# 2. Download a program

Try to create a project and download the program.

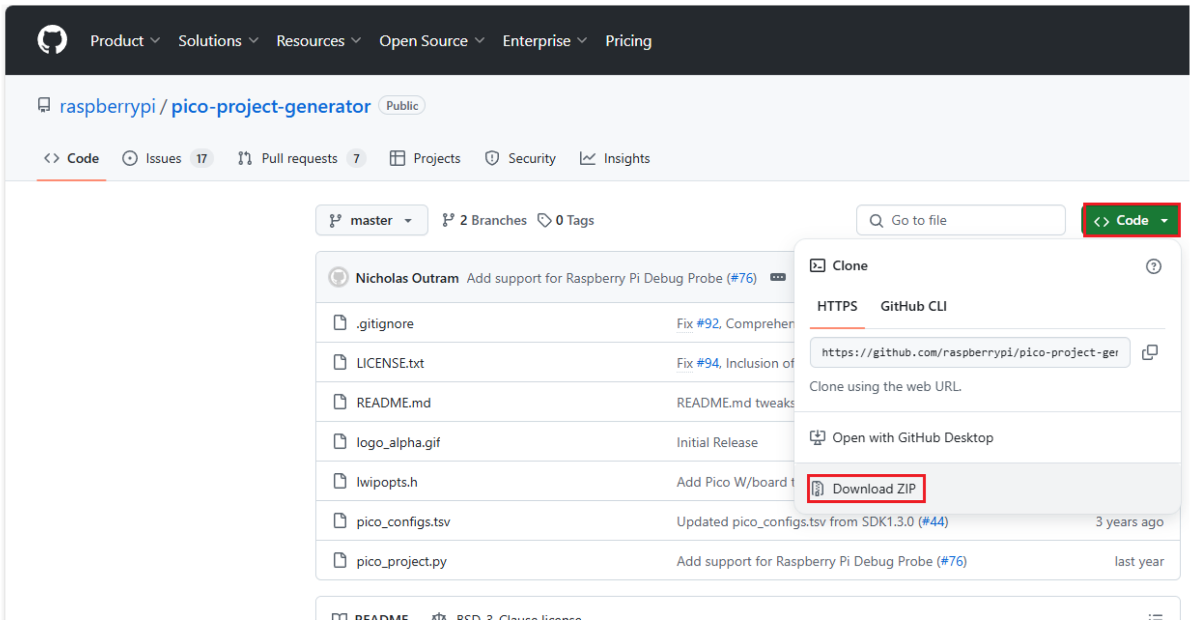
## Project Generator

If you don't want to configure the project manually, you can use the project configuration script to generate the project file.

You can go to this

https://github.com/raspberrypi/pico-project-generator

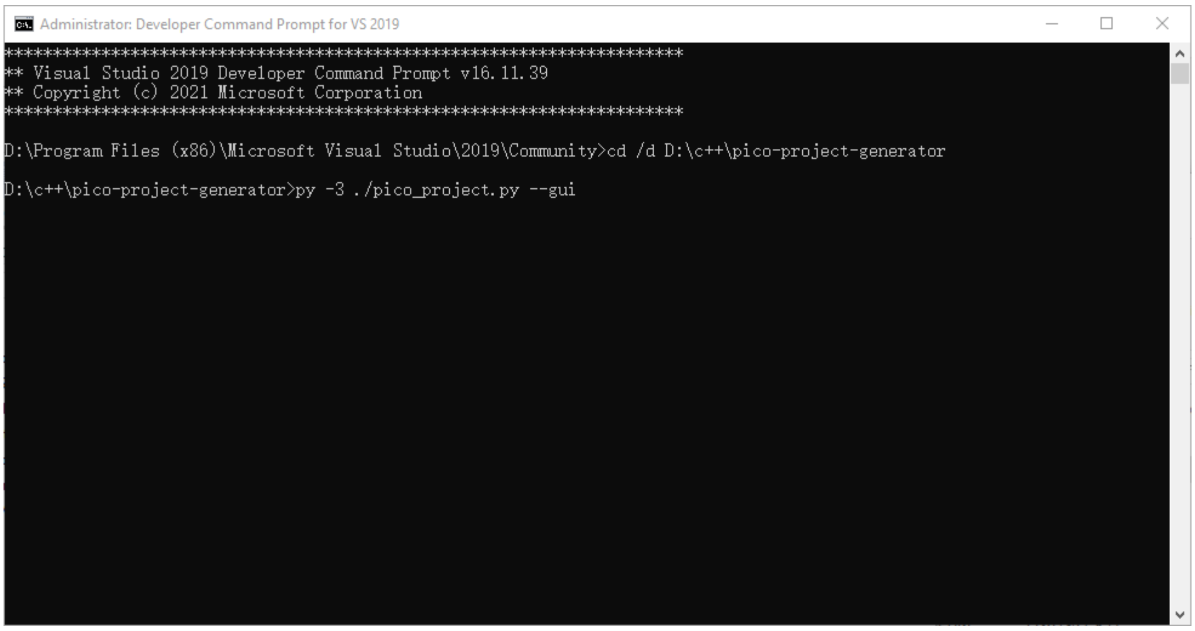
Use Git or directly package and download the file and unzip it,



| | | |
|---|---|---|
| Nicholas Outram | Add support for Raspberry Pi Debug Probe (#76) | |
| .gitignore | Fix #92, Comprehen | |
| LICENSE.txt | Fix #94, Inclusion of | |
| README.md | README.md tweaks | |
| logo_alpha.gif | Initial Release | |
| lwipopts.h | Add Pico W/board t | |
| pico_configs.tsv | Updated pico_configs.tsv from SDK1.3.0 (#44) | 3 years ago |
| pico_project.py | Add support for Raspberry Pi Debug Probe (#76) | last year |

Open Windows Terminal in the folder and run:

According to the previous pico-project-generator installation path, enter the folder, and then enter py -3 ./pico_project.py --gui

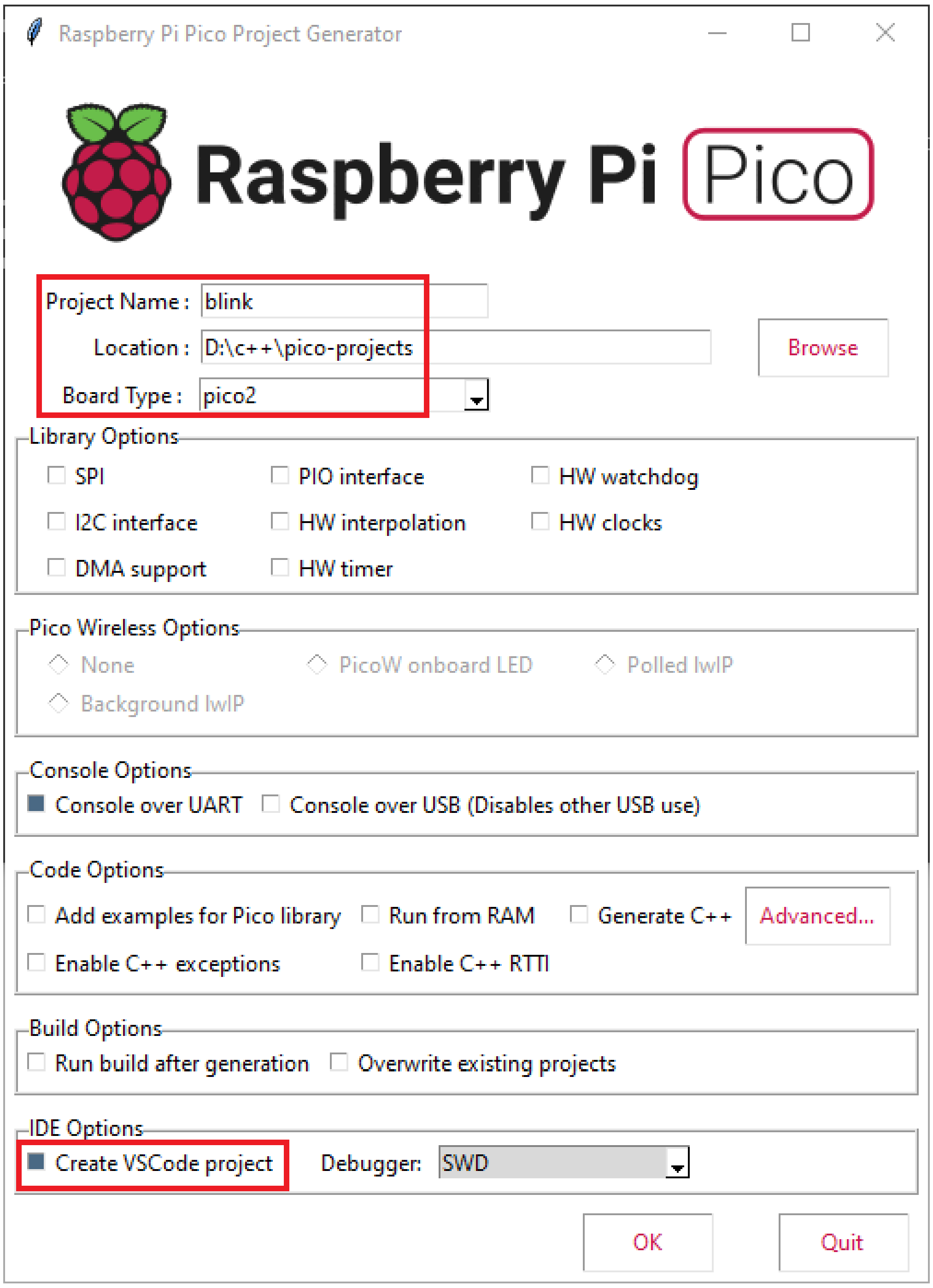


```
Administrator: Developer Command Prompt for VS 2019
**********************************************************************
** Visual Studio 2019 Developer Command Prompt v16.11.39
** Copyright (c) 2021 Microsoft Corporation
**********************************************************************

D:\Program Files (x86)\Microsoft Visual Studio\2019\Community>cd /d D:\c++\pico-project-generator

D:\c++\pico-project-generator>py -3 ./pico_project.py --gui
```

After starting the tool generator, select the project name, select the file path where the project is stored, select pico2 for Board Type, and select create VSCODE in the lower left corner project, and then click OK.


Raspberry Pi Pico Project Generator

Raspberry Pi Pico

Project Name : blink
Location : D:\c++\pico-projects
Board Type : pico2

Library Options
- [ ] SPI
- [ ] PIO interface
- [ ] HW watchdog
- [ ] I2C interface
- [ ] HW interpolation
- [ ] HW clocks
- [ ] DMA support
- [ ] HW timer

Pico Wireless Options
- ◇ None
- ◇ PicoW onboard LED
- ◇ Polled lwIP
- ◇ Background lwIP

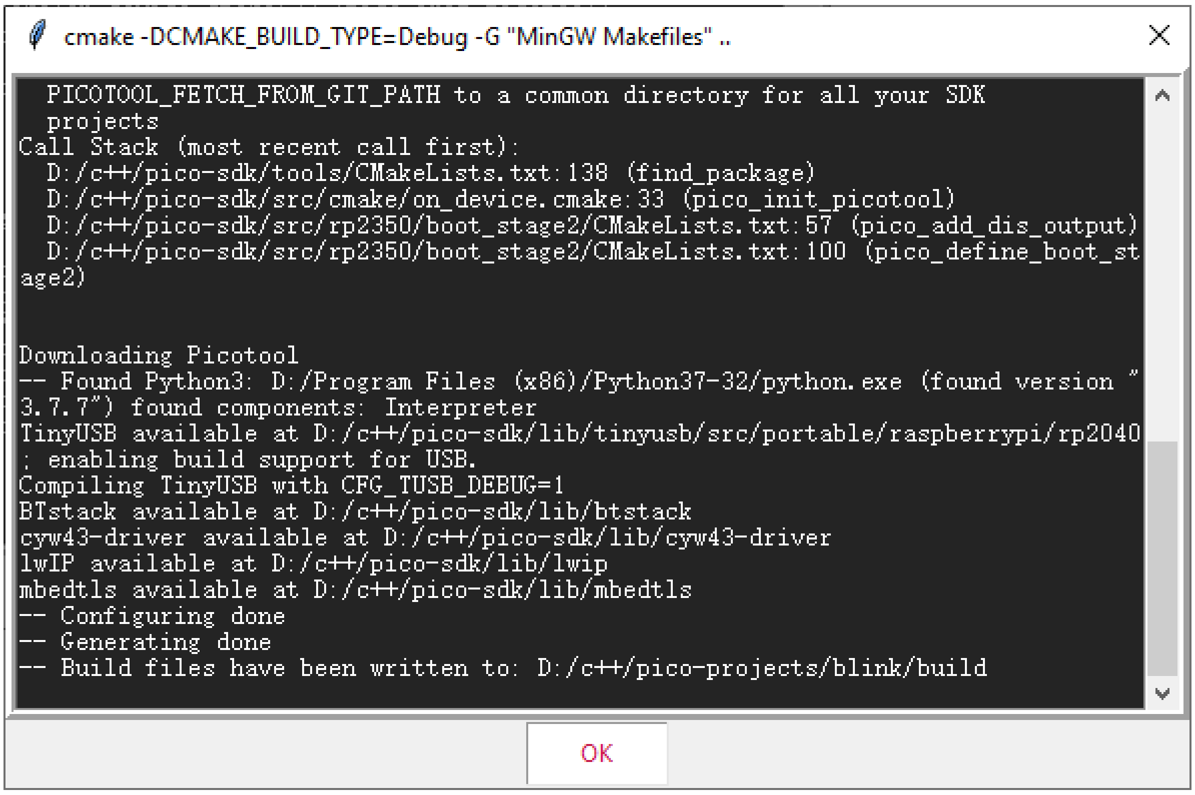
Console Options
- [x] Console over UART
- [ ] Console over USB (Disables other USB use)

Code Options
- [ ] Add examples for Pico library
- [ ] Run from RAM
- [ ] Generate C++
- [ ] Enable C++ exceptions
- [ ] Enable C++ RTTI

Build Options
- [ ] Run build after generation
- [ ] Overwrite existing projects

IDE Options
- [x] Create VSCode project
- Debugger: SWD

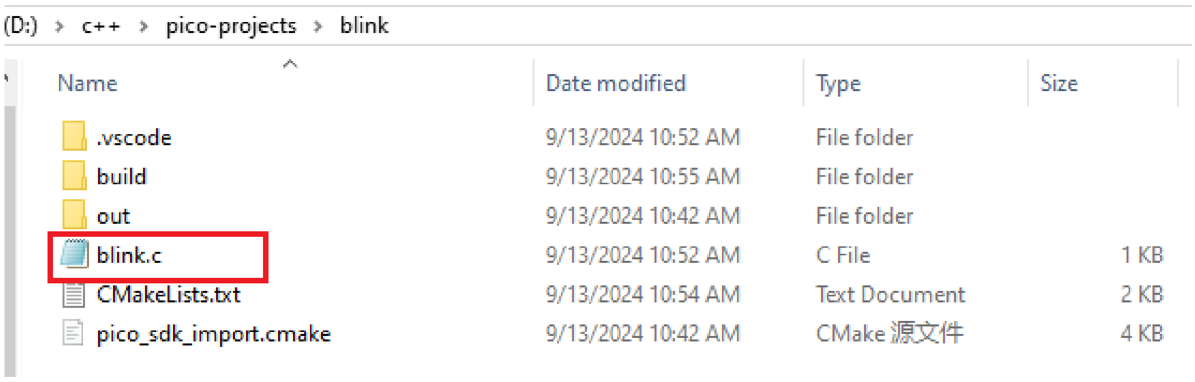
Click OK again.



```
  PICOTOOL_FETCH_FROM_GIT_PATH to a common directory for all your SDK
  projects
Call Stack (most recent call first):
  D:/c++/pico-sdk/tools/CMakeLists.txt:138 (find_package)
  D:/c++/pico-sdk/src/cmake/on_device.cmake:33 (pico_init_picotool)
  D:/c++/pico-sdk/src/rp2350/boot_stage2/CMakeLists.txt:57 (pico_add_dis_output)
  D:/c++/pico-sdk/src/rp2350/boot_stage2/CMakeLists.txt:100 (pico_define_boot_st
age2)


Downloading Picotool
-- Found Python3: D:/Program Files (x86)/Python37-32/python.exe (found version "
3.7.7") found components: Interpreter
TinyUSB available at D:/c++/pico-sdk/lib/tinyusb/src/portable/raspberrypi/rp2040
; enabling build support for USB.
Compiling TinyUSB with CFG_TUSB_DEBUG=1
BTstack available at D:/c++/pico-sdk/lib/btstack
cyw43-driver available at D:/c++/pico-sdk/lib/cyw43-driver
lwIP available at D:/c++/pico-sdk/lib/lwip
mbedtls available at D:/c++/pico-sdk/lib/mbedtls
-- Configuring done
-- Generating done
-- Build files have been written to: D:/c++/pico-projects/blink/build
```

After clicking the .C file, copy the code into it and save it



(D:) › c++ › pico-projects › blink

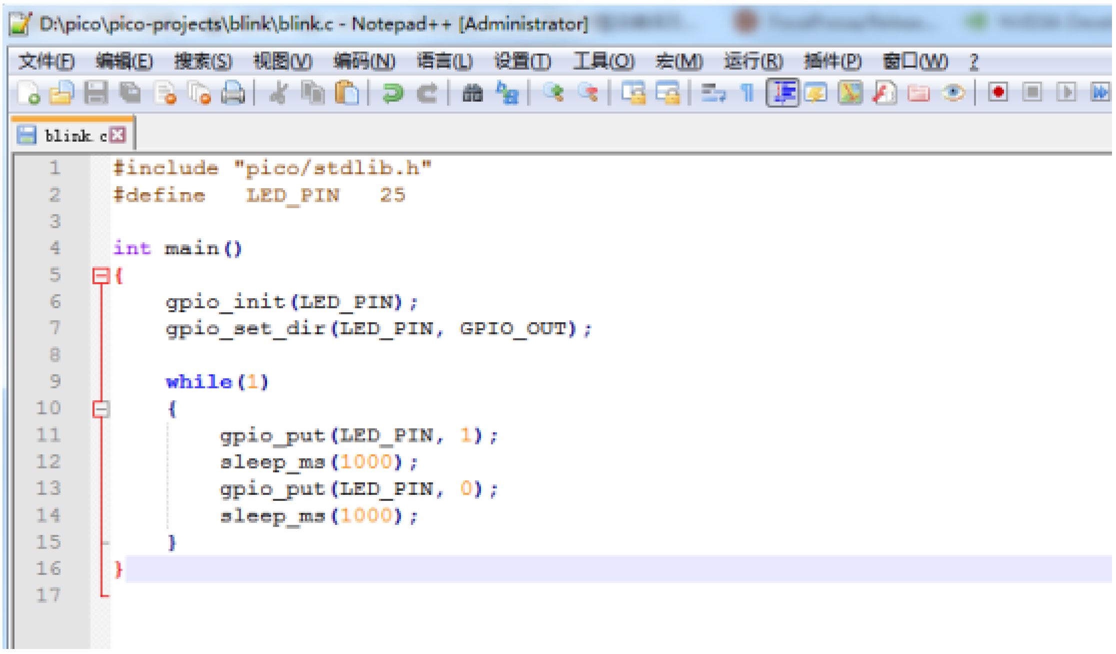
| Name | Date modified | Type | Size |
|---|---|---|---|
| .vscode | 9/13/2024 10:52 AM | File folder | |
| build | 9/13/2024 10:55 AM | File folder | |
| out | 9/13/2024 10:42 AM | File folder | |
| blink.c | 9/13/2024 10:52 AM | C File | 1 KB |
| CMakeLists.txt | 9/13/2024 10:54 AM | Text Document | 2 KB |
| pico_sdk_import.cmake | 9/13/2024 10:42 AM | CMake 源文件 | 4 KB |

```c
#include "pico/stdlib.h"
#define LED_PIN 25

int main()
{
gpio_init(LED_PIN);
gpio_set_dir(LED_PIN, GPIO_OUT);

while (1)
{
gpio_put(LED_PIN, 1);
sleep_ms(1000);
gpio_put(LED_PIN, 0);

sleep_ms(1000);
}
}
```


D:\pico\pico-projects\blink\blink.c - Notepad++ [Administrator]
文件(F) 编辑(E) 搜索(S) 视图(V) 编码(N) 语言(L) 设置(T) 工具(O) 宏(M) 运行(R) 插件(P) 窗口(W) ?
blink.c

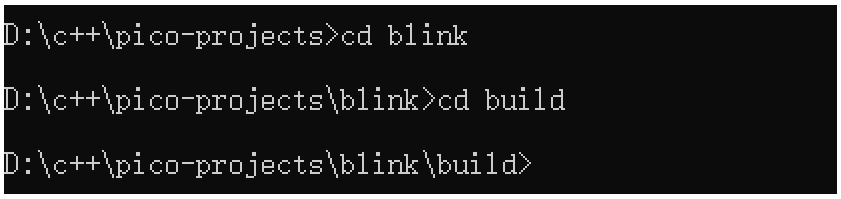
```c
#include "pico/stdlib.h"
#define   LED_PIN   25

int main()
{
    gpio_init(LED_PIN);
    gpio_set_dir(LED_PIN, GPIO_OUT);

    while(1)
    {
        gpio_put(LED_PIN, 1);
        sleep_ms(1000);
        gpio_put(LED_PIN, 0);
        sleep_ms(1000);
    }
}
```

After entering the bulid path of the project, delete all files in the folder

Enter the project path through the command line



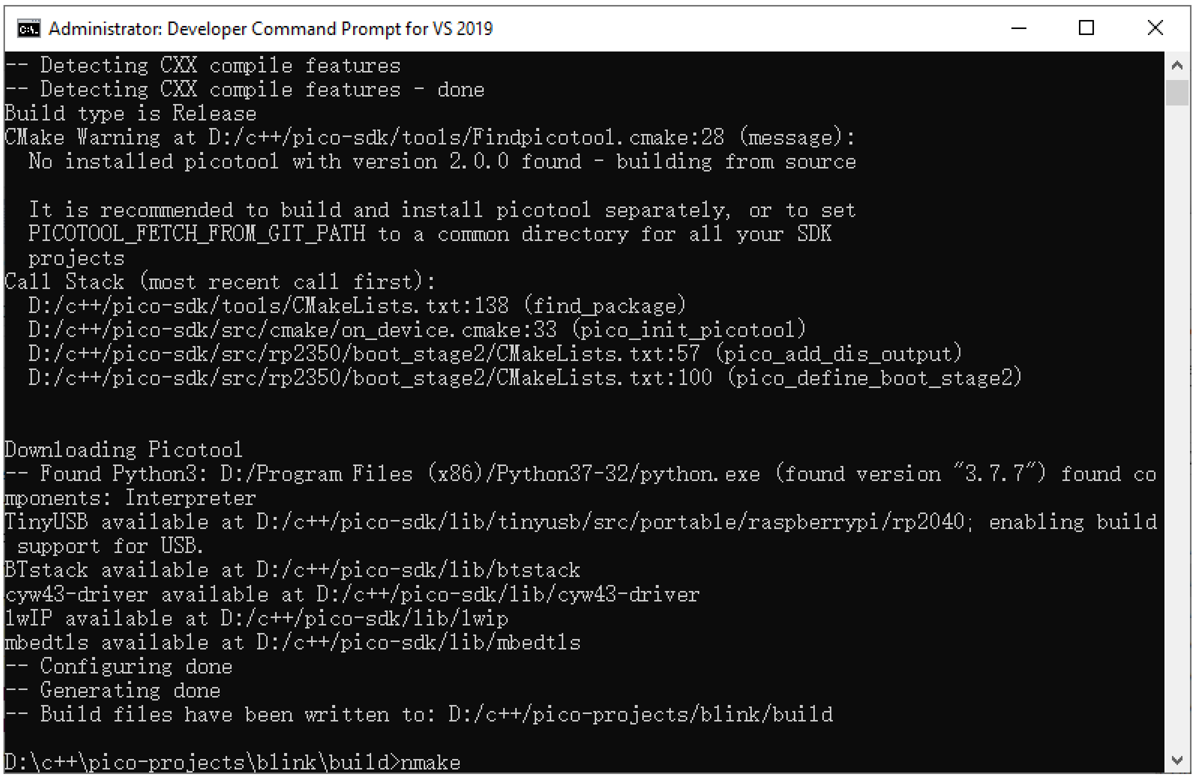
```
D:\c++\pico-projects>cd blink

D:\c++\pico-projects\blink>cd build

D:\c++\pico-projects\blink\build>
```

Enter the following command to compile

```
cmake .. -G "NMake Makefiles"
nmake
```



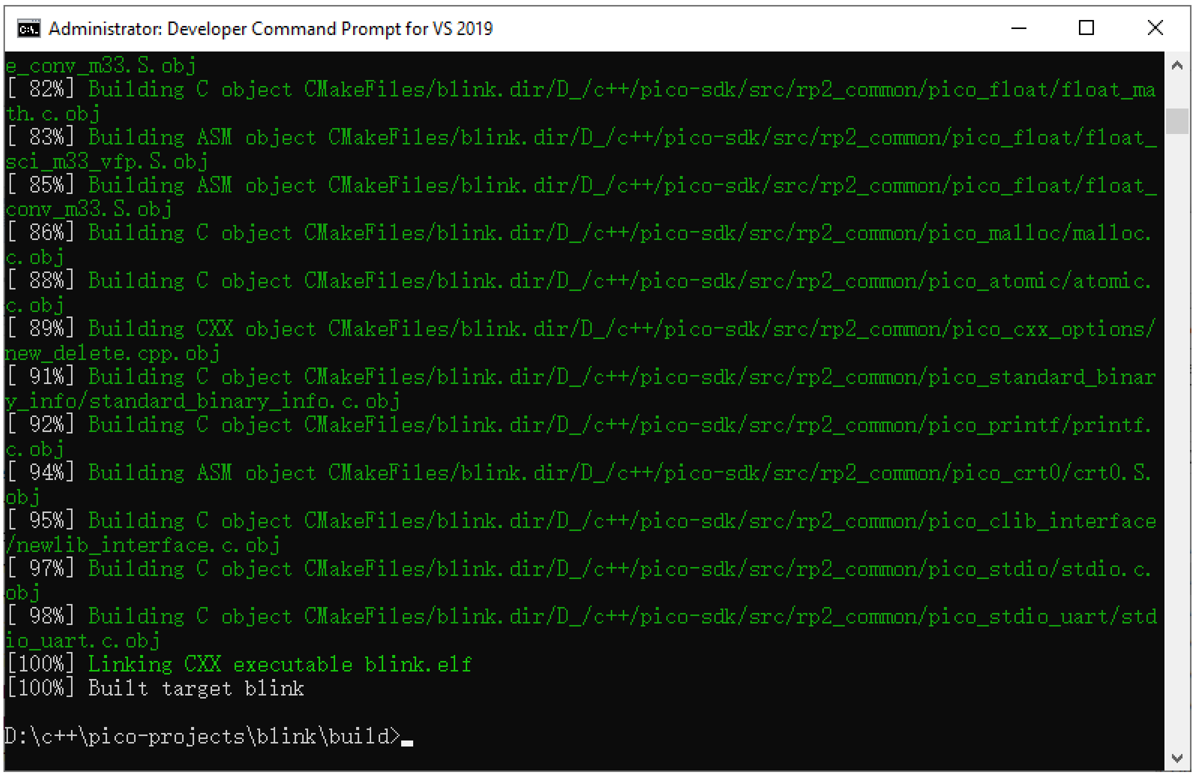
```
-- Detecting CXX compile features
-- Detecting CXX compile features - done
Build type is Release
CMake Warning at D:/c++/pico-sdk/tools/Findpicotool.cmake:28 (message):
  No installed picotool with version 2.0.0 found - building from source

  It is recommended to build and install picotool separately, or to set
  PICOTOOL_FETCH_FROM_GIT_PATH to a common directory for all your SDK
  projects
Call Stack (most recent call first):
  D:/c++/pico-sdk/tools/CMakeLists.txt:138 (find_package)
  D:/c++/pico-sdk/src/cmake/on_device.cmake:33 (pico_init_picotool)
  D:/c++/pico-sdk/src/rp2350/boot_stage2/CMakeLists.txt:57 (pico_add_dis_output)
  D:/c++/pico-sdk/src/rp2350/boot_stage2/CMakeLists.txt:100 (pico_define_boot_stage2)


Downloading Picotool
-- Found Python3: D:/Program Files (x86)/Python37-32/python.exe (found version "3.7.7") found co
mponents: Interpreter
TinyUSB available at D:/c++/pico-sdk/lib/tinyusb/src/portable/raspberrypi/rp2040; enabling build
 support for USB.
BTstack available at D:/c++/pico-sdk/lib/btstack
cyw43-driver available at D:/c++/pico-sdk/lib/cyw43-driver
lwIP available at D:/c++/pico-sdk/lib/lwip
mbedtls available at D:/c++/pico-sdk/lib/mbedtls
-- Configuring done
-- Generating done
-- Build files have been written to: D:/c++/pico-projects/blink/build

D:\c++\pico-projects\blink\build>nmake
```

After the compilation is complete, files in formats such as .bin .hex .elf .uf2 can be generated in the build directory.



```
e_conv_m33.S.obj
[ 82%] Building C object CMakeFiles/blink.dir/D_/c++/pico-sdk/src/rp2_common/pico_float/float_ma
th.c.obj
[ 83%] Building ASM object CMakeFiles/blink.dir/D_/c++/pico-sdk/src/rp2_common/pico_float/float_
sci_m33_vfp.S.obj
[ 85%] Building ASM object CMakeFiles/blink.dir/D_/c++/pico-sdk/src/rp2_common/pico_float/float_
conv_m33.S.obj
[ 86%] Building C object CMakeFiles/blink.dir/D_/c++/pico-sdk/src/rp2_common/pico_malloc/malloc.
c.obj
[ 88%] Building C object CMakeFiles/blink.dir/D_/c++/pico-sdk/src/rp2_common/pico_atomic/atomic.
c.obj
[ 89%] Building CXX object CMakeFiles/blink.dir/D_/c++/pico-sdk/src/rp2_common/pico_cxx_options/
new_delete.cpp.obj
[ 91%] Building C object CMakeFiles/blink.dir/D_/c++/pico-sdk/src/rp2_common/pico_standard_binar
y_info/standard_binary_info.c.obj
[ 92%] Building C object CMakeFiles/blink.dir/D_/c++/pico-sdk/src/rp2_common/pico_printf/printf.
c.obj
[ 94%] Building ASM object CMakeFiles/blink.dir/D_/c++/pico-sdk/src/rp2_common/pico_crt0/crt0.S.
obj
[ 95%] Building C object CMakeFiles/blink.dir/D_/c++/pico-sdk/src/rp2_common/pico_clib_interface
/newlib_interface.c.obj
[ 97%] Building C object CMakeFiles/blink.dir/D_/c++/pico-sdk/src/rp2_common/pico_stdio/stdio.c.
obj
[ 98%] Building C object CMakeFiles/blink.dir/D_/c++/pico-sdk/src/rp2_common/pico_stdio_uart/std
io_uart.c.obj
[100%] Linking CXX executable blink.elf
[100%] Built target blink

D:\c++\pico-projects\blink\build>_
```

Drag the u2f in the above file into the disk recognized by Pico (Note: When burning for the first time, it is an empty code. Pico 2 can directly recognize the disk when connected to USB. When there is an executable program in it, you need to hold down the BOOTSEL button and then connect USB) After dragging, the disk is disconnected and execution begins (the blink compiled file used here has the onboard LED flashing)