ESP32-IDF configuration tool

1、Introduction to ESP32-IDF configuration tool

In the root directory of the ESP32-IDF project, there is an sdkconfig file, which is used to store the configuration content of ESP32-IDF.

2、Activate the ESP-IDF development environment

Run the following command in the esp-idf tool directory

```
source ~/esp/esp-idf/export.sh
```

Note: Every time you open a new terminal, you need to activate the ESP-IDF development environment before you can compile the ESP-IDF project.

3、New Construction

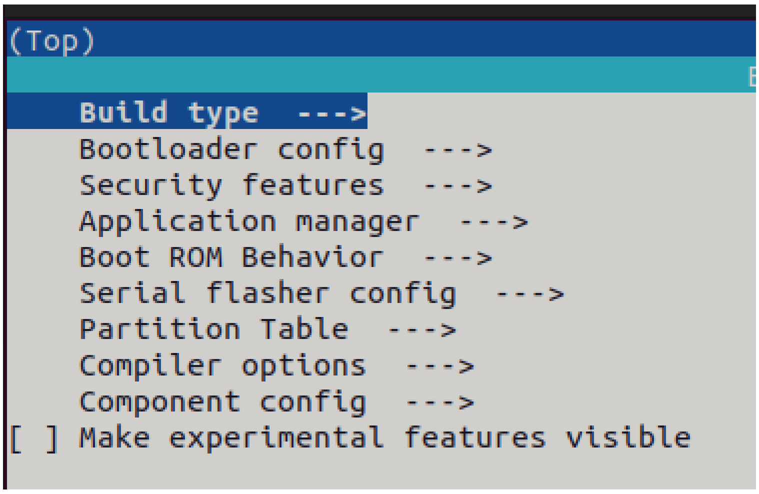
Create a new blank project to use as a basic project.

```
mkdir ~/esp/Samples/esp32_samples
cd ~/esp/Samples/esp32_samples
idf.py create-project main
cd main
```

4、Configuration project

Set the target chip to esp32s3 chip and open the configuration interface.

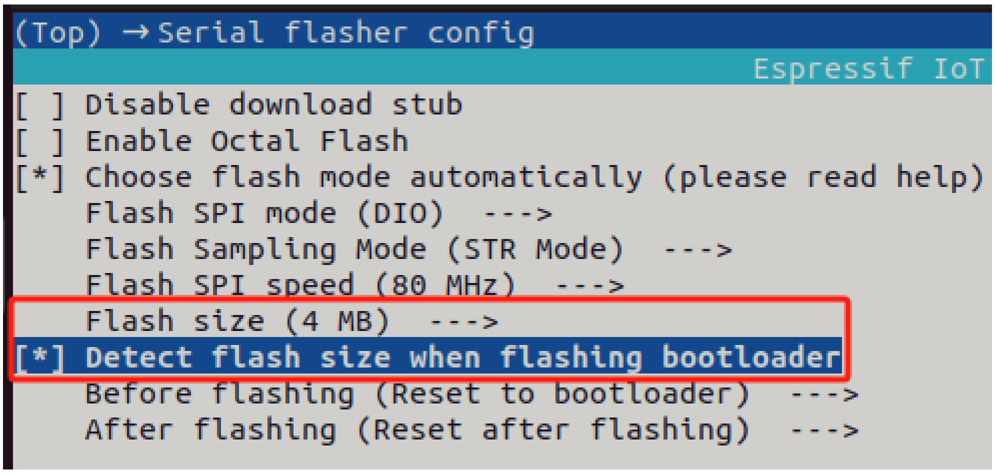
```
idf.py set-target esp32s3
idf.py menuconfig
```



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

1. Configure to automatically detect the flash size during burning.

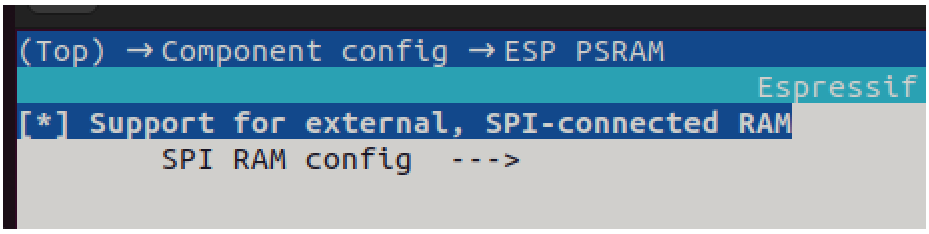
In the Serial flasher config, change the Flash size to 4MB and check Detect flash size when flashing bootloader.



```
(Top) → Serial flasher config
                                                Espressif IoT
[ ] Disable download stub
[ ] Enable Octal Flash
[*] Choose flash mode automatically (please read help)
    Flash SPI mode (DIO)  --->
    Flash Sampling Mode (STR Mode)  --->
    Flash SPI speed (80 MHz)  --->
    Flash size (4 MB)  --->
[*] Detect flash size when flashing bootloader
    Before flashing (Reset to bootloader)  --->
    After flashing (Reset after flashing)  --->
```

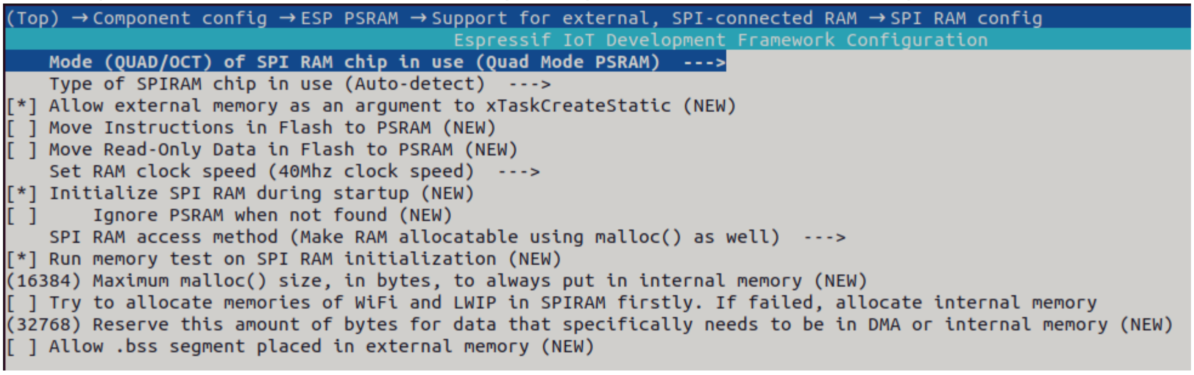
2. Configure external PSRAM

Open Component config->ESP PSRAM and enable Support for external, SPI-connected RAM.



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

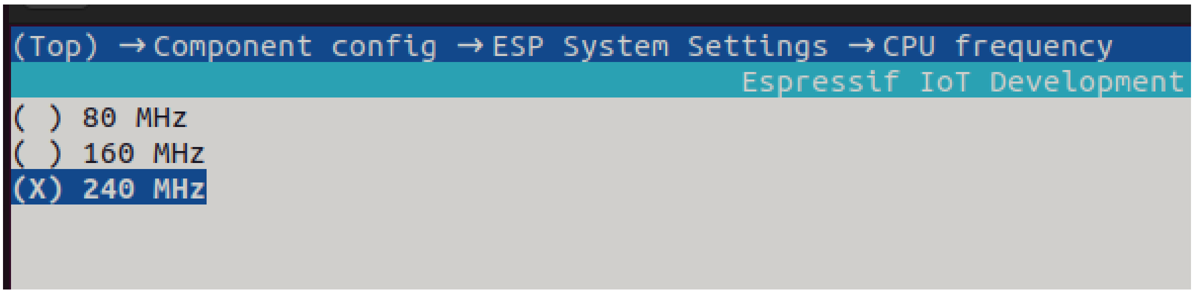
Just keep the default in the SPI RAM config configuration as shown in the figure below.



```
(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 (Quad Mode PSRAM)  --->
    Type of SPIRAM chip in use (Auto-detect)  --->
[*] Allow external memory as an argument to xTaskCreateStatic (NEW)
[ ] Move Instructions in Flash to PSRAM (NEW)
[ ] Move Read-Only Data in Flash to PSRAM (NEW)
    Set RAM clock speed (40Mhz clock speed)  --->
[*] Initialize SPI RAM during startup (NEW)
[ ]     Ignore PSRAM when not found (NEW)
    SPI RAM access method (Make RAM allocatable using malloc() as well)  --->
[*] Run memory test on SPI RAM initialization (NEW)
(16384) Maximum malloc() size, in bytes, to always put in internal memory (NEW)
[ ] 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 (NEW)
[ ] Allow .bss segment placed in external memory (NEW)
```

3. Configure the CPU frequency to 240MHz

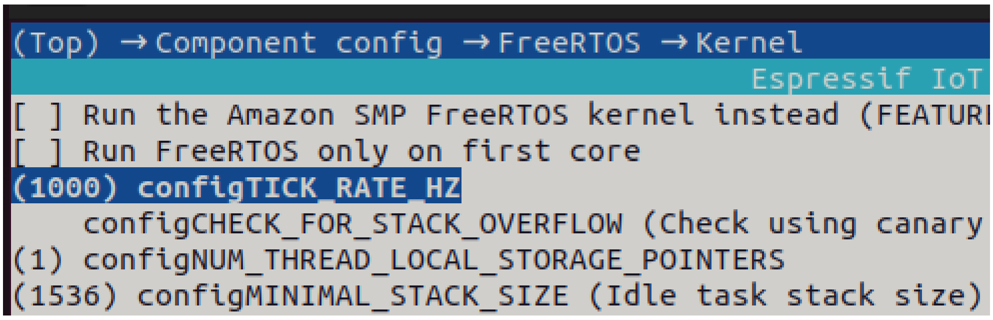
Open Component config->ESP System Settings->CPU frequency and change the CPU frequency to 240MHz.



```
(Top) →Component config →ESP System Settings →CPU frequency
                                    Espressif IoT Development
( ) 80 MHz
( ) 160 MHz
(X) 240 MHz
```

4.Configure the frequency of freertos to 1000hz

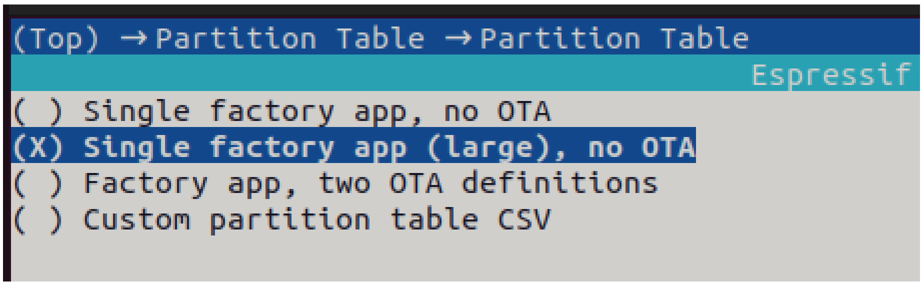
Open Component config->FreeRTOS->Kernel, find configTICK_RATE_HZ, and change the value to 1000.



```
(Top) →Component config →FreeRTOS →Kernel
                                               Espressif IoT
[ ] Run the Amazon SMP FreeRTOS kernel instead (FEATURI
[ ] Run FreeRTOS only on first core
(1000) configTICK_RATE_HZ
    configCHECK_FOR_STACK_OVERFLOW (Check using canary
(1) configNUM_THREAD_LOCAL_STORAGE_POINTERS
(1536) configMINIMAL_STACK_SIZE (Idle task stack size)
```

5.Configure partition table to increase program space

Open Partition Table->Partition Table, select Single factory app (large), no OTA.



```
(Top) →Partition Table →Partition Table
                                                Espressif
( ) Single factory app, no OTA
(X) Single factory app (large), no OTA
( ) Factory app, two OTA definitions
( ) Custom partition table CSV
```