Lab 1: Qt6 Testcase Compile Using MXE

YT0798 Cross Development, Thomas More

Gilles Van pellicom

October 15, 2025

Contents

Notes on the Process

Reproducability

- **Cross-compilation environment:**
  - **Platform:** Ubuntu 24.04.3 LTS Server
  - **Hardware:** AMD EPYC 7502P, 128 GB DDR4
  - **Build system:** GNU Make 4.3
  - **Project generator:** CMake 3.28.3
  - **Compiler:** GCC 13.3.0
  - **Cross-compile toolchain:** MXE v2023.10
- **Development environment:**
  - **Platform:** macOS Tahoe 26.1
  - **Hardware:** M3 Pro, 36 GB LPDDR5
  - **Build system:** GNU Make 4.4
  - **Project generator:** CMake 4.1.2
  - **Compiler:** Apple Clang 18.0.0
  - **IDE:** CLion 2025.2
- **Testing environment:**
  - **Platform:** Windows 11 24H2 (build 26100.6584)
  - **Hardware:** Intel i7-6700K, 32 GB DDR4

Chapter 1

Testcase

1.1 Requirements and Fulfillment

The test case chosen is a calculator application developed in Qt6, meeting the following evaluation criteria:

- **Standard widgets:** Implemented using `QPushButton`, `QLineEdit`, and `QComboBox` for number input, operation selection, and mode switching.
- **Custom widget drawn with primitives:** `DisplayWidget` renders the calculator display with rounded rectangle background, custom typography, and right-aligned text using `QPainter` primitives.
- **Dialog for selecting a file or color:** Settings dialog includes `QColorDialog` for accent color selection.
- **File I/O or settings storage:** Selected accent color stored persistently using nlohmann JSON library with file I/O.
- **Variable content using `QStackedWidget`:** Switches between normal and scientific calculator modes, similar to Android's fragment-based navigation.

1.2 Portability Considerations

Since I usually work with portability in mind, the application required no modifications for cross-compilation with MXE. Portability practices used include but are not limited to:

- **Cross-compatible JSON library:** Utilized the nlohmann JSON library, which is platform-agnostic.
- **Dynamic file paths:** Used `QFile f(settingsFilePath())` instead of static paths to handle file I/O portably.
- **OS-agnostic color picker:** Leveraged Qt's `QColorDialog` to handle color selection, allowing Qt to manage platform-specific details seamlessly.

Chapter 2

Setup and Dependency Installation

2.1 System Update and Package Installation

Install all required tools for compilation of MXE and Qt6:

```
sudo apt update
sudo apt install -y autoconf automake autopoint bash bison bzip2 flex g++ \
g++-multilib gettext git gperf intltool libffi-dev libgdk-pixbuf2.0-dev \
libtool libltdl-dev libssl-dev libxml-parser-perl make python3 python3-pip
```

MXE requires `mako` for `mako-render` during the Qt6 build. Install it using `pip3`:

```
pip3 install mako
```

MXE assumes `python` to be the command to trigger python. Most systems have moved on to `python3`. A simple fix for this is:

```
sudo apt install python-is-python3
```

Now MXE will correctly be able to exectute python scripts.

2.2 Cloning and Building MXE

Clone the MXE repository and build Qt6 for the static Windows 64-bit target:

```
git clone https://github.com/mxe/mxe.git
cd ~/mxe
make qt6 -j$(nproc)
```

2.3 Environment Configuration

Add MXE tools to the PATH for access to cross-compilation utilities inside .bashrc, for persistence:

```
export PATH=~/mxe/usr/bin:$PATH
```

Chapter 3

Project Compilation

3.1 Preamble

Pull the project from git since the compiler machine is not my workspace. Navigate into the pulled project:

```
git clone https://github.com/GillesVanPellicom/TM_crossdev_2025/
cd ~/TM_crossdev_2025
```

Create appropriate build directory:

```
mkdir build-win64-static
cd build-win64-static
```

3.2 CMake Configuration

Configure the project using MXE's CMake wrapper for the `x86_64-w64-mingw32.static` target, which sets the toolchain and locates Qt6 automatically:

```
x86_64-w64-mingw32.static-cmake ..
```

3.3 Building the Executable

Compile the project to generate `crossdev.exe`:

```
make -j$(nproc)
```

3.4 Outcome

The build produces `crossdev.exe`, a standalone Windows 64-bit executable statically linked with Qt6. Manual testing revealed a fully working windows executable.