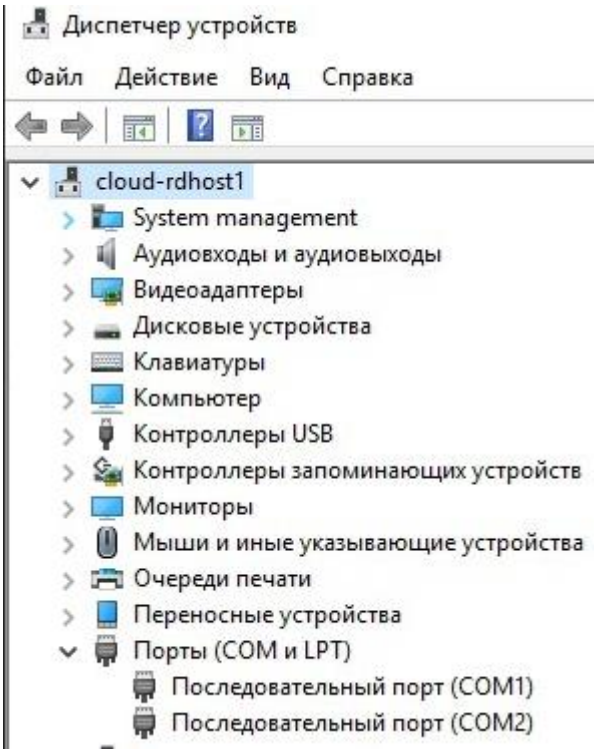
3	Редактор

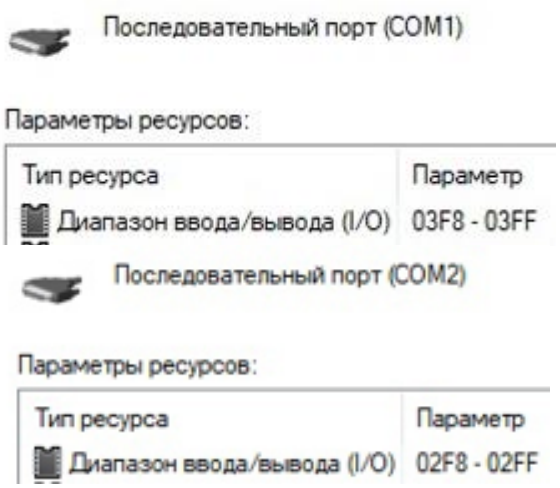
9. Изменить один символ в посылке и найти контрольные суммы, результаты занести в протокол.
10. Поменять местами два любых символа в посылке и найти контрольные суммы, результаты занести в протокол.
11. Передать любое вещественное число типа float, содержащее 5 знаков после запятой, с помощью стокового и бинарного протоколов, результаты занести в протокол.

3. Ход работы

1) Написать и отладить программу, определяющую базовые адреса последовательных портов

Для просмотра существующих последовательных портов COM перейдем в диспетчер устройств. Как видно из рисунков ниже на данном компьютере есть два COM порта.





Вывод программы:

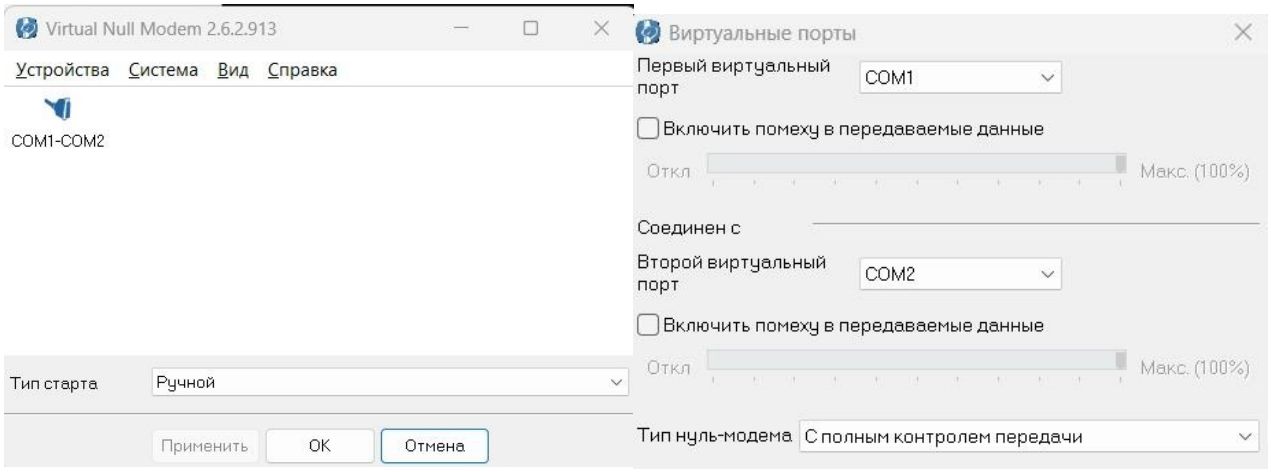
```
Были найдены следующие последовательные порты:
        Name = COM2     Address = 0x000002F8
        Name = COM1     Address = 0x000003F8
```

2) Написать программу для передачи и приема данных через COM-порт. Основные требования:

- возможность конфигурирования порта (скорость передачи, контроль четности, паритет четности);
- возможность передачи пакета по строковому (ASCII или UTF-8) и по бинарному протоколам в зависимости от выбора пользователя, тип протокола включить в передаваемый пакет данных (например, 0 – строковый, 1 – бинарный);
- возможность передачи символов кириллицы;

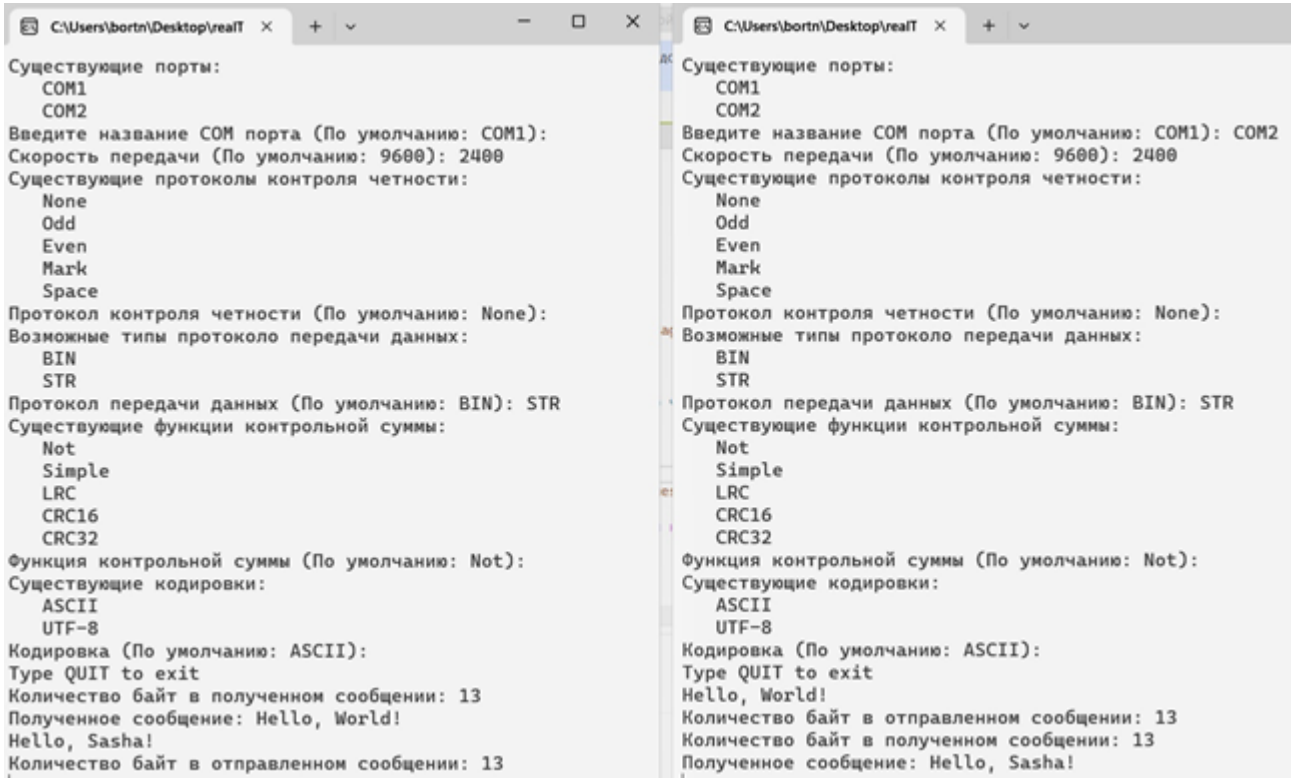
- возможность вывода переданных и полученных данных, а также количества переданных и принятых байтов;

3) Соединить COM порты компьютера нуль – модемным кабелем. Если в компьютерном классе установлены компьютеры с одним COM портом, то соединить два соседних компьютера. Если COM-порты отсутствуют, то используйте виртуальные порты.



4) Тестирование обмена данными

Запустить 2 копии программы, одну подключить к порту COM1, вторую – к COM2. Провести обмен данными.
На скорости 2400 бит/с, без контроля четности.



В окне настройки порта COM1 включить контроль четности в состояние «четное», провести обмен данными, пояснить результаты.
С контролем четности (even - четное) на одном приложении с портом COM1.

```
Существующие порты:
   COM1
   COM2
Введите название COM порта (По умолчанию: COM1): COM1
Скорость передачи (По умолчанию: 9600): 2400
Существующие протоколы контроля четности:
   None
   Odd
   Even
   Mark
   Space
Протокол контроля четности (По умолчанию: None):Even
Возможные типы протоколо передачи данных:
   BIN
   STR
Протокол передачи данных (По умолчанию: BIN): STR
Существующие функции контрольной суммы:
   Not
   Simple
   LRC
   CRC16
   CRC32
Функция контрольной суммы (По умолчанию: Not):
Существующие кодировки:
   ASCII
   UTF-8
Кодировка (По умолчанию: ASCII):
Type QUIT to exit
Hello, Sasha!
Количество байт в отправленном сообщении: 13
```

```
Существующие порты:
   COM1
   COM2
Введите название COM порта (По умолчанию: COM1): COM2
Скорость передачи (По умолчанию: 9600): 2400
Существующие протоколы контроля четности:
   None
   Odd
   Even
   Mark
   Space
Протокол контроля четности (По умолчанию: None):
Возможные типы протоколо передачи данных:
   BIN
   STR
Протокол передачи данных (По умолчанию: BIN): STR
Существующие функции контрольной суммы:
   Not
   Simple
   LRC
   CRC16
   CRC32
Функция контрольной суммы (По умолчанию: Not):
Существующие кодировки:
   ASCII
   UTF-8
Кодировка (По умолчанию: ASCII):
Type QUIT to exit
Hello, World!
Количество байт в отправленном сообщении: 13
```

Вывод: Передача данных не произошла, т.к. контроль четности на обоих COM портах не совпадает.

С контролем четности на двух приложениях.

```
Существующие порты:
   COM1
   COM2
Введите название COM порта (По умолчанию: COM1):
Скорость передачи (По умолчанию: 9600): 2400
Существующие протоколы контроля четности:
   None
   Odd
   Even
   Mark
   Space
Протокол контроля четности (По умолчанию: None):Even
Возможные типы протоколо передачи данных:
   BIN
   STR
Протокол передачи данных (По умолчанию: BIN): STR
Существующие функции контрольной суммы:
   Not
   Simple
   LRC
   CRC16
   CRC32
Функция контрольной суммы (По умолчанию: Not):
Существующие кодировки:
   ASCII
   UTF-8
Кодировка (По умолчанию: ASCII):
Type QUIT to exit
Количество байт в полученном сообщении: 13
Полученное сообщение: Hello, World!
Hello, Nikita!
Количество байт в отправленном сообщении: 14
```

```
Существующие порты:
   COM1
   COM2
Введите название COM порта (По умолчанию: COM1): COM2
Скорость передачи (По умолчанию: 9600): 2400
Существующие протоколы контроля четности:
   None
   Odd
   Even
   Mark
   Space
Протокол контроля четности (По умолчанию: None):Even
Возможные типы протоколо передачи данных:
   BIN
   STR
Протокол передачи данных (По умолчанию: BIN): STR
Существующие функции контрольной суммы:
   Not
   Simple
   LRC
   CRC16
   CRC32
Функция контрольной суммы (По умолчанию: Not):
Существующие кодировки:
   ASCII
   UTF-8
Кодировка (По умолчанию: ASCII):
Type QUIT to exit
Hello, World!
Количество байт в отправленном сообщении: 13
Количество байт в полученном сообщении: 14
Полученное сообщение: Hello, Nikita!
```

В окне настройки порта COM1 отключить контроль четности и включить скорость передачи данных 9600 бит/с, провести обмен данными и пояснить результаты.
С разной скоростью. На приложении с COM1 установлена скорость 9600 бит/с, а на COM2 2400 бит/с.

```
Существующие порты:
   COM1
   COM2
Введите название COM порта (По умолчанию: COM1):
Скорость передачи (По умолчанию: 9600):
Существующие протоколы контроля четности:
   None
   Odd
   Even
   Mark
   Space
Протокол контроля четности (По умолчанию: None):
Возможные типы протоколо передачи данных:
   BIN
   STR
Протокол передачи данных (По умолчанию: BIN): STR
Существующие функции контрольной суммы:
   Not
   Simple
   LRC
   CRC16
   CRC32
Функция контрольной суммы (По умолчанию: Not):
Существующие кодировки:
   ASCII
   UTF-8
Кодировка (По умолчанию: ASCII):
Type QUIT to exit
Hello, Ira!
Количество байт в отправленном сообщении: 11
```

```
Существующие порты:
   COM1
   COM2
Введите название COM порта (По умолчанию: COM1): COM2
Скорость передачи (По умолчанию: 9600): 2400
Существующие протоколы контроля четности:
   None
   Odd
   Even
   Mark
   Space
Протокол контроля четности (По умолчанию: None):None
Возможные типы протоколо передачи данных:
   BIN
   STR
Протокол передачи данных (По умолчанию: BIN): STR
Существующие функции контрольной суммы:
   Not
   Simple
   LRC
   CRC16
   CRC32
Функция контрольной суммы (По умолчанию: Not):
Существующие кодировки:
   ASCII
   UTF-8
Кодировка (По умолчанию: ASCII):
Type QUIT to exit
Hello, World!
Количество байт в отправленном сообщении: 13
```

С одинаковой скоростью (9600 бит/с)

```
Существующие порты:
   COM1
   COM2
Введите название COM порта (По умолчанию: COM1):
Скорость передачи (По умолчанию: 9600):
Существующие протоколы контроля четности:
   None
   Odd
   Even
   Mark
   Space
Протокол контроля четности (По умолчанию: None):
Возможные типы протоколо передачи данных:
   BIN
   STR
Протокол передачи данных (По умолчанию: BIN): STR
Существующие функции контрольной суммы:
   Not
   Simple
   LRC
   CRC16
   CRC32
Функция контрольной суммы (По умолчанию: Not):
Существующие кодировки:
   ASCII
   UTF-8
Кодировка (По умолчанию: ASCII):
Type QUIT to exit
Количество байт в полученном сообщении: 13
Полученное сообщение: Hello, World!
Hello, Ira!
Количество байт в отправленном сообщении: 11
```

```
Существующие порты:
   COM1
   COM2
Введите название COM порта (По умолчанию: COM1): COM2
Скорость передачи (По умолчанию: 9600):
Существующие протоколы контроля четности:
   None
   Odd
   Even
   Mark
   Space
Протокол контроля четности (По умолчанию: None):
Возможные типы протоколо передачи данных:
   BIN
   STR
Протокол передачи данных (По умолчанию: BIN): STR
Существующие функции контрольной суммы:
   Not
   Simple
   LRC
   CRC16
   CRC32
Функция контрольной суммы (По умолчанию: Not):
Существующие кодировки:
   ASCII
   UTF-8
Кодировка (По умолчанию: ASCII):
Type QUIT to exit
Hello, World!
Количество байт в отправленном сообщении: 13
Количество байт в полученном сообщении: 11
Полученное сообщение: Hello, Ira!
```

Вывод: Программа-приемник не смогла принять сообщение из-за разной скорости передачи данных. При одинаковой скорости всё работает корректно.

Написать и отладить функции вычисления простой контрольной суммы, LRC, CRC16 и CRC32. Добавить эти функции к программе, разработанной в п. 4.2, вычисляя контрольные суммы при передаче и при приеме данных, включить контрольную сумму в передаваемый пакет данных. Провести тестирование функций для передачи символьной строки.

С **простой** контрольной суммой

```
Протокол передачи данных (По умолчанию: BIN): STR
Существующие функции контрольной суммы:
   Not
   Simple
   LRC
   CRC16
   CRC32
Функция контрольной суммы (По умолчанию: Not): Simple
Существующие кодировки:
   ASCII
   UTF-8
Кодировка (По умолчанию: ASCII): UTF-8
Type QUIT to exit
Контрольная сумма посылки: 0x17
Количество байт в полученном сообщении: 16
Полученное сообщение: Редактор
Редактор
Количество байт в отправленном сообщении: 16
Контрольная сумма посылки: 0x17
```

```
Протокол передачи данных (По умолчанию: BIN): STR
Существующие функции контрольной суммы:
   Not
   Simple
   LRC
   CRC16
   CRC32
Функция контрольной суммы (По умолчанию: Not): Simple
Существующие кодировки:
   ASCII
   UTF-8
Кодировка (По умолчанию: ASCII): UTF-8
Type QUIT to exit
Редактор
Количество байт в отправленном сообщении: 16
Контрольная сумма посылки: 0x17
Контрольная сумма посылки: 0x17
Количество байт в полученном сообщении: 16
Полученное сообщение: Редактор
```

Изменение одного символа:

```
Редуктор
Количество байт в отправленном сообщении: 16
Контрольная сумма посылки: 0x25
Контрольная сумма посылки: 0x25
Количество байт в полученном сообщении: 16
Полученное сообщение: Редуктор
```

```
Контрольная сумма посылки: 0x25
Количество байт в полученном сообщении: 16
Полученное сообщение: Редуктор
Редуктор
Количество байт в отправленном сообщении: 16
Контрольная сумма посылки: 0x25
```

Изменение двух символов в посылке:

```
Контрольная сумма посылки: 0x17
Количество байт в полученном сообщении: 16
Полученное сообщение: Редоктар
Редоктар
Количество байт в отправленном сообщении: 16
Контрольная сумма посылки: 0x17
```

```
Редоктар
Количество байт в отправленном сообщении: 16
Контрольная сумма посылки: 0x17
Контрольная сумма посылки: 0x17
Количество байт в полученном сообщении: 16
Полученное сообщение: Редоктар
```

С контрольной суммой **LRC**

```
Редактор
Количество байт в отправленном сообщении: 16
Контрольная сумма посылки: 0x40
Контрольная сумма посылки: 0x40
Количество байт в полученном сообщении: 16
Полученное сообщение: Редактор
```

```
Контрольная сумма посылки: 0x40
Количество байт в полученном сообщении: 16
Полученное сообщение: Редактор
Редактор
Количество байт в отправленном сообщении: 16
Контрольная сумма посылки: 0x40
```

```
Редуктор
Количество байт в отправленном сообщении: 16
Контрольная сумма посылки: 0x6C
Контрольная сумма посылки: 0x6C
Количество байт в полученном сообщении: 16
Полученное сообщение: Редуктор
```

```
Контрольная сумма посылки: 0x6C
Количество байт в полученном сообщении: 16
Полученное сообщение: Редуктор
Редуктор
Количество байт в отправленном сообщении: 16
Контрольная сумма посылки: 0x6C
```

```
Контрольная сумма посылки: 0x40
Количество байт в полученном сообщении: 16
Полученное сообщение: Редоктар
Редоктар
Количество байт в отправленном сообщении: 16
Контрольная сумма посылки: 0x40
```

```
Редоктар
Количество байт в отправленном сообщении: 16
Контрольная сумма посылки: 0x40
Контрольная сумма посылки: 0x40
Количество байт в полученном сообщении: 16
Полученное сообщение: Редоктар
```

С контрольной суммой **CRC16**

```
Редактор
Количество байт в отправленном сообщении: 16
Контрольная сумма посылки: 0x39-7E
Контрольная сумма посылки: 0x39-7E
Количество байт в полученном сообщении: 16
Полученное сообщение: Редактор
Контрольная сумма посылки: 0x3A-C8
Количество байт в полученном сообщении: 16
Полученное сообщение: Редуктор
Редуктор
Количество байт в отправленном сообщении: 16
Контрольная сумма посылки: 0x3A-C8
Редоктар
Количество байт в отправленном сообщении: 16
Контрольная сумма посылки: 0x4F-16
Контрольная сумма посылки: 0x4F-16
Количество байт в полученном сообщении: 16
Полученное сообщение: Редоктар
```

```
Контрольная сумма посылки: 0x39-7E
Количество байт в полученном сообщении: 16
Полученное сообщение: Редактор
Редактор
Количество байт в отправленном сообщении: 16
Контрольная сумма посылки: 0x39-7E
Редуктор
Количество байт в отправленном сообщении: 16
Контрольная сумма посылки: 0x3A-C8
Контрольная сумма посылки: 0x3A-C8
Количество байт в полученном сообщении: 16
Полученное сообщение: Редуктор
Контрольная сумма посылки: 0x4F-16
Количество байт в полученном сообщении: 16
Полученное сообщение: Редоктар
Редоктар
Количество байт в отправленном сообщении: 16
Контрольная сумма посылки: 0x4F-16
```

С контрольной суммой **CRC32**

```
Контрольная сумма посылки: 0xFA-DF-D1-08
Количество байт в полученном сообщении: 16
Полученное сообщение: Редактор
Редактор
Количество байт в отправленном сообщении: 16
Контрольная сумма посылки: 0xFA-DF-D1-08
Редуктор
Количество байт в отправленном сообщении: 16
Контрольная сумма посылки: 0xD3-48-AF-F0
Контрольная сумма посылки: 0xD3-48-AF-F0
Количество байт в полученном сообщении: 16
Полученное сообщение: Редуктор
Контрольная сумма посылки: 0x62-29-8D-CA
Количество байт в полученном сообщении: 16
Полученное сообщение: Редоктар
Редоктар
Количество байт в отправленном сообщении: 16
Контрольная сумма посылки: 0x62-29-8D-CA
```

```
Редактор
Количество байт в отправленном сообщении: 16
Контрольная сумма посылки: 0xFA-DF-D1-08
Контрольная сумма посылки: 0xFA-DF-D1-08
Количество байт в полученном сообщении: 16
Полученное сообщение: Редактор
Контрольная сумма посылки: 0xD3-48-AF-F0
Количество байт в полученном сообщении: 16
Полученное сообщение: Редуктор
Редуктор
Количество байт в отправленном сообщении: 16
Контрольная сумма посылки: 0xD3-48-AF-F0
Редоктар
Количество байт в отправленном сообщении: 16
Контрольная сумма посылки: 0x62-29-8D-CA
Контрольная сумма посылки: 0x62-29-8D-CA
Количество байт в полученном сообщении: 16
Полученное сообщение: Редоктар
```

Контрольная сумма	Значение «Редактор»	Значение «Редуктор»	Значение «Редоктар»
Простая	0x17	0x25	0x17
LRC	0x40	0x6C	0x40
CRC16	0x39-7E	0x3A-C8	0x4F-16
CRC32	0xFA-DF-D1-08	0xD3-48-AF-F0	0x62-29-8D-CA

Вывод: Алгоритмы простой контрольной суммы и LRC смогли выявить замену символа в сообщении, но не смогли выявить ошибку при перемешивании символов. Алгоритмы CRC16 и CRC32 показали разную контрольную сумму во всех трех случаях, поэтому можно считать их более устойчивыми к выявлению помех, чем LRC и простой алгоритм.

Передать любое вещественное число типа float, содержащее 5 знаков после запятой, с помощью стокового и бинарного протоколов, результаты занести в протокол.
Отправка числа типа Double (8 байт). Бинарный формат:

```
Возможные типы протоколо передачи данных:
   BIN
   STR
Протокол передачи данных (По умолчанию: BIN):
Существующие функции контрольной суммы:
   Not
   Simple
   LRC
   CRC16
   CRC32
Функция контрольной суммы (По умолчанию: Not):
Существующие кодировки:
   ASCII
   UTF-8
Кодировка (По умолчанию: ASCII):
Type QUIT to exit
Количество байт в полученном сообщении: 8
Полученное сообщение: 3,14151
3,14151
Количество байт в отправленном сообщении: 8
```

```
Возможные типы протоколо передачи данных:
   BIN
   STR
Протокол передачи данных (По умолчанию: BIN):
Существующие функции контрольной суммы:
   Not
   Simple
   LRC
   CRC16
   CRC32
Функция контрольной суммы (По умолчанию: Not):
Существующие кодировки:
   ASCII
   UTF-8
Кодировка (По умолчанию: ASCII):
Type QUIT to exit
3,14151
Количество байт в отправленном сообщении: 8
Количество байт в полученном сообщении: 8
Полученное сообщение: 3,14151
```

Строковый формат:

```
Возможные типы протоколо передачи данных:
   BIN
   STR
Протокол передачи данных (По умолчанию: BIN): STR
Существующие функции контрольной суммы:
   Not
   Simple
   LRC
   CRC16
   CRC32
Функция контрольной суммы (По умолчанию: Not):
Существующие кодировки:
   ASCII
   UTF-8
Кодировка (По умолчанию: ASCII):
Type QUIT to exit
Количество байт в полученном сообщении: 7
Полученное сообщение: 3,14151
3,14151
Количество байт в отправленном сообщении: 7
```

```
Возможные типы протоколо передачи данных:
   BIN
   STR
Протокол передачи данных (По умолчанию: BIN): STR
Существующие функции контрольной суммы:
   Not
   Simple
   LRC
   CRC16
   CRC32
Функция контрольной суммы (По умолчанию: Not):
Существующие кодировки:
   ASCII
   UTF-8
Кодировка (По умолчанию: ASCII):
Type QUIT to exit
3,14151
Количество байт в отправленном сообщении: 7
Количество байт в полученном сообщении: 7
Полученное сообщение: 3,14151
```

Вывод: При отправки числа через бинарный протокол сообщение всегда будет занимать 8 байт, а количество байт при отправке числа через символьный протокол зависит от количества символов.

4. Вывод

При выполнении лабораторной работы были изучены принципы работы и методы программирования COM портов, а также методы обнаружения помех при передачи сообщений.

5. Текст программы

1) Программа для обнаружения COM портов и их базовых адресов.

```
using System;
using System.IO.Ports;
using System.Management;
using System.Text;
using System.Threading;
using System.Xml.Linq;

namespace Program
{
    class Program
    {

        public static void Main()
        {
            // Получение списка имен последовательных портов.
            string[] ports = SerialPort.GetPortNames();

            Console.WriteLine("Были найдены следующие последовательные порты:");

            // Вывод каждого имени порта на консоль.
            foreach (string port in ports)
            {
                Console.WriteLine(GetComPortInformation(port));
            }

            Console.ReadLine();
        }

        // Возвращает информацию о последовательном порте с указанным именем
        public static string GetComPortInformation(string name)
        {
            ManagementObjectSearcher mbs;
            StringBuilder sb = new StringBuilder(2000);
            sb.Append("\tName = " + name);

            // Попытка получить данные из Win32_PortResource,
            // Которые соответствуют строке из Win32_SerialPort с именем девайса =
имени порта (#COM1)
            mbs = new ManagementObjectSearcher("ASSOCIATORS OF {Win32_SerialPort.De-
viceID='" + name +
                                       "'} WHERE RESULTCLASS  = Win32_PortResource");

            ManagementObjectCollection mbsResult = mbs.Get();

            if (mbsResult.Count == 0)
                return "Информация не найдена";

            // Структура Win32_PortResource:
            // Name - строка с диапазоном адресов, использующихся COM портом (#
"0x000002F8-0x000002FF")
            // Также начальный и конечный адрес можно получить по свойствам
StartingAddress и EndingAddress в десятичном виде
            foreach (ManagementObject mo in mbsResult)
            {
                var addr = mo["Name"].ToString().Split('-')[0];
                sb.Append("\tAddress = " + addr);
```

```
            }

            return sb.ToString();
        }
    }


}
```

2) Программа для передачи данных по COM портам.

```
using System;
using System.Collections.Generic;
using System.IO.Ports;
using System.Linq;
using System.Text;

public class PortChat
{
    static bool _continue;
    static SerialPort _serialPort;

    // typeProtocol: 0 - bin, 1 - string
    static bool typeProtocol = false;
    static string encoding = "ASCII";
    static CheckSumAlg checkSumAlg = CheckSumAlg.Not;

    public enum CheckSumAlg
    {
        Not = 0,
        Simple = 1,
        LRC = 2,
        CRC16 = 3,
        CRC32 = 4
    }


    public static void Main()
    {

        // Создание объекта последовательного порта с настройками по умолчанию
        _serialPort = new SerialPort();

        // PortName - наименование последовательного порта
        SetPortName(ref _serialPort);

        // BaudRate - Возвращает или задает скорость передачи для последовательного
порта (бит в секунду).
        SetPortBaudRate(ref _serialPort);

        // Parity - протокол контроля четности
        // Отвечается за контроль чертности и паритет четности
        // Контроль четности: Even(четное) и Odd(нечетное)
        // Паритет четности: Mark(бит четности = 1) и Space(бит четности = 0)
        SetPortParity(ref _serialPort);

        // Устанавливаем таймаут для чтения и записи
        _serialPort.ReadTimeout = 50000;
        _serialPort.WriteTimeout = 50000;
```

```
        SetTypeProtocol();
        SetCheckSumAlg();
        SetEncoding();

        // Подписка на обработчик получения сообщений
        _serialPort.DataReceived += GetMessage;
        // Открывает соединение последовательного порта
        _serialPort.Open();
        _continue = true;


        string message;
        StringComparer stringComparer = StringComparer.OrdinalIgnoreCase;
        Console.WriteLine("Type QUIT to exit");
        while (_continue)
        {
            message = Console.ReadLine();

            if (stringComparer.Equals("QUIT", message))
            {
                _continue = false;
            }
            else
            {
                SendMessage(message);
            }
        }

        _serialPort.Close();
    }

    // Отображает существующие порты и задает выбранный порт
    public static void SetPortName(ref SerialPort _serialPort)
    {
        string newPortName;

        Console.WriteLine("Существующие порты:");
        foreach (string s in SerialPort.GetPortNames())
        {
            Console.WriteLine("   {0}", s);
        }

        Console.Write("Введите название COM порта (По умолчанию: {0}): ", _serial-
Port.PortName);
        newPortName = Console.ReadLine();

        if (!(newPortName == "") && newPortName.ToLower().StartsWith("com"))
        {
            _serialPort.PortName = newPortName;
        }
    }



    // Задает скорость передачи для последовательного порта.
    public static void SetPortBaudRate(ref SerialPort _serialPort)
    {
        string newBaudRate;

        Console.Write("Скорость передачи (По умолчанию: {0}): ", _serialPort.BaudRate);
        newBaudRate = Console.ReadLine();
```

```
        if (newBaudRate != "")
        {
            _serialPort.BaudRate = int.Parse(newBaudRate);
        }
    }



    // Отображает существующие протоколы контроля четности и задает выбранный протокол
    public static void SetPortParity(ref SerialPort _serialPort)
    {
        string newParity;

        Console.WriteLine("Существующие протоколы контроля четности:");
        foreach (string s in Enum.GetNames(typeof(Parity)))
        {
            Console.WriteLine("   {0}", s);
        }

        Console.Write("Протокол контроля четности (По умолчанию: {0}):", _serial-
Port.Parity.ToString());
        newParity = Console.ReadLine();

        if (newParity != "")
        {
            _serialPort.Parity = (Parity)Enum.Parse(typeof(Parity), newParity, true);
        }
    }

    public static void SetTypeProtocol()
    {
        Console.WriteLine("Возможные типы протоколо передачи данных:");
        Console.WriteLine("   BIN");
        Console.WriteLine("   STR");
        Console.Write("Протокол передачи данных (По умолчанию: BIN): ");
        string inputType = Console.ReadLine();
        StringComparer stringComparer = StringComparer.OrdinalIgnoreCase;

        if (inputType != "")
        {
            typeProtocol = !stringComparer.Equals("BIN", inputType);
        }
    }

    public static void SetCheckSumAlg()
    {
        Console.WriteLine("Существующие функции контрольной суммы:");
        foreach (string s in Enum.GetNames(typeof(CheckSumAlg)))
        {
            Console.WriteLine("   {0}", s);
        }

        Console.Write("Функция контрольной суммы (По умолчанию: Not): ");
        string inputCheckSumAlg = Console.ReadLine();

        if (inputCheckSumAlg != "")
        {
            checkSumAlg = (CheckSumAlg)Enum.Parse(typeof(CheckSumAlg), inputCheck-
SumAlg, true);
        }
    }
```

```
    public static void SetEncoding()
    {
        Console.WriteLine("Существующие кодировки:");
        Console.WriteLine("   ASCII");
        Console.WriteLine("   UTF-8");

        Console.Write("Кодировка (По умолчанию: ASCII): ");
        string inputEncoding = Console.ReadLine();
        StringComparer stringComparer = StringComparer.OrdinalIgnoreCase;

        if (inputEncoding != "")
        {
            if (stringComparer.Equals("UTF-8", inputEncoding))
            {
                _serialPort.Encoding = Encoding.UTF8;
            }
        }
    }

    public static byte CalcSimpleCheckSum(byte[] messageBytes)
    {
        byte checkSum = 0;
        foreach (byte b in messageBytes)
        {
            checkSum ^= b;
        }
        return checkSum;
    }

    public static byte CalcLongRedCheck(byte[] messageBytes)
    {
        byte checkSum = 0;

        for (int i = 0; i < messageBytes.Length; i++)
        {
            checkSum = (byte)((checkSum + messageBytes[i]) % 255);
        }

        checkSum = (byte)(255 - checkSum);
        checkSum = (byte)((checkSum + 1) % 255);

        return checkSum;

    }


    public static ushort CalcCRC16Ira(byte[] messageBytes)
    {
        // Задаем контрольную сумму из 2 байт
        ushort checkSum = 0;

        // Начинаем обрабатывать сообщение побайтно
        for (int i = 0; i < messageBytes.Length; i++)
        {
            checkSum ^= (ushort)(messageBytes[i] << 8); // сдвиг вправо

            for (int j = 0; j < 8; j++)
            {
                if ((ushort)(checkSum & (ushort)0x8000u) != 0) // если младший бит не
равен 0, 32768 = 1000 0000 0000 0000
                    checkSum = (ushort)((ushort)(checkSum << 1) ^ (ushort)0x1021u); //
исключающее или регистра, 4129 = 0001 0000 0010 0001
```

```
                else // если нет
                    checkSum <<= 1; // то сдвиг
            }
        }
        return checkSum;
    }

    public static uint CalcCRC32(byte[] messageBytes)
    {
        var crcTable = new uint[256];
        uint checkSum;
        for (uint i = 0; i < 256; i++)
        {
            checkSum = i;
            for (uint j = 0; j < 8; j++)
                checkSum = (checkSum & 1) != 0 ? (checkSum >> 1) ^ 0xEDB88320 : check-
Sum >> 1;
            crcTable[i] = checkSum;
        }
        checkSum = messageBytes.Aggregate(0xFFFFFFFF, (current, s) => crcTable[(current
^ s) & 0xFF] ^ (current >> 8));
        checkSum ^= 0xFFFFFFFF;
        return checkSum;
    }

    // Получение сообщений
    private static void GetMessage(object sender, SerialDataReceivedEventArgs e)
    {
        // Чтение полученных байтов
        var receivedMessage = new byte[_serialPort.BytesToRead];
        _serialPort.Read(receivedMessage, 0, _serialPort.BytesToRead);

        // Проверка контрольной суммы
        if (CheckControlSum(receivedMessage)) ;
        WriteMessageToConsole(receivedMessage);// Вывод сообщения на экран
    }

private static void WriteMessageToConsole(byte[] messageBytes)
    {
        switch (checkSumAlg)
        {
            case CheckSumAlg.Not:
                break;
            case CheckSumAlg.Simple:
                {

                    // Вычиселние сообщения без контрольной суммы
                    Array.Resize(ref messageBytes, messageBytes.Length - 1);
                    break;
                }
            case CheckSumAlg.LRC:
                {
                    // Вычиселние сообщения без контрольной суммы
                    Array.Resize(ref messageBytes, messageBytes.Length - 1);
                    break;
                }
            case CheckSumAlg.CRC16:
                {

                    // Вычиселние сообщения без контрольной суммы
                    Array.Resize(ref messageBytes, messageBytes.Length - 2);
                    break;
```

```
                }
            case CheckSumAlg.CRC32:
                {

                    // Вычиселние сообщения без контрольной суммы
                    Array.Resize(ref messageBytes, messageBytes.Length - 4);
                    break;
                }
        }


        // Вывод сообщения на экран
        Console.WriteLine("Количество байт в полученном сообщении: " + message-
Bytes.Length.ToString());
        if (!typeProtocol)
        {
            Console.WriteLine("Полученное сообщение: " + BitConverter.ToDouble(message-
Bytes, 0).ToString());
        }
        else
        {
            Console.WriteLine("Полученное сообщение: " + _serialPort.Encod-
ing.GetString(messageBytes));
        }

        return;
    }


    private static bool CheckControlSum(byte[] messageBytes)
    {
        bool flagCheckSumEqual = false;
        switch (checkSumAlg)
        {
            case CheckSumAlg.Not:
                return true;
            case CheckSumAlg.Simple:
                {

                    // Последний байт в сообщении - контрольная сумма
                    byte checkSumInMessage = messageBytes[messageBytes.Length - 1];

                    // Вычиселние сообщения без контрольной суммы
                    byte[] array = new byte[messageBytes.Length];
                    messageBytes.CopyTo(array, 0);

                    Array.Resize(ref array, array.Length - 1);
                    byte[] checkSum = new byte[] { CalcSimpleCheckSum(array) };
                    Console.WriteLine("Контрольная сумма посылки: " + $"0x{BitCon-
verter.ToString(checkSum)}");
                    // Сравнение двух сумм
                    if (checkSum[0] == checkSumInMessage)
                    {
                        flagCheckSumEqual = true;
                    }
                    break;
                }
            case CheckSumAlg.LRC:
                {
                    // Последний байт в сообщении - контрольная сумма
                    byte checkSumInMessage = messageBytes[messageBytes.Length - 1];
```

```
                    // Вычиселние сообщения без контрольной суммы
                    byte[] array = new byte[messageBytes.Length];
                    messageBytes.CopyTo(array, 0);

                    Array.Resize(ref array, array.Length - 1);
                    byte[] checkSum = new byte[] { CalcLongRedCheck(array) };
                    Console.WriteLine("Контрольная сумма посылки: " + $"0x{BitCon-
verter.ToString(checkSum)}");
                    // Сравнение двух сумм
                    if (checkSum[0] == checkSumInMessage)
                    {
                        flagCheckSumEqual = true;
                    }
                    break;
                }
            case CheckSumAlg.CRC16:
                {
                    // Последние 2 байта в сообщении - контрольная сумма
                    byte[] checkSumInMessage = new byte[2];

                    for (int i = 0; i < checkSumInMessage.Length; i++)
                    {
                        checkSumInMessage[i] = messageBytes[messageBytes.Length -
checkSumInMessage.Length + i];
                    }

                    // Вычиселние сообщения без контрольной суммы
                    byte[] array = new byte[messageBytes.Length];
                    messageBytes.CopyTo(array, 0);
                    Array.Resize(ref array, array.Length - 2);

                    //byte[] checkSum = BitConverter.GetBytes(CalcCRC16Sasha(array));
                    //Console.WriteLine("Контрольная сумма посылки-Саша: " +
$"0x{BitConverter.ToString(checkSum)}");

                    byte[] checkSum = BitConverter.GetBytes(CalcCRC16Ira(array));
                    Console.WriteLine("Контрольная сумма посылки: " + $"0x{BitCon-
verter.ToString(checkSum)}");

                    if (checkSum.Equals(checkSumInMessage))
                    {
                        flagCheckSumEqual = true;
                    }

                    break;
                }
            case CheckSumAlg.CRC32:
                {

                    // Последние 2 байта в сообщении - контрольная сумма
                    byte[] checkSumInMessage = new byte[4];

                    for (int i = 0; i < checkSumInMessage.Length; i++)
                    {
                        checkSumInMessage[i] = messageBytes[messageBytes.Length -
checkSumInMessage.Length + i];
                    }

                    // Вычиселние сообщения без контрольной суммы
                    byte[] array = new byte[messageBytes.Length];
                    messageBytes.CopyTo(array, 0);
                    Array.Resize(ref array, array.Length - 4);
```

```
                    //byte[] checkSum = BitConverter.GetBytes(CalcCRC16Sasha(array));
                    //Console.WriteLine("Контрольная сумма посылки-Саша: " +
$"0x{BitConverter.ToString(checkSum)}");

                    byte[] checkSum = BitConverter.GetBytes(CalcCRC32(array));
                    Console.WriteLine("Контрольная сумма посылки: " + $"0x{BitCon-
verter.ToString(checkSum)}");

                    if (checkSum.Equals(checkSumInMessage))
                    {
                        flagCheckSumEqual = true;
                    }
                    break;
                }
            }

            return flagCheckSumEqual;
        }


        // Вычисление контрольной суммы
        private static byte[] CalcCheckSum(byte[] messageBytes)
        {
            byte[] checkSum;

            switch (checkSumAlg)
            {
                case CheckSumAlg.Not:
                    return new byte[0];
                case CheckSumAlg.Simple:
                    {
                        //контрольная сумма состоит из 1 байта
                        checkSum = new byte[1];
                        checkSum[0] = CalcSimpleCheckSum(messageBytes);
                        Console.WriteLine("Контрольная сумма посылки: " + $"0x{
BitConverter.ToString(checkSum)}");
                        return checkSum;
                    }
                case CheckSumAlg.LRC:
                    {
                        // контрольная сумма состоит из 1 байта
                        checkSum = new byte[1];
                        checkSum[0] = CalcLongRedCheck(messageBytes);
                        Console.WriteLine("Контрольная сумма посылки: " + $"0x{  BitCon-
verter.ToString(checkSum)}");
                        return checkSum;
                    }
                case CheckSumAlg.CRC16:
                    {

                        checkSum = BitConverter.GetBytes(CalcCRC16Ira(messageBytes));
                        Console.WriteLine("Контрольная сумма посылки: " + $"0x{ BitCon-
verter.ToString(checkSum)}");
                        return checkSum;
                    }
                case CheckSumAlg.CRC32:
                    {
                        checkSum = BitConverter.GetBytes(CalcCRC32(messageBytes));
                        Console.WriteLine("Контрольная сумма посылки: " + $"0x{  BitCon-
verter.ToString(checkSum)}");
                        return checkSum;
```

```
                }
            default:
                throw new Exception("Выбран нереализованный алгоритм подсчета контроль-
ной суммы");
        }
    }


    // Отправка сообщения
    private static void SendMessage(string message)
    {

        byte[] messageBytes;
        // Если формат данных бинарный
        if (!typeProtocol)
            try
            {
                messageBytes = BitConverter.GetBytes(Convert.ToDouble(message));
            }
            catch (FormatException)
            {
                throw new Exception("В биннарном формате передаются только числа");
            }
        else
            messageBytes = _serialPort.Encoding.GetBytes(message);

        Console.WriteLine("Количество байт в отправленном сообщении: " +
messageBytes.Length);
        // Добавление контрольной суммы
        // Переведем массив байт в список, чтобы было проще добавить байты контрольной
суммы
        List<byte> messageListByte = messageBytes.ToList();

        byte[] checkSum = CalcCheckSum(messageBytes);

        for (int i = 0; i < checkSum.Length; i++)
        {
            messageListByte.Add(checkSum[i]);
        }


        byte[] newMessageBytes = messageListByte.ToArray();

        // Отправка сообщения
        _serialPort.Write(newMessageBytes, 0, newMessageBytes.Length);
    }
}
```