ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Московский институт электроники и математики им. А. Н. Тихонова
Департамент компьютерной инженерии

Дисциплина: **Проектный семинар «Практическая реализация современных технологий создания информационных систем»**

ПРОЕКТ

«Комплекс, измеряющий частоту колебаний симметричного транзисторного мультивибратора»

по направлению **"Системы контроля и управления"**

Студенты: Пыжов Илья Игоревич, Татаринова Полина Юрьевна

Группа: БИВ225

Вариант: 2

Дата сдачи: 08.06.2024

Преподаватель: Тув Александр Леонидович

МОСКВА 2024

Оглавление

Задание

Разработать комплекс, измеряющий частоту колебаний симметричного транзисторного мультивибратора (питание 7 Вольт). Частота колебаний мультивибратора задаётся переменными резисторами (изменять симметрично). Измеренную частоту индицировать в окне приложения ПК в виде числа с плавающей точкой. Приложение должно позволять задавать пороги (нижний и верхний) допустимой частоты и сохранять их в XML файле. В случае, если измеренная частота в пределах нормы – отображать ее значение зеленым цветом, при превышении – красным, если частота ниже нижнего порога, то синим.

Предполагаемая схемотехника и перечень элементов:

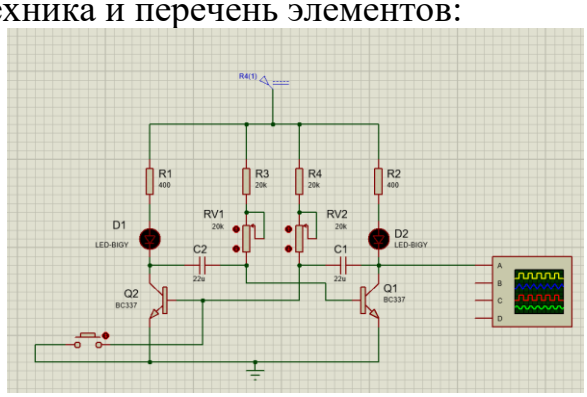

Рисунок 1 — Предполагаемая схемотехника

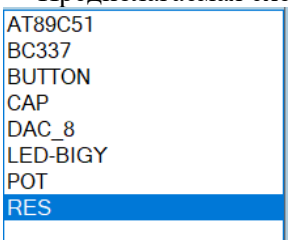

Рисунок 2 — Перечень элементов

Выход мультивибратора (точка, подключенная к осциллографу) подключается на ножке порта P1.

Блок-схема

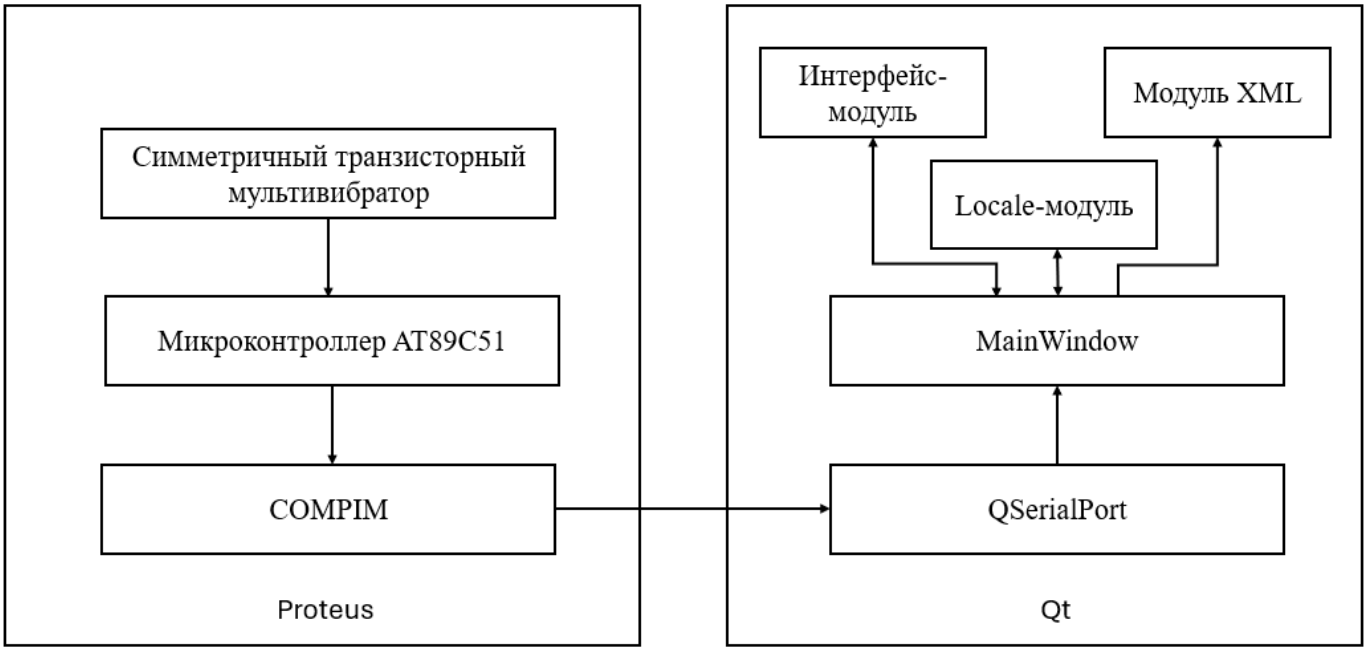

Рисунок 3 — Блок-схема

Работа в Proteus

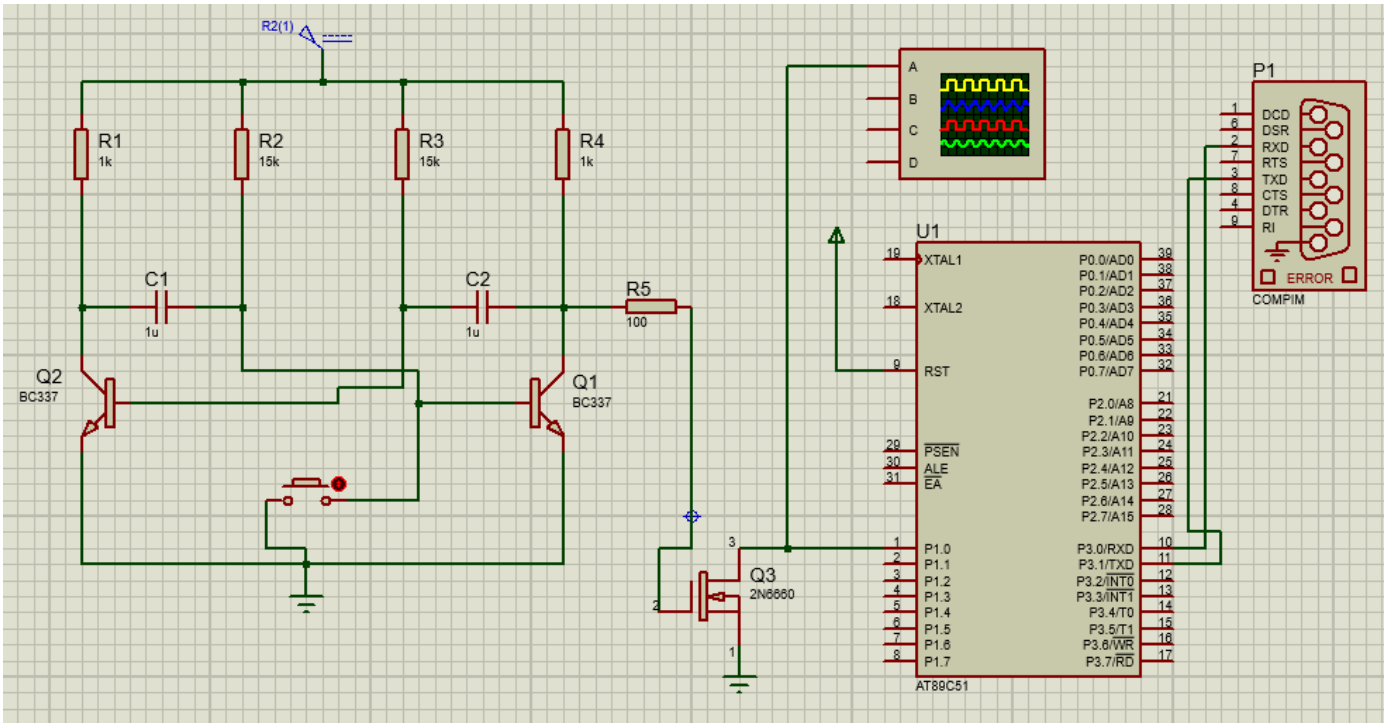

Рисунок 4 — Схема в Proteus

Алгоритм решения в Proteus

```
#include <mcs51reg.h>

typedef unsigned char uint8_t;
typedef unsigned int uint16_t;
typedef unsigned long uint32_t;

sbit output_pin = P1^0;

void delay(uint32_t ms);
void init_serial(void);
void putc(uint8_t input);

void init_serial() {
   TMOD |= 0x20;//!!!!!!!
   SCON = 0x50;
   TH1 = 0xFD;
   TL1 = 0xFD;
   TR1 = 1;
}

void putc(uint8_t input) {
   SBUF = input;
   while (!TI);
   TI = 0;
}

void delay(uint32_t ms) {
   uint32_t i, j;
   for (i = 0; i < ms; i++)
```

```
      for (j = 0; j < 120; j++);
}

void main() {
   TMOD = 0x01;
   init_serial();//!!!!!!! здесь необходимо заменить TMOD = 0x20; на TMOD |= 0x20;, так как иначе переписывается инициализация таймера 0.
   P1 = 0xFF;
   while (1) {
                  TR0 = 0;
                  // Измеряем период сигнала
                  TH0 = 0; // Сбросим старший байт
                  TL0 = 0; // Сбросим младший байт
                  while (sbit output_pin == 1);
                  while (sbit output_pin == 1);
                  TR0 = 1; // Запустим таймер
                  while (sbit output_pin == 0); // Ждем конца сигнала
                  TR0 = 0; // Остановим таймер
      putc(TH0); // Отправка старшего байта
      putc(TL0);
      delay(10);
   }
}
```

Алгоритм решения в Qt

Файл project.pro

```
QT += core gui serialport
CONFIG += c++11

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

TARGET = MyApp
TEMPLATE = app

SOURCES += main.cpp \
        mainwindow.cpp

HEADERS += mainwindow.h

FORMS += \
   mainwindow.ui

TRANSLATIONS += \
   translations/project_en_US.ts \
   translations/project_ru_RU.ts \
   translations/project_de_DE.ts
```

Файл mainwimdow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QSerialPort>
#include <QLabel>
#include <QDoubleSpinBox>
#include <QTranslator>
#include <QDoubleSpinBox>

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
   Q_OBJECT

public:
   MainWindow(QWidget *parent = nullptr);
   ~MainWindow();

private slots:
   void readSerialData();
   void loadSettings();
   void saveSettings();
   void updateThresholds();
   void changeLanguage(int index);
```

```
private:
    void setupUI();
    void setupSerialPort();
    void updateFrequencyDisplay(double freq);

    Ui::MainWindow *ui;
    QSerialPort *serial;
    QTranslator translator;
    double lowerThreshold;
    double upperThreshold;
    QString currentLocale;
};

#endif // MAINWINDOW_H
```

Файл main.cpp

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

int main(int argc, char *argv[]) {
    QApplication app(argc, argv);
    MainWindow window;
    window.show();
    return app.exec();
}
```

Файл mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QSettings>
#include <QDebug>
#include <limits>
#include <QTranslator>
#include <QXmlStreamWriter>
#include <QFile>
#include <QCoreApplication>
#include <QDir>

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent), ui(new Ui::MainWindow), serial(new QSerialPort(this)) {
    ui->setupUi(this);
    setupUI();
    setupSerialPort();
    loadSettings();

    ui->languageComboBox->addItem("English", "en_US");
    ui->languageComboBox->addItem("Русский", "ru_RU");
    ui->languageComboBox->addItem("Deutsch", "de_DE");

    ui->label->setText(tr("Frequency, (Hz)"));
    ui->label_2->setText(tr("Upper Threshold, (Hz)"));
    ui->label_3->setText(tr("Lower Threshold, (Hz)"));

    // Связываем выбор языка с функцией смены языка
```

```
    connect(ui->languageComboBox, QOverload<int>::of(&QComboBox::currentIndexChanged),
            this, &MainWindow::changeLanguage);
}

MainWindow::~MainWindow() {
    delete ui;
}


void MainWindow::changeLanguage(int index)
{
    currentLocale = ui->languageComboBox->itemData(index).toString();
    if (translator.load("C:\\Qt\\project\\translations\\project_" + currentLocale + ".qm")) {
        qApp->installTranslator(&translator);
        ui->retranslateUi(this);
    }
}


void MainWindow::setupUI() {
    // Устанавливаем минимальные и максимальные значения для doubleSpinBox и doubleSpinBox_2
    ui->doubleSpinBox->setRange(std::numeric_limits<double>::lowest(),
std::numeric_limits<double>::max());
    ui->doubleSpinBox_2->setRange(std::numeric_limits<double>::lowest(),
std::numeric_limits<double>::max());

    // Подключаем сигналы doubleSpinBox и doubleSpinBox_2 к слоту обновления порогов
    connect(ui->doubleSpinBox, QOverload<double>::of(&QDoubleSpinBox::valueChanged), this,
&MainWindow::updateThresholds);
    connect(ui->doubleSpinBox_2, QOverload<double>::of(&QDoubleSpinBox::valueChanged), this,
&MainWindow::updateThresholds);
}

void MainWindow::setupSerialPort() {
    serial->setPortName("COM4"); // Убедитесь, что выбран правильный COM-порт
    serial->setBaudRate(QSerialPort::Baud9600);
    serial->setDataBits(QSerialPort::Data8);
    serial->setParity(QSerialPort::NoParity);
    serial->setStopBits(QSerialPort::OneStop);
    serial->setFlowControl(QSerialPort::NoFlowControl);

    connect(serial, &QSerialPort::readyRead, this, &MainWindow::readSerialData);

    serial->open(QIODevice::ReadOnly); // Открываем порт только для чтения
}

void MainWindow::loadSettings() {
    QSettings settings("C:/Qt/project/build/Desktop_Qt_6_7_1_MinGW_64_bit-Debug/settings.xml",
QSettings::NativeFormat);
    lowerThreshold = settings.value("lowerThreshold", 0).toUInt();
    upperThreshold = settings.value("upperThreshold", 1000).toUInt();
    currentLocale = settings.value("language", "en_US").toString(); // Загрузка текущего языка
    ui->doubleSpinBox_2->setValue(lowerThreshold); // Устанавливаем значение нижнего порога в
текстовое поле
```

```
    ui->doubleSpinBox->setValue(upperThreshold); // Устанавливаем значение верхнего порога в
текстовое поле
    // Устанавливаем текущий язык
    int index = ui->languageComboBox->findData(currentLocale);
    if (index != -1) {
        ui->languageComboBox->setCurrentIndex(index);
        changeLanguage(index);
    }
}

void MainWindow::saveSettings() {
    QSettings settings("C:/Qt/project/build/Desktop_Qt_6_7_1_MinGW_64_bit-Debug/settings.xml",
QSettings::NativeFormat);
    settings.setValue("lowerThreshold", QString::number(lowerThreshold));
    settings.setValue("upperThreshold", QString::number(upperThreshold));
    settings.setValue("language", currentLocale); // Сохранение текущего языка
    // Записываем пороговые значения в XML файл
    QFile file("C:/Qt/project/build/Desktop_Qt_6_7_1_MinGW_64_bit-Debug/settings.xml");
    if (file.open(QIODevice::WriteOnly | QIODevice::Text)) {
        QTextStream out(&file);
        out << "<settings>\n";
        out << "    <lowerThreshold>" << lowerThreshold << "</lowerThreshold>\n";
        out << "    <upperThreshold>" << upperThreshold << "</upperThreshold>\n";
        out << "    <windowGeometry>" << QString("%1X%2").arg(width()).arg(height())<<
"</windowGeometry>\n";
        out << "    <windowPosition>" << QString("%1X%2").arg(geometry().x()).arg(geometry().y())<<
"</windowPosition>\n";
        out << "    <locale>" << currentLocale << "</locale>\n";
        out << "</settings>";
        file.close();
    }
}

void MainWindow::updateThresholds() {
    QLocale locale(currentLocale);
    ui->doubleSpinBox_2->setLocale(locale);
    ui->doubleSpinBox->setLocale(locale);
    lowerThreshold = ui->doubleSpinBox_2->value(); // Получаем значение нижнего порога из
текстового поля
    upperThreshold = ui->doubleSpinBox->value(); // Получаем значение верхнего порога из текстового
поля
}

void MainWindow::readSerialData() {
    QByteArray data = serial->readAll();
    int dataSize = data.size();

    if (dataSize % 2 != 0) {
        return; // Если данные неполные, игнорируем их
    }

    for (int i = 0; i < dataSize; i += 2) {
        char byte1 = data[i+1];
        char byte2 = data[i];
```

```
        quint16 value = (static_cast<quint16>(static_cast<uint8_t>(byte1)) << 8) |
static_cast<uint8_t>(byte2);

        qDebug() << "Raw data received:" << QByteArray::fromRawData(&byte1, 1).toHex() <<
QByteArray::fromRawData(&byte2, 1).toHex();
        qDebug() << "Interpreted value:" << value;

        double frequency = 1000000.0*0.92 / (value * 2);
        updateFrequencyDisplay(frequency);
    }
}

void MainWindow::updateFrequencyDisplay(double freq) {
    QString color = "green";
    QLocale locale(currentLocale);
    QString fre = "";
    ui->label_4->setLocale(locale);
    fre = locale.toString(freq);
    ui->label_4->setText(QString("<font color=\"%1\">%2</font>").arg(color).arg(fre));
    ui->doubleSpinBox_2->setLocale(locale);
    ui->doubleSpinBox->setLocale(locale);
    saveSettings();
}
```

Итоговый интерфейс

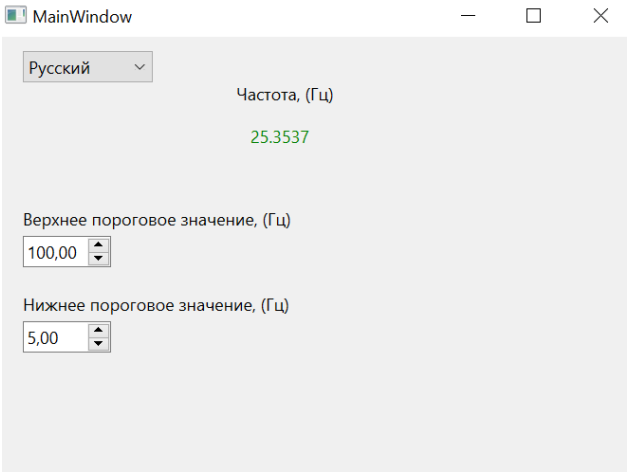

Рисунок 5 — Итоговый интерфейс

Работа в VSPE

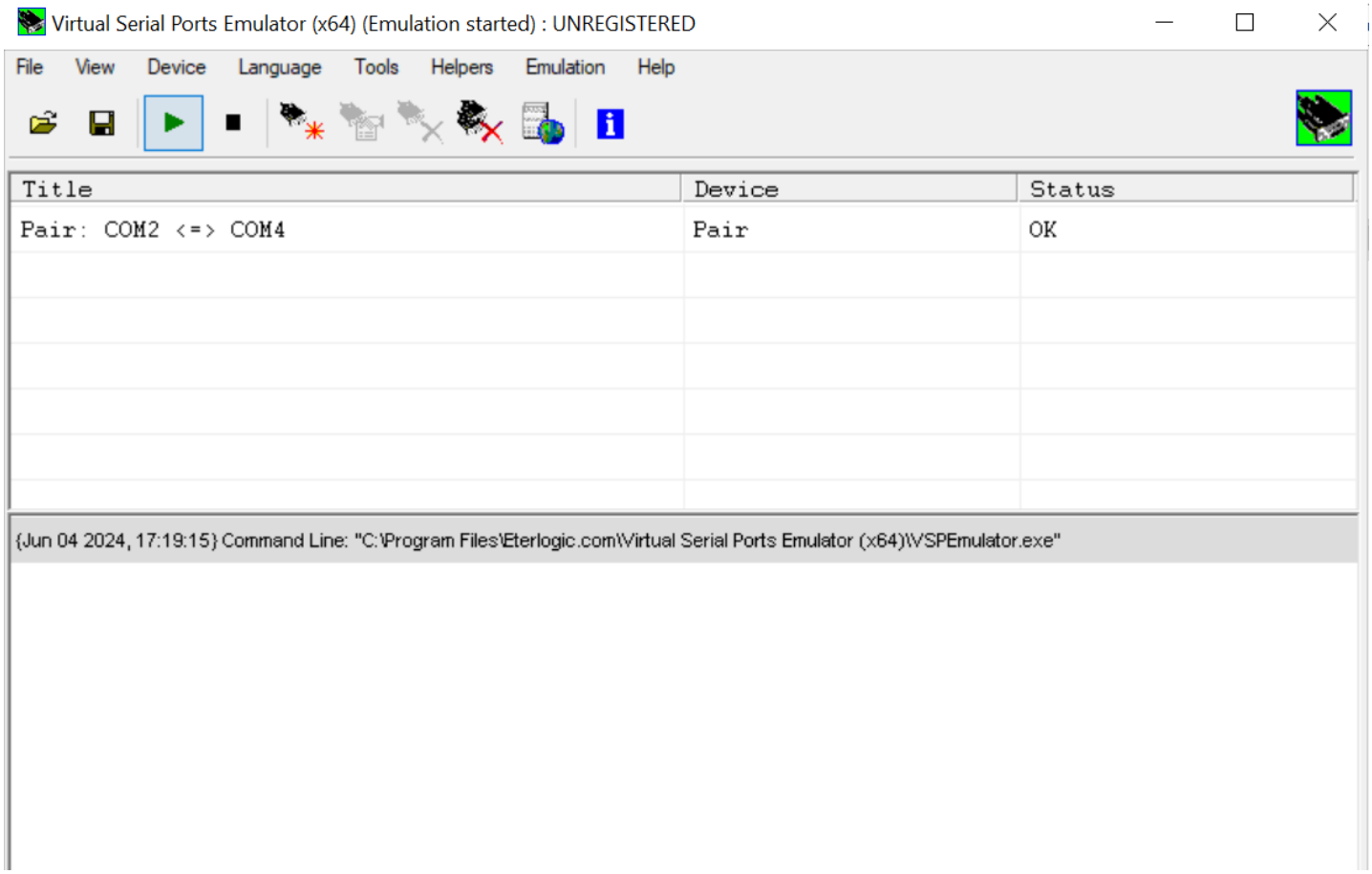

Рисунок 6 — Созданная пара портов COM2 COM4