Compiling and running of mycobotCpp

1 Downloading

1.1 Downloading source code

Download MycobotCpp on github.

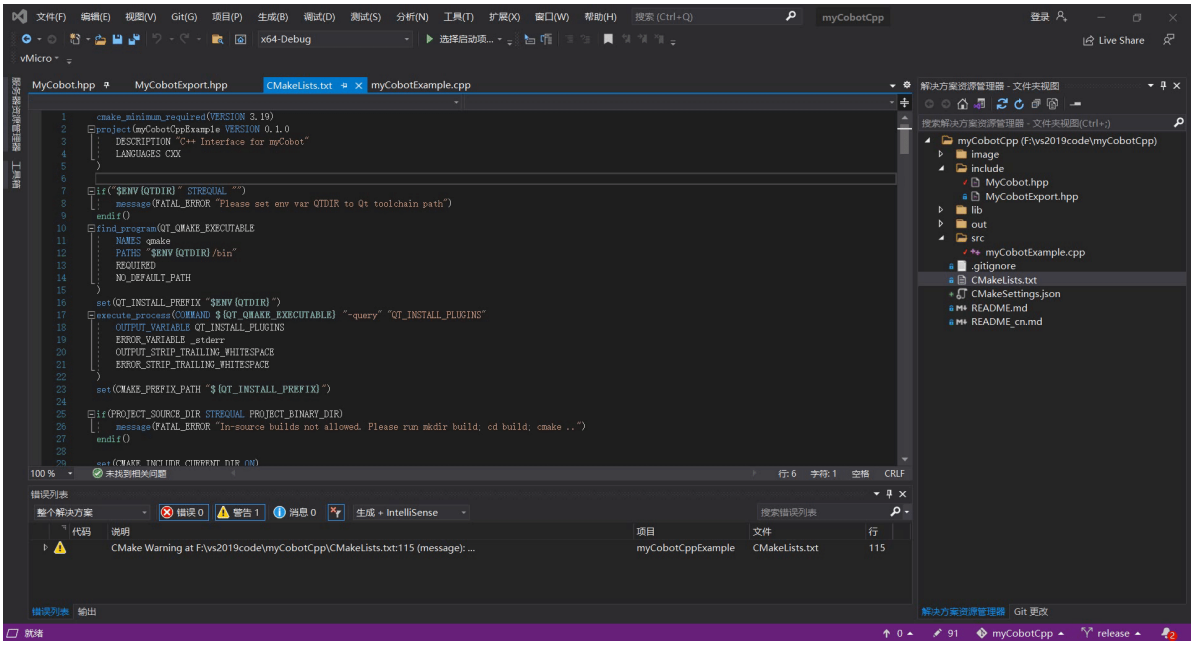
1.2 Dynamic library downloading

Dependency library downloading (Download the latest version, and pay attention to select Windows or Linux; the suffix.zip is the library required by Windows, and .tar.gz is the library required by Linux).

2 Running in Windows

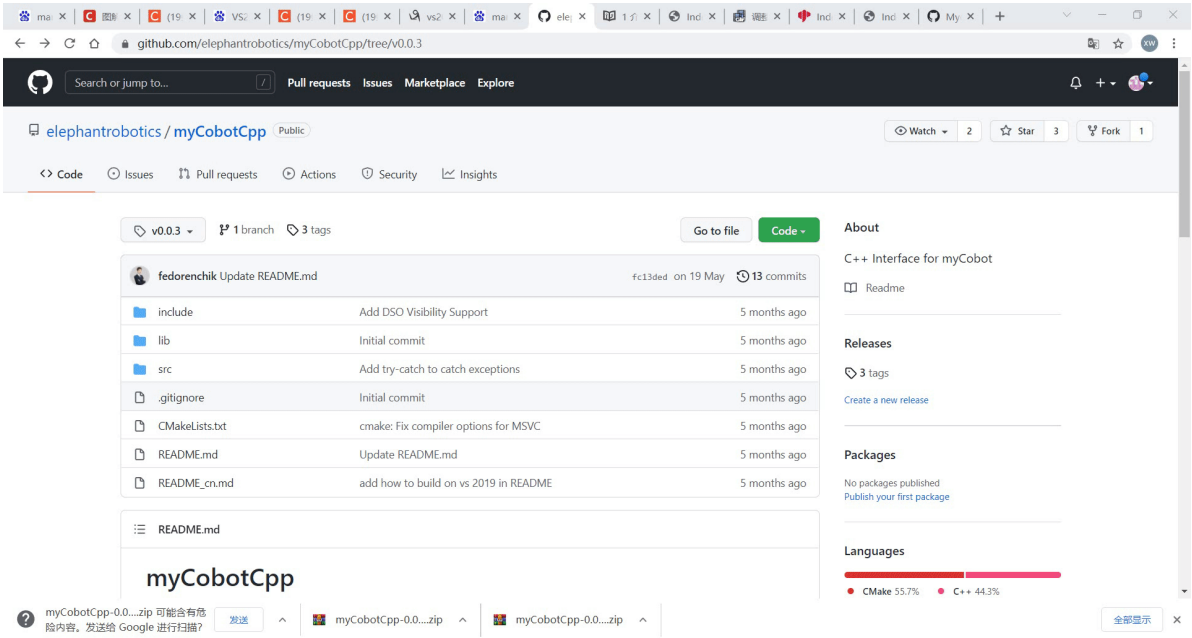
2.1 Compiling

In vs2019, open MycobotCpp, select x64-Release to compile (which is next to the startup item. If there is no x64-Release, click the drop-down box -> Manage configuration to add it), at the same time select release in the configuration of cmake setting, and finally click Generate. Note: Be sure to select x64-Release to compile, as shown in the following figure.



2.2 Running

- Add library files: add .lib to myCobotCpp/lib. Add .lib and .dll to the directory, of myCobotCppExample.exe, at the same level, such as out/build/x64-Release/bin (The two files .lib and .dll are in the myCobotCpp-0.0.3-windows-msvc-x86_64. zipped package).
- Running: Select the startup item (which is next to the green play button and also to the running button), drop down to select myCobotCppEXample.exe (bin.exe), and click Run, as shown in the following figure:



2.3 Common problems and solutions.

Run-time error:

- If myCobotCpp.dll is missing, put myCobotCpp.dll previously saved in the lib directory to the directory where mycobotcppexample.exe is located.
- If QT5Core.dll is reported to be missing, open the qt command (search for QT in the menu bar) and select msvc 2017 64 -bit, execute the directory where windeployqt –release myCobotCppExample.exe is located (for example: windeployqt –release D:201964-Release). If the VS installation path cannot be found after executing the command here, check the settings of VS environment variable.
- After executing the above steps, if the qt5serialport.dll file is reported to be missing, copy the file in the qt installation directory (such as: D:5.12.10\5.12.102017_64) to the directory where myCobotCppExample.exe is located.

3 Running in Linux

3.1 Compiling and building

- mkdir build && cd build
- cmake ..
- cmake –build .

3.2 Run

- copy all .so files to the lib directory;
- command line run: ./bin/myCobotCppExample (Here it is running in the build directory).

4 For example, running on Ubuntu20.04

4.1 Compiling

- mkdir build && cd build
- cmake ..
- cmake –build .

4.2 run

- copy all .so files to the lib directory (Note: after downloading, unzip it. Do not unzip it in Windows, and then copy to Ubuntu. Unzip it directly in Ubuntu, such as: tar -xvf and then drag the file directly to the terminal);
- Soft-link libQt5SerialPort.so.5 (in the QT installation directory, such as: /home/"username"/Qt5.12.10/5.12.10 /gcc_64/lib) to mycobotcpp/build/bin (Do not copy directly). The command is as follows (Be careful to choose your path): ln -s /home/"username"/Qt5.12.10/5.12.10/gcc_64/ lib/libQt5SerialPort.so.5 /home/“username”/myCobotCpp/build/bin/libQt5SerialPort.so.5

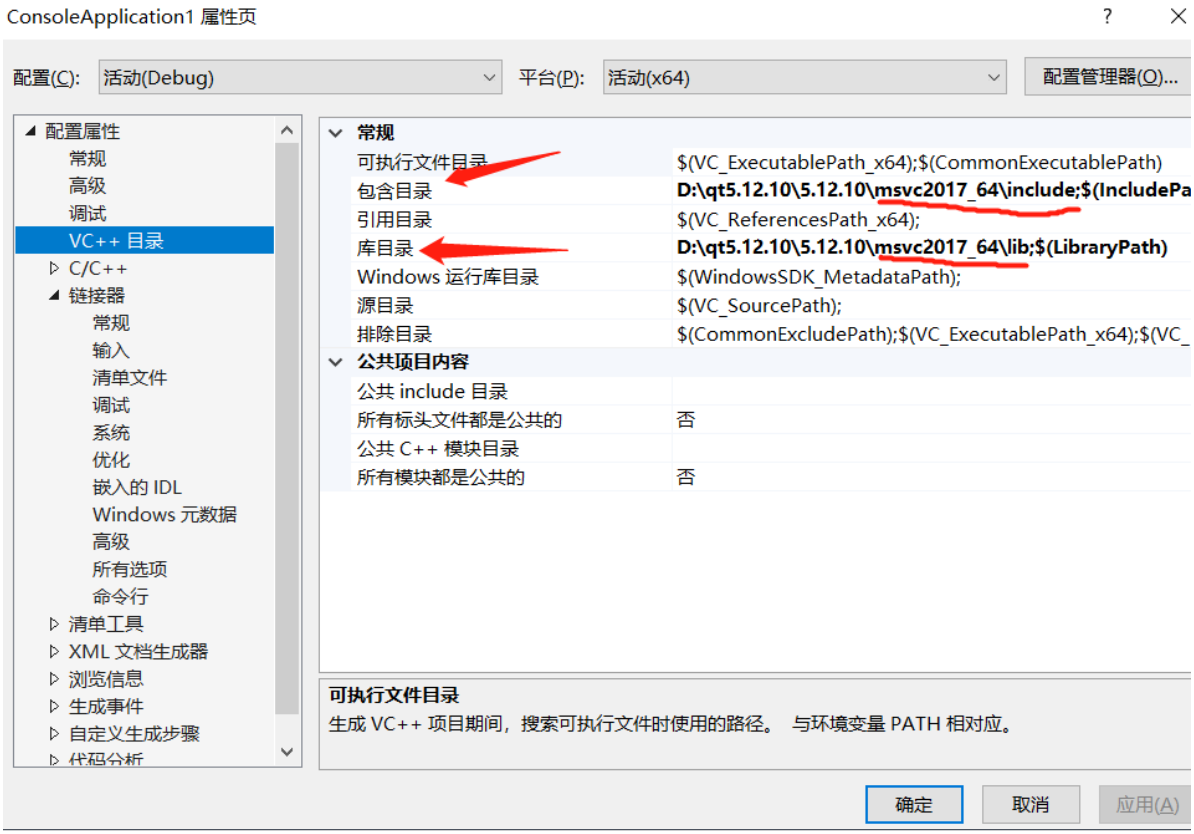
4.3 Common problems and solutions

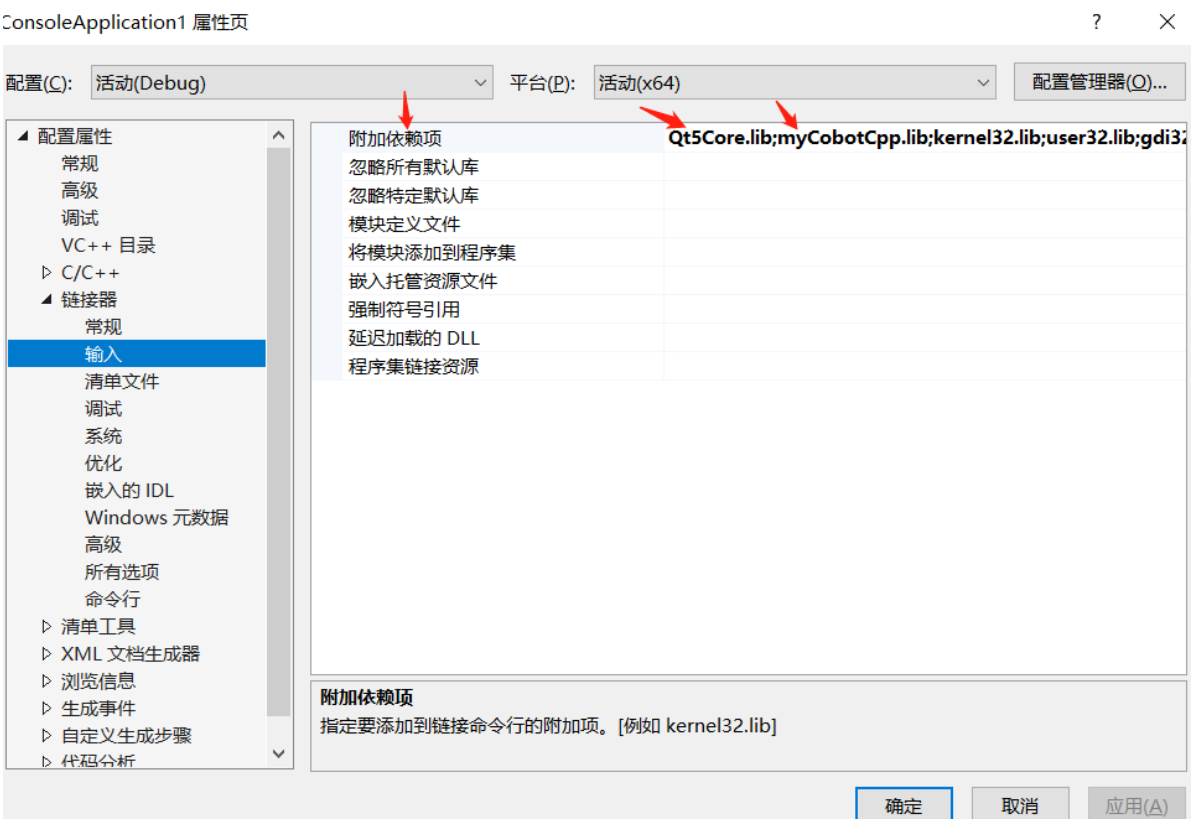
- Error during compilation: QTDIR is not found. Solution: Check whether QTDIR is configured correctly. You may check it in the command line output: echo $QTDIR
- run-time error: Serial port problem: The serial port cannot be opened. Solution: Modify the permission for the serial port of the robot arm. You cannot directly chmod..., because the permission must be modified every time of restart. Modify the file directly:
  - cd /etc/udev/rules.d
  - sudo gedit 20-usb-serial.rules
  - Add in the file: KERNEL=="ttyUSB*" MODE="0777"
- The file cannot be found: such as libQt5SerialPort.so.5 could not be opened or found.

  Solution: Check the running step 2 above.

Notice

If you don't use cmake to compile, and if you use it directly in MFC, do configurations as shown in the figure below.


ConsoleApplication1 属性页
配置(C): 活动(Debug)
平台(P): 活动(x64)
配置管理器(O)...
配置属性
常规
高级
调试
VC++ 目录
C/C++
链接器
常规
输入
清单文件
调试
系统
优化
嵌入的 IDL
Windows 元数据
高级
所有选项
命令行
清单工具
XML 文档生成器
浏览信息
生成事件
自定义生成步骤
代码分析
常规
可执行文件目录 $(VC_ExecutablePath_x64);$(CommonExecutablePath)
包含目录 D:\qt5.12.10\5.12.10\msvc2017_64\include;$(IncludePa
引用目录 $(VC_ReferencesPath_x64);
库目录 D:\qt5.12.10\5.12.10\msvc2017_64\lib;$(LibraryPath)
Windows 运行库目录 $(WindowsSDK_MetadataPath);
源目录 $(VC_SourcePath);
排除目录 $(CommonExcludePath);$(VC_ExecutablePath_x64);$(VC_
公共项目内容
公共 include 目录
所有标头文件都是公共的 否
公共 C++ 模块目录
所有模块都是公共的 否
可执行文件目录
生成 VC++ 项目期间，搜索可执行文件时使用的路径。 与环境变量 PATH 相对应。
确定
取消
应用(A)


ConsoleApplication1 属性页
配置(C): 活动(Debug)
平台(P): 活动(x64)
配置管理器(O)...
配置属性
常规
高级
调试
VC++ 目录
C/C++
链接器
常规
输入
清单文件
调试
系统
优化
嵌入的 IDL
Windows 元数据
高级
所有选项
命令行
清单工具
XML 文档生成器
浏览信息
生成事件
自定义生成步骤
代码分析
附加依赖项 Qt5Core.lib;myCobotCpp.lib;kernel32.lib;user32.lib;gdi3
忽略所有默认库
忽略特定默认库
模块定义文件
将模块添加到程序集
嵌入托管资源文件
强制符号引用
延迟加载的 DLL
程序集链接资源
附加依赖项
指定要添加到链接命令行的附加项。[例如 kernel32.lib]
确定
取消
应用(A)