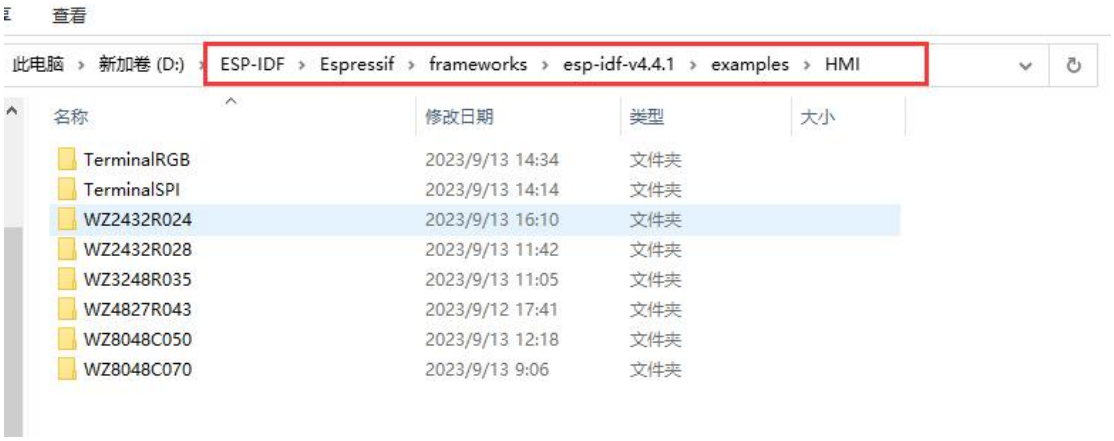
WZ8048C070 Use a tutorial

Place the downloaded project under the IDF directory (as shown below):



Let's first learn at the use of commands：

cd xxx---Moving to the xxx directory, xxx represents the name of the directory, for example: cd example

idf.py set-target esp32s3---Set the target chip for example: esp32s3

idf.py fullclean---Delete the entire build directory, including all the CMake configuration output files.

idf.py clean---It removes the building output files from the building directory and cleans up the entire project..

idf.py menuconfig---Configure the target chip

idf.py build---Compile a private code base

idf.py -p com3 flash---Download the program to the target chip

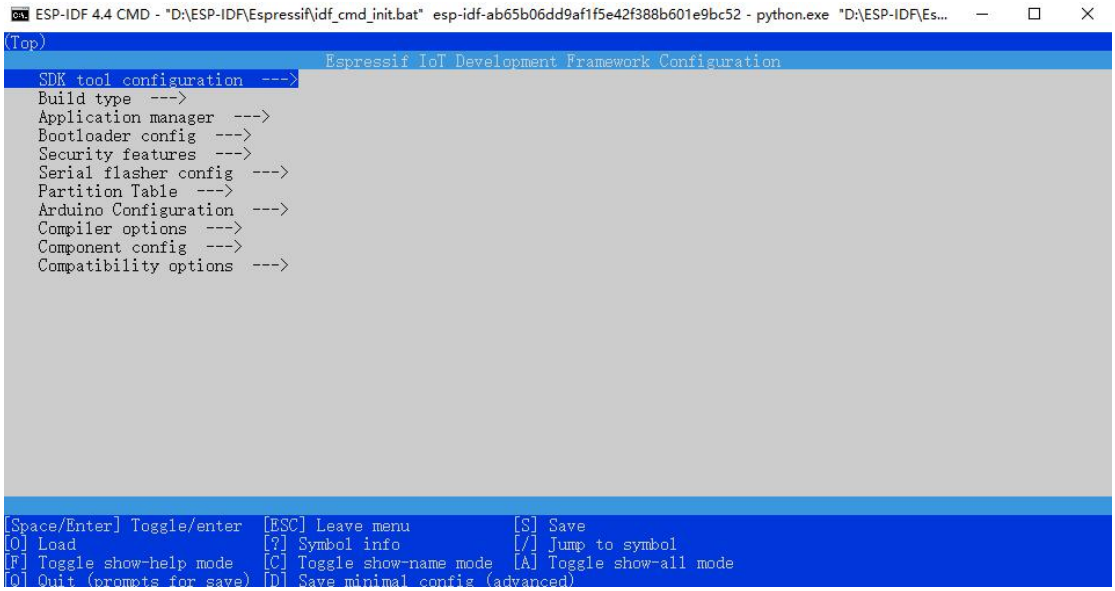
idf.py -p com3 flash monitor---Once compile burn and open monitoring

Now we open the terminal and go to the WZ8048C070 project catalog

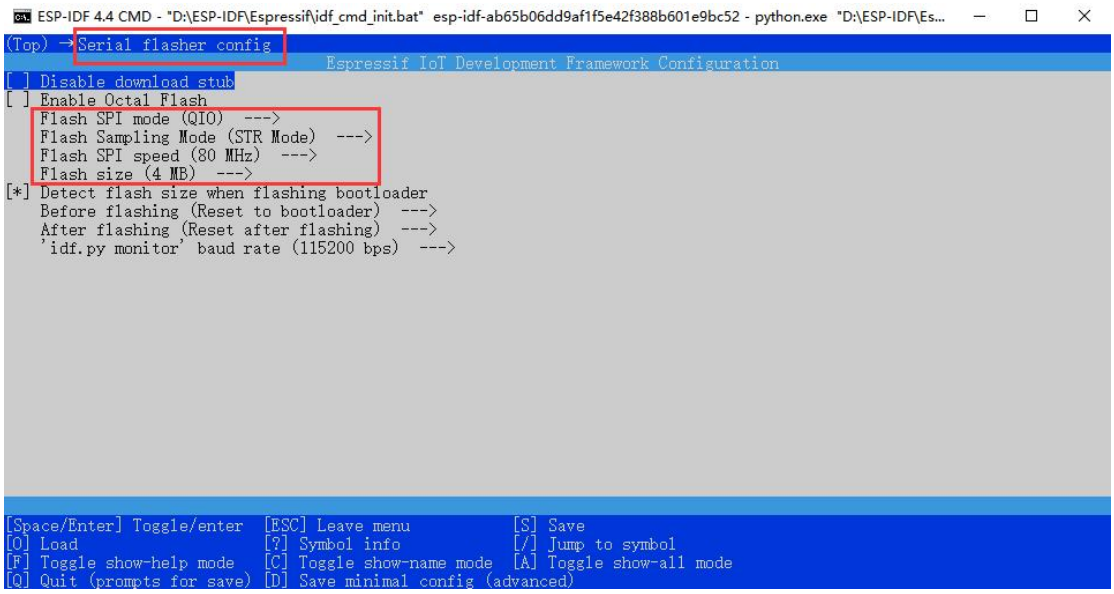
```
D:\ESP-IDF\Espressif\frameworks\esp-idf-v4.4.1\examples\HMI>cd WZ8048C070

D:\ESP-IDF\Espressif\frameworks\esp-idf-v4.4.1\examples\HMI\WZ8048C070>_
```

Now we have to empty the project idf.py fullclean once first, and then go into the configuration

```
D:\ESP-IDF\Espressif\frameworks\esp-idf-v4.4.1\examples\HMI\WZ8048C070>idf.py fullclean
Executing action: fullclean
Done
```



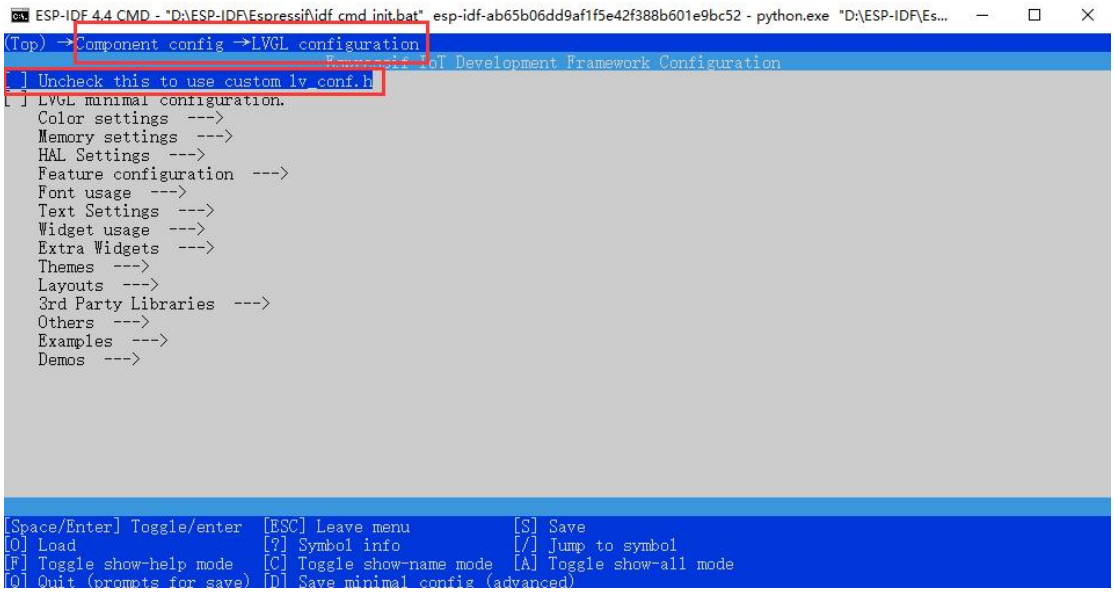
Now modify the options by following the following steps:



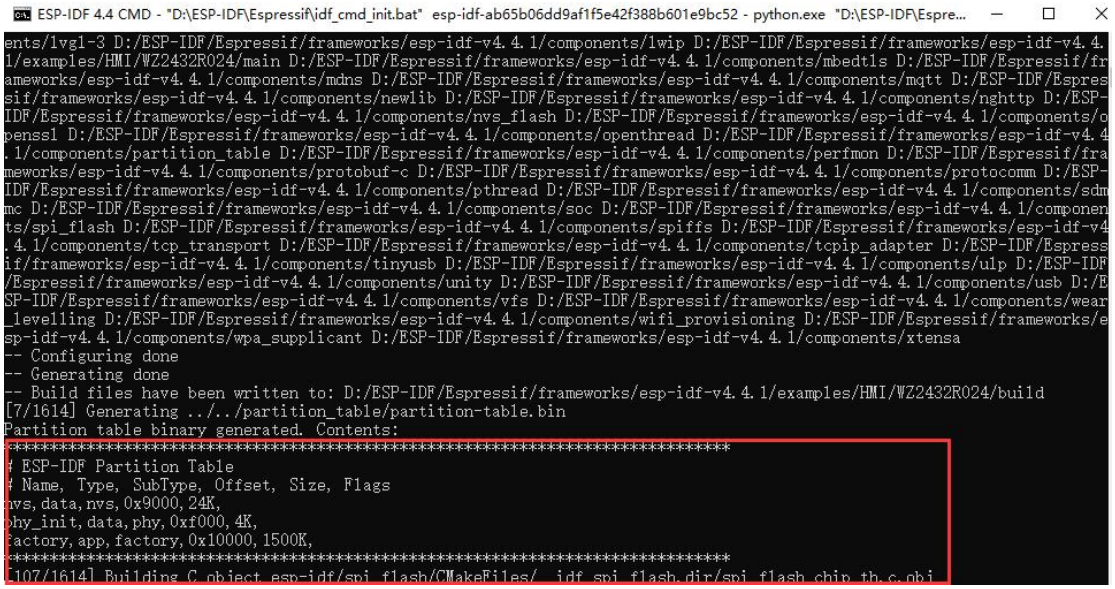
```
ESP-IDF 4.4 CMD - "D:\ESP-IDF\Espressif\idf_cmd_init.bat" esp-idf-ab65b06dd9af1f5e42f388b601e9bc52 - python.exe "D:\ESP-IDF\Es...
(Top) → Partition Table
                    Espressif IoT Development Framework Configuration
    Partition Table (Custom partition table CSV)  --->
(huge_app.csv) Custom partition CSV file
(0x8000) Offset of partition table
[*] Generate an MD5 checksum for the partition table

[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)
```

```
ESP-IDF 4.4 CMD - "D:\ESP-IDF\Espressif\idf_cmd_init.bat" esp-idf-ab65b06dd9af1f5e42f388b601e9bc52 - python.exe "D:\ESP-IDF\Es...
(Top) → Component config → ESP32S3-Specific
                    Espressif IoT Development Framework Configuration
    CPU frequency (160 MHz)  --->
    Cache config      >
[*] Support for external, SPI-connected RAM
        SPI RAM config  --->
[ ] Use TRAX tracing feature
[ ] Enable Ultra Low Power (ULP) Coprocessor
[*] Make exception and panic handlers JTAG/OCD aware
[*] Hardware brownout detect & reset
        Brownout voltage level (2.44V)  --->
    Timers used for gettimeofday function (RTC and high-resolution timer)  --->
    RTC clock source (Internal 150kHz RC oscillator)  --->
(1024) Number of cycles for RTC_SLOW_CLK calibration
(2000) Extra delay in deep sleep wake stub (in us)
[ ] No Binary Blobs
[ ] Place RTC_DATA_ATTR and RTC_RODATA_ATTR variables into RTC fast memory segment
[ ] Use fixed static RAM size

[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)
```

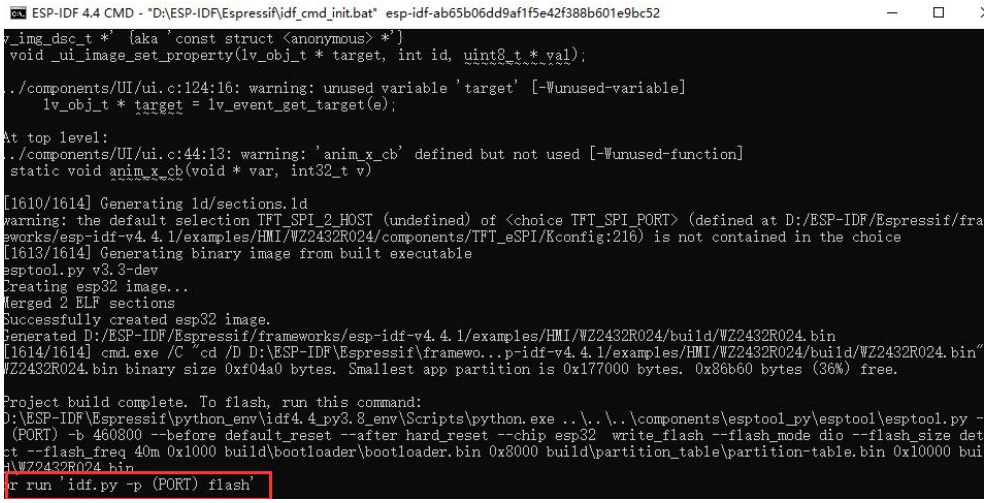
```
ESP-IDF 4.4 CMD - "D:\ESP-IDF\Espressif\idf_cmd_init.bat" esp-idf-ab65b06dd9af1f5e42f388b601e9bc52 - python.exe "D:\ESP-IDF\Es...
(Top) → Component config → ESP32S3-Specific → Support for external, SPI-connected RAM → SPI RAM config
                    Espressif IoT Development Framework Configuration
    Mode (QUAD/OCT) of SPI RAM chip in use (Octal Mode PSRAM)  --->
    Type of SPIRAM chip in use (Auto-detect)  --->
    PSRAM Clock and CS IO for ESP32S3  --->
[ ] Cache fetch instructions from SPI RAM
[ ] Cache load read only data from SPI RAM
    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

[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)
```



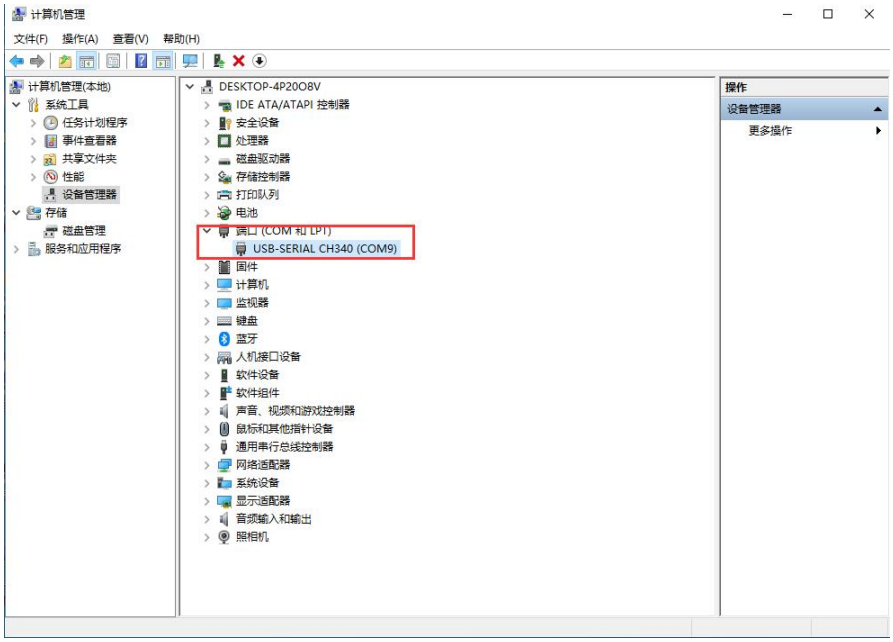
Save the exit after setup, and then execute the idf.py build



Waiting for the compilation to complete, the following figure interface appears:



Perform the idf.py-p com9 flash



success!