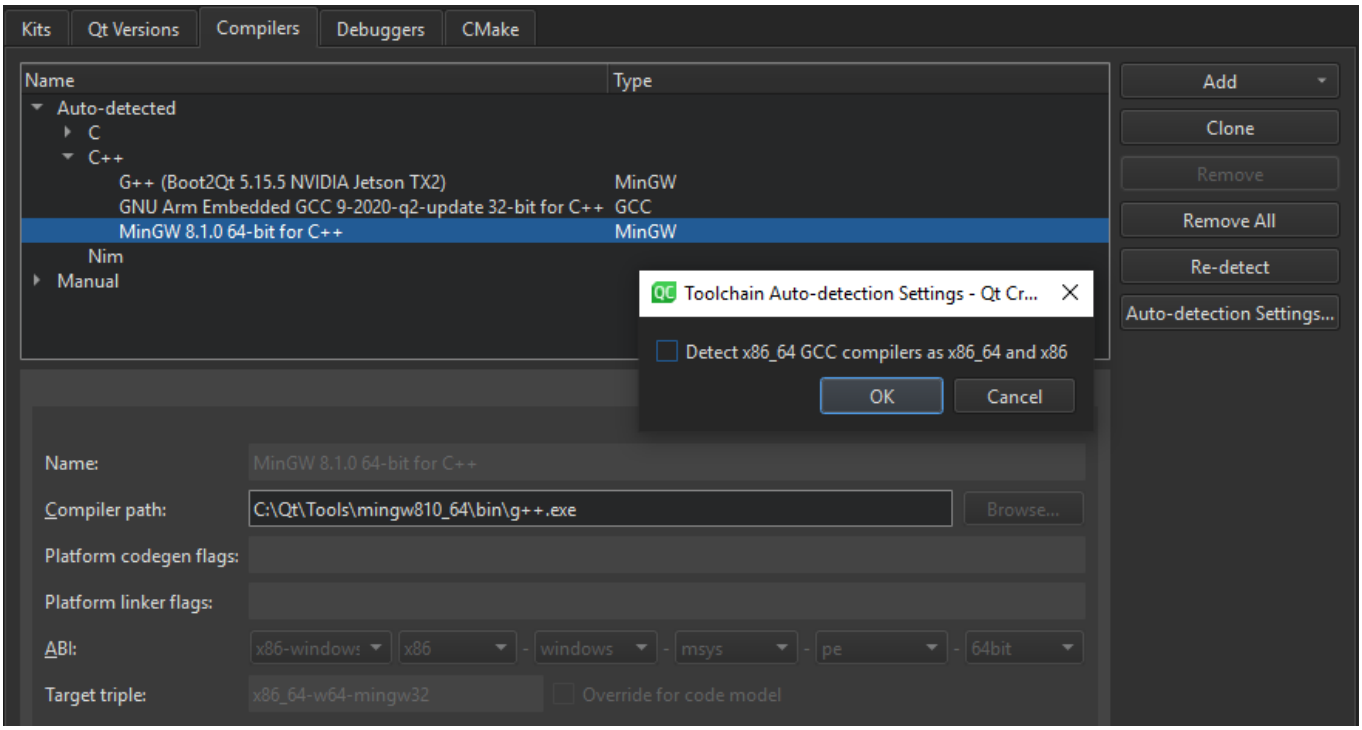
添加编译器

Qt在各种32位和64位平台上受支持，通常可以在每个平台上使用GCC，供应商提供的编译器或第三方编译器构建。在Qt Creator中，工具包指定编译器和其他必要的工具，用于构建应用程序并在特定平台上运行该应用程序。

Qt Creator会自动检测由您的系统或Qt安装程序注册的编译器，并将它们列在**"编辑>首选项">套件>编译器中**：



您可以添加以下编译器，以通过使用其他编译器或使用自动检测到的编译器的其他版本来生成应用程序：

- 叮当声是适用于 Windows、Linux 和 macOS 的 LLVM 编译器的 C、C++、目标 C 和目标C++前端。
- clang-cl 是 Clang 的替代命令行接口，它与可视C++编译器 兼容。`cl.exe`
- GNU 编译器集合 （GCC） 是一个适用于 Linux 和 macOS 的编译器。
- ICC（英特尔C++编译器）是一组 C 和 C++ 编译器。目前只有适用于 Linux 和 macOS 的 GCC 兼容变体受 Qt 创建者支持。
- MinGW（用于视窗的极简主义 GNU）是 GCC 和 GNU 二元压缩的原生软件端口，用于在视窗上开发原生微软视窗应用程序。MinGW 与 Qt 创建者和 Qt Windows 一起分发。
- MSVC（微软视觉C++编译器）是一个C++编译器，随微软视觉工作室一起安装。
- Nim 是适用于 Windows、Linux 和 macOS 的 Nim 编译器。
- QCC 是用于为 QNX 编译C++应用程序的接口。

- IAREW 是一组来自各种 IAR 嵌入式工作台开发环境的 C 和 C++ 裸机编译器。

  > **注意：** 当前支持的体系结构有 、 、 和 。8051AVRARMSTM8MSP430

- KEIL 是一组来自各种 KEIL 开发环境的 C 和 C++ 裸机编译器。

  > **注意：** 当前支持的体系结构是 和 。8051ARM

- SDCC 是一个可重定向的、针对各种架构优化 C 裸机编译器的功能。

  > **注意：** 当前支持的体系结构是 和 。8051STM8

脚本编译器是用于编译到 Web 组件的工具链。

重新检测编译器

当Qt Creator找到一个x86_64 GCC编译器时，它会为本机x86_64目标设置一个实例。如果计划在不使用专用交叉编译器的情况下创建 32 位 x86 二进制文件，请选择"**自动检测设置**">**将x86_64 GCC 编译器检测为x86_64和x86。**然后选择"**重新检测**"以刷新自动检测到的编译器的列表。

若要删除手动添加的编译器，请选择"**删除**"或"**全部删除**"。

指定编译器设置

要使用 GCC、MinGW、Clang 或 QCC 构建应用程序，请指定编译器所在目录的路径，然后从可用版本列表中选择应用程序二进制接口 （ABI） 版本。您还可以创建自定义 ABI 定义。对于 QCC，还要在 **SPD** 路径字段中指定 QNX 软件开发平台 （SDP） 的路径。

To enable Microsoft Visual C++ Compilers (MSVC) and clang-cl to find system headers, libraries, and the linker, Qt Creator executes them inside a command prompt where the environment has been set up using . For these compilers, you also specify the path to the script that sets up the command prompt in the **Initialization** field.`vcvarsall.bat`

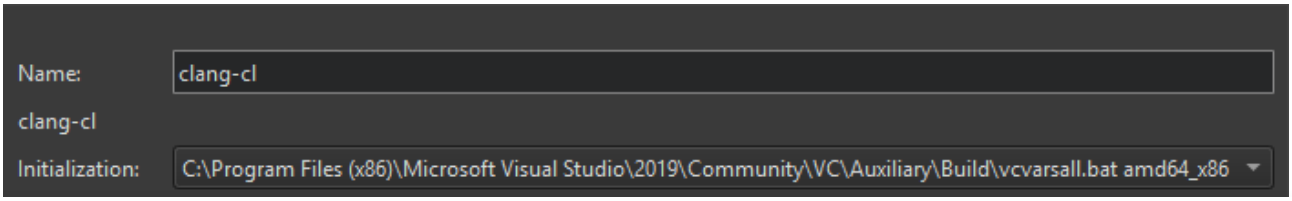
You specify the compiler to use for each kit in **Edit** > **Preferences** > **Kits**.

To add a C or C++ compiler, select **Edit** > **Preferences** > **Kits** > **Compilers** > **Add**. Select a compiler in the list, and then select **C** or **C++**.

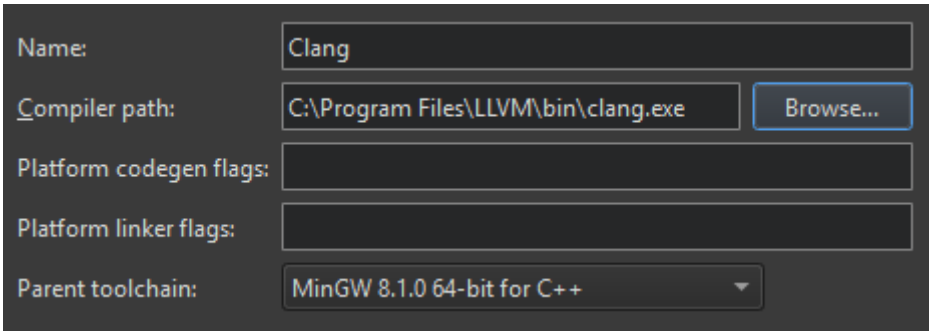
To clone the selected compiler, select **Clone**.

The settings to specify depend on the compiler:

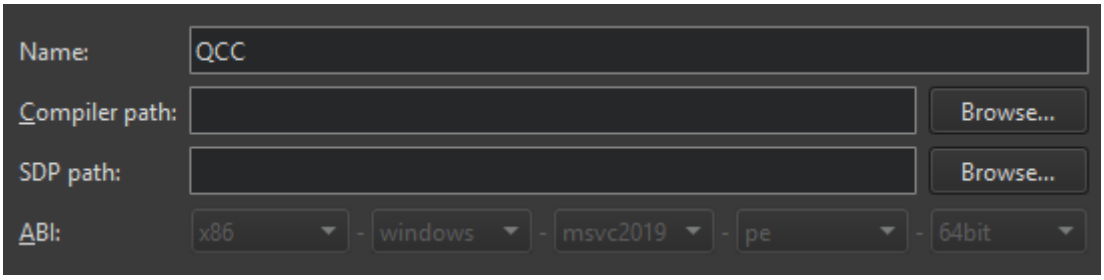
- In the **Name** field, enter a name for the compiler to identify it in Qt Creator.



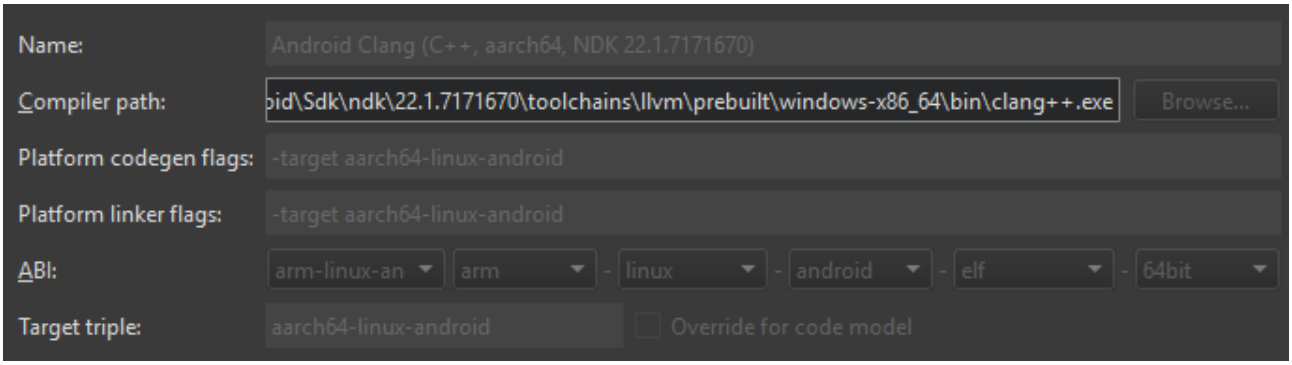
- In the **Initialization** field, select the file for setting up the command prompt to use.`vcvarsall.bat`
- In the **Compiler path** field, enter the path to the directory where the compiler is located.
- In the **Platform codegen flags** field, check the flags passed to the compiler that specify the architecture on the target platform.
- In the **Platform linker flags** field, check the flags passed to the linker that specify the architecture on the target platform. The linker flags are used only when building with Qbs.



- In the **Parent toolchain** field, select a MinGW compiler, which is needed because Clang does not have its own standard library.
- In the **SPD path** field, specify the path to the QNX Software Development Platform (SDP).



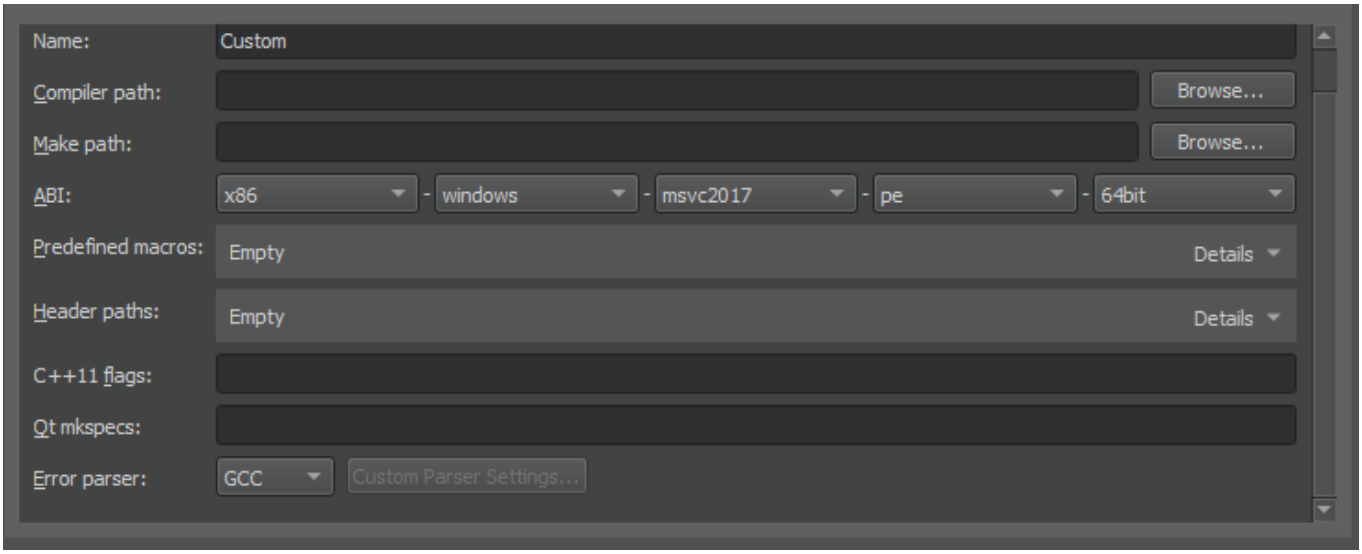
- In the **ABI** field, provide an identification for the target architecture. This is used to warn about ABI mismatches within the kits.
- In the **Target triple** field, specify the GCC target architecture. If services provided by the code model fail because Clang does not understand the target architecture, select **Override for code model**.



Adding Nim Compilers

To build an application using the Nim Compiler, select **Edit** > **Preferences** > **Kits** > **Compilers** > **Add** > **Nim**, and specify the path to the directory where the compiler is located.

To add a compiler that is not listed above or a remote compiler, use the **Custom** option and specify the paths to the directories where the compiler and make tool are located and options for the compiler.



To add other compilers:

1. Select **Edit** > **Preferences** > **Kits** > **Compilers** > **Add** > **Custom** > **C** or **C++**.
2. In the **Name** field, enter a name for the compiler.
3. In the **Compiler path** field, enter the path to the directory where the compiler is located.
4. In the **Make path** field, enter the path to the directory where the make tool is located.
5. In the **ABI** field, specify the ABI version.
6. In the **Predefined macros** field, specify the macros that the compiler enables by default. Specify each macro on a separate line, in the following format: MACRO[=value].
7. In the **Header paths** field, specify the paths to directories that the compiler checks for headers. Specify each path on a separate line.
8. In the **C++11 flags** field, specify the flags that turn on C++11 support in the compiler.
9. In the **Qt mkspecs** field, specify the path to the directory where mkspecs are located. Usually, the path is specified relative to the Qt mkspecs directory.
10. In the **Error parser** field, select the error parser to use. You can add custom output parsers to the list. For more information, see Using Custom Output Parsers.

Troubleshooting MinGW Compilation Errors

If error messages displayed in Compile Output contain paths where slashes are missing (for example,), check your PATH variable. At the command line, enter the following commands:`C:QtSDK`

```
where sh.exe
where make.exe
where mingw32-make.exe
```

If these commands show paths, they have been added to the global PATH variable during the installation of a tool chain based on Cygwin or MinGW, even though this is against Windows conventions.

```
C:\Windows\System32\cmd.exe /K C:\path_to\myenv.bat
```

where the /K parameter carries out the command specified in the bat file.

Create the myenv.bat file at *path_to*, which should be in a convenient location. In the file, specify the paths to the tool chains. For example,

```
set PATH=C:\path1;C:\path2;%PATH%
```

where *path1* and *path2* are paths to the tool chains.

Finally, remove the paths from the global PATH, reboot the computer, and run the commands again to verify that the global PATH is now clean.`where`

You can use the shell link to run the tools in the third-party tool chains.

‹ Adding Qt Versions

Adding Debuggers ›

© 2022 The Qt Company Ltd. Documentation contributions included herein are the copyrights of their respective owners. The documentation provided herein is licensed under the terms of the GNU Free Documentation License version 1.3 as published by the Free Software Foundation. Qt and respective logos are trademarks of The Qt Company Ltd in Finland and/or other countries worldwide. All other trademarks are property of their respective owners.

Qt The Qt Company

Contact Us

Company

About Us
Investors
Newsroom
Careers
Office Locations

Licensing

Terms & Conditions
Open Source
FAQ

Support

Support Services
Professional Services
Partners
Training

For Customers

Support Center
Downloads
Qt Login
Contact Us
Customer Success

© 2022 The Qt Company