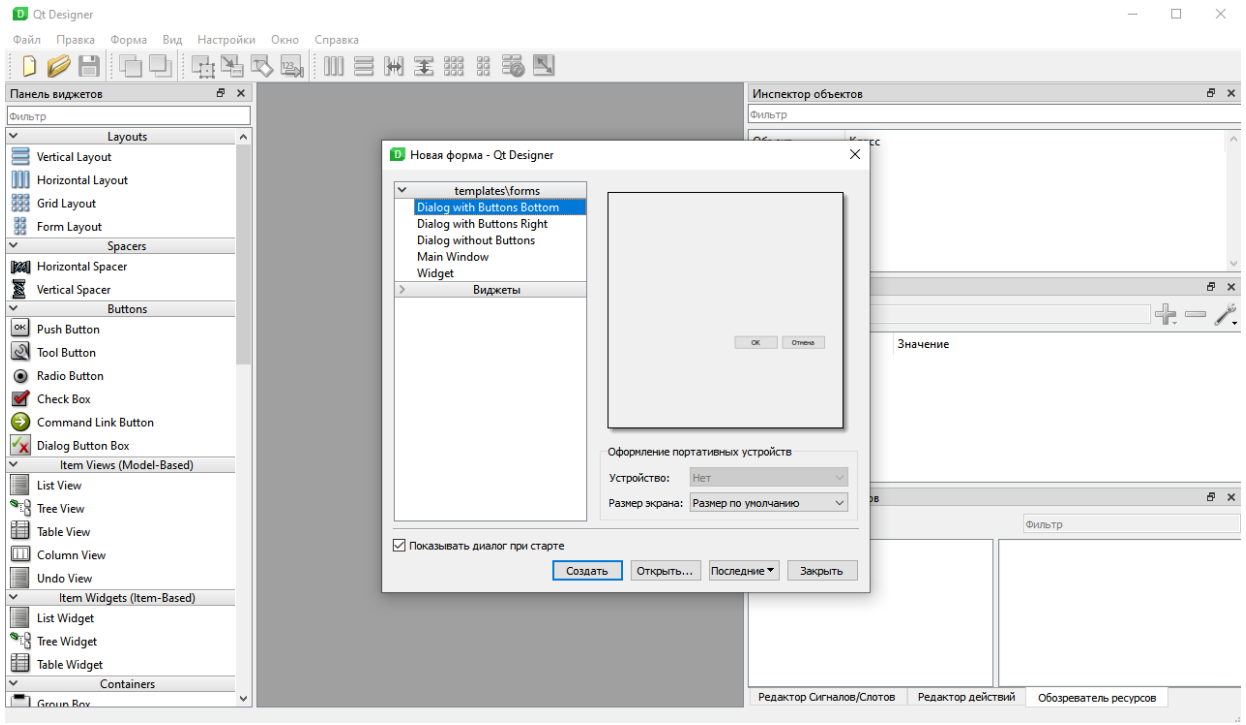
Работа в QtDesigner.

Лекция + практика

Что такое QtDesigner

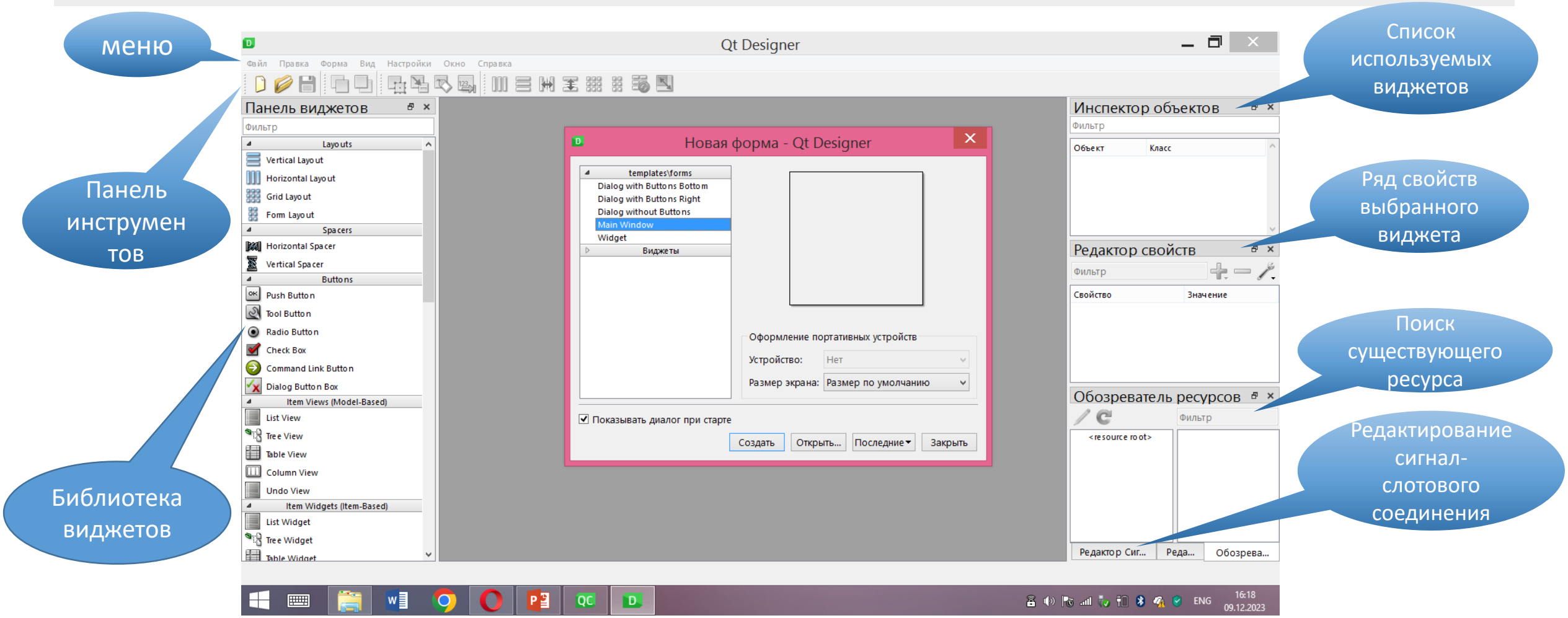
Программа Qt Designer - это средство быстрой разработки приложений (*Rapid Application Development*, RAD). Прежде всего, этот инструмент предназначен для дизайнеров, и принцип его работы отвечает принципу **WYSIWYG** (*What You See is What You Get*, «что видишь, то и получишь»). Он предоставляет возможность быстро создавать прототипы приложений, которые базируются на диалоговых окнах, а также могут иметь главное окно, меню, строку состояния и панель инструментов. Созданные в программе Qt Designer файлы описания интерфейса можно конвертировать в исходный код на языке C++. Кроме виджетов, уже содержащихся в библиотеке Qt, программа Qt Designer может дополняться виджетами, созданными самим разработчиком .



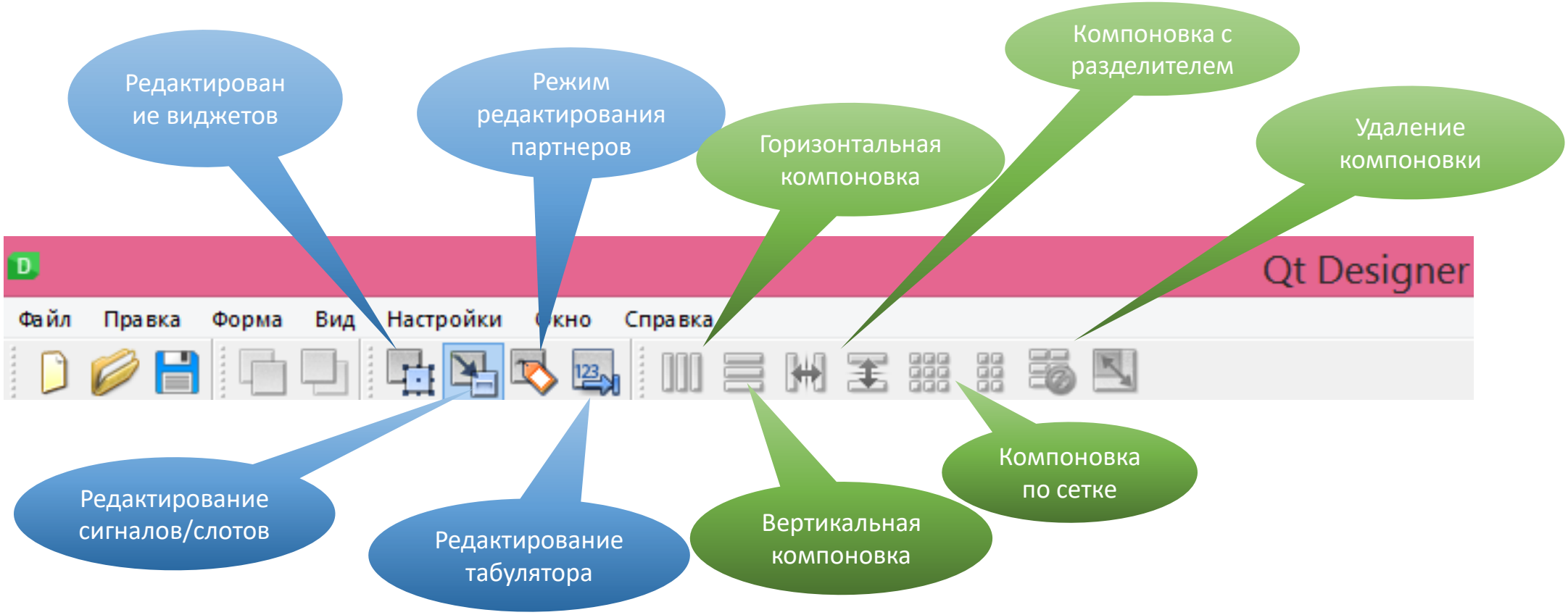
Запуск программы из Qt/[ver]/bin/designer.exe

Создание новой формы

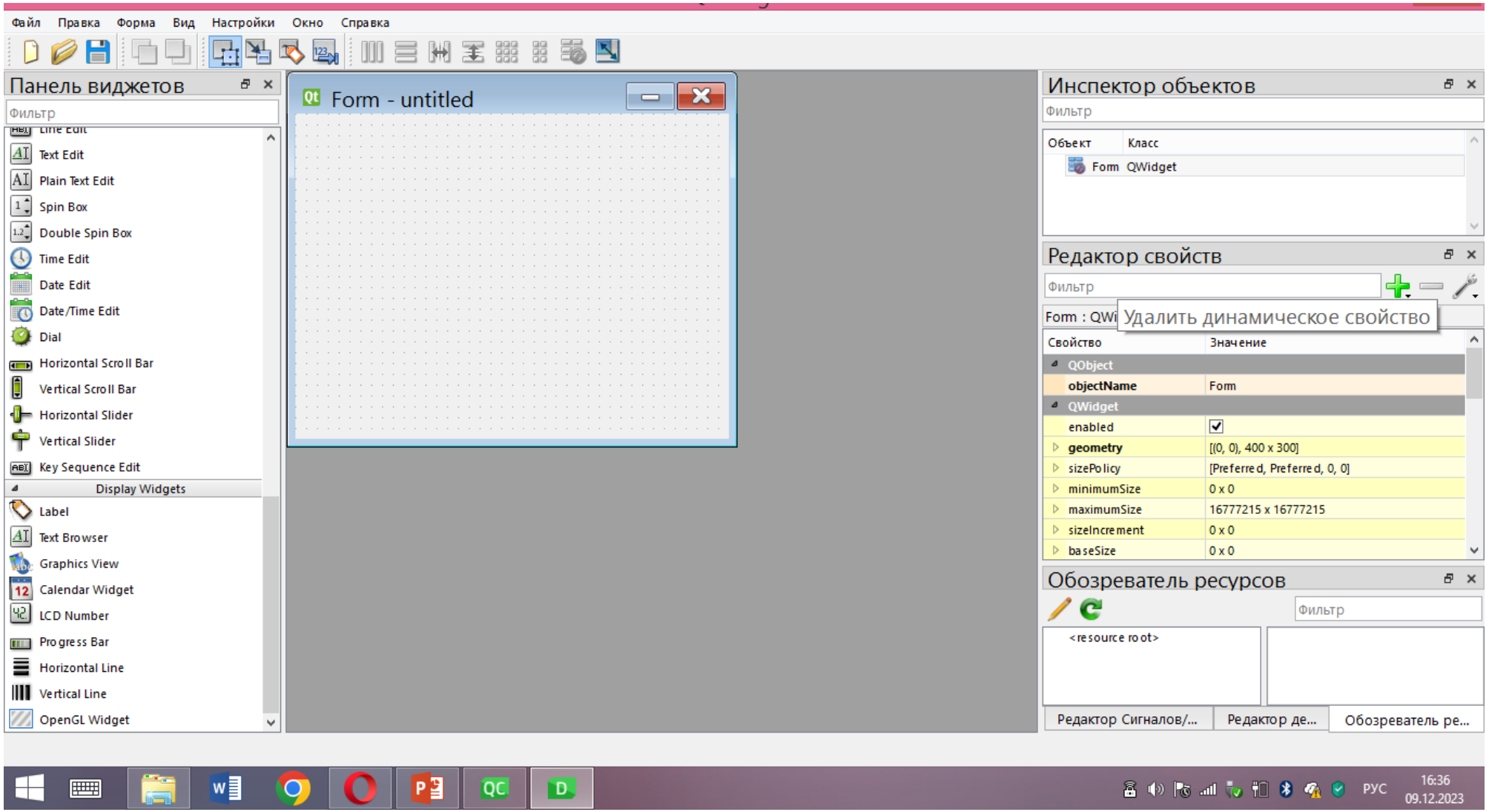
Окно программы Qt Designer содержит меню и панели инструментов. Панель инструментов предоставляет общие операции для редактирования формы: скопировать, вставить и т. д, а также и *режимы редактирования* и *размещения*.



Режимы редактирования и размещения

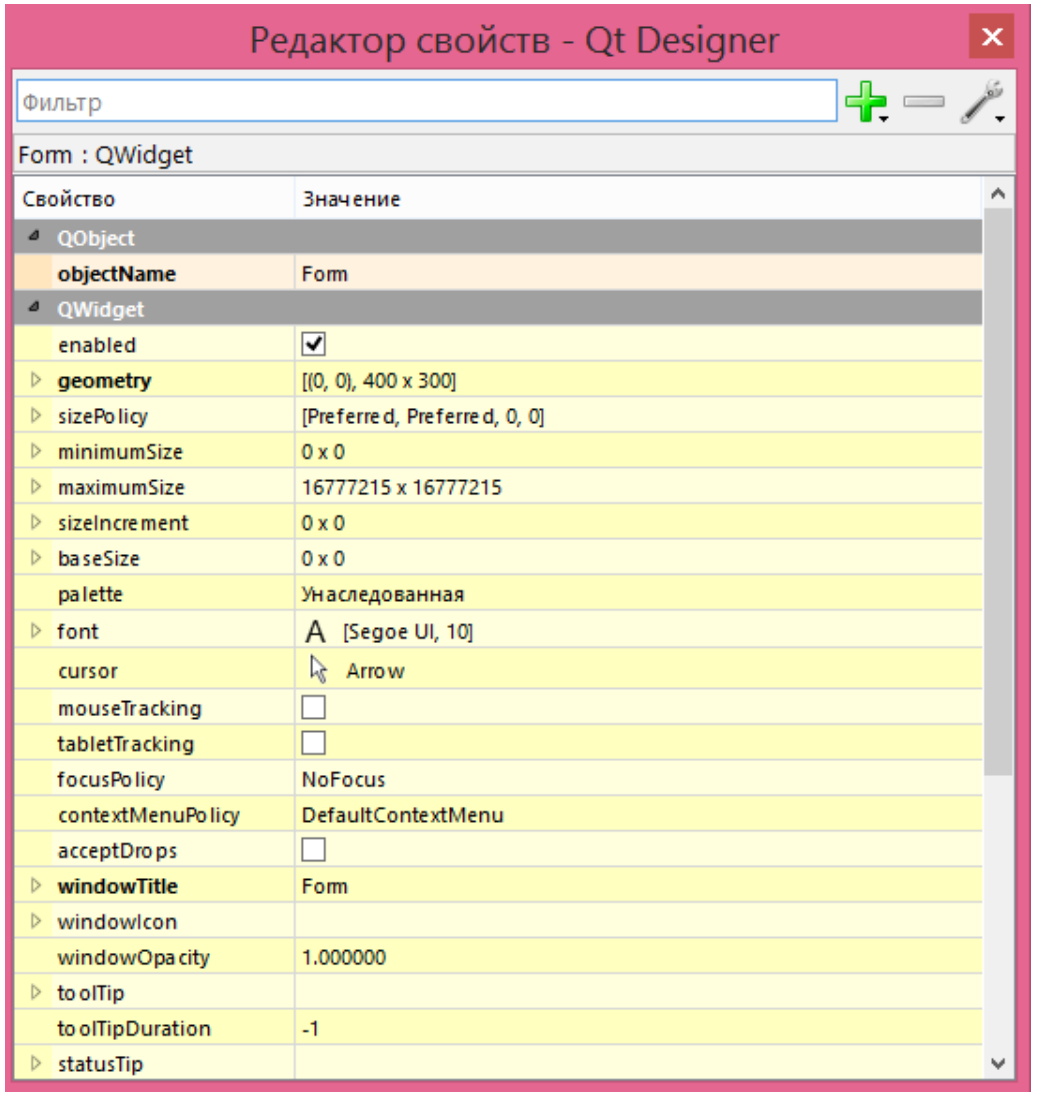


Окно виджета



Окно «Редактора свойств» виджета

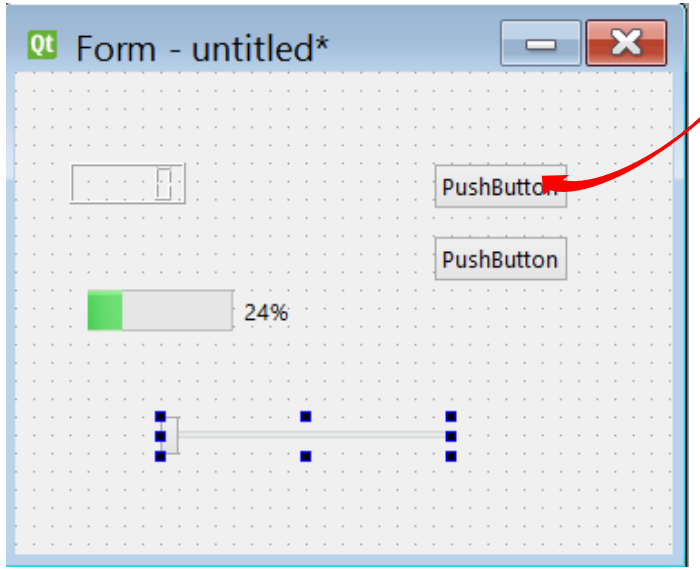
«Редактор свойств» отображает свойства выделенного виджета, расположенного на форме, а если ни один из виджетов не выделен, то отображаются свойства самой формы.

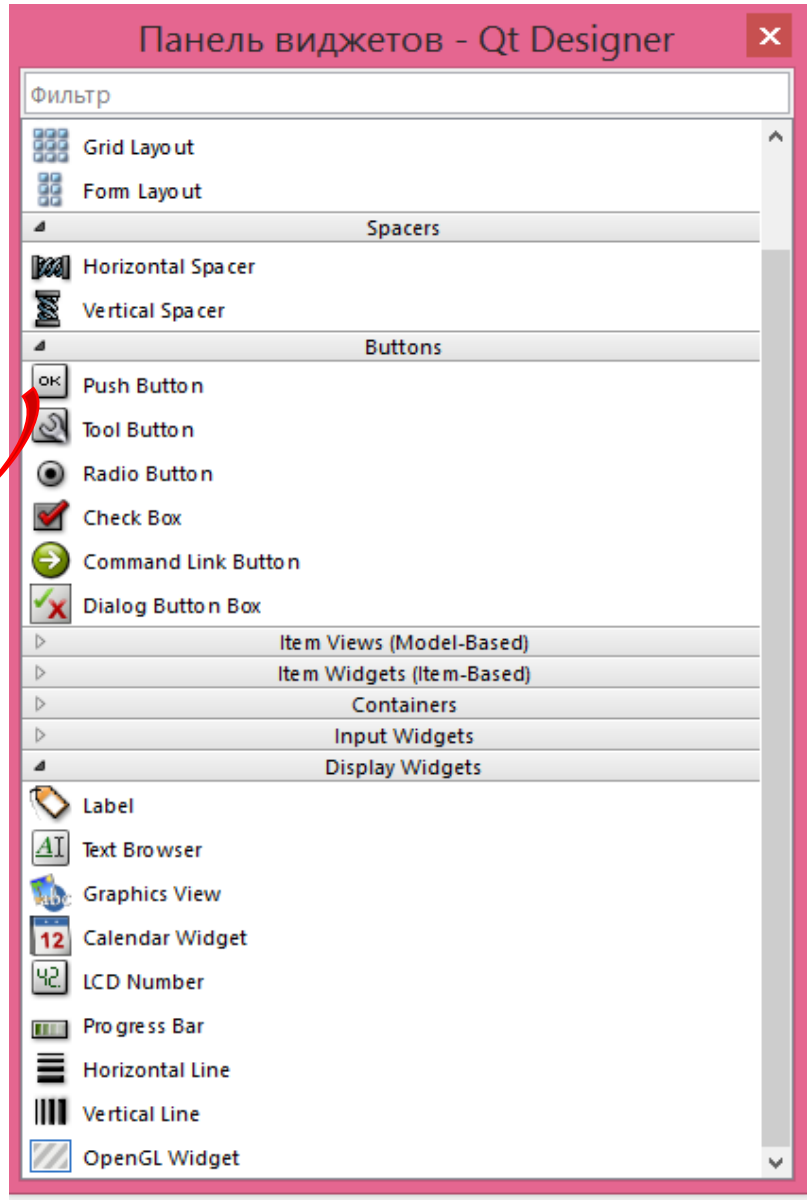


Добавление виджетов

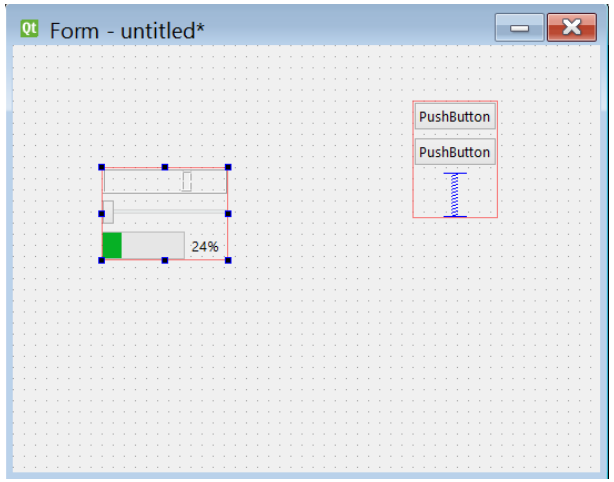
Чтобы добавить в диалоговое окно новые виджеты, нужно воспользоваться **панелью виджетов** *Widget Box* (Виджеты). Выберите нужный вам виджет, нажмите на нем левую кнопку мыши и перетащите его в область формы виджета окна на нужное вам место.

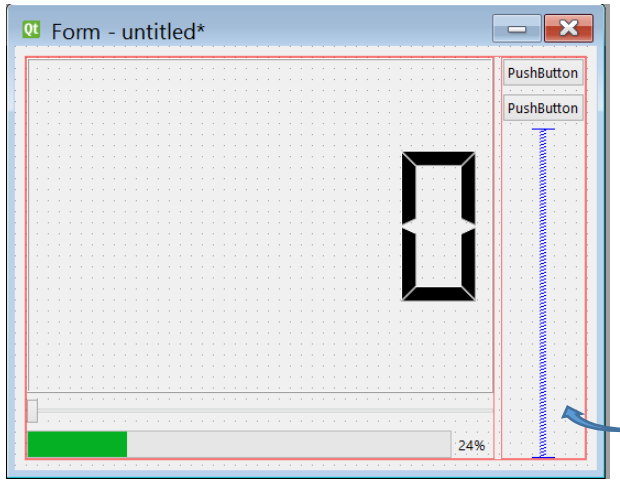
Выберите один из виджетов на форме (кнопку например), щелкнув на ней левой кнопкой мыши в окне Инспектор Объектов или в самом диалоговом окне введите в поле **text** Редактора свойств текст **&Reset**, а в поле **objectName** - **m _ pcmdReset**. Мы изменили надпись кнопки и её имя, которое будет использоваться в дальнейшем в программе.





Компоновка





В места, которые должны оставаться свободными, следует поместить «**заполнитель пространства**» (**Spacer**).

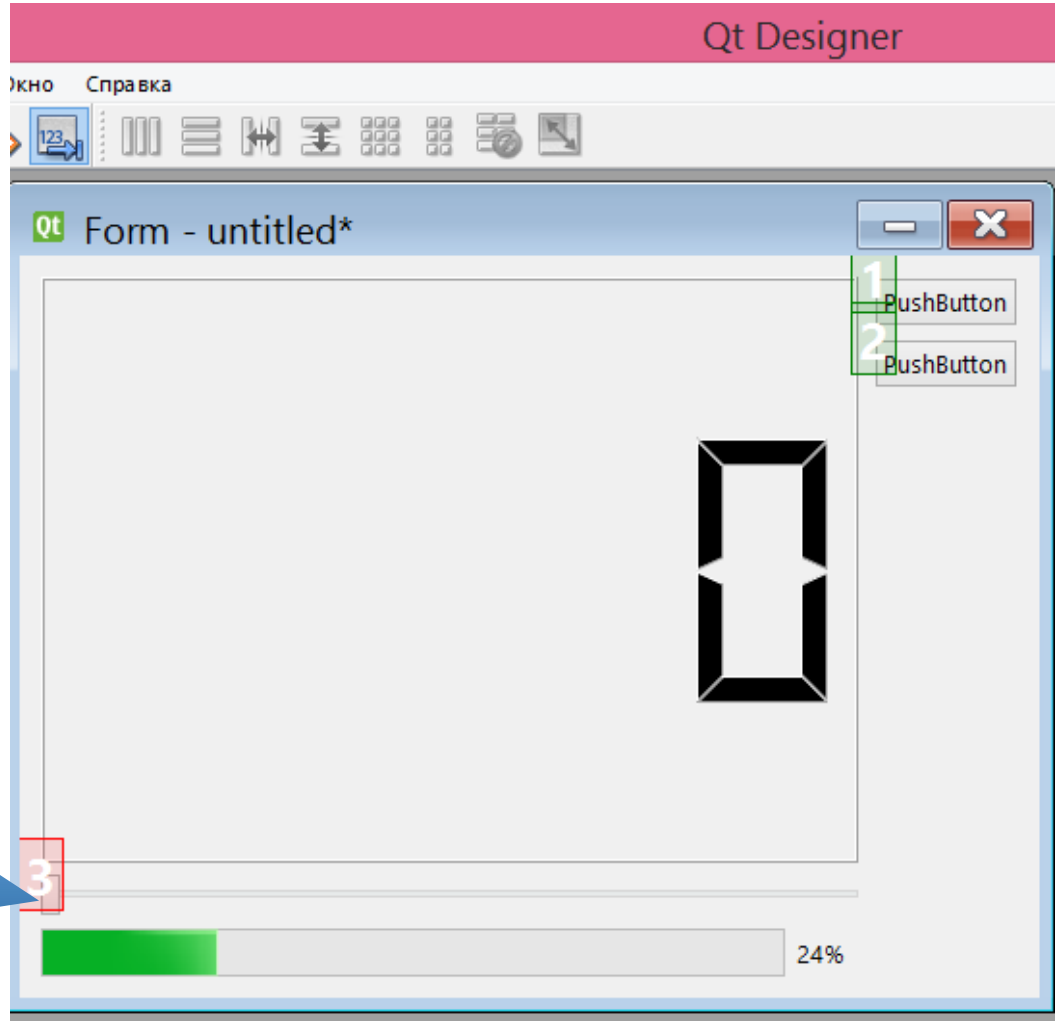
Добавив все нужные виджеты, займемся их размещением . Для этого поместите указатель мыши в область диалогового окна, нажмите левую кнопку и, не отпуская, переместите указатель - вы увидите рамку. Охватите этой рамкой виджет ползунка и виджет электронного индикатора. Отпустите левую кнопку мыши, и эти виджеты окажутся выделенными. Нажмите на выделенных элементах правую кнопку мыши, и вы увидите контекстное меню, в котором выберите команду **Компоновка -> Скомпоновать по вертикали** или просто нажмите комбинацию «горячих» клавиш **<Ctrl>+<2>.**

В завершении - поручите диалоговому окну размещать свои элементы в горизонтальном порядке. Для этого щелкните на одной из пустых областей диалогового окна указателем мыши или выберите опцию **My Form** в окне «Инспектор объектов».
Затем нажмите правую кнопку мыши и выберите из контекстного меню команду Компоновка -> Скомпоновать по горизонтали или нажмите комбинацию «горячих» клавиш <Ctrl>+<1>.

Табулятор

Порядок следования табулятора нужен для навигации по окну при помощи клавиатуры - нажатие на клавишу **<Tab>** перемещает фокус от одного виджета к другому. Для определения порядка следования табулятора нужно установить соответствующий режим, для чего выберите команду меню:
Правка -> Изменение порядка переключений

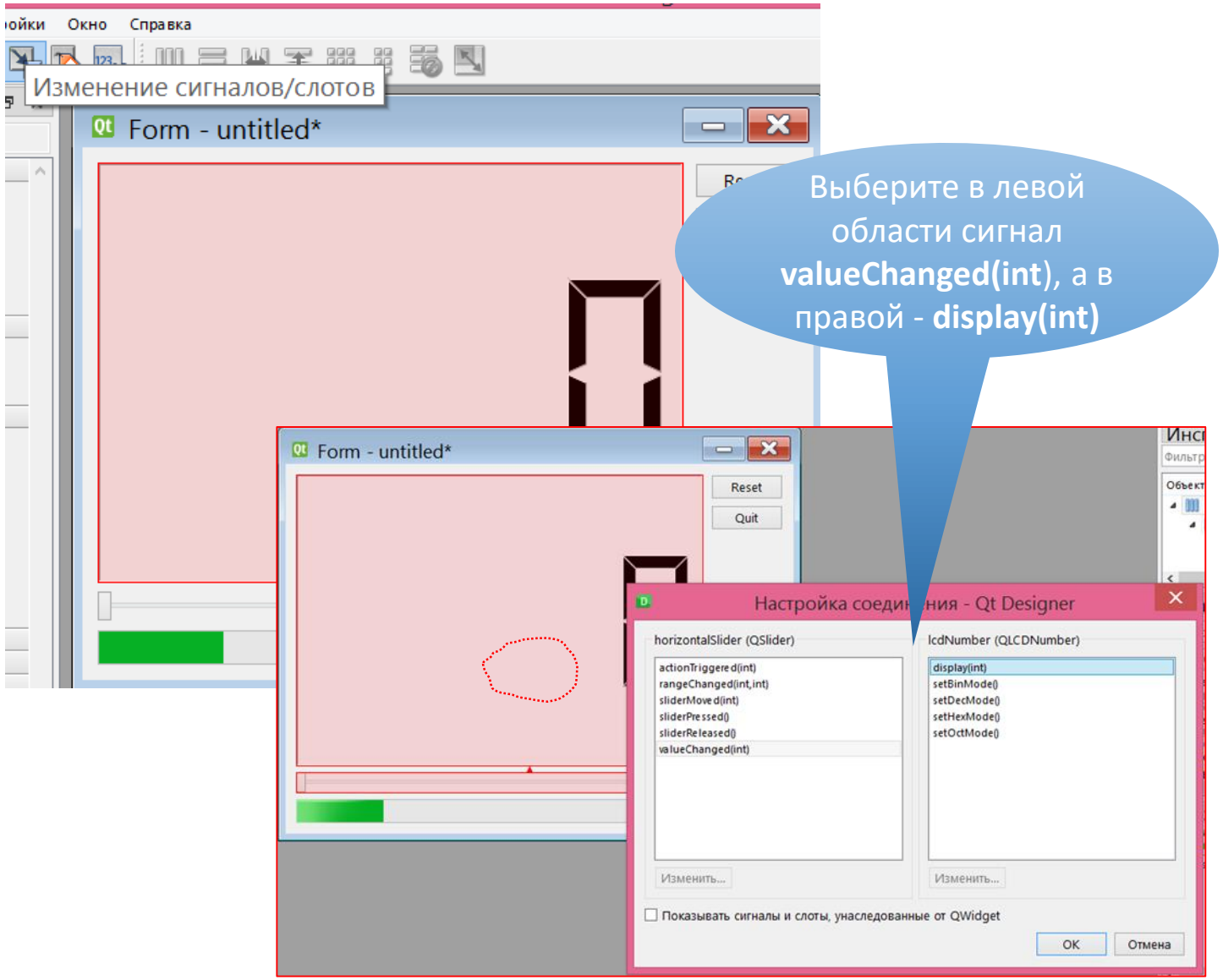
Нажимая указателем мыши на виджеты, обозначенные прямоугольниками, можно менять порядок следования табулятора.



Сигналы и слоты

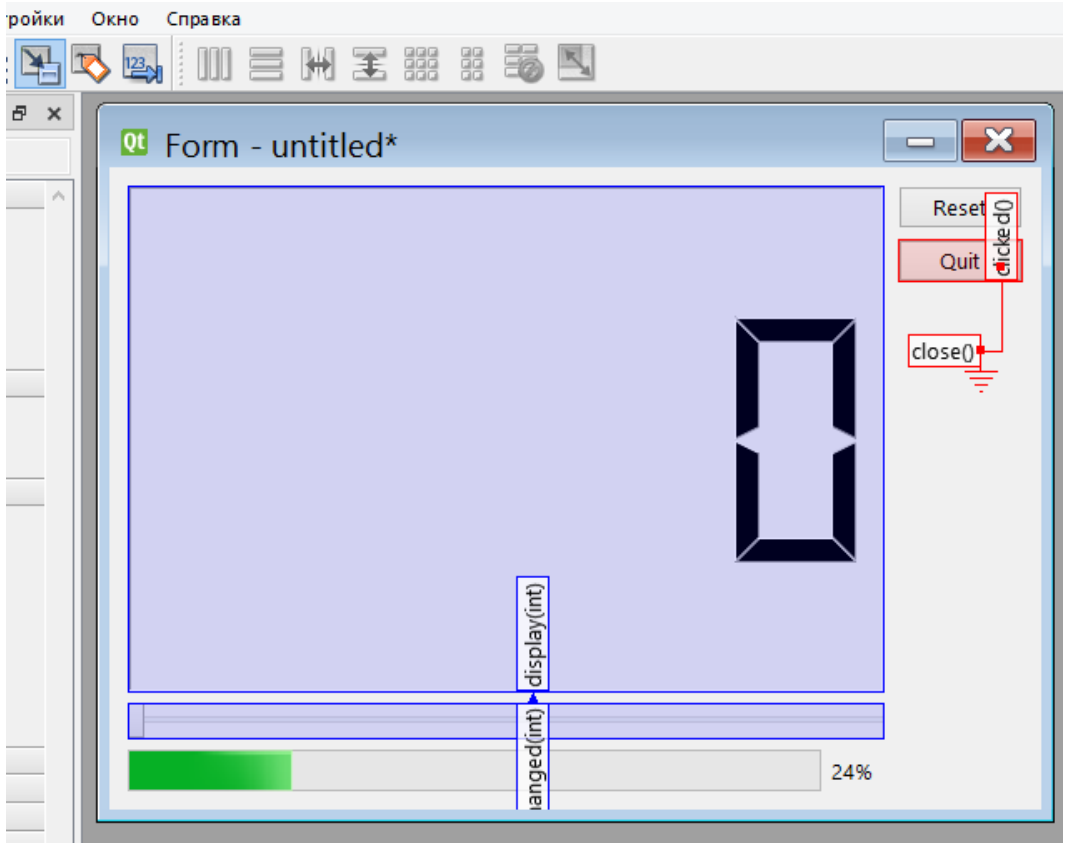
Для соединения **сигналов** одного виджета со **слотами** другого нужно установить соответствующий режим, для чего выберите команду меню:
Правка ->Изменение сигналов/слотов или нажмите клавишу <F4>.
Чтобы выполнить соединение, просто задержите указатель мыши на нужном вам элементе до тех пор, пока он не будет выделен, а затем нажмите левую кнопку и переместитесь к тому элементу, с которым должно быть осуществлено соединение. Вы увидите красную стрелку, показывающую в его сторону.

После отпускания кнопки мыши будет показано диалоговое окно «**Настройка соединения**», предлагающее выбрать сигналы и слоты для связываемых виджетов.



Сигналы и слоты

Чтобы полюбоваться на созданный нами виджет в действии, выберите команду меню:
Форма -> Препросмотр .. . . Теперь вы можете поэкспериментировать с изменениями положения ползунка, которые приведут к изменению информации, отображаемой виджетом электронного индикатора. Можете проверить установленный порядок следования табулятора и проследить за изменениями размеров виджетов, находящихся в компоновщике, при изменении размеров основного окна.

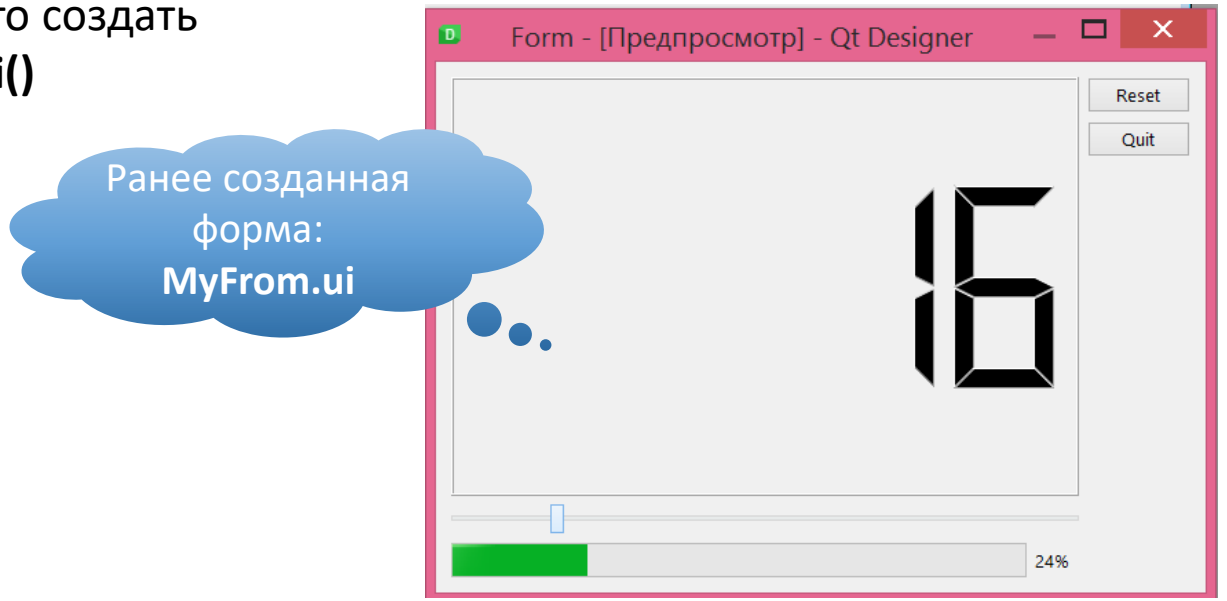


Если вы остались довольны просмотром виджета, то вам остается только сохранить его. Для этого нужно выбрать в меню команду:
Файл -> Сохранить как ...
Затем выбрать в диалоговом окне путь для сохранения формы и задать ее имя - например, «**MyFrom.ui**».

Использование ранее созданных форм

Первый способ называют прямым (*direct approach*) - он является и самым простым. Все, что нам нужно, - это создать виджет и установить в методе: **Ui::MyForm::setupUi()** созданную ранее форму.

```
#include "ui_MyForm.h"
#include <QtWidgets>
int main(int argc, char* argv[]) {
    QApplication app(argc, argv);
    QWidget* form = new QWidget;
    Ui::MyForm ui;
    ui.setupUi(form);
    form->show () ;
  return app.exec();
}
```



Если в вашей форме есть, например, виджеты, которые нуждаются в соединении со слотами, реализованными другими классами проекта, то этот способ - не для вас. Для нашего конкретного случая он не подходит, потому что нам нужно реализовать слот, с которым будет соединена кнопка **Reset** ().

Использование ранее созданных форм

Второй способ реализуется при помощи наследования (*inheritance approach*). В этом случае мы наследуем класс от класса, за базу которого была взята наша форма, - в нашем примере это QWidget.

Здесь мы используем форму ui::MyForm в качестве атрибута нашего класса и устанавливаем ее в конструкторе с помощью метода Ui:: MyForm::setupUi(). Затем соединяем кнопку Reset (Сброс) (указатель m_cmReset) с реализованным нами слотом *slotReset()*. Если бы мы вдруг пришли к выводу, что наша форма нуждается еще в паре элементов, то просто добавили бы их при помощи программы *Qt Designer*, после чего реализовали бы недостающие слоты в нашем классе. То есть, дизайн графического интерфейса отделен от программной реализации.

```
#include "ui_MyForm.h“

class MyForm : public QWidget {
    Q_OBJECT
private:
    Ui::MyForm m_ui;
public:
     MyForm(QWidget* pwgt = 0) : QWidget (pwgt)
     {
       m_ui.setupUi(this);
       connect(m_ui.m_cmdReset, SIGNAL(clicked()),
                                          SLOT(slotReset() ));
      }
public slots:
      void slotReset (){
            m_ui.m_sld->setValue(0);
            m_ui.m_lcd->display(0);
      }
};
```

Использование ранее созданных форм

Третий способ реализуется при помощи множественного наследования (*multiple inheritance approach*). По своей сути он очень похож на второй способ, но его достоинство заключается в том, что мы можем напрямую обращаться ко всем виджетам формы.

```
#include "ui_MyForm.h“

class MyForm : public QWidget, public Ui::MyForm {
        Q_OBJECT
public:
        MyForm(QWidget* pwgt = 0) :QWidget(pwgt){
                setupUi(this);
                connect(m_cmdReset, SIGNAL(clicked() ),
                                        SLOT(slotReset()));
        }
public slots:
        void slotReset (){
                m_sld->setValue(0);
                m_lcd->display(0);
        }
};
#endif //_MyForm_h_
```

Компиляция

Первым делом надо создать проектный файл и не забыть включить все используемые формы в секции FORMS. В нашем случае она только одна, и pro-файл должен выглядеть следующим образом:

```
TEMPLATE   =app
HEADERS      +=MyForm.h
FORMS         +=MyForm.ui
SOURCES      +=MyForm.cpp
QT                +=widgets
win32:TARGET   =../MyForm
```

После этого в каталогах, содержащих проектный файл, файл формы (MyForm.ui) и другие исходные файлы, выполните команды создания make-файла и компиляции:

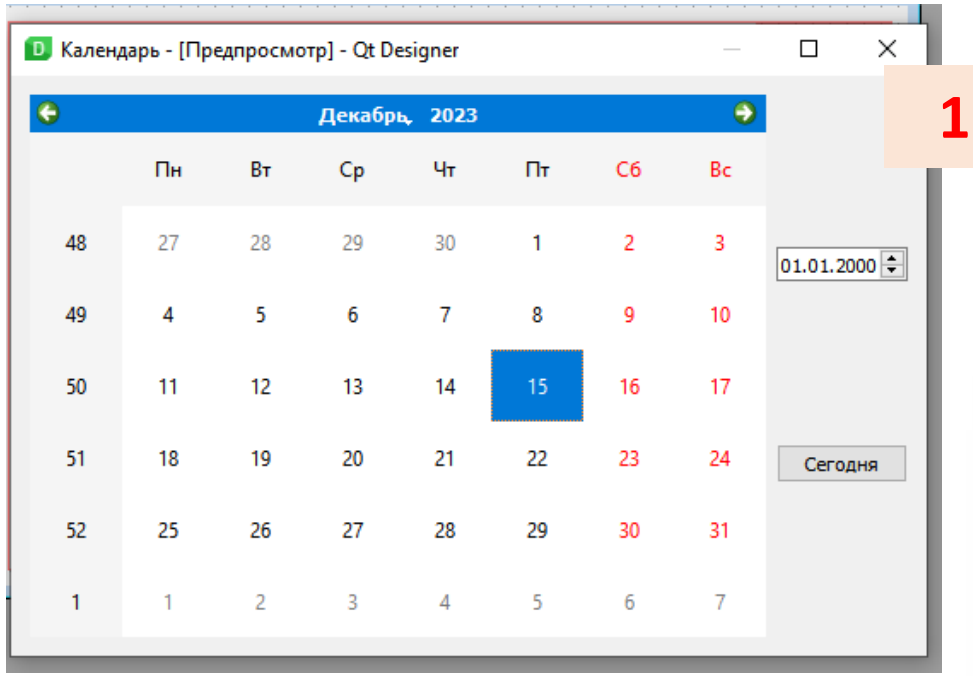
qmake

make

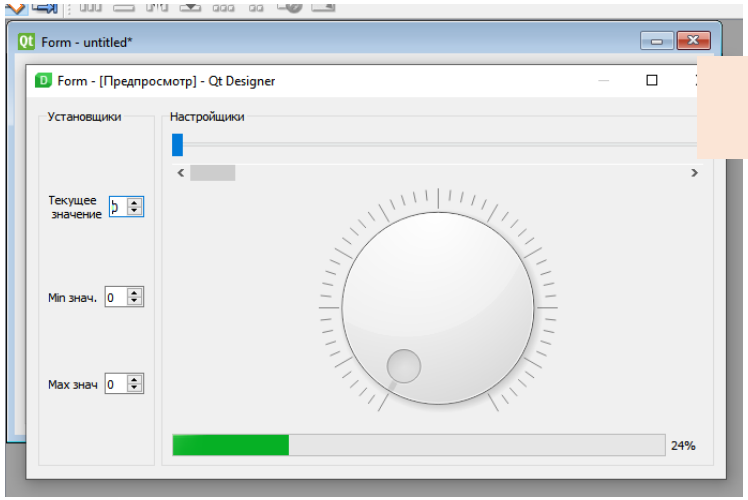
В процессе компиляции из ui-файла будет автоматически создан h-файл с префиксом ui_.

По завершении компилирования будет создана исполняемая программа, отображающая
окно.

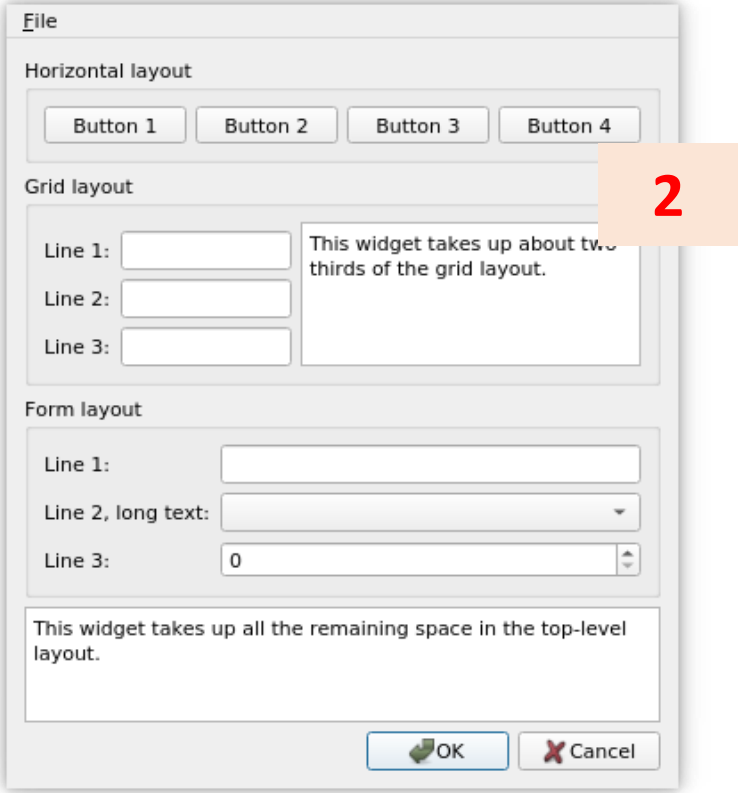
Практика.



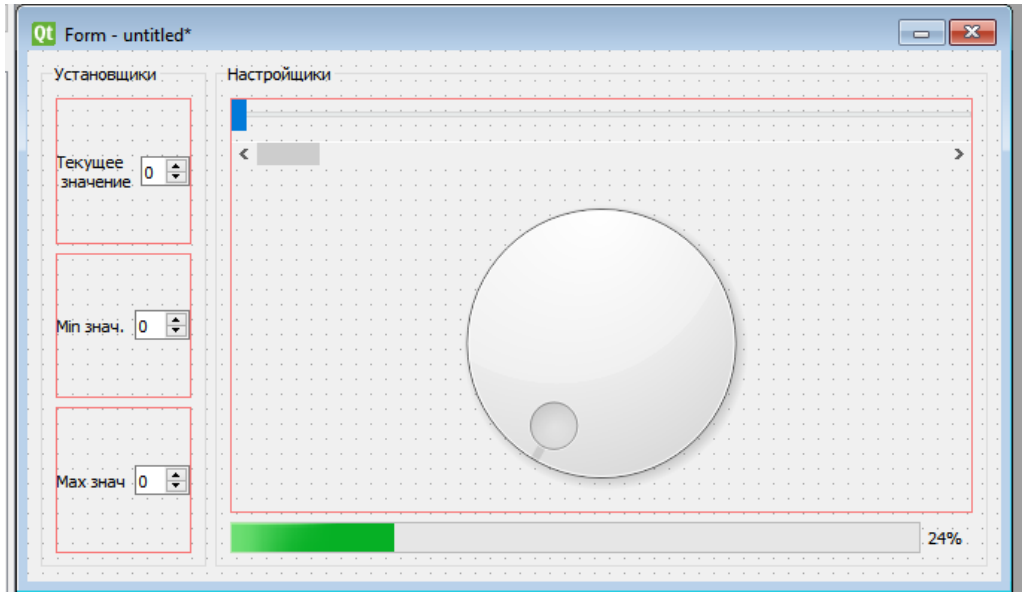
1

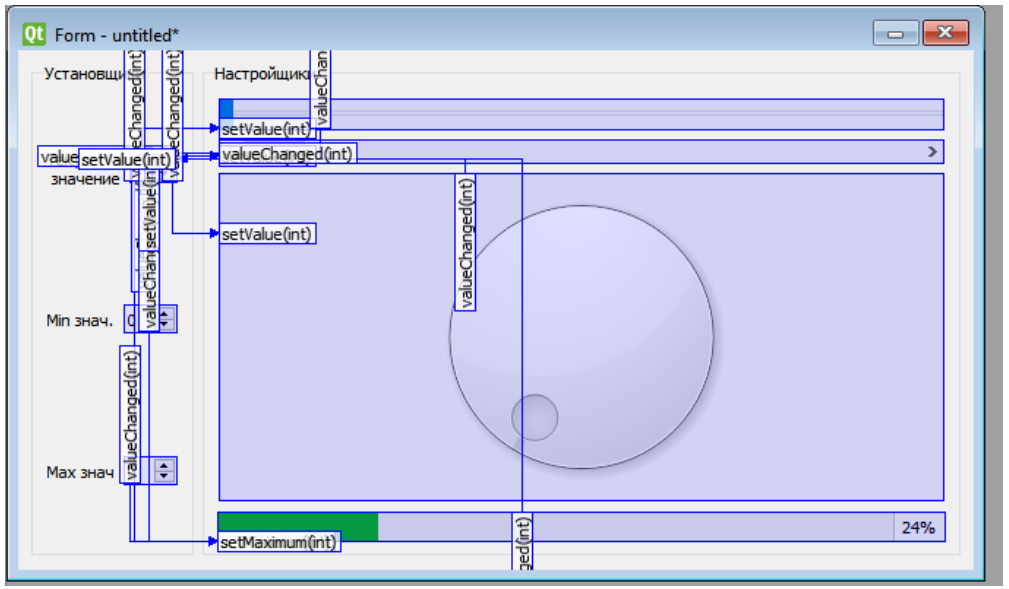


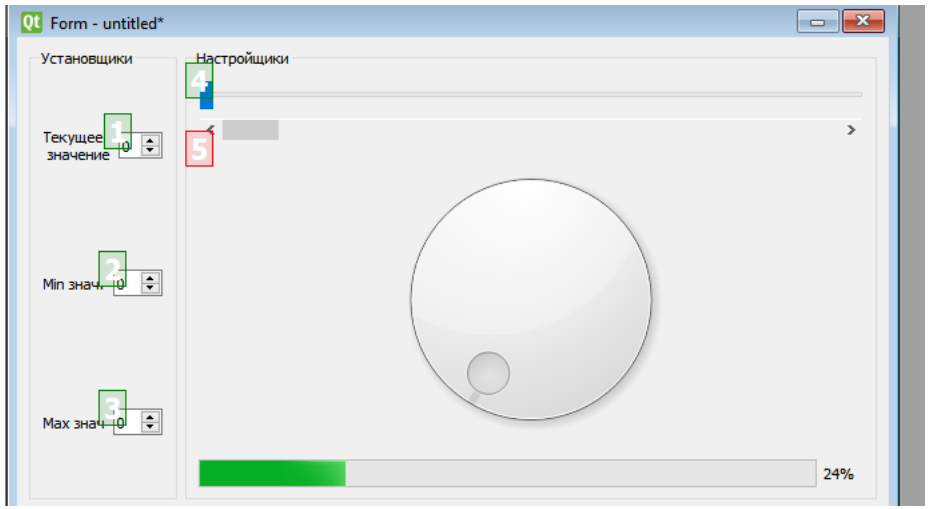
3



2


Form - untitled*
Установщики
Настройщики
Текущее значение
0
Min знач.
0
Max знач
0
24%


Form - untitled*
Установщики
Настройщики
setValue(int)
valueChanged(int)
Текущее значение
Min знач.
Max знач
setMaximum(int)
24%


Form - untitled*
Установщики
Настройщики
Текущее значение
Min знач.
Max знач
24%

Типы окон

конструктор

QWidget(QWidget* pwgt =O, Qt::WindowFlags f = 0)

wgt.setWindowFlags(Qt::Window | Qt::WindoWTitleHint | Qt::WindowStaysOnTopHint);

метод

```
#include "ui_MyForm.h“

class MyForm : public QWidget {
    Q_OBJECT
private:
    Ui::MyForm m_ui;
public:
     MyForm(QWidget* pwgt = 0) : QWidget (pwgt)
     {
       m_ui.setupUi(this);
       m_ui.setWindowFlags(Qt::Window | Qt::WindowTitleHint);
       connect(m_ui.m_cmdReset, SIGNAL(clicked()),
                                          SLOT(slotReset() ));
      }
public slots:
void slotReset (){
                m_ui.m_sld->setValue(0);
                m_ui.m_lcd->display(0);
     }
};
```