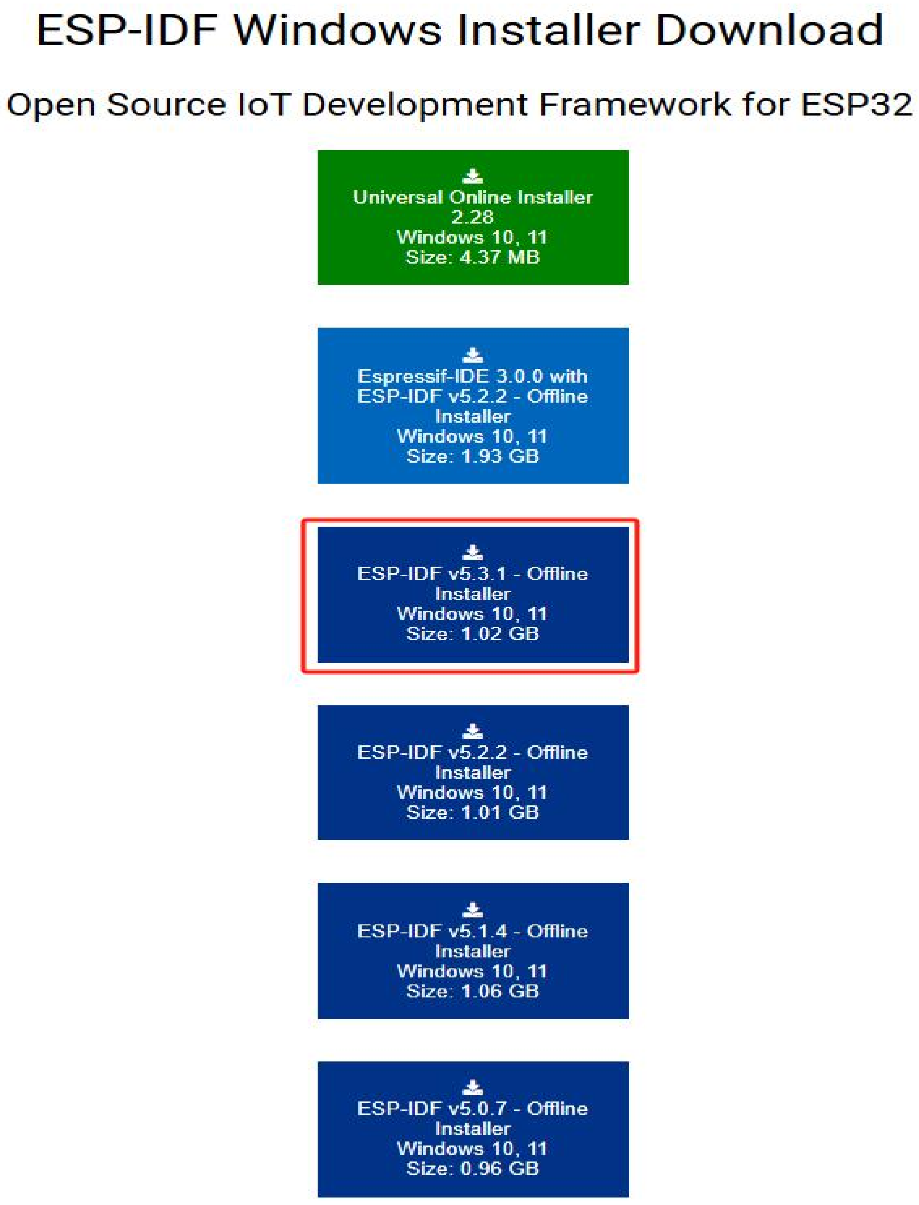
ESP-IDF Tutorial

To help you more quickly understand the use of ESP-IDF and configuration catalogue

Software Install

1. Install:

Browser search ESP-IDF download : dl.espressif.com/dl/esp-idf/?idf=4.4


ESP-IDF Windows Installer Download

Open Source IoT Development Framework for ESP32

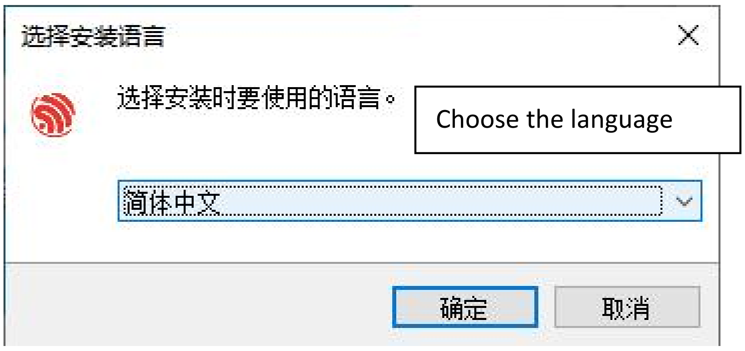
Universal Online Installer 2.28 Windows 10, 11 Size: 4.37 MB
Espressif-IDE 3.0.0 with ESP-IDF v5.2.2 - Offline Installer Windows 10, 11 Size: 1.93 GB
ESP-IDF v5.3.1 - Offline Installer Windows 10, 11 Size: 1.02 GB
ESP-IDF v5.2.2 - Offline Installer Windows 10, 11 Size: 1.01 GB
ESP-IDF v5.1.4 - Offline Installer Windows 10, 11 Size: 1.06 GB
ESP-IDF v5.0.7 - Offline Installer Windows 10, 11 Size: 0.96 GB

Generally, we use ESP-IDF only for compiling files. For editing, we can use VSCode, so we just need to download "ESP-IDF vx.x.x-Offline", x.x.x is the version number, the default download is the latest. You can also download "Espressif-IDE x.x.x with ESP-IDF vx.x.x -Offline" for IDE and IDF

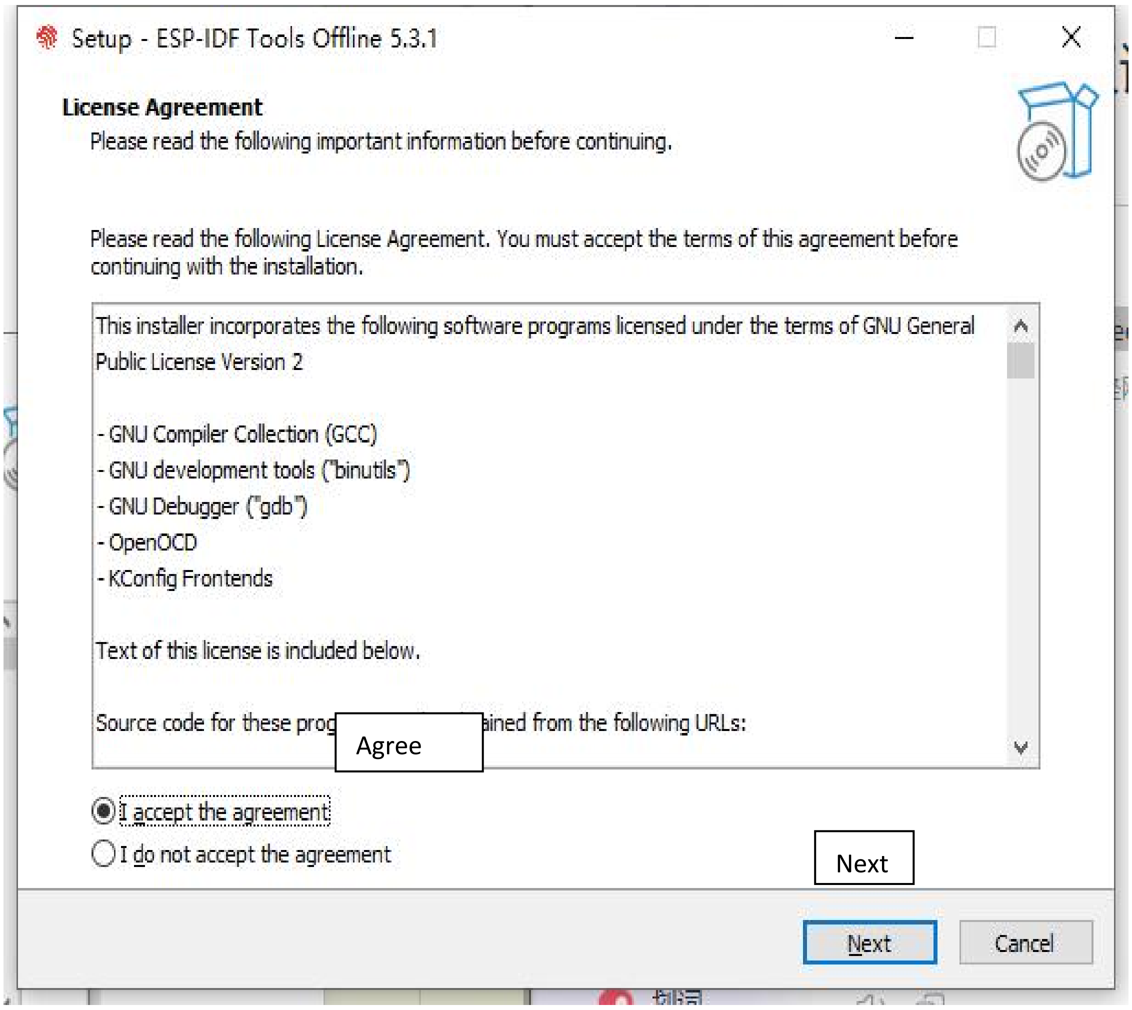
2. installation procedure

Here we will use "ESP-IDF v5.3.1-Offline" as an example to install the demo, the other versions of the same steps

(1) Step 1：Choose the language



(2) Step 2: License the agreement


Setup - ESP-IDF Tools Offline 5.3.1

License Agreement

Please read the following important information before continuing.

Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.

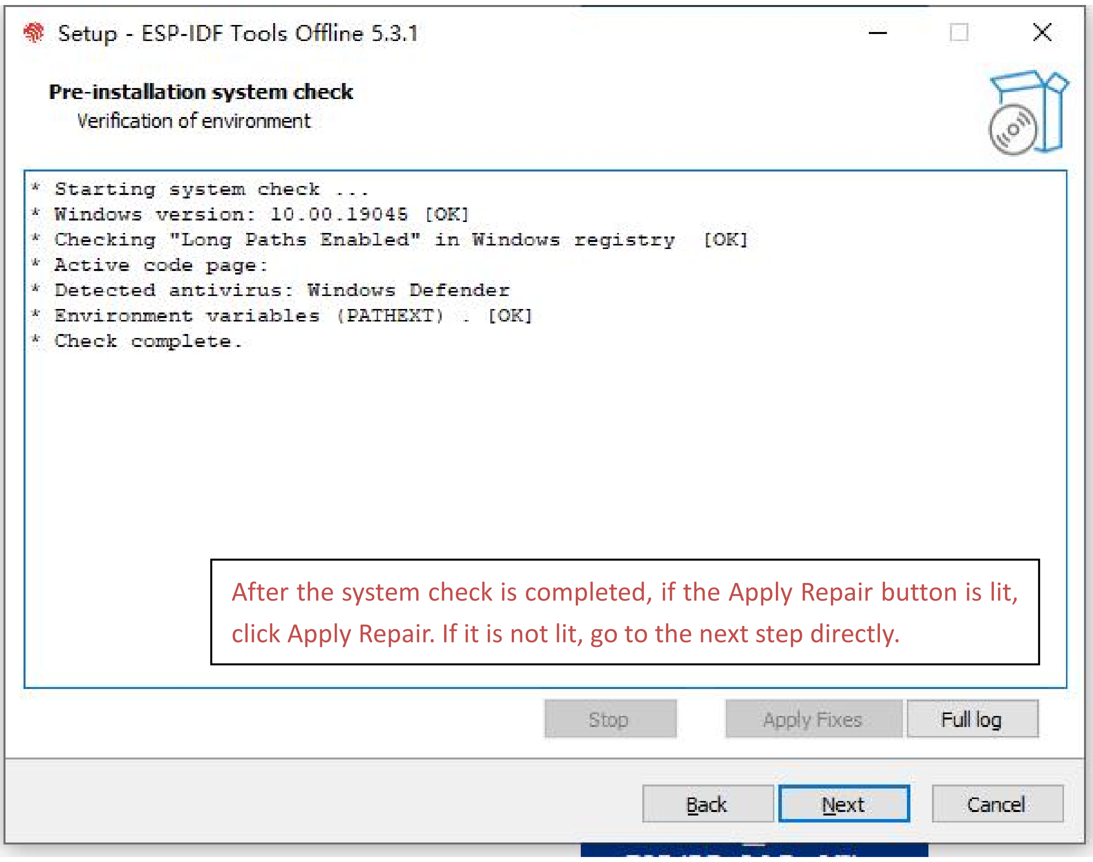
This installer incorporates the following software programs licensed under the terms of GNU General Public License Version 2

- GNU Compiler Collection (GCC)
- GNU development tools ("binutils")
- GNU Debugger ("gdb")
- OpenOCD
- KConfig Frontends

Text of this license is included below.

Source code for these prog ained from the following URLs:

(3) Step 3: Environment validation

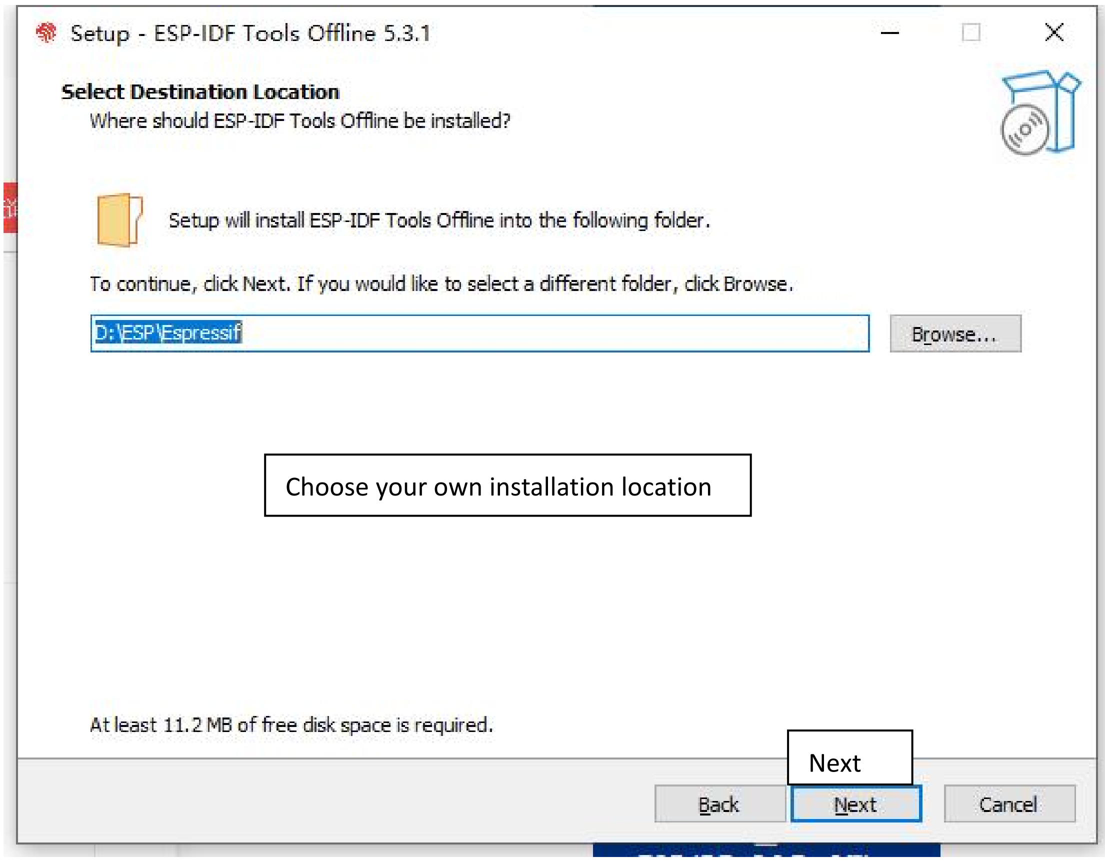

Setup - ESP-IDF Tools Offline 5.3.1

Pre-installation system check
Verification of environment

```
* Starting system check ...
* Windows version: 10.00.19045 [OK]
* Checking "Long Paths Enabled" in Windows registry  [OK]
* Active code page:
* Detected antivirus: Windows Defender
* Environment variables (PATHEXT) . [OK]
* Check complete.
```

After the system check is completed, if the Apply Repair button is lit, click Apply Repair. If it is not lit, go to the next step directly.

(4) Step 4: Select an installation path


Setup - ESP-IDF Tools Offline 5.3.1

Select Destination Location
Where should ESP-IDF Tools Offline be installed?

Setup will install ESP-IDF Tools Offline into the following folder.

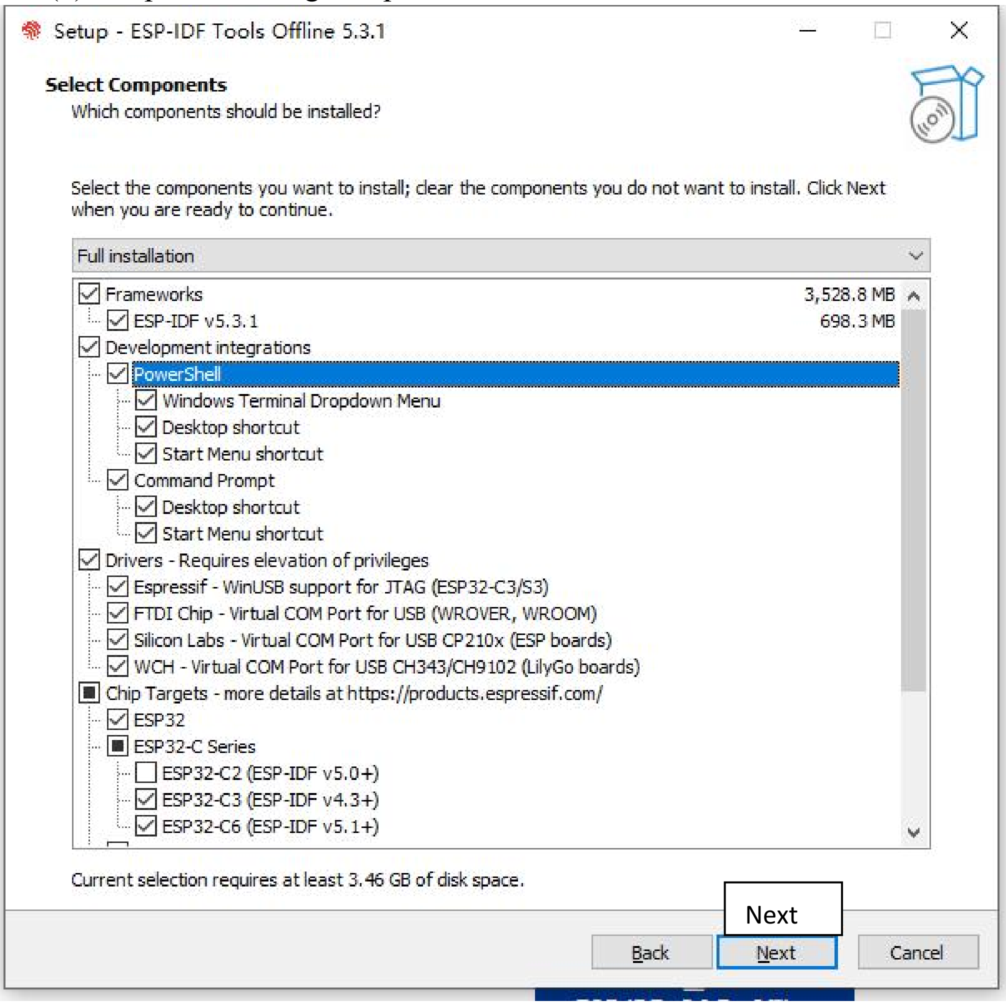
To continue, click Next. If you would like to select a different folder, click Browse.

D:\ESP\Espressif

Choose your own installation location

At least 11.2 MB of free disk space is required.

(5) Step 5: Selecting components



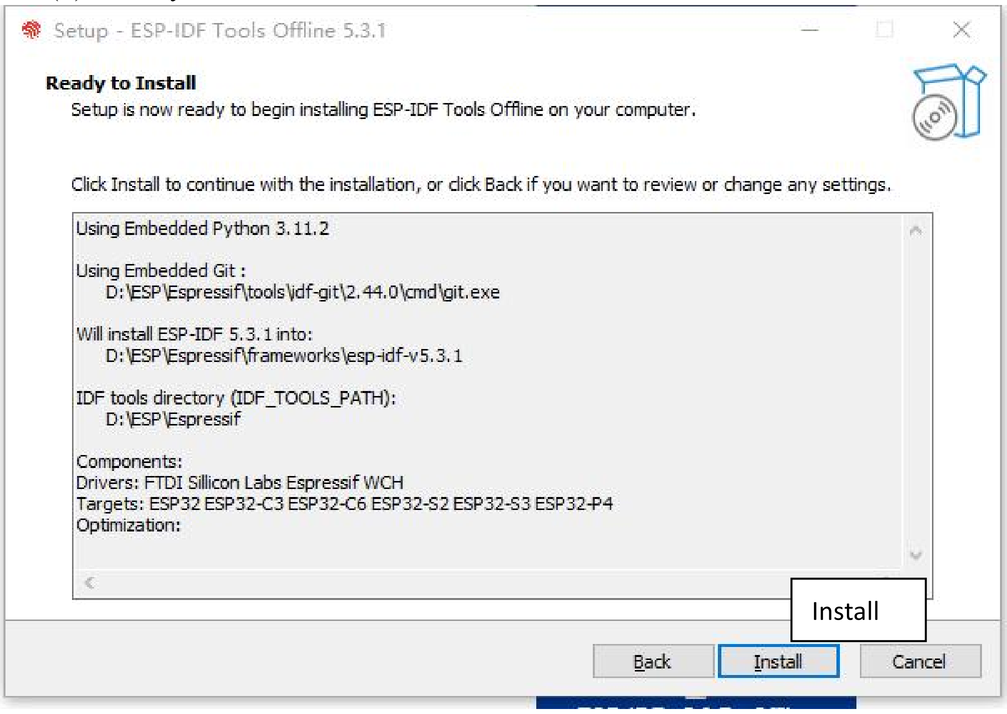
Select Components
Which components should be installed?

Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

- [✓] Frameworks — 3,528.8 MB
  - [✓] ESP-IDF v5.3.1 — 698.3 MB
- [✓] Development integrations
  - [✓] PowerShell
    - [✓] Windows Terminal Dropdown Menu
    - [✓] Desktop shortcut
    - [✓] Start Menu shortcut
  - [✓] Command Prompt
    - [✓] Desktop shortcut
    - [✓] Start Menu shortcut
- [✓] Drivers - Requires elevation of privileges
  - [✓] Espressif - WinUSB support for JTAG (ESP32-C3/S3)
  - [✓] FTDI Chip - Virtual COM Port for USB (WROVER, WROOM)
  - [✓] Silicon Labs - Virtual COM Port for USB CP210x (ESP boards)
  - [✓] WCH - Virtual COM Port for USB CH343/CH9102 (LilyGo boards)
- [■] Chip Targets - more details at https://products.espressif.com/
  - [✓] ESP32
  - [■] ESP32-C Series
    - [ ] ESP32-C2 (ESP-IDF v5.0+)
    - [✓] ESP32-C3 (ESP-IDF v4.3+)
    - [✓] ESP32-C6 (ESP-IDF v5.1+)

Current selection requires at least 3.46 GB of disk space.

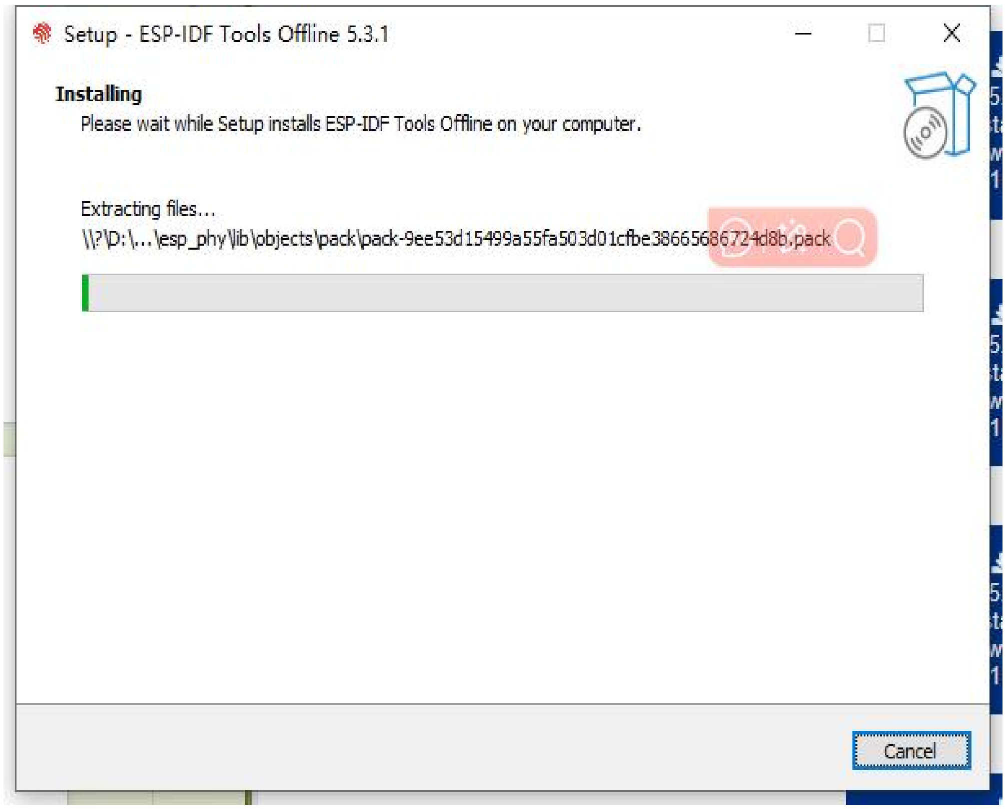
(6) Ready to installation



Ready to Install
Setup is now ready to begin installing ESP-IDF Tools Offline on your computer.

Click Install to continue with the installation, or click Back if you want to review or change any settings.

```
Using Embedded Python 3.11.2

Using Embedded Git :
    D:\ESP\Espressif\tools\idf-git\2.44.0\cmd\git.exe

Will install ESP-IDF 5.3.1 into:
    D:\ESP\Espressif\frameworks\esp-idf-v5.3.1

IDF tools directory (IDF_TOOLS_PATH):
    D:\ESP\Espressif

Components:
Drivers: FTDI Silicon Labs Espressif WCH
Targets: ESP32 ESP32-C3 ESP32-C6 ESP32-S2 ESP32-S3 ESP32-P4
Optimization:
```

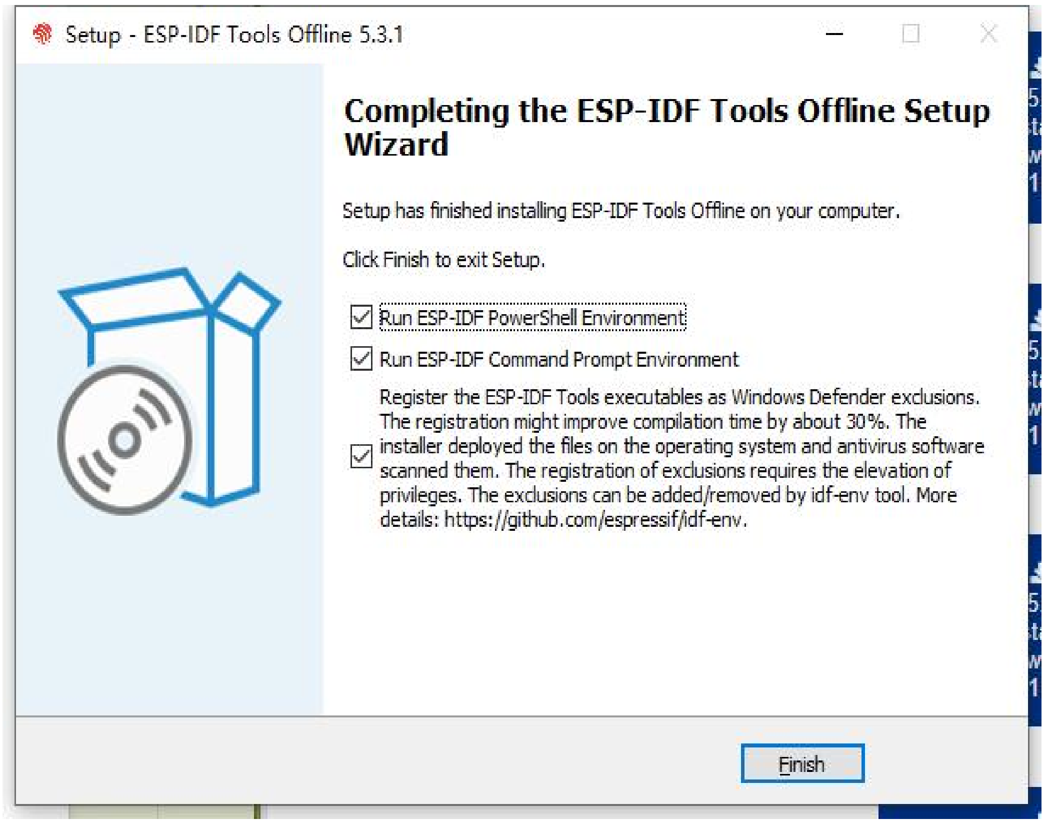


Setup - ESP-IDF Tools Offline 5.3.1

Installing

Please wait while Setup installs ESP-IDF Tools Offline on your computer.

Extracting files...
\\?\D:\...\esp_phy\lib\objects\pack\pack-9ee53d15499a55fa503d01cfbe38665686724d8b.pack



Setup - ESP-IDF Tools Offline 5.3.1

Completing the ESP-IDF Tools Offline Setup Wizard

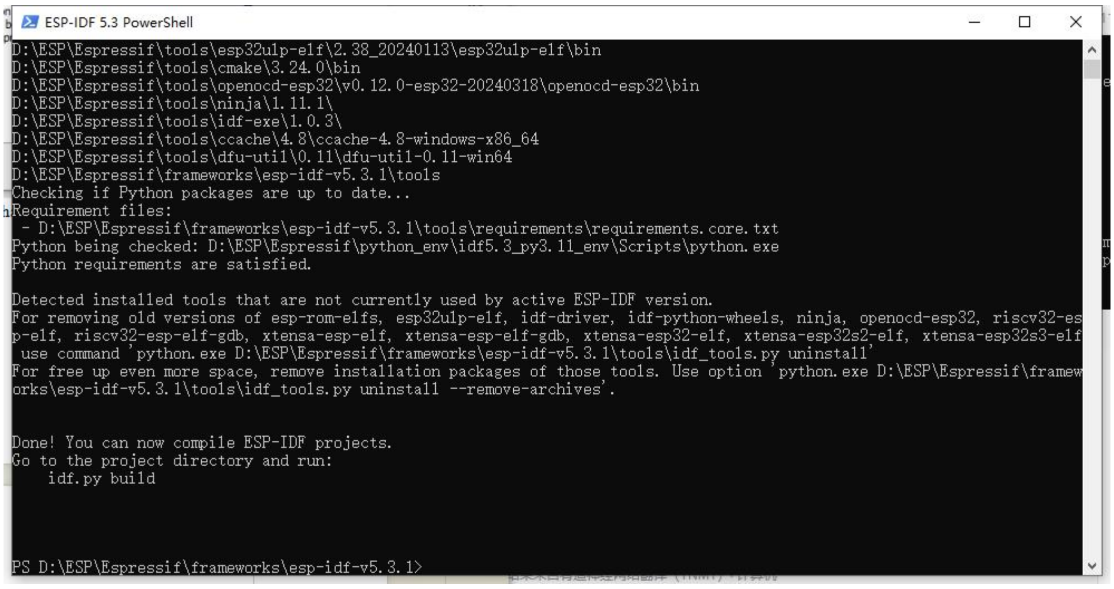
Setup has finished installing ESP-IDF Tools Offline on your computer.

Click Finish to exit Setup.

- [✓] Run ESP-IDF PowerShell Environment
- [✓] Run ESP-IDF Command Prompt Environment
- [✓] Register the ESP-IDF Tools executables as Windows Defender exclusions. The registration might improve compilation time by about 30%. The installer deployed the files on the operating system and antivirus software scanned them. The registration of exclusions requires the elevation of privileges. The exclusions can be added/removed by idf-env tool. More details: https://github.com/espressif/idf-env.

(7) Two Windows should appear indicating that the installation is complete, as shown in the following listing

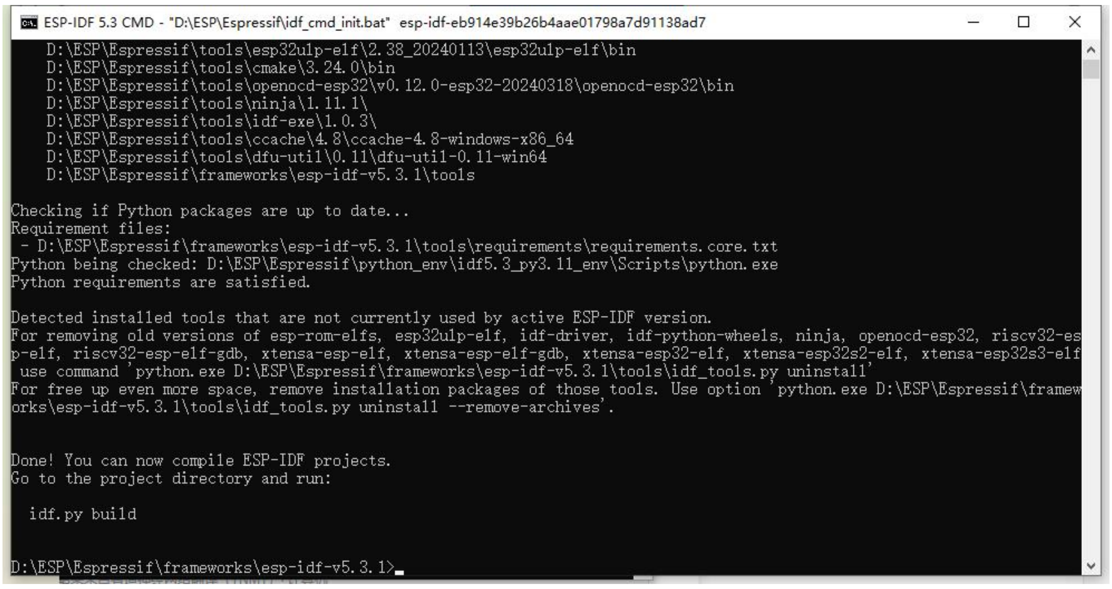
1.



```
D:\ESP\Espressif\tools\esp32ulp-elf\2.38_20240113\esp32ulp-elf\bin
D:\ESP\Espressif\tools\cmake\3.24.0\bin
D:\ESP\Espressif\tools\openocd-esp32\v0.12.0-esp32-20240318\openocd-esp32\bin
D:\ESP\Espressif\tools\ninja\1.11.1\
D:\ESP\Espressif\tools\idf-exe\1.0.3\
D:\ESP\Espressif\tools\ccache\4.8\ccache-4.8-windows-x86_64
D:\ESP\Espressif\tools\dfu-util\0.11\dfu-util-0.11-win64
D:\ESP\Espressif\frameworks\esp-idf-v5.3.1\tools
Checking if Python packages are up to date...
Requirement files:
 - D:\ESP\Espressif\frameworks\esp-idf-v5.3.1\tools\requirements\requirements.core.txt
Python being checked: D:\ESP\Espressif\python_env\idf5.3_py3.11_env\Scripts\python.exe
Python requirements are satisfied.

Detected installed tools that are not currently used by active ESP-IDF version.
For removing old versions of esp-rom-elfs, esp32ulp-elf, idf-driver, idf-python-wheels, ninja, openocd-esp32, riscv32-es
p-elf, riscv32-esp-elf-gdb, xtensa-esp-elf, xtensa-esp-elf-gdb, xtensa-esp32-elf, xtensa-esp32s2-elf, xtensa-esp32s3-elf
 use command 'python.exe D:\ESP\Espressif\frameworks\esp-idf-v5.3.1\tools\idf_tools.py uninstall'
For free up even more space, remove installation packages of those tools. Use option 'python.exe D:\ESP\Espressif\framew
orks\esp-idf-v5.3.1\tools\idf_tools.py uninstall --remove-archives'.


Done! You can now compile ESP-IDF projects.
Go to the project directory and run:
    idf.py build


PS D:\ESP\Espressif\frameworks\esp-idf-v5.3.1>
```

Corresponding icon:

2.


ESP-IDF 5.3 CMD - "D:\ESP\Espressif\idf_cmd_init.bat" esp-idf-eb914e39b26b4aae01798a7d91138ad7

```
    D:\ESP\Espressif\tools\esp32ulp-elf\2.38_20240113\esp32ulp-elf\bin
    D:\ESP\Espressif\tools\cmake\3.24.0\bin
    D:\ESP\Espressif\tools\openocd-esp32\v0.12.0-esp32-20240318\openocd-esp32\bin
    D:\ESP\Espressif\tools\ninja\1.11.1\
    D:\ESP\Espressif\tools\idf-exe\1.0.3\
    D:\ESP\Espressif\tools\ccache\4.8\ccache-4.8-windows-x86_64
    D:\ESP\Espressif\tools\dfu-util\0.11\dfu-util-0.11-win64
    D:\ESP\Espressif\frameworks\esp-idf-v5.3.1\tools

Checking if Python packages are up to date...
Requirement files:
 - D:\ESP\Espressif\frameworks\esp-idf-v5.3.1\tools\requirements\requirements.core.txt
Python being checked: D:\ESP\Espressif\python_env\idf5.3_py3.11_env\Scripts\python.exe
Python requirements are satisfied.

Detected installed tools that are not currently used by active ESP-IDF version.
For removing old versions of esp-rom-elfs, esp32ulp-elf, idf-driver, idf-python-wheels, ninja, openocd-esp32, riscv32-es
p-elf, riscv32-esp-elf-gdb, xtensa-esp-elf, xtensa-esp-elf-gdb, xtensa-esp32-elf, xtensa-esp32s2-elf, xtensa-esp32s3-elf
 use command 'python.exe D:\ESP\Espressif\frameworks\esp-idf-v5.3.1\tools\idf_tools.py uninstall'
For free up even more space, remove installation packages of those tools. Use option 'python.exe D:\ESP\Espressif\framew
orks\esp-idf-v5.3.1\tools\idf_tools.py uninstall --remove-archives'.


Done! You can now compile ESP-IDF projects.
Go to the project directory and run:

  idf.py build


D:\ESP\Espressif\frameworks\esp-idf-v5.3.1>_
```

Corresponding icon:

Software Operation Process

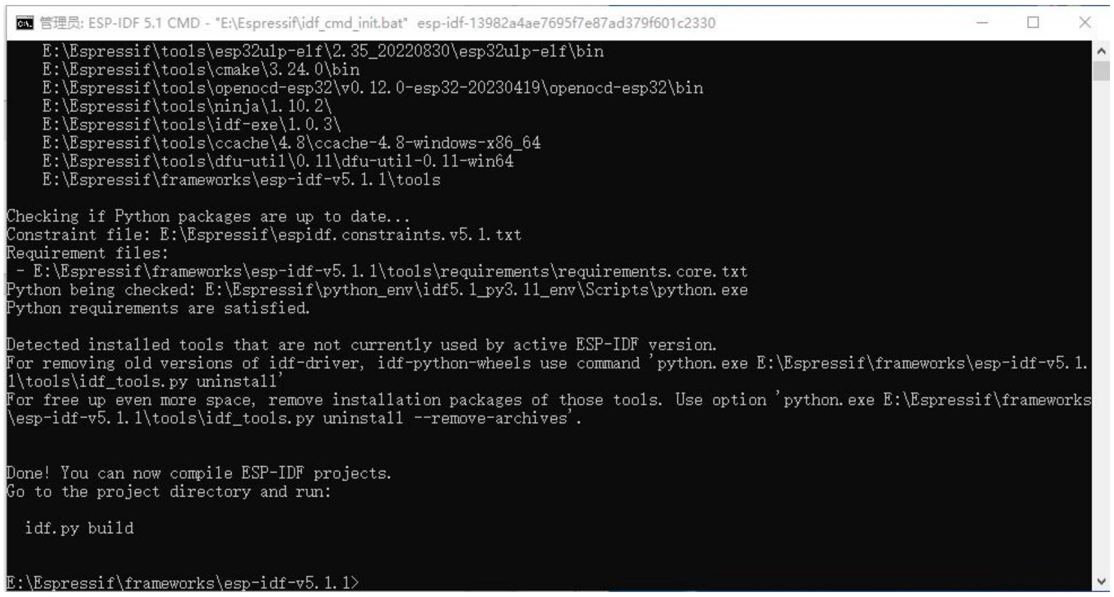
1. Common idf directives

(1) Set the chip model：idf.py set-target esp32xx
    ① Choose the corresponding model according to the chip you use, including C2,C3,C6,S2,S3, etc. If you use S3, it is： idf.py set-target esp32S3
(2) Open the configuration menu：idf.py menuconfig
(3) Compile：idf.py build
(4) Burn the program to the specified serial port：idf.py -p COM3 flash
(5) Open the serial port monitor： idf.py -p COM3 monitor
(6) Clear the configuration and build files：idf.py fullclean
(7) Compile files clearly：idf.py clean
(8) view folders：dir
(9) Return to the previous directory：cd ..
(10) Go to the specified folder:cd path

2. Configure the project build environment

For IDF project files generally we put in a specific location, here I take my own as an example my project files stored in D:\ESP\Espressif\frameworks\esp-idf-v5.3.1\examples\get-started,Locate your folder and place the project in it.

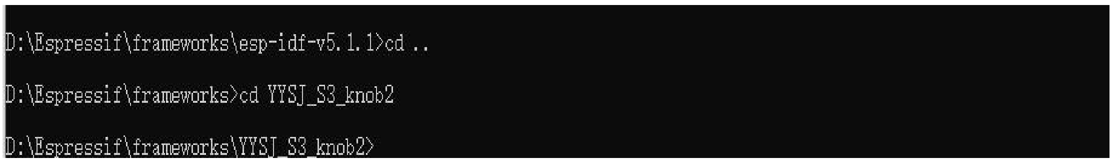
(11) Open Software,After opening, directly locate the installation directory



管理员: ESP-IDF 5.1 CMD - "E:\Espressif\idf_cmd_init.bat" esp-idf-13982a4ae7695f7e87ad379f601c2330

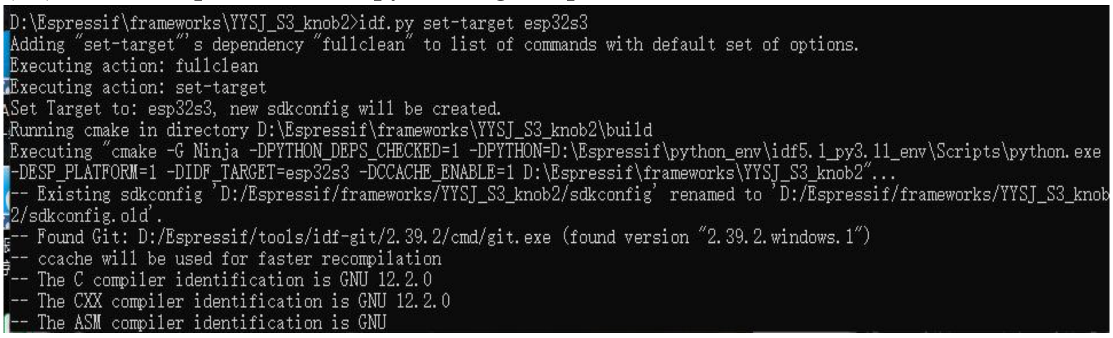
```
    E:\Espressif\tools\esp32ulp-elf\2.35_20220830\esp32ulp-elf\bin
    E:\Espressif\tools\cmake\3.24.0\bin
    E:\Espressif\tools\openocd-esp32\v0.12.0-esp32-20230419\openocd-esp32\bin
    E:\Espressif\tools\ninja\1.10.2\
    E:\Espressif\tools\idf-exe\1.0.3\
    E:\Espressif\tools\ccache\4.8\ccache-4.8-windows-x86_64
    E:\Espressif\tools\dfu-util\0.11\dfu-util-0.11-win64
    E:\Espressif\frameworks\esp-idf-v5.1.1\tools

Checking if Python packages are up to date...
Constraint file: E:\Espressif\espidf.constraints.v5.1.txt
Requirement files:
 - E:\Espressif\frameworks\esp-idf-v5.1.1\tools\requirements\requirements.core.txt
Python being checked: E:\Espressif\python_env\idf5.1_py3.11_env\Scripts\python.exe
Python requirements are satisfied.

Detected installed tools that are not currently used by active ESP-IDF version.
For removing old versions of idf-driver, idf-python-wheels use command 'python.exe E:\Espressif\frameworks\esp-idf-v5.1.
1\tools\idf_tools.py uninstall'
For free up even more space, remove installation packages of those tools. Use option 'python.exe E:\Espressif\frameworks
\esp-idf-v5.1.1\tools\idf_tools.py uninstall --remove-archives'.


Done! You can now compile ESP-IDF projects.
Go to the project directory and run:

  idf.py build


E:\Espressif\frameworks\esp-idf-v5.1.1>
```

(12)Go to the project path



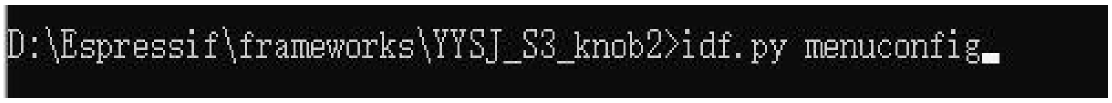
```
D:\Espressif\frameworks\esp-idf-v5.1.1>cd ..

D:\Espressif\frameworks>cd YYSJ_S3_knob2

D:\Espressif\frameworks\YYSJ_S3_knob2>
```

(13)Set the chip model：idf.py set-target esp32xx



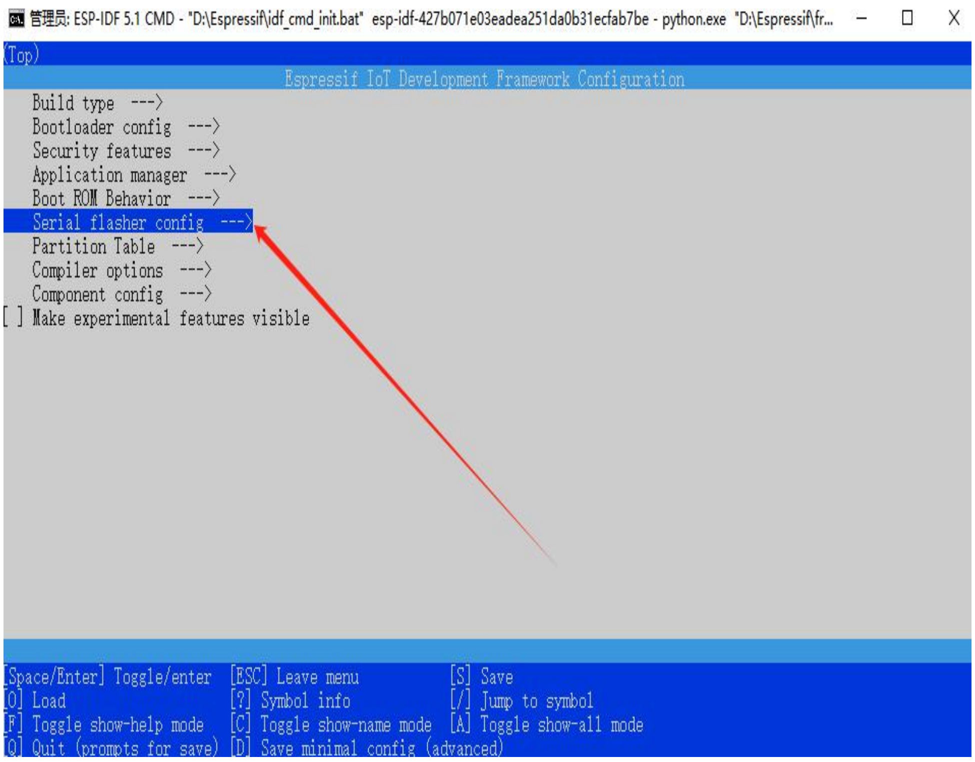
```
D:\Espressif\frameworks\YYSJ_S3_knob2>idf.py set-target esp32s3
Adding "set-target"'s dependency "fullclean" to list of commands with default set of options.
Executing action: fullclean
Executing action: set-target
Set Target to: esp32s3, new sdkconfig will be created.
Running cmake in directory D:\Espressif\frameworks\YYSJ_S3_knob2\build
Executing "cmake -G Ninja -DPYTHON_DEPS_CHECKED=1 -DPYTHON=D:\Espressif\python_env\idf5.1_py3.11_env\Scripts\python.exe
-DESP_PLATFORM=1 -DIDF_TARGET=esp32s3 -DCCACHE_ENABLE=1 D:\Espressif\frameworks\YYSJ_S3_knob2"...
-- Existing sdkconfig 'D:/Espressif/frameworks/YYSJ_S3_knob2/sdkconfig' renamed to 'D:/Espressif/frameworks/YYSJ_S3_knob
2/sdkconfig.old'.
-- Found Git: D:/Espressif/tools/idf-git/2.39.2/cmd/git.exe (found version "2.39.2.windows.1")
-- ccache will be used for faster recompilation
-- The C compiler identification is GNU 12.2.0
-- The CXX compiler identification is GNU 12.2.0
-- The ASM compiler identification is GNU
```

(14)Open the configuration menu,Change Uart port flash configuration：

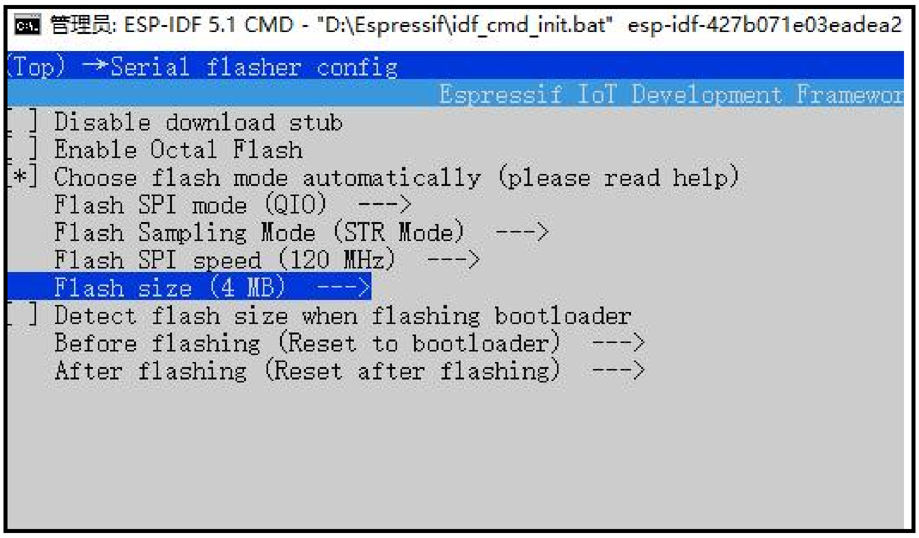


```
D:\Espressif\frameworks\YYSJ_S3_knob2>idf.py menuconfig
```

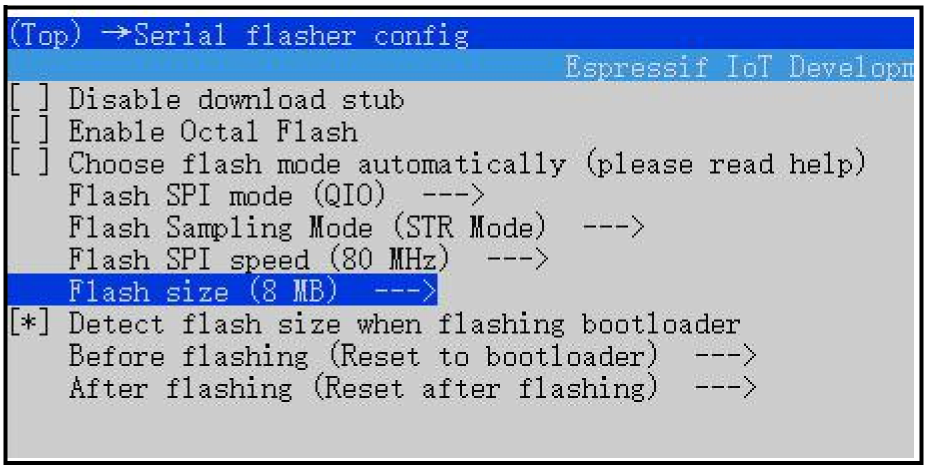
After entering the configuration menu, the flash should be configured firstly. Generally, the flash size should be configured to 4MB or 8MB according to the flash size of the chip, and the Flash SPI speed should be configured to 80MH or 120MHz, generally 80



```
管理员: ESP-IDF 5.1 CMD - "D:\Espressif\idf_cmd_init.bat" esp-idf-427b071e03eadea251da0b31ecfab7be - python.exe "D:\Espressif\fr...
(Top)
                    Espressif IoT Development Framework Configuration
    Build type  --->
    Bootloader config  --->
    Security features  --->
    Application manager  --->
    Boot ROM Behavior  --->
    Serial flasher config  --->
    Partition Table  --->
    Compiler options  --->
    Component config  --->
[ ] Make experimental features visible

[Space/Enter] Toggle/enter   [ESC] Leave menu           [S] Save
[O] Load                     [?] Symbol info            [/] Jump to symbol
[F] Toggle show-help mode    [C] Toggle show-name mode  [A] Toggle show-all mode
[Q] Quit (prompts for save)  [D] Save minimal config (advanced)
```

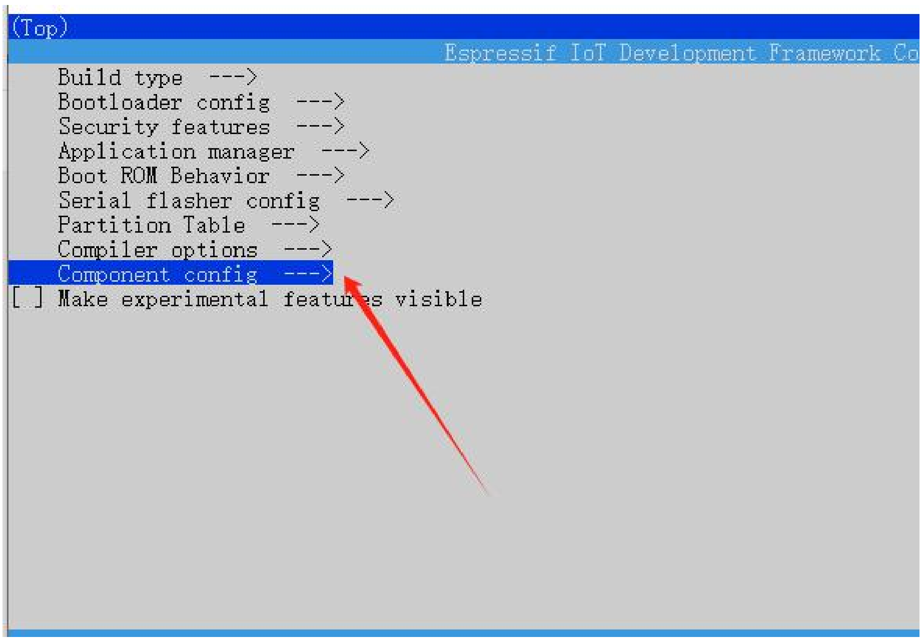


```
管理员: ESP-IDF 5.1 CMD - "D:\Espressif\idf_cmd_init.bat"  esp-idf-427b071e03eadea2
(Top) →Serial flasher config
                                    Espressif IoT Development Framewor
[ ] Disable download stub
[ ] Enable Octal Flash
[*] Choose flash mode automatically (please read help)
    Flash SPI mode (QIO)  --->
    Flash Sampling Mode (STR Mode)  --->
    Flash SPI speed (120 MHz)  --->
    Flash size (4 MB)  --->
[ ] Detect flash size when flashing bootloader
    Before flashing (Reset to bootloader)  --->
    After flashing (Reset after flashing)  --->
```

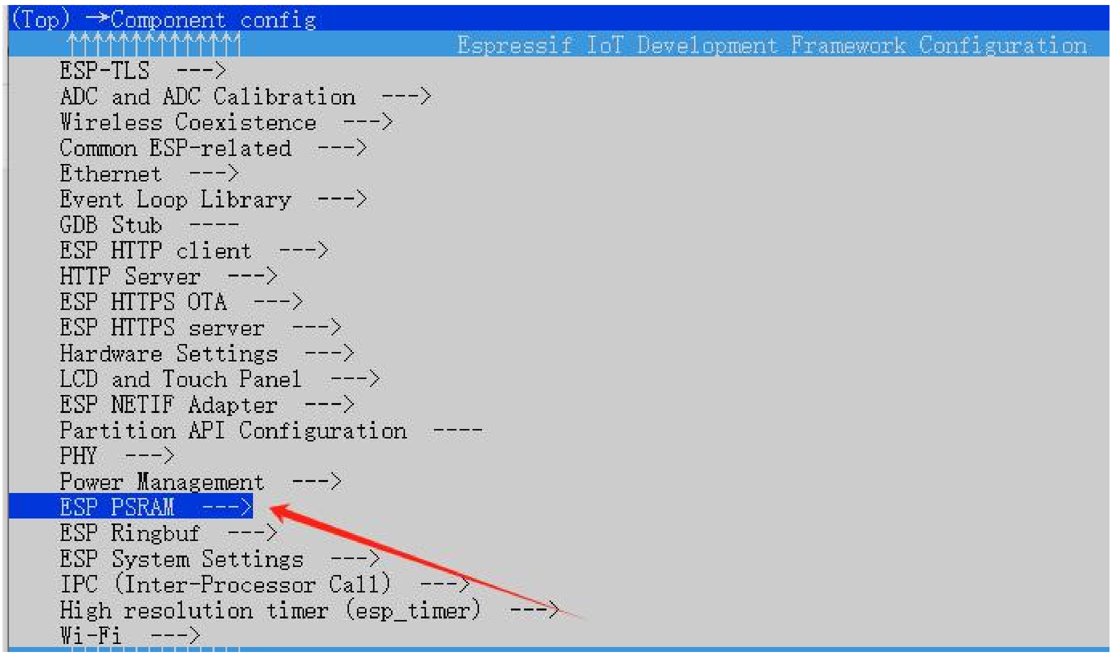


```
(Top) →Serial flasher config
                                    Espressif IoT Developm
[ ] Disable download stub
[ ] Enable Octal Flash
[ ] Choose flash mode automatically (please read help)
    Flash SPI mode (QIO)  --->
    Flash Sampling Mode (STR Mode)  --->
    Flash SPI speed (80 MHz)  --->
    Flash size (8 MB)  --->
[*] Detect flash size when flashing bootloader
    Before flashing (Reset to bootloader)  --->
    After flashing (Reset after flashing)  --->
```

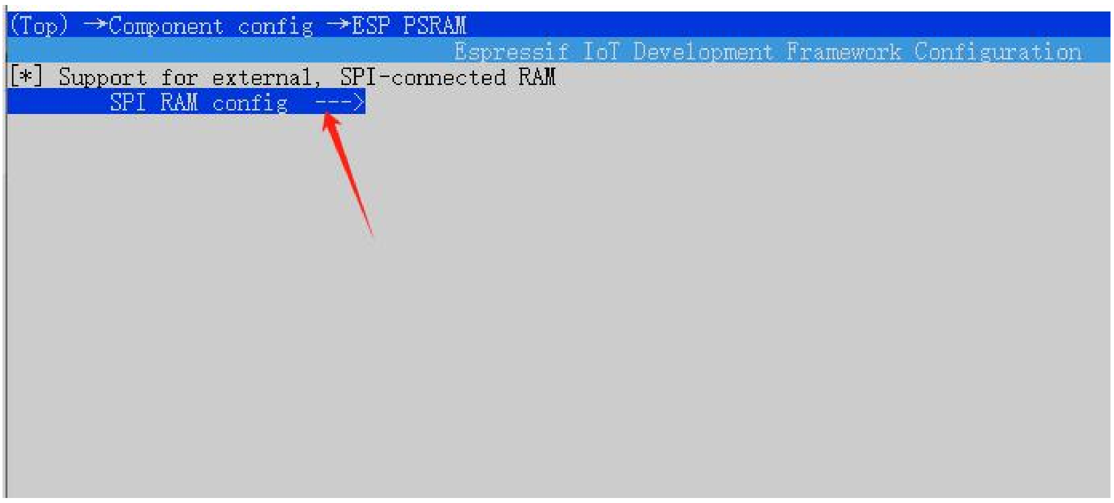
psram configuration:



```
(Top)
                                   Espressif IoT Development Framework Co
    Build type  --->
    Bootloader config  --->
    Security features  --->
    Application manager  --->
    Boot ROM Behavior  --->
    Serial flasher config  --->
    Partition Table  --->
    Compiler options  --->
    Component config  --->
[ ] Make experimental features visible
```

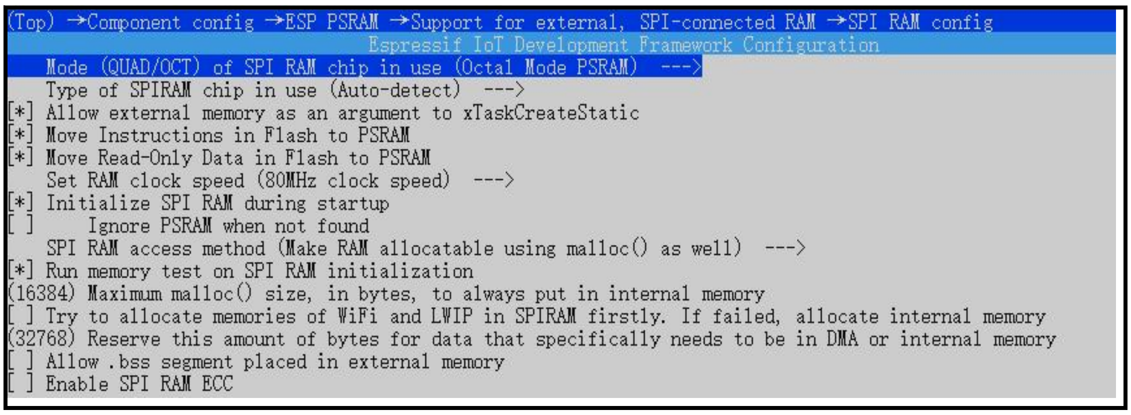


```
(Top) →Component config
                                    Espressif IoT Development Framework Configuration
    ESP-TLS  --->
    ADC and ADC Calibration  --->
    Wireless Coexistence  --->
    Common ESP-related  --->
    Ethernet  --->
    Event Loop Library  --->
    GDB Stub  ----
    ESP HTTP client  --->
    HTTP Server  --->
    ESP HTTPS OTA  --->
    ESP HTTPS server  --->
    Hardware Settings  --->
    LCD and Touch Panel  --->
    ESP NETIF Adapter  --->
    Partition API Configuration  ----
    PHY  --->
    Power Management  --->
    ESP PSRAM  --->
    ESP Ringbuf  --->
    ESP System Settings  --->
    IPC (Inter-Processor Call)  --->
    High resolution timer (esp_timer)  --->
    Wi-Fi  --->
```



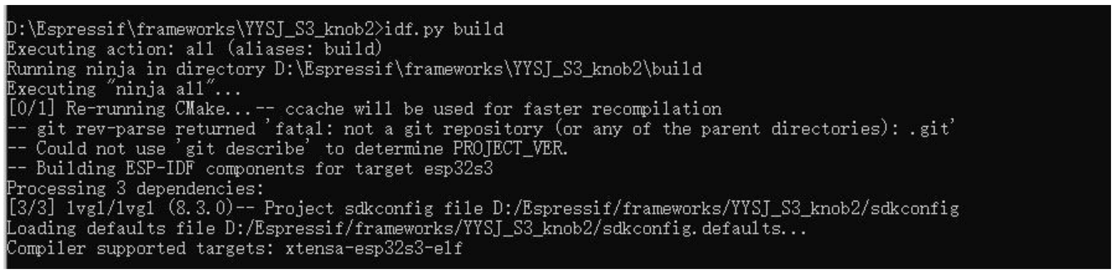
```
(Top) →Component config →ESP PSRAM
                              Espressif IoT Development Framework Configuration
[*] Support for external, SPI-connected RAM
        SPI RAM config  --->
```

Mode (OAD/OCT) is configured according to the external PSRAM of the chip. The four lines are OAD and the eight lines are OCT； Generally, RAM clock speed and Flash SPI speed are set to the same value



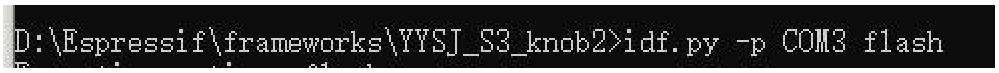
```
(Top) →Component config →ESP PSRAM →Support for external, SPI-connected RAM →SPI RAM config
                              Espressif IoT Development Framework Configuration
   Mode (QUAD/OCT) of SPI RAM chip in use (Octal Mode PSRAM)  --->
   Type of SPIRAM chip in use (Auto-detect)  --->
[*] Allow external memory as an argument to xTaskCreateStatic
[*] Move Instructions in Flash to PSRAM
[*] Move Read-Only Data in Flash to PSRAM
   Set RAM clock speed (80MHz clock speed)  --->
[*] Initialize SPI RAM during startup
[ ]     Ignore PSRAM when not found
   SPI RAM access method (Make RAM allocatable using malloc() as well)  --->
[*] Run memory test on SPI RAM initialization
(16384) Maximum malloc() size, in bytes, to always put in internal memory
[ ] Try to allocate memories of WiFi and LWIP in SPIRAM firstly. If failed, allocate internal memory
(32768) Reserve this amount of bytes for data that specifically needs to be in DMA or internal memory
[ ] Allow .bss segment placed in external memory
[ ] Enable SPI RAM ECC
```

(15) Compile



```
D:\Espressif\frameworks\YYSJ_S3_knob2>idf.py build
Executing action: all (aliases: build)
Running ninja in directory D:\Espressif\frameworks\YYSJ_S3_knob2\build
Executing "ninja all"...
[0/1] Re-running CMake...-- ccache will be used for faster recompilation
-- git rev-parse returned 'fatal: not a git repository (or any of the parent directories): .git'
-- Could not use 'git describe' to determine PROJECT_VER.
-- Building ESP-IDF components for target esp32s3
Processing 3 dependencies:
[3/3] lvgl/lvgl (8.3.0)-- Project sdkconfig file D:/Espressif/frameworks/YYSJ_S3_knob2/sdkconfig
Loading defaults file D:/Espressif/frameworks/YYSJ_S3_knob2/sdkconfig.defaults...
Compiler supported targets: xtensa-esp32s3-elf
```

(16) Download the program to the specified Uart port



```
D:\Espressif\frameworks\YYSJ_S3_knob2>idf.py -p COM3 flash
```

Note: If the program has already been compiled, nothing else needs to be done; just go to the last step

3. FAQ

Question：How to burn when the burn does not go in？

Answer：Check the burn log to see the error

(1) such as A fatal error occurred: Wrong --chip argument => idf.py set-target esp32xx, modify the chip model execute the command: idf.py set-target esp32xx.

(2) If the display cannot detect the serial port, please check whether there are other events occupied the serial port, if not occupied, see whether the serial port driver is installed.