# Связка Windows+Eclipse+QT+MinGW+CMake:

- 1. <u>Скачиваем QT</u>, сейчас актуальна версия <u>Qt 5.3.1 for Windows 32-bit (MinGW 4.8.2, OpenGL, 735 MB)</u>
- 2. <u>Скачиваем СМаке</u>, сейчас актуальная версия <u>cmake-3.0.0-win32-x86.exe</u> (генерирует файлы управления сборкой из файлов CMakeLists.txt)
- 3. Прописываем пути в PATH до cmake (можно задать при установке), до mingw (идет вместе с Qt, сейчас ставится в C:\Qt\Qt5.3.1\Tools\mingw482\_32\bin), до qmake (C:\Qt\Qt5.3.1\5.3\mingw482\_32\bin),
- 4. Проверяем из консоли, что пути выставлены правильно:

#### >cmake --version

cmake version 3.0.0

CMake suite maintained and supported by Kitware (kitware.com/cmake).

## >g++ --version

g++ (i686-posix-dwarf-rev3, Built by MinGW-W64 project) 4.8.2 Copyright (C) 2013 Free Software Foundation, Inc. This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

#### >qmake --version

QMake version 3.0

Using Qt version 5.3.1 in C:\Qt\Qt5.3.1\\5.3\mingw482\_32\lib

- 5. Добавляем переменную окружения QTDIR=C:\Qt\Qt5.3.1\5.3\mingw482\_32 (понадобится потом в Eclipse),
- 6. Создаём тестовый файл **CMakeLists.txt** в обычном **блокноте** и сохраним его в папке C:\HelloWorld:

```
cmake minimum required (VERSION 2.8.8)
cmake policy (SET CMP0020 NEW)
SET (CMAKE AUTOMOC ON)
SET (CMAKE BINARY DIR bin)
SET (CMAKE INCLUDE CURRENT DIR ON)
FIND PACKAGE (Qt5Widgets REQUIRED)
SET (PROJECT
                      "HelloWorld")
FILE (GLOB HEADERS
                       "${PROJECT}/*.h")
FILE (GLOB SOURCES
                       "${PROJECT}/*.cpp")
FILE (GLOB FORMS "${PROJECT}/*.ui")
FILE (GLOB RESOURCES"${PROJECT}/*.rc")
QT5 WRAP UI (UI HEADERS ${FORMS})
ADD EXECUTABLE (${PROJECT} WIN32 ${HEADERS} ${SOURCES} ${UI HEADERS} ${RESOURCES})
QT5 USE MODULES (${PROJECT} Widgets)
```

#### 7. mainwindow.h:

Сохраняем файл mainwindow.h в C:\HelloWorld\HelloWorld

#### 8. mainwindow.cpp:

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
MainWindow::MainWindow(QWidget *parent):
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
}
MainWindow::~MainWindow()
{
    delete ui;
}
```

Сохраняем файл mainwindow.cpp в C:\HelloWorld\HelloWorld

#### 9. main.cpp:

```
#include "mainwindow.h"
#include <QApplication>
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    return a.exec();
}
```

Сохраняем файл main.cpp в C:\HelloWorld\HelloWorld

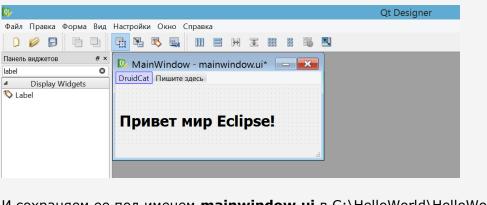
## 10. mainwindow.ui

```
А теперь самое интересное, запускаем QT Designer. Находится он в:

C:\Qt\Qt5.3.1\5.3\mingw482_32\bin\designer.exe

Появляется окно новой формы, выберем Main Window.
```

И создадим незатейливую форму, как нарисовано на картинке ниже.



И сохраняем ее под именем mainwindow.ui в C:\HelloWorld\HelloWorld



11. Создаём папку **Debug** для сборки в папке C:\HelloWorld



Заходим в папку **Debug** через терминал и запускаем стаке командой:

#### cmake -G "MinGW Makefiles" -DCMAKE\_BUILD\_TYPE=Debug ..

После этой команды, cmake в папке **Debug** сгенерирует Makefiles для MinGW. Хочу подметить, что название папки Debug я взял потому, что в этой папке будет создаваться отладочная версия программы, так как cmake мы даем команду на создание отладочной версии программы (**=Debug** ..). Ну и визуально очень наглядно, все выглядит как VS или QT Creator.

```
Microsoft Windows [Uersion 6.3.9600]
(c) Kopnopaция Майкросо+Т (Microsoft Corporation), 2013. Все права защищены.

C:\Users\DruidCat>cd c:\HelloWorld\Debug

c:\HelloWorld\Debug>cmake -G "MinGW Makefiles" -DCMAKE_BUILD_TYPE=Debug ...

-- The C compiler identification is GNU 4.8.2

-- The CXX compiler identification is GNU 4.8.2

-- Check for working C compiler: C:/Qt/Qt5.3.1/Tools/mingw482_32/bin/gcc
.exe

-- Check for working C compiler: C:/Qt/Qt5.3.1/Tools/mingw482_32/bin/gcc
.exe -- works

-- Detecting C compiler ABI info

-- Detecting C compiler ABI info - done

-- Check for working CXX compiler: C:/Qt/Qt5.3.1/Tools/mingw482_32/bin/g

++.exe

-- Check for working CXX compiler: C:/Qt/Qt5.3.1/Tools/mingw482_32/bin/g

++.exe

-- Check for working CXX compiler: C:/Qt/Qt5.3.1/Tools/mingw482_32/bin/g

++.exe

-- Check for working CXX compiler: C:/Qt/Qt5.3.1/Tools/mingw482_32/bin/g

+-.exe

-- Detecting CXX compiler ABI info

-- Configuring done

-- Generating done

-- Generating done

-- Build files have been written to: C:/HelloWorld/Debug

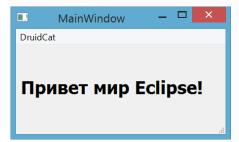
c:\HelloWorld\Debug>_
```

12. Запускаем сборку через mingw32-make:

#### mingw32-make

```
c:\HelloWorld\Debug>mingw32-make
Scanning dependencies of target HelloWorld_automoc
[ 20%] Automatic moc for target HelloWorld
Generating moc_mainwindow.cpp
[ 20%] Built target HelloWorld_automoc
[ 40%] Generating ui_mainwindow h
Scanning dependencies of target HelloWorld
[ 60%] Building CXX object CMakeFiles/HelloWorld.dir/HelloWorld/main.cpp.obj
[ 80%] Building CXX object CMakeFiles/HelloWorld.dir/HelloWorld/mainwindow.cpp.obj
[ 100%] Building CXX object CMakeFiles/HelloWorld.dir/HelloWorld_automoc.cpp.obj
Linking CXX executable HelloWorld.exe
[ 100%] Built target HelloWorld
c:\HelloWorld\Debug>______
```

Запускаем HelloWorld.exe, чтобы проверить собранный exe.



Всё это нужно сделать один раз, чтобы проверить корректность установки Qt/CMake/MinGW.

Теперь про то как будем работать в Eclipse, тестировал на Kepler и Luna:

1. Запускаем Eclipse вашем workspace.

2. Сразу отключаем Project->Build Automatically (это только для Java работает HOРМально): Project Run Window Help Open Project Close Project Build All Ctrl+B **Build Configurations Build Working Set** Build Automatically Make Target Properties 3. Создаём новый проект через File->New->C++ Project (приходится снимать галочку почему-то): Toolchains: Project type: ▶ B GNU Autotools Cygwin GCC ▶ Executable **GNU Autotools Toolchain** ▶ Shared Library Linux GCC ▶ Static Library MacOSX GCC Microsoft Visual C++ Empty Project MinGW GCC Hello World C++ Makefile Project Solaris GCC Show project types and toolchains only if they are supported on the platform 4. Создаём директорию **Debug** через контекстное меню:

▶ 🗂 Project...

Folder

File from Template

li File

File E

**☆** ▼

Proje

⊿ 😂 H

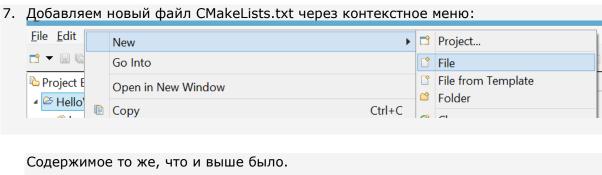
New

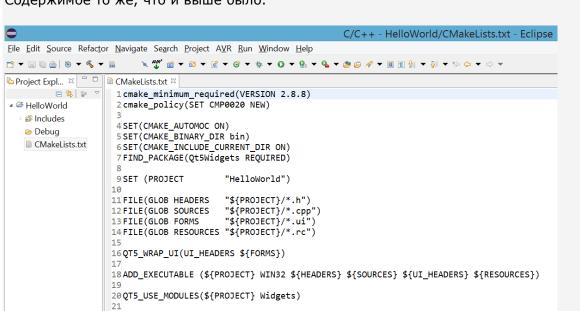
Go Into

Open in New Window

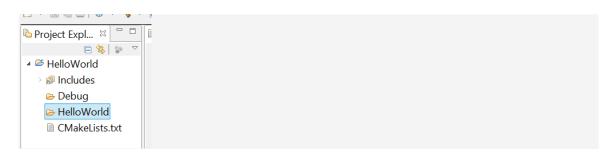
5. В настройках проекта (C/C++ Build) выставляем команду mingw32-make для сборки в директории Debug: Configuration: Default [ Active ] ✓ Manage Configurations... C/C++ Build ≜ Toolchain "MinGW GCC" is not detected. Refer to "C/C++ Development User Guide", "Before you begin" how to ins ▷ C/C++ General Linux Tools Path Project References Run/Debug Settin ▶ Task Repository Builder type: External builder WikiText Use default build command Build command: mingw32-make Variables... Makefile generation ☐ Generate Makefiles automatically ✓ Expand Env. Variable Refs in Makefiles **Build location** Build directory: \${workspace\_loc:/HelloWorld}/Debug Workspace... File system... Variables...

6. В настройках проекта (Paths and Symbols) добавляем необходимые в вашем проекте пути до Qt: Code Analysis 🕒 Includes # Symbols 🛎 Libraries 🕒 Library Paths 🐸 Source Location 🕏 Output Location 🗈 References Documentation Include directories Add... File Types Languages Formatter Assembly \${QTDIR}/include Edit... Indexer GNU C \${QTDIR}/include/QtCore Delete Language Mappings GNU C++ ♣\${QTDIR}/include/QtGui Paths and Symbols \${QTDIR}/include/QtOpenGL Export Preprocessor Include \${QTDIR}/include/QtSql **Profiling Categories** \${QTDIR}/include/QtWidgets Move Up XL C/C++ Language \${QTDIR}/include/QtSerialPort Move Down Linux Tools Path Project References

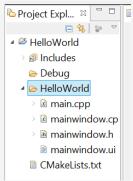




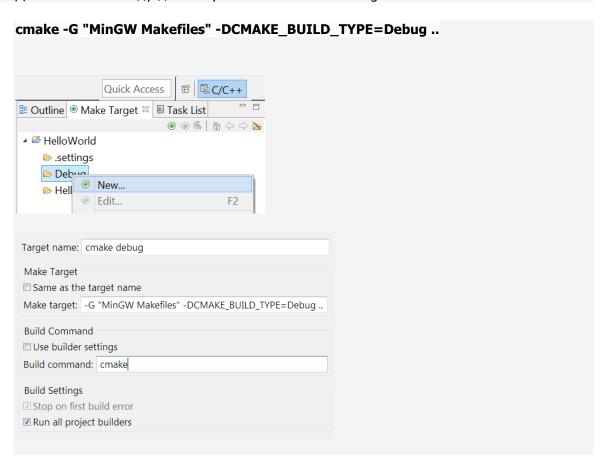
8. Создаём директорию HelloWorld через контекстное меню:



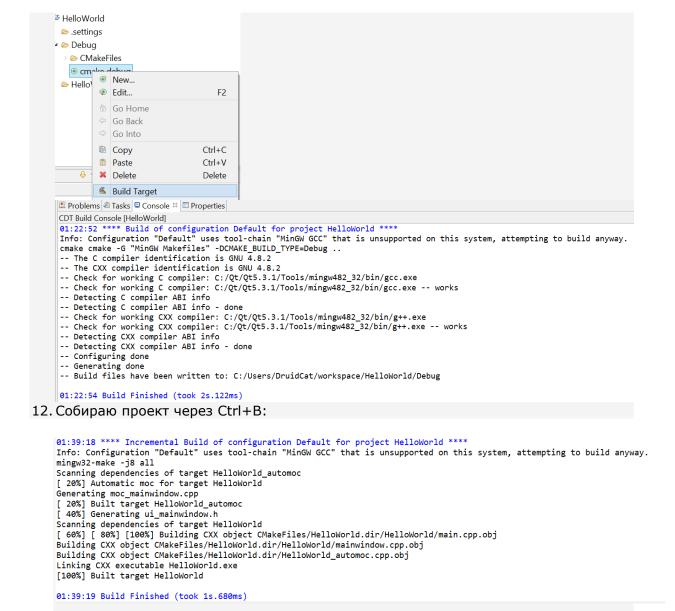
9. Добавляем в эту директорию (HelloWorld) файлы mainwindow.h mainwindow.cpp mainwindow.ui main.cpp с тем же содержанием, что и выше было:



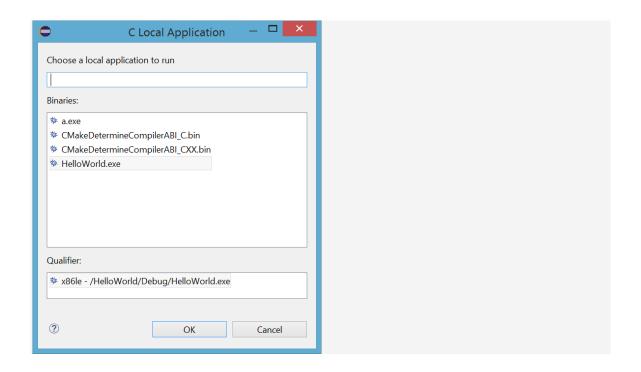
10. Добавляем команду для запуска CMake в Make Target:



#### 11. Запускаем cmake:



13. Запускаю полученный ехе файл через Ctrl+F11 (при этом создается новая конфигурация для запуска):



14. Теперь поговорим о файле **mainwindow.ui**, чтоб пользовать QT Designer в Eclipse настраиваем ассоциацию ui файлов с QT Designer: Window->Preferences->General->Editors->File Associations->Add

Добавляем тип файла \*.ui \_ \_ Preferences type filter text **File Associations** ■ General See 'Content Types' for content-type based file associations. Appearance File types: Compare/Patch ■ ChangeLog Add.. Content Types ■ Editors Add File Type **Remove** File Associations Define a New File Type Text Editors Globalization Enter file type to add: (\*.doc or report.doc for example)

Жмём **O**к, в появившемся окне **File types:** выбираем **\*.ui**, далее в **Associated editors:** жмём на **Add**, в появившемся окне жмём на **Browse..** и в появившемся окне указываем путь к QT Designer, в моем случае, это:

OK

Cancel

C:\Qt\Qt5.3.1\5.3\mingw482\_32\bin\designer.exe

File type:

?

Keys

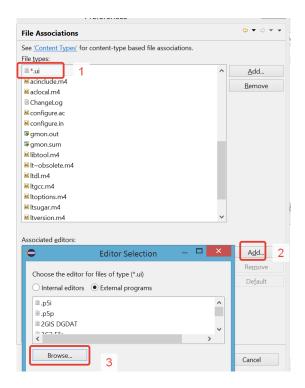
Security

Network Connections

▶ Startup and Shutdow

Perspectives Search

Теперь, вы можете писать исходный код QT в вашей любимой IDE Eclipse и использовать QT Designer.



15. И еще один момент относительно файла **CMakeLists.txt**, если хотите раскрыть весть потенциал вашего проекта, по изучайте структуру этого файла. Лично я написал этот файл под себя, надеюсь вам это пригодится. Если захотите подгрузить библиотеки QtSerialPort QtSql и т.д., не забывайте их добавлять в **CMakeLists.txt**:

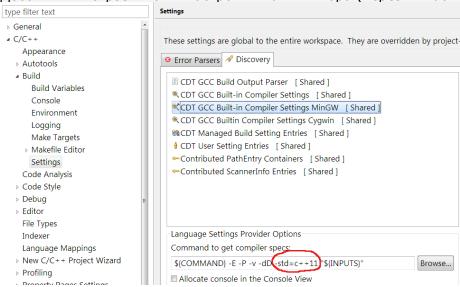
QT5\_USE\_MODULES(\${PROJECT} Widgets SerialPort Sql)

Учтите, регистор букв Widgets SerialPort Sql (и т.д.) крайне важен. Если вы, например, напишите sql, у вас проект не соберется. И не забывайте указать путь этих библиотек в настройках проекта (Пункт №6).

15. Чтобы использовать C++11 нужно добавить в **CMakeLists.txt** строку:

set(CMAKE\_CXX\_FLAGS "\${CMAKE\_CXX\_FLAGS} -std=c++11")

Добавить в Eclipse ключ в настройки компилятора (через Window->Preferences):



PS: Данная инструкция актуальна и для Linux, только необходимо изменить некоторые параметры, но если вы поняли как настроить на Windows, то под Linux тем более поймете.

# Данную инструкцию написали: kamre и DruidCat.