Министерство образования Республики Беларусь

Учреждение образования

“Белорусский государственный университет информатики и радиоэлектроники”

Факультет компьютерных систем и сетей  
Кафедра электронных вычислительных машин  
Дисциплина: Основы компьютерных сетей

Отчет по лабораторной работе № 2

на тему

“Программная реализация пакетной передачи и алгоритма бит-стаффинга”

Выполнил:  
студент группы 150501 Климович А.Н.

Проверил:  
старший преподаватель Глецевич И.И.

Минск 2023

**1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

**1.1 Исходные данные**

Для написания и отладки программы использовался Qt Creator. В качестве языка программирования был выбран язык C++ и фреймворк Qt для создания графического интерфейса.

Для эмуляции COM-портов использовался Virtual Serial Ports Emulator, задействующий архитектуру UART 16550.

Программа запускалась на ОС Windows 10.

**1.2 Пакеты**

Для именования порции информации, передаваемой по каналам компьютерных сетей, используется обобщенный термин *пакет*. Пакет содержит последовательно сформированные станцией-передатчиком *поля*, предназначенные для их интерпретации в станции-приемнике. Формат пакета представлен на рисунке 1.1.

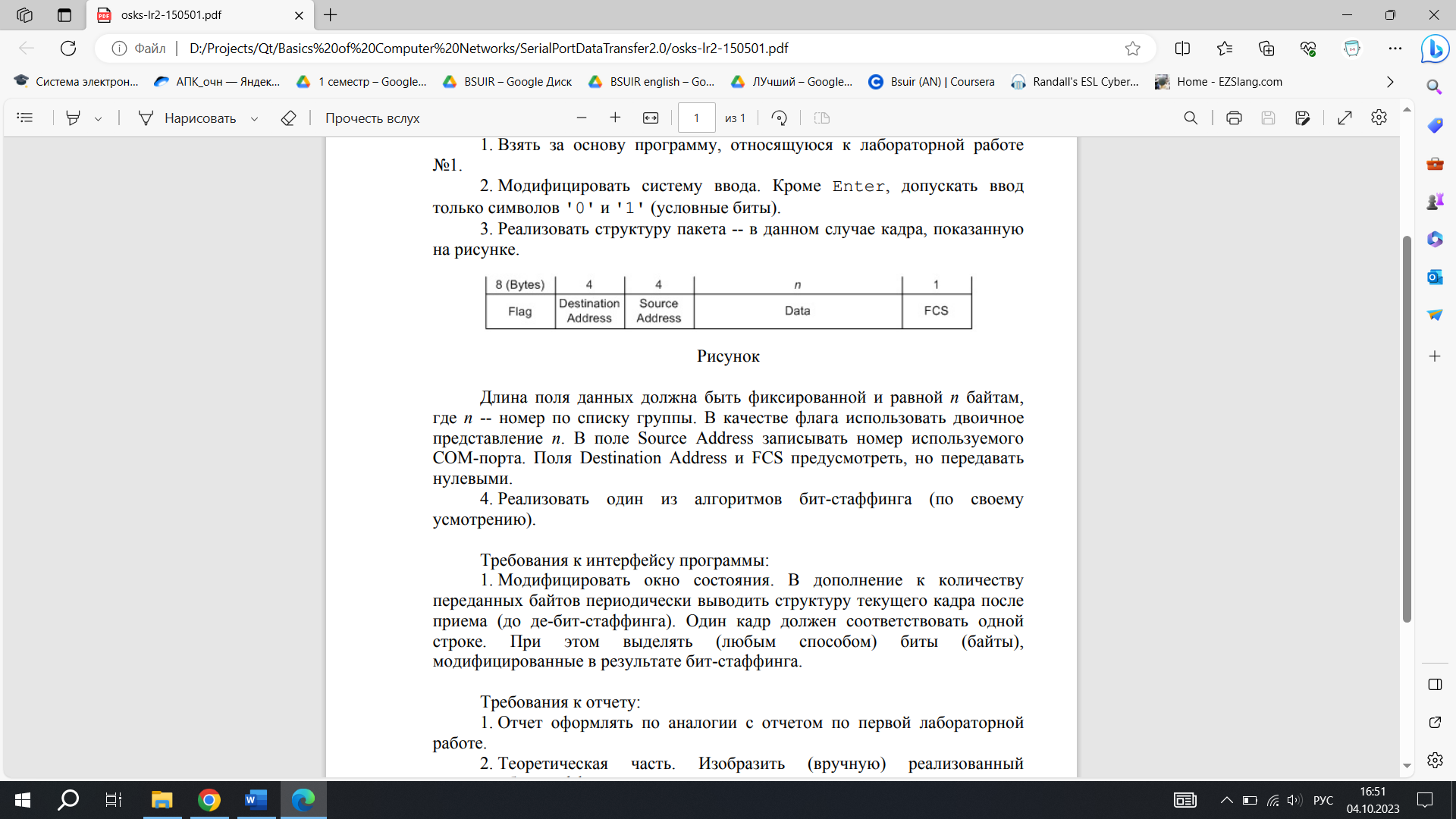


Рисунок 1.1 – Формат пакета

Назначение полей:

1. Flag (флаг начала пакета) – позволяет определить начало пакета.

2. Destination Address (адрес назначения) – позволяет указать станцию, для которой предназначен пакет.

3. Source Address (адрес источника) – позволяет указать станцию, сгенерировавшую пакет.

4. Data (данные) – полезное наполнение пакета.

5. FCS (Frame Control Sequence – контрольная сумма) – позволяет проверить целостность пакета.

**1.3 Алгоритм бит-стаффинга на стороне передатчика**



**1.4 Алгоритм бит-стаффинга на стороне приемника**



**2 ПРАКТИЧЕСКАЯ ЧАСТЬ**

**2.1 Изменения и дополнения кода программы**

Файл MainWindow.cpp:

void MainWindow::sendMessage()

{

port.setDataBits(QSerialPort::DataBits(ui->byteSize\_spinBox->value()));

QString newPortName = "COM" + ui->COMPort\_comboBox->currentText();

if(newPortName != this->port.portName())

{

if(canChangePort(portList, newPortName) == true)

{

port.close();

this->port.setPortName(newPortName);

if(!port.open(QIODevice::ReadWrite))

{

QMessageBox::critical(this, "Error", "Cannot change port");

exit(1);

}

}

else

{

QMessageBox::critical(this, "Error", "Port is not available");

ui->COMPort\_comboBox->setCurrentText(port.portName().sliced(3, port.portName().length() - 3));

return;

}

}

QString message = ui->input\_textEdit->toPlainText(); // получаем текст из входной строки

if(!message.isEmpty()) // если сообщение не пустое

{

QString messageLength = QString::number(message.toUtf8().size());

unsigned long countOfFrames = message.length() / STUDENT\_LIST\_NUMBER;

unsigned long position = 0;

for(size\_t i = 0; i < countOfFrames; ++i)

{

Frame frame(message.mid(position, STUDENT\_LIST\_NUMBER), port);

QString modifiedFrame = frame.doBitStaffing();

// Передаем кадр посимвольно

for(auto bit : modifiedFrame)

port.write(QString(bit).toStdString().c\_str());

position += STUDENT\_LIST\_NUMBER;

}

// Если сообщение по длине меньше STUDENT\_LIST\_NUMBER, то есть меньше одного кадра

if(messageLength.toInt() % STUDENT\_LIST\_NUMBER != 0)

{

QString lastMessage = message.mid(position);

while(lastMessage.length() < STUDENT\_LIST\_NUMBER)

lastMessage = lastMessage + '?';

Frame frame(lastMessage, port);

QString modifiedFrame = frame.doBitStaffing();

messageLength = QString::number(modifiedFrame.length());

// Передаем кадр посимвольно

for(auto bit : modifiedFrame)

port.write(QString(bit).toStdString().c\_str());

}

ui->status\_textBrowser->append("Bytes sent: " + messageLength);

ui->input\_textEdit->clear(); // очищаем строку для ввода сообщений

}

}

void MainWindow::receiveMessage()

{

QByteArray receivedBytes = port.readAll();

QString receivedData = "";

Frame frame("", port);

QStringList listOfFrames = QString(receivedBytes).split(QRegularExpression(frame.getFlag()));

char bitSymbol = frame.getBitSymbolForBitStaffing(frame.getFlag());

QString modifiedFlag = frame.getFlag().mid(0, 7) + bitSymbol + frame.getFlag().mid(7, 1);

QString highlightFlag = modifiedFlag.mid(0, 7) + "<font color=\"#FF0000\">" + modifiedFlag[7] + "</font>" + modifiedFlag[8];

for(int i = 1; i < listOfFrames.length(); ++i)

{

listOfFrames[i].replace(modifiedFlag, highlightFlag);

ui->status\_textBrowser->append(frame.getFlag() + listOfFrames[i]);

listOfFrames[i].replace(highlightFlag, modifiedFlag);

frame.setFrame(frame.doDeBitStaffing(frame.getFlag() + listOfFrames[i]));

receivedData += frame.getData().remove('?');

}

ui->output\_textBrowser->append(receivedData);

}

void MainWindow::on\_input\_textEdit\_textChanged()

{

QString text = ui->input\_textEdit->toPlainText();

text.remove(QRegularExpression("[^01]"));

if(ui->input\_textEdit->toPlainText() != text)

{

ui->input\_textEdit->blockSignals(true);

ui->input\_textEdit->setText(text);

ui->input\_textEdit->moveCursor(QTextCursor::EndOfBlock);

ui->input\_textEdit->blockSignals(false);

}

}

Файл Frame.h:

#ifndef FRAME\_H

#define FRAME\_H

#include "SerialPortFunctions.h"

#define STUDENT\_LIST\_NUMBER 11 // номер студента по списку

class Frame

{

private:

QString frame;

QString flag;

QString sourceAddress;

QString destinationAddress;

QString data;

QString fcs;

public:

Frame();

Frame(QString data, QSerialPort& port);

QString doBitStaffing();

QString doDeBitStaffing(QString modifiedFrame);

char getBitSymbolForBitStaffing(QString flag);

QString getFrame() const;

QString getFlag() const;

QString getSourceAddress() const;

QString getDestinationAddress() const;

QString getData() const;

QString getFcs() const;

void setFlag(const QString &newFlag);

void setFrame(const QString &newFrame);

void setSourceAddress(const QString &newSourceAddress);

void setDestinationAddress(const QString &newDestinationAddress);

void setData(const QString &newData);

void setFcs(const QString &newFcs);

protected:

void updateFrame();

};

#endif // FRAME\_H

Файл Frame.cpp:

#include "Frame.h"

void Frame::setFrame(const QString &newFrame)

{

frame = newFrame;

flag = frame.mid(0, 8);

destinationAddress = frame.mid(8, 4);

sourceAddress = frame.mid(12, 4);

data = frame.mid(16, 11);

fcs = frame.mid(27, 1);

}

void Frame::setFlag(const QString &newFlag) { flag = newFlag; updateFrame(); }

void Frame::setSourceAddress(const QString &newSourceAddress) { sourceAddress = newSourceAddress; updateFrame(); }

void Frame::setDestinationAddress(const QString &newDestinationAddress) { destinationAddress = newDestinationAddress; updateFrame(); }

void Frame::setData(const QString &newData) { data = newData; updateFrame(); }

void Frame::setFcs(const QString &newFcs) { fcs = newFcs; updateFrame(); }

QString Frame::getFrame() const { return frame; }

QString Frame::getFlag() const { return flag; }

QString Frame::getSourceAddress() const { return sourceAddress; }

QString Frame::getDestinationAddress() const { return destinationAddress; }

QString Frame::getData() const { return data; }

QString Frame::getFcs() const { return fcs; }

Frame::Frame()

{

flag = toBinary(QString::number(STUDENT\_LIST\_NUMBER), 8);

sourceAddress = "0000";

destinationAddress = "0000";

fcs = "0";

data = "00000000000";

updateFrame();

}

Frame::Frame(QString newData, QSerialPort& port)

{

flag = toBinary(QString::number(STUDENT\_LIST\_NUMBER), 8);

sourceAddress = toBinary(port.portName().mid(3, port.portName().length() - 3), 4);

destinationAddress = "0000";

fcs = "0";

data = newData;

updateFrame();

}

QString Frame::doBitStaffing()

{

QString frameWithoutFlag = frame.mid(8, frame.length() - 8);

int pos = frameWithoutFlag.indexOf(flag.mid(0, 7), 0);

char bitSymbol = getBitSymbolForBitStaffing(flag);

while (pos != -1) {

frameWithoutFlag.insert(pos + 7, bitSymbol);

pos = frameWithoutFlag.indexOf(flag.mid(0, 7), pos + 1);

}

return flag + frameWithoutFlag;

}

QString Frame::doDeBitStaffing(QString modifiedFrame)

{

QString flag = modifiedFrame.mid(0, 8);

int pos = modifiedFrame.indexOf(flag.mid(0, 7), 8);

while (pos != -1) {

modifiedFrame.remove(pos + 7, 1);

pos = modifiedFrame.indexOf(flag.mid(0, 7), pos + 1);

}

return modifiedFrame;

}

char Frame::getBitSymbolForBitStaffing(QString flag)

{

char bitSymbol;

if((flag[6] == '1' && flag[7] == '1')

|| (flag[6] == '0' && flag[7] == '1'))

{

bitSymbol = '0';

}

else

bitSymbol = '1';

return bitSymbol;

}

void Frame::updateFrame()

{

frame = flag + destinationAddress + sourceAddress + data + fcs;

}