#define TIMEOUT 100

**Функция подключения COM-порта и установление нужного значения бодрейта.**

void COM::Init(const string& port, int baudrate)

{

//открытие порта

m\_Handle = CreateFile(

port.c\_str(), //наименование порта, например COM1

GENERIC\_READ | GENERIC\_WRITE,

0,

NULL,

OPEN\_EXISTING,

FILE\_ATTRIBUTE\_NORMAL,

NULL);

if (m\_Handle == (HANDLE)-1) {

m\_Handle = 0;

throw Exception();

}

//устанавливаем размеры входного и выходного буферов

SetupComm(m\_Handle, 1500, 1500);

//настраиваем тайм-ауты для текущего устройства

COMMTIMEOUTS CommTimeOuts;

CommTimeOuts.ReadIntervalTimeout = 0xFFFFFFFF;

CommTimeOuts.ReadTotalTimeoutMultiplier = 0;

CommTimeOuts.ReadTotalTimeoutConstant = TIMEOUT;

CommTimeOuts.WriteTotalTimeoutMultiplier = 0;

CommTimeOuts.WriteTotalTimeoutConstant = TIMEOUT;

if (!SetCommTimeouts(m\_Handle, &CommTimeOuts)) {

m\_Handle = 0;

throw Exception();

}

//настраиваем параметры работы для текущего устройства

DCB ComDCM;

memset(&ComDCM, 0, sizeof(ComDCM));

ComDCM.DCBlength = sizeof(DCB);

GetCommState(m\_Handle, &ComDCM); // текущие значения

ComDCM.BaudRate = DWORD(baudrate); // скорость в бодах

ComDCM.ByteSize = 8; // размер байта

ComDCM.Parity = NOPARITY; // паритет

ComDCM.StopBits = ONESTOPBIT; // количество стоп бит

ComDCM.fAbortOnError = TRUE;

ComDCM.fDtrControl = DTR\_CONTROL\_DISABLE; // сброс DTR бита

ComDCM.fRtsControl = RTS\_CONTROL\_TOGGLE; // автоустановка RTS бита

ComDCM.fBinary = TRUE; //бинарный режим всегда

ComDCM.fParity = FALSE; //паритет

ComDCM.fInX = ComDCM.fOutX = FALSE;

ComDCM.XonChar = 0;

ComDCM.XoffChar = uint8\_t(0xff);

ComDCM.fErrorChar = FALSE;

ComDCM.fNull = FALSE;

ComDCM.fOutxCtsFlow = FALSE;

ComDCM.fOutxDsrFlow = FALSE;

ComDCM.XonLim = 128;

ComDCM.XoffLim = 128;

//установка параметров работы для текущего устройства

if (!SetCommState(m\_Handle, &ComDCM)) {

CloseHandle(m\_Handle);

m\_Handle = 0;

throw Exception();

}

}

**Функция отключения COM-порта.**

void COM::Disconnect()

{

if (m\_Handle != 0) {

CloseHandle(m\_Handle);

m\_Handle = 0;

}

}

**Функция записи в COM-порт. Выполняется через запись данных в файл устройства при этом будет записано столько байт data, сколько будет указано в datasize.**

void COM::Write(const uint8\_t\* data, int datasize)

{

if (m\_Handle == 0) { throw Exception(); }

DWORD feedback;

if (!WriteFile(m\_Handle, data, datasize, &feedback, 0) ||

feedback != datasize) {

CloseHandle(m\_Handle);

m\_Handle = 0;

throw Exception("write");

}

}

**Функция записи строки в COM-порт. Запись происходит побайтно и работает до тех пор пока не встретится символ конца строки(‘\0’).**

void COM::WriteStr(uint8\_t\* data)

{

int i;

for (i = 0; \*(data + i) != '\0'; ++i)

{

Write((data + i), 1);

}

Write((data + i), 1);

}

**Функция чтения из COM-порта. Выполняется через чтения данных из файла устройства при этом будет записано столько байт data, сколько будет указано в datasize.**

void COM::Read(uint8\_t\* data, int bufsize) {

DWORD feedback = 0;

int attempts = 5; //количество запросов на чтение

while (bufsize && attempts) {

if (attempts) attempts--;

if (!ReadFile(m\_Handle, data, bufsize, &feedback, NULL))

{

CloseHandle(m\_Handle);

m\_Handle = 0;

throw Exception("read");

}

bufsize -= feedback;

data += feedback;

}

if (attempts == 0)

{

throw Exception("read2");

}

}

**Функция чтения строки из COM-порта. Чтение происходит побайтно и работает до тех пор пока не встретится символ конца строки(‘\0’).**

void COM::ReadStr(uint8\_t\* data) {

int i;

uint8\_t temp='a';

for (i = 0; temp != '\0'; ++i)

{

Read(&temp, 1);

\*(data + i) = temp;

}

\*(data + i) = '\0';

}