Getting Started

Prerequisites

Two things must be installed before the Motion Helper can be built:

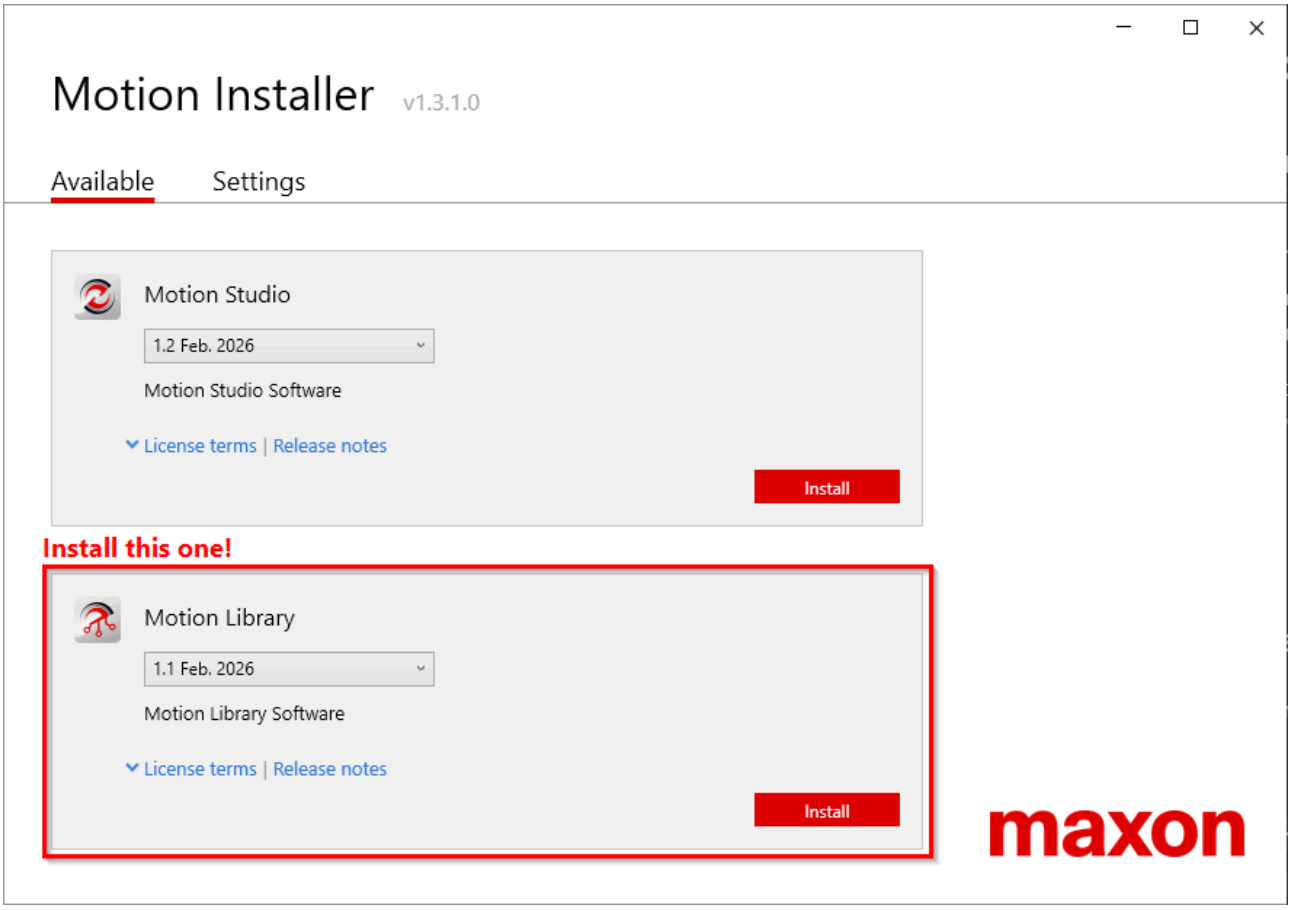
- the Motion Library - the communication library from maxon
- Visual Studio 2022 or 2026 - the C++ build tools

Installing the Motion Library

The Motion Library (DLL, import library and header) is only distributed with the *Motion Studio* installer.

1. Download the Motion Studio installer from the maxon product page, unzip it and start it: https://www.maxongroup.com/en/drives-and-systems/controls/current-and-speed-controllers#software-included-242572

2. Select the package *Motion Library* for installation.



3. Click "Install" and wait until it has finished.

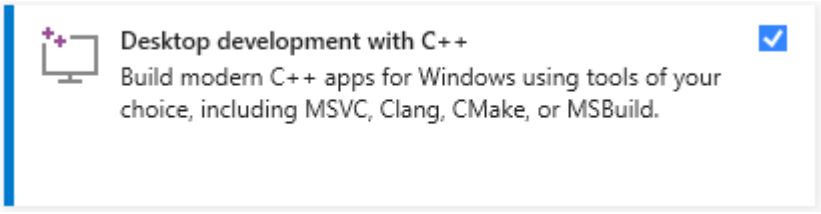
By default the library is installed to `C:\Program Files (x86)\maxon\Motor and Motion Control\Software\Motion Library`, and the CMake module finds it there without further configuration.

> **Note**
>
> If you install to a different location, note the path of the `Motion Library` folder (the one that contains `Windows Motion Library`, not `Windows Motion Library` itself). You will pass it to CMake as `MOTION_LIBRARY_PATH`, see below.

Installing Visual Studio

Visual Studio 2022 or 2026 with the workload **Desktop development with C++**. Both are supported; any edition works, including Community.

1. Download Visual Studio: https://visualstudio.microsoft.com/downloads/
2. In the installer, select the workload "**Desktop development with C++**". It brings the compiler, CMake and Ninja.



3. Wait until it has finished.

Using the Motion Helper in your own project

The Motion Helper is designed for CMake projects. The quickest way to your own program is to copy the example folder `examples/profile_velocity_mode/` and edit `main.cpp`. The steps below describe the same thing for an existing project.

1. Unpack the archive into a folder of your project, e.g. `third_party/motion_helper/`:

```
C:\dev\my_motion\
    ├── CMakeLists.txt
    ├── main.cpp
    └── third_party\
        └── motion_helper\
            ├── motion_helper.cmake
            ├── source\
            └── ...
```

2. In your `CMakeLists.txt`, include `motion_helper.cmake`, link the target `maxon_MotionHelper` and let the module copy the Motion Library DLL next to your executable. A complete minimal file:

```
cmake_minimum_required(VERSION 3.21)

project(my_motion LANGUAGES CXX)

set(CMAKE_CXX_STANDARD 17)
set(CMAKE_CXX_STANDARD_REQUIRED ON)

add_executable(my_motion main.cpp)

# Finds the Motion Library and defines the targets maxon_MotionLibrary and
maxon_MotionHelper.
include("${CMAKE_CURRENT_LIST_DIR}/third_party/motion_helper/motion_helper.cmake")

# Linking the helper brings the Motion Library (header and import library) along.
target_link_libraries(my_motion PRIVATE maxon_MotionHelper)

# The Motion Library DLL is not on the PATH; this copies it next to the executable.
maxon_copy_motion_library_dll(my_motion)
```

The `include()` may come before or after `add_executable()`; the two lines that name your target come after it.

3. In `main.cpp`, include `MotionDevicePVM.h` and follow the sequence of the example.

Motion Library in a non-default location

If the Motion Library is not installed in its default location, tell CMake where it is - either with a CMake variable before the `include()`:

```
set(MOTION_LIBRARY_PATH "D:/tools/maxon/Software/Motion Library")
include(".../motion_helper.cmake")
```

or with an environment variable `MOTION_LIBRARY_PATH`, or on the command line with `-DMOTION_LIBRARY_PATH=...`. The path points at the `Motion Library` folder, **not** at `Windows Motion Library`. A path that is set but wrong stops the configure step with an error.

Building from the command line

The CMake presets pin the compiler to `cl.exe`, so they need a **Developer PowerShell for VS** (Start menu), not a plain shell. The `x86` presets need the *x86 Native Tools Command Prompt*. See Examples.