**Конструкторы класса**

CMyDate(); // конструктор без параметров

CMyDate(int d, int m, int y); // конструктор с тремя параметрами

// реализация конструкторов и методов класса

// конструктор без параметров (конструктор по умолчанию)

CMyDate::CMyDate()

{

// установить дату 01.01.2001 day = 1; month = 1; year = 2001;

}

// конструктор с тремя параметрами

CMyDate::CMyDate(int d, int m, int y)

{

day = d; month = m; year = y;

}

**kbhit() – возвращает истину, если нажата какая то клавиша на клавиатуре**

**Скелет для работы с LCard Digital**

Подключить библиотеку (инклуды,либ)

#include <stdlib.h>

#include <stdio.h>

#include <QMainWindow>

#include "Lusbapi.h" // заголовочный файл библиотеки Lusbapi

#include <conio.h>

QT\_BEGIN\_NAMESPACE

namespace **Ui** { class **MainWindow**; }

QT\_END\_NAMESPACE

class **MainWindow** : public QMainWindow

{

Q\_OBJECT

public:

**MainWindow**(QWidget \*parent = nullptr);

~***MainWindow***();

private:

Ui::MainWindow \*ui;

public:

ILE140 \*pModule; // указатель на интерфейс модуля

MODULE\_DESCRIPTION\_E140 md; // структура c информацией о модуле

char ModuleName[7]; // название модуля

BYTE UsbSpeed; // скорость работы шины USB

WORD TtlIn, TtlOut; // состояния цифровых линий

DWORD DllVersion; // версия библиотеки

public slots:

void **Print**(QString data);

private slots:

void **on\_pushButton\_clicked**();

void **on\_pushButton\_2\_clicked**();

void **on\_pushButton\_3\_clicked**();

};

WORD i;

PCHAR pMname = PCHAR("e140");

if((DllVersion=GetDllVersion()) != CURRENT\_VERSION\_LUSBAPI) // проверим версию DLL библиотеки

{

Print("Неправильная версия Dll!");

}

pModule = static\_cast<ILE140 \*>(CreateLInstance(pMname)); // получим указатель на интерфейс модуля

if(!pModule)

{

Print("Не могу получить указатель на интерфейс");

}

// if(!pModule->OpenLDevice(0)) // попробуем обнаружить какой-нибудь модуль в нулевом виртуальном слоте

// {

// Print("Не могу получить доступ к модулю!");

// }

// попробуем обнаружить модуль E14-140 в первых WORD MAX\_VIRTUAL\_SLOTS\_QUANTITY\_LUSBAPI виртуальных слотах

for(i = 0x0; i < MAX\_VIRTUAL\_SLOTS\_QUANTITY\_LUSBAPI; i++) if(pModule->*OpenLDevice*(i)) break;

if(!pModule->*GetUsbSpeed*(*&UsbSpeed*)) // попробуем получить скорость работы шины USB

{

Print("Не могу узнать скорость работы USB!\n");

}

if(!pModule->*GetModuleName*(ModuleName)) // прочитаем название модуля в нулевом виртуальном слоте

{

Print("Не могу прочитать название модуля!\n");

}

if(strcmp(ModuleName, "E140")) // проверим: этот модуль - 'E14-140'?

{

Print(" В нулевом виртуальном слоте не 'E14-140'\n");

}

if(!pModule->*GET\_MODULE\_DESCRIPTION*(*&md*)) // попробуем прочитать информацию о модуле

{

Print("Не выполнена функция GET\_MODULE\_DESCRIPTION ()!");

}

//connect(ui->pushButton,SIGNAL());

// char chars[51]; //сообщение о готовности к работе (вероятно не нужно, так как логика моей проги немного другая и сообщение по умолчанию не работает для кьюта

// memcpy( chars, md.Module.SerialNumber,51);//переделал формирование сообщения с серийником

// Print("Модуль E14-140 (серийный номер "+QString(chars)+") полностью готов к работе!");

// далее можно располагать функции для непосредственного

// управления модулем

//. . . . . .

pModule->*ENABLE\_TTL\_OUT*(true);

TtlOut = 0x0; //инициализация состояния выходных линий по умолчанию

// выйдем из программы

// return 0;

}

MainWindow::~***MainWindow***()

{

delete ui;

}

void MainWindow::**Print**(QString data)

{

ui->consol->textCursor().insertText(data+'\r'); // Вывод текста в консоль

ui->consol->moveCursor(QTextCursor::End);//Scroll

}

void MainWindow::**on\_pushButton\_clicked**()

{

//TtlOut =0b1111111111111111; //для примера присвоение логических единиц на все внешние цифровые линии

TtlOut =0x01;

pModule->*TTL\_OUT*(TtlOut);

}

void MainWindow::**on\_pushButton\_2\_clicked**()

{

TtlOut =0x0;

pModule->*TTL\_OUT*(TtlOut);

}

void MainWindow::**on\_pushButton\_3\_clicked**()

{

if(!pModule->*ReleaseLInstance*()) // завершим работу с модулем

{

Print("Не выполнена функция ReleaseLInstance()!");

}

}

**Русский язык в Визуал Студио**

**Setlocale (LC\_ALL,”ru”);**

**Шаблонная функция**

template<typename T1, typename T2>

void Sum (T1 a, T2 b) {Sum = a+b;}

**Указатель на функцию в качестве параметра**

Int foo1(int a){return a-1}; Int foo2(int a){return a+1};

void main()

int (\*foopointer)(int a);

foopointer = foo2;

**Нультерминатор –** показывает компилятору конец строки \0

**Условная компиляция**

#ifdef Debug {} #endif выполнится, если определен в дефайне Debug, #else – выполнится по аналогии с обычным if

#ifndef – наоброт, если не поределен

**Тернарный оператор –**

Запись условного оператора if в сокращенном виде: (a<10)?(действие //?-аналог if): (действие//: - аналог else)

**Принципы ООП –** инкапсуляция, наследование и полиморфизм

**Int main vs void main** (первое более правильно, так как возвращает инт значение после завершения – 0 для операционки – программа завершилась успешно

**Создание файла и запись в него данных**

#include <QCoreApplication>

#include <QFile> //Подключаем для работы с классом QFile

#include <QDate>

#include <QByteArray>

#include <QString>

#include <QString>

#include <QDesktopWidget>

#include <QScreen>

#include <QMetaEnum>

#include <unistd.h>

#include <errno.h>

QFile file;

QString fname = QDate::currentDate().toString("dd.MM.yyyy")+".txt"; //присвоение переменной значения текущей даты (CurrentDateTime для добавления еще и времени) с припиской txt

file.setFileName(fname); // присвоение объекту класса QFile file в качестве имени значения переменной

//Создание файла и запись в него данных

void MainWindow::**writeInFile**()

{

if (file.exists()){//Проверка - существует ли файл

if (file.*open*(QIODevice::WriteOnly | QIODevice::Append)) { // Append - для записи в конец файла

QByteArray data = ui->lineEdit->text().toLocal8Bit()+'\r';

file.write(data);

file.*close*();}}

else {if (file.*open*(QIODevice::WriteOnly | QIODevice::Append))//Если файл только создается, то в первую строчку записываем название параметра

{file.write("Значение напряжения \n");

QByteArray data = ui->lineEdit->text().toLocal8Bit()+'\r';

file.write(data);

file.*close*();}

}

}

//+++++++++++++[Процедура вывода данных в консоль]+++++++++++

void MainWindow::**Print**(QString data)

{

ui->consol->textCursor().insertText(data+'\r'); // Вывод текста в консоль

ui->consol->moveCursor(QTextCursor::End);//Scroll

}

**//++++++++Работа с портом++++++++++++**

В pro file

QT += serialport

**#include <QtSerialPort/QSerialPortInfo>**

//Получение списка доступных портов при помощи QSerialPortInfo

{QSerialPort port;

Port.setPort(info);

If (port.open(QIODevice::ReadWrite))

{qDebug()<<”Название”+info.portName()+” “+ info.description()+info.manufacturer();

}

}

// Закрытие соединения при выходе из формы

Delete ui;

Serial->close;

Delete serial;

**QList<QSerialPortInfo> serialPortInfo = QSerialPortInfo::availablePorts();**

**qDebug() << «Количество последовательных портов:» << serialPortInfo.count();**

**// Отображение списка скоростей передачи, поддерживаемых целевым последовательным портом**

**QList<qint32> baudRates = QSerialPortInfo::standardBaudRates();**

**qDebug() << baudRates;**

**qDebug() << «Описание последовательного порта:» << serialPortInfo.at(0).description();**

**qDebug() << "hasProductIdentifier(): " << serialPortInfo.at(0).hasProductIdentifier();**

**qDebug() << "hasVendorIdentifier(): " << serialPortInfo.at(0).hasVendorIdentifier();**

**qDebug() << "isBusy: " << serialPortInfo.at(0).isBusy();**

**qDebug() << "manufacturer: " << serialPortInfo.at(0).manufacturer();**

**qDebug() << "portName: " << serialPortInfo.at(0).portName();**

**qDebug() << "productIdentifier: " << serialPortInfo.at(0).productIdentifier();**

**qDebug() << "serialNumber: " << serialPortInfo.at(0).serialNumber();**

**qDebug() << "vendorIdentifier: " << serialPortInfo.at(0).vendorIdentifier();**

**qDebug() << "systemLocation: " << serialPortInfo.at(0).systemLocation();**

H файл

struct PortRT

{

QString name;

qint32 baudRate;

QSerialPort::DataBits dataBits;

QSerialPort::Parity parity;

QSerialPort::StopBits stopBits;

QSerialPort::FlowControl flowControl;

};

QSerialPort thisPort;

PortRT SettingsPort;

Cpp файл

Port \*PortNew = new Port();

//Запись всех доступных ком портов в комбобокс

void MainWindow::**on\_Btn\_Serch\_clicked**()

{

ui->PortNameBox->clear();

foreach (const QSerialPortInfo &info, QSerialPortInfo::availablePorts())

{

ui->PortNameBox->addItem(info.portName());

}

}

//Сигнал для записи параметров и послед. Отправки в ком порт

void MainWindow::**on\_BtnSave\_clicked**()

{

int baudrate = 9600;

int databits = 8;

int parity = 0; //DataToInt (QSerialPort::NoParity)

int stopbits = 1; //DataToInt (QSerialPort::OneStop)

int flowcontrol = 0; //DataToInt (QSerialPort::NoFlowControl)

savesettings(ui->PortNameBox->currentText(), baudrate, databits, parity, stopbits, flowcontrol);

}

//По кнопке коннект соединение с портом

void Port :: **ConnectPort**(void)//процедура подключения

{

thisPort.setPortName(SettingsPort.name);

if (thisPort.open(QIODevice::ReadWrite))

{

if (thisPort.setBaudRate(SettingsPort.baudRate)

&& thisPort.setDataBits(SettingsPort.dataBits)//DataBits

&& thisPort.setParity(SettingsPort.parity)

&& thisPort.setStopBits(SettingsPort.stopBits)

&& thisPort.setFlowControl(SettingsPort.flowControl))

{

if (thisPort.isOpen())

{

error\_((SettingsPort.name+ " >> Открыт!\r").toLocal8Bit());

}

} else

{

thisPort.close();

error\_(thisPort.errorString().toLocal8Bit());

}

}

else

{

thisPort.close();

error\_(thisPort.errorString().toLocal8Bit());

}

}

void Port::handleError(QSerialPort::SerialPortError error)//проверка ошибок при работе

{

if ( (thisPort.isOpen()) && (error == QSerialPort::ResourceError))

{

error\_(thisPort.errorString().toLocal8Bit());

DisconnectPort();

}

}

void Port::**DisconnectPort**()//Отключаем порт

{

if(thisPort.isOpen())

{

thisPort.close();

error\_(SettingsPort.name.toLocal8Bit() + " >> Закрыт!\r");

}

}

void Port :: **WriteToPort**(QByteArray data){//Запись данных в порт

if(thisPort.isOpen()){

thisPort.write(data);

}

}

void Port :: **ReadInPort**(){//Чтение данных из порта

QByteArray data;

data.append(thisPort.readAll());

outPort(data);

//((QString)(adr.toInt())).toLatin1().toHex()

}

void MainWindow::**on\_cEnterText\_returnPressed**()

{

QByteArray data; // Текстовая переменная

data = ui->cEnterText->text().toLocal8Bit().toHex() + '\r'; // Присвоение "data" значения из EnterText

writeData(data); // Отправка данных в порт

Print(data); // Вывод данных в консоль

}

void MainWindow::**Print**(QString data)

{

ui->consol->textCursor().insertText(data+'\r'); // Вывод текста в консоль

ui->consol->moveCursor(QTextCursor::End);//Scroll

}

Другой пример по компорту

void takePacketsFromComPort();

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

Это значит, что ты при приёме пакета с СОМ порта сразу попадаешь в эту функцию и в ней, чтобы получить пакет, можно написать это:

void MainWindow::takePacketsFromComPort()

{

       QString packets = serial->readAll();

}

А если с СОМ порта пакет придёт большой, то в один раз ты его не получишь а будешь получать 3-4 раза и тем самым переменная будет перезаписывать сама себя. В таком случае нужно комбинировать пакет. То-есть ты получаешь один паке, а вместо него приходит 3-4 пакета, тогда его нужно соединить в один пакет.  
Вместо этого:

void MainWindow::takePacketsFromComPort()

{

       QString packets = serial->readAll();

}

private:

    static bool m\_synchronized;

в .cpp файле пишем это:

bool MainWindow::m\_synchronized = false;

void MainWindow::takePacketsFromComPort()

{

**for**(;;)

    {

        const auto bytesAvailable = serial->bytesAvailable();

**if**(!m\_synchronized) {

**if**(bytesAvailable < 4)

                return;

            const auto head = serial->peek(4);

**if**(head != "FCD>") { //в head должна быть запись начала пакета. В моём случае начало пакета это FCD> у тебя будет другое

                serial->read(1);

                continue;

            } **else** {

                m\_synchronized = true;

            }

        } **else** {

            const auto probe = serial->peek(bytesAvailable);

            const auto tailIndex = probe.indexOf("**\r**");

**if**(tailIndex == -1)

                return;

            packet = serial->read(tailIndex + 2);

            qDebug() << "Ответ записи значений: " << packet;

            m\_synchronized = false;

**if**(tailIndex != -1 && probe.indexOf("**\r**"))

            {

                serial->close();

            }

        }

    }

}

**// +++++++++++Таймер+++++++++**

#include <QTimer>

QTimer \*timer;

Timer = new QTimer();

Timer->start(2000); //в скобках частота срабатывания timeout сигнала в мс.

**//++++++++++Структуры** struct **и списки List+++++**

struct **Settings**

{

QString name;

qint32 baudRate;

QSerialPort::DataBits dataBits;

QSerialPort::Parity parity;

QSerialPort::StopBits stopBits;

QSerialPort::FlowControl flowControl;

};

Settings SettingsPort;

QList<Settings> ListOfSettings;

int set\_number;

struct **mode** {

int number = 0;

float u1, u2, i1, i2;

int timer = 1000;

};

QList<mode> ListOfModes;

int mode\_number;

mode\_number = 0;

mode m;

m.number = 1;

m.i1 = 100;

m.i2 = 200;

m.u1 = 120;

m.u2 = 220;

m.timer = 5000;

ListOfModes.append(m);

m.number = 2;

m.i1 = 110;

m.i2 = 210;

m.u1 = 130;

m.u2 = 230;

m.timer = 1000;

ListOfModes.append(m);

m.number = 3;

m.i1 = 310;

m.i2 = 310;

m.u1 = 330;

m.u2 = 330;

m.timer = 5000;

ListOfModes.append(m);

m.number = 4;

m.i1 = 10;

m.i2 = 10;

m.u1 = 30;

m.u2 = 30;

m.timer = 2000;

ListOfModes.append(m);

connect(timer,SIGNAL(timeout()),this,SLOT(NextMode()));

void MainWindow::**NextMode**()

{

if(mode\_number >= ListOfModes.size())

{

ui->btntimer->setEnabled(true);

return;

}

MainWindow::ui->lbl\_etap->setText(QString::number(ListOfModes.at(mode\_number).number));

MainWindow::ui->lbl\_u1->setText(QString::number(ListOfModes.at(mode\_number).u1));

MainWindow::ui->lbl\_u2->setText(QString::number(ListOfModes.at(mode\_number).u2));

MainWindow::ui->lbl\_i1->setText(QString::number(ListOfModes.at(mode\_number).i1));

MainWindow::ui->lbl\_i2->setText(QString::number(ListOfModes.at(mode\_number).i2));

ui->progressBar->setValue(ui->progressBar->value()+1) ;

timer->start(ListOfModes.at(mode\_number).timer);

mode\_number++;

}

**//++++++Расчет контрольной суммы+++++++**

unsigned short **Crc16**(unsigned char \*pcBlock, unsigned short len)

{

const unsigned short Crc16Table[256] = {

0x0000, 0x1021, 0x2042, 0x3063, 0x4084, 0x50A5, 0x60C6, 0x70E7,

0x8108, 0x9129, 0xA14A, 0xB16B, 0xC18C, 0xD1AD, 0xE1CE, 0xF1EF,

0x1231, 0x0210, 0x3273, 0x2252, 0x52B5, 0x4294, 0x72F7, 0x62D6,

0x9339, 0x8318, 0xB37B, 0xA35A, 0xD3BD, 0xC39C, 0xF3FF, 0xE3DE,

0x2462, 0x3443, 0x0420, 0x1401, 0x64E6, 0x74C7, 0x44A4, 0x5485,

0xA56A, 0xB54B, 0x8528, 0x9509, 0xE5EE, 0xF5CF, 0xC5AC, 0xD58D,

0x3653, 0x2672, 0x1611, 0x0630, 0x76D7, 0x66F6, 0x5695, 0x46B4,

0xB75B, 0xA77A, 0x9719, 0x8738, 0xF7DF, 0xE7FE, 0xD79D, 0xC7BC,

0x48C4, 0x58E5, 0x6886, 0x78A7, 0x0840, 0x1861, 0x2802, 0x3823,

0xC9CC, 0xD9ED, 0xE98E, 0xF9AF, 0x8948, 0x9969, 0xA90A, 0xB92B,

0x5AF5, 0x4AD4, 0x7AB7, 0x6A96, 0x1A71, 0x0A50, 0x3A33, 0x2A12,

0xDBFD, 0xCBDC, 0xFBBF, 0xEB9E, 0x9B79, 0x8B58, 0xBB3B, 0xAB1A,

0x6CA6, 0x7C87, 0x4CE4, 0x5CC5, 0x2C22, 0x3C03, 0x0C60, 0x1C41,

0xEDAE, 0xFD8F, 0xCDEC, 0xDDCD, 0xAD2A, 0xBD0B, 0x8D68, 0x9D49,

0x7E97, 0x6EB6, 0x5ED5, 0x4EF4, 0x3E13, 0x2E32, 0x1E51, 0x0E70,

0xFF9F, 0xEFBE, 0xDFDD, 0xCFFC, 0xBF1B, 0xAF3A, 0x9F59, 0x8F78,

0x9188, 0x81A9, 0xB1CA, 0xA1EB, 0xD10C, 0xC12D, 0xF14E, 0xE16F,

0x1080, 0x00A1, 0x30C2, 0x20E3, 0x5004, 0x4025, 0x7046, 0x6067,

0x83B9, 0x9398, 0xA3FB, 0xB3DA, 0xC33D, 0xD31C, 0xE37F, 0xF35E,

0x02B1, 0x1290, 0x22F3, 0x32D2, 0x4235, 0x5214, 0x6277, 0x7256,

0xB5EA, 0xA5CB, 0x95A8, 0x8589, 0xF56E, 0xE54F, 0xD52C, 0xC50D,

0x34E2, 0x24C3, 0x14A0, 0x0481, 0x7466, 0x6447, 0x5424, 0x4405,

0xA7DB, 0xB7FA, 0x8799, 0x97B8, 0xE75F, 0xF77E, 0xC71D, 0xD73C,

0x26D3, 0x36F2, 0x0691, 0x16B0, 0x6657, 0x7676, 0x4615, 0x5634,

0xD94C, 0xC96D, 0xF90E, 0xE92F, 0x99C8, 0x89E9, 0xB98A, 0xA9AB,

0x5844, 0x4865, 0x7806, 0x6827, 0x18C0, 0x08E1, 0x3882, 0x28A3,

0xCB7D, 0xDB5C, 0xEB3F, 0xFB1E, 0x8BF9, 0x9BD8, 0xABBB, 0xBB9A,

0x4A75, 0x5A54, 0x6A37, 0x7A16, 0x0AF1, 0x1AD0, 0x2AB3, 0x3A92,

0xFD2E, 0xED0F, 0xDD6C, 0xCD4D, 0xBDAA, 0xAD8B, 0x9DE8, 0x8DC9,

0x7C26, 0x6C07, 0x5C64, 0x4C45, 0x3CA2, 0x2C83, 0x1CE0, 0x0CC1,

0xEF1F, 0xFF3E, 0xCF5D, 0xDF7C, 0xAF9B, 0xBFBA, 0x8FD9, 0x9FF8,

0x6E17, 0x7E36, 0x4E55, 0x5E74, 0x2E93, 0x3EB2, 0x0ED1, 0x1EF0

};

unsigned short crc = 0xFFFF;

while (len--)

crc = (crc << 8) ^ Crc16Table[(crc >> 8) ^ \*pcBlock++];

return crc;

}

unsigned char pcBlock[4];

pcBlock[0] = startByte;

pcBlock[1] = outAdr;

pcBlock[2] = inAdr;

pcBlock[3] = messType;

unsigned short len = 4;

unsigned short iCrc16 = Crc16(*pcBlock*,len); //

unsigned short data[5] = {startByte, outAdr,inAdr,messType,iCrc16};

unsigned char upper = Crc16(*pcBlock*,len)>>8; //получение старшего байта контрольной суммы

unsigned short lower1 = Crc16(*pcBlock*,len)<<8;

unsigned char lower = lower1>>8; //получение младшего байта контрольной суммы

// unsigned short full = (upper\*256)+lower; // получение общего значения контрольной суммы из старшего и младшего байтов

unsigned short full = (unsigned short) (upper<<8) | lower; // получение общего значения контрольной суммы из старшего и младшего байтов с помощью побитового сложения

//+++++++++Цикл For

for (int i=0;i<5;i++)

//+++++++++массивы

Размер массива а: sizeof(a); или std::size(a);

Указатель на первый элeменет std::begin(a); последний - std::end(a);

//+++++++++++Ссылки

**int** x;

**int** &y = x;

**int** z = y;

Ссылки на массив

**void** **f** (**int** (&x)[5])

{

// sizeof (x) здесь равен 5 \* sizeof (int)

}

**int** **main** (**void**)

{

**int** x[5];

f (x); // OK

f (x + 0); // Нельзя

**int** y[7];

f (y); // Нельзя, не тот размер

}

//++++++++++Указатели

**int** x;

**int** \*y = &x; // От любой переменной можно взять адрес при помощи операции взятия адреса "&". Эта операция возвращает указатель

**int** z = \*y; // Указатель можно разыменовать при помощи операции разыменовывания "\*". Это операция возвращает тот объект, на который указывает указатель

Передача в функцию:

**int** (\*a)[2]; // Это указатель на массив. Самый настоящий. Он имеет тип int (\*TYPE)[2]

**int** b[2];

**int** \*c = b; // Это не указатель на массив. Это просто указатель. Указатель на первый элемент некоего массива

**int** \*d = **new** **int**[4]; // И это не указатель на массив. Это указатель

//+++++++++++Примеры гет и сет функций

class MainWindow : public QMainWindow{

public:

void setAge(int age){

\_ui->ageInput->setValue(age);

}

int age() const{

return \_ui->ageInput->value();

}

void setName(const QString &name){

\_ui->nameInput->setText(name);

}

QString name() const{

\_ui->nameInput->text();

}

private:

Ui::MainWindow \*\_ui;

};

**Qt + MVP + QThread. Строим свой велосипед**

[**https://habr.com/ru/post/246445/**](https://habr.com/ru/post/246445/)

**++++++++++++++QByteArray**

**databuf = QByteArray(reinterpret\_cast<char\*>(buf), 10); //перевод из чар в QByteArray**

**databuf = QByteArray((char\*)buf, 10);**

**----------------------------------------**

**QByteArray data;**

**Data.reserve(10); //резервирует максимальный замер массива на 10 байт**

**Size – размер фактический массива**

**Capacity – максимально возможный размер**

**---------------------------**

**Внести в QByteArray стринг строку**

**QByteArray data (QString(“Bill”).toLocal8Bit());**

**------------------------------**

**Data.resize – изменение размера**

**Truncate(n) – обрезает массив до размера n**

**Data.clear – очищает массив**

**https://www.youtube.com/watch?v=IDh022l5A40**

**+++++++++++++++++++++работа с таблицами**

[**https://www.youtube.com/watch?v=AUCP3OSkxg8**](https://www.youtube.com/watch?v=AUCP3OSkxg8)

ui->tblRT->setRowCount(10); //задание количества строк таблицы

ui->tblRT->setColumnCount(3); //задание количества колонок таблицы

ui->tblRT->setHorizontalHeaderLabels(QStringList()<<"МК1"<<"МК2"<<"МК3"); //заполнение заголовков столбцов

ui->tblRT->setVerticalHeaderLabels(QStringList()<<"Ток 2 на выходе СБ2"<<"Ток 1 на выходе СБ2"<<"Ток 2 на выходе СБ1" //заполнение заголовков строк

<<"Ток 1 на выходе СБ1"<<"Ток нагрузки 2 моноблока"<<"Ток нагрузки 1 моноблока"<<"Температура 2 корпуса прибора"

<<"Температура 1 корпуса прибора"<<"Состояние РТ СБ2"<<"Состояние РТ СБ1");

// QTableWidgetItem \*itm1\_2 = new QTableWidgetItem(tr("%1").arg(5)); //создание итема таблицы для заполнения

// ui->tblRT->setItem(1,2,itm1\_2); //заполнение указанной ячейки (строки, столбцы,итем для заполнения)

// itm1\_2->setTextColor(Qt::red); //задание цвета у текста определенной ячейки таблицы

// itm1\_2->setBackgroundColor(Qt::black); //задание цвета самой определенной ячейки таблицы

**+++++++++++++++Работа с ini файлами и настройками проекта**

#include <QSettings>

void MainWindow::**LoadSettings**()

{

QString dis, serial;

QSettings setting("ports.ini", QSettings::IniFormat); //ports.ini файл должен быть в одной папке с exe

setting.beginGroup("MK1-osn");// [MK1-osn] в ини файле

QString status = setting.value("work","0").toString();

if ( status == "on") {

dis = setting.value("description","0").toString();

serial = setting.value("serialNumber","0").toString();

}

setting.endGroup();

setting.beginGroup("MK2-osn");// [MK2-osn] в ини файле

QString status1 = setting.value("work","0").toString();

if ( status1 == "on"){

dis = setting.value("description","0").toString();

serial = setting.value("serialNumber","0").toString();

}

setting.endGroup();

QString name = getPortName(dis,serial);

emit savesettings(name, baudrate, databits, parity, stopbits, flowcontrol);

}

**------Создание экзешника со всеми длл в одной папке-----**

Проекты – Сборка – Сборка, этапы – добавить этап

Нужно создать два особых этапа обработки

1. Команда: xcopy

Параметры: "%{ActiveProject:RunConfig:Executable:NativeFilePath}" "..\Work\" /y /i

Рабочий каталог: каталог с нашей прогой

1. Команда: windeployqt.exe

Параметры: --no-translations --no-system-d3d-compiler --no-opengl-sw --no-angle "..\Work\%{ActiveProject:RunConfig:Executable:FileName}"

Рабочий каталог: каталог с нашей прогой

**------------------Добавление картинок------------**

#include <QPixmap>

QPixmap pix("D:/GitHub/KPA\_BSWV/redbtn.png"); //указание расположения картинки и создание объекта класса

ui->redLbl->setPixmap(pix.scaled(45,45,Qt::KeepAspectRatio)); //присвоение лейблу этой картинки с уменьшением ее размеров

**----------------------Работа с чекбоксом-----------**

void MainWindow::**AcpVisible**()

{

if (ui->cbAcp->isChecked()){

ui->tblAcp->*setVisible*(true);

setMinimumSize(1132,725);

resize(972,613);

}

else {

ui->tblAcp->*setVisible*(false);

setMinimumSize(1132,382);

resize(972,321);

}

}

**Escape-последовательности**

\r возврат каретки в начало строки

\n новая строка

\t горизонтальная табуляция

\v вертикальная табуляция

\» двойные кавычки

\’ апостроф

\\ обратный слеш

\0 нулевой символ

\? знак вопроса

\a сигнал бипера (спикера) компьютера

**Continue – позволяет пропустить итерацию цикла**

**Static переменные – общие переменные для всех экземпляров класса и обратившись и изменив из одного экземпляра эта переменная поменяется в другом**

#include <conio.h>

kbhit() //нажатие какой либо кнопки на клаве (можно в проге завершать выполнение по нажатию (!kbhit())

**---------------------LCard и работа с библиотеками-----------------**

**Скелет работы программы**

#include #include #include "Lusbapi.h" // заголовочный файл библиотеки Lusbapi

ILE140 \*pModule; // указатель на интерфейс модуля

MODULE\_DESCRIPTION\_E140 md; // структура c информацией о модуле

char ModuleName[7]; // название модуля

BYTE UsbSpeed; // скорость работы шины USB

**int main(void)**

**{**

**// проверим версию DLL библиотеки**

**if(GetDllVersion() != CURRENT\_VERSION\_LUSBAPI)**

**{**

**printf("Неправильная версия Dll!");**

**return 1; //выйдем из программы с ошибкой**

**}**

**// получим указатель на интерфейс модуля**

**pModule = static\_cast<ILE140 \*>(CreateLInstance("e140"));**

**if(!pModule)**

**{**

**printf("Не могу получить указатель на интерфейс");**

**return 1; //выйдем из программы с ошибкой**

**}**

**// попробуем обнаружить какой-нибудь модуль**

**// в нулевом виртуальном слоте**

**if(!pModule->OpenLDevice(0))**

**{**

**printf("Не могу получить доступ к модулю!");**

**return 1; //выйдем из программы с ошибкой**

**}**

**// попробуем получить скорость работы шины USB**

**if(!pModule->GetUsbSpeed(&UsbSpeed))**

**{**

**printf("Не могу узнать скорость работы USB!\n");**

**return 1; //выйдем из программы с ошибкой**

**}**

**// прочитаем название модуля в нулевом виртуальном слоте**

**if(!pModule->GetModuleName(ModuleName))**

**{**

**printf("Не могу прочитать название модуля!\n");**

**return 1; //выйдем из программы с ошибкой**

**}**

**// проверим: этот модуль - 'E14-140'?**

**if(strcmp(ModuleName, "E140"))**

**{**

**printf(" В нулевом виртуальном слоте не 'E14-140'\n");**

**return 1; //выйдем из программы с ошибкой**

**}**

**// попробуем прочитать информацию о модуле**

**if(!pModule->GET\_MODULE\_DESCRIPTION(&md))**

**{**

**printf("Не выполнена функция GET\_MODULE\_DESCRIPTION ()!");**

**return 1; //выйдем из программы с ошибкой**

**}**

**printf("Модуль E14-140 (серийный номер %s) полностью готов к\**

**работе!", md.Module.SerialNumber);**

**// далее можно располагать функции для непосредственного**

**// управления модулем**

**. . . . . .**

**// завершим работу с модулем**

**if(!pModule->ReleaseLInstance())**

**{**

**printf("Не выполнена функция ReleaseLInstance()!");**

**return 1; //выйдем из программы с ошибкой**

**}**

**// выйдем из программы**

**return 0;**

**}**

TtlOut = 0b0; //инициализация состояния выходных линий по умолчанию

TtlOut =0b1111111111111111; //для примера присвоение логических единиц на все внешние цифровые линии

pModule->*TTL\_OUT*(TtlOut);