

***Package na***

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## FreeRTOS(教程非常详细）

<https://blog.csdn.net/qq_61672347/article/details/125748646>

统一链接

### FreeRTOS移植到STM32

#### 一、找一个STM32的裸机工程模板

随便找的一个裸机程序

#### 二、去官网上下载FreeRTOS V9.0.0 源码

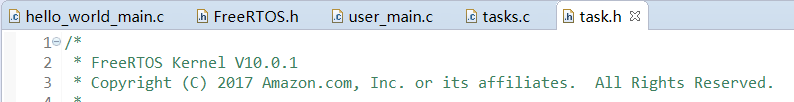
在移植之前，我们首先要获取到 FreeRTOS 的官方的源码包。这里我们提供两个下载 链 接 ，

一 个 是 官 网 ： http://www.freertos.org/

另 外 一 个 是 代 码 托 管 网 站 ： <https://sourceforge.net/projects/freertos/files/FreeRTOS/>。

虽然不是最新版本的源码包但是因为内核很稳定， 并且网上资料很多所以我们选用V9.0.0 版本

我们打开 FreeRTOS 的代码托管网站，就可以看到 FreeRTOS 的源码及其版本信息了， 具体见图



而在8266的最新版本的SDK中使用的是10.0.1的内核

##### v9.0.0 2018-8

##### v10.0.1 2017-12

/\*

\* FreeRTOS Kernel V10.0.1

这里尽可能的选择和8266一样的内核，但是相对来说V9.0.0的人用的多。看项目需要



去上面下载。exe的文件或者zip文件都是可行的，exe文件是7z的压缩文件，

解压完成后就会得到一个完整的Freertos源码包



FreeRTOS 包含 Demo 例程和内核源码（比较重要，我们就需要提取该目录下的大部分 文件），具体见图 13-5。

***1、FreeRTOS 文件夹下的 Source 文件夹里面包含的是 FreeRTOS 内 核的源代码***，我们移植 FreeRTOS 的时候就需要这部分源代码；

***2、FreeRTOS 文件夹下的 Demo 文件夹里面包含了 FreeRTOS 官方为各个单片机移植好的工程代码***，FreeRTOS 为了 推广自己，会给各种半导体厂商的评估板写好完整的工程程序，这些程序就放在 Demo 这 个目录下，这部分 Demo 非常有参考价值。我们把 FreeRTOS 到 STM32 的时候， ***FreeRTOSConfig.h*** 这个头文件就是从这里拷贝过来的，我们这里不做详解

|  |
| --- |
|  |

#### 三、往裸机工程添加 FreeRTOS 源码

##### 1、创建FreeRTOS文件夹

首先在我们的STM32 裸 机工程模板根目录下新建一个文件夹， 命名 为 “FreeRTOS”，并且在 FreeRTOS 文件夹下新建两个空文件夹，分别命名为“src” 与“port”。

1）src 文件夹用于保存 FreeRTOS 中的核心源文件（内核），也就是我们常说的 ‘.c 文件’。

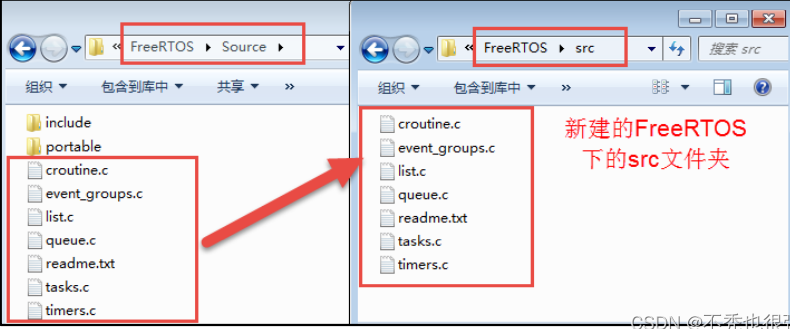
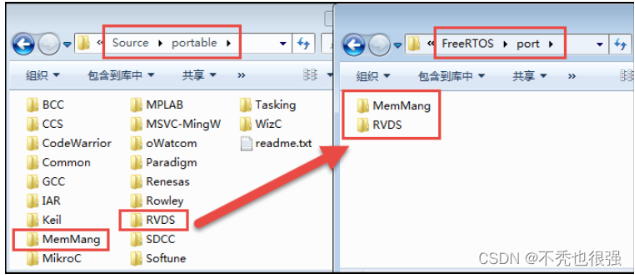
2）port 文件夹用于保存内存管理以及处理器架构相关代码，这些代码 FreeRTOS 官方已经提供给我们的，直接使用即可。

在前面已经说了，FreeRTOS 是软件，我们的开发版是硬件，软硬件必须有桥梁来连接，这些与处理器架构相 关的代码，可以称之为 RTOS 硬件接口层，它们位于 FreeRTOS/Source/Portable 文 件夹下。

##### 2、进行文件拷贝

###### 源文件

打开 FreeRTOS V9.0.0 源码，在“FreeRTOSv9.0.0\FreeRTOS\Source”目录下找到 所有的‘.c 文件’，将它们拷贝到我们新建的 src 文件夹中，

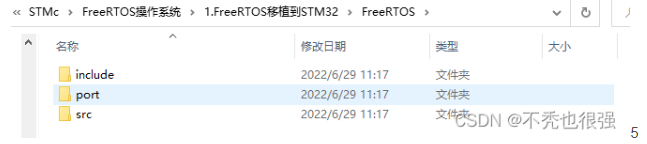
 

V9.0.0 源码，在“FreeRTOSv9.0.0\FreeRTOS\Source\portable”目 录下找到“MemMang”文件夹与“RVDS”文件夹，将它们拷贝到我们新建的 port 文件夹中，

###### 头文件

打开 FreeRTOS V9.0.0 源码，在“***FreeRTOSv9.0.0\ FreeRTOS\Source***”目录下找到 “***include***”文件夹，它是我们需要用到 FreeRTOS 的一些头文件，将它直接拷贝 到我们新建的 FreeRTOS 文件夹中，完成这一步之后就可以看到我们新建的 FreeRTOS 文件夹已经有 3 个文件夹，这 3个文件夹就包含 FreeRTOS 的核心文件， 至此，FreeRTOS 的源码就提取完成，

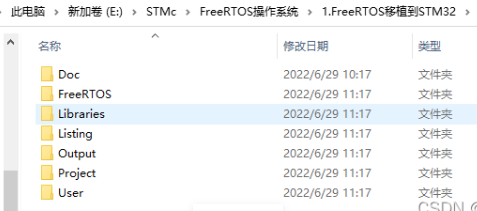
###### 拷贝完成后的FreeRTOS文件夹



##### 3、拷贝 FreeRTOS 到裸机工程根目录

表示这个FreeRTOS文件夹拷贝到任何的keil工程都是可以使用的，包含了具体的源码

鉴于 FreeRTOS 容量很小，我们直接将刚刚提取的整个 FreeRTOS 文件夹拷贝到我们 的 STM32 裸机工程里面，让整个 FreeRTOS 跟随我们的工程一起发布，使用这种方法打包 的 FreeRTOS 工程，即使是将工程拷贝到一台没有安装 FreeRTOS 支持包（MDK 中有 FreeRTOS 的支持包）的电脑上面都是可以直接使用的，因为工程已经包含了 FreeRTOS 的 源码。

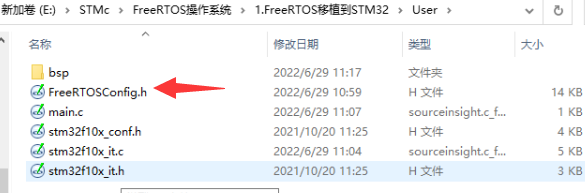


FreeRTOS 文件夹下就是我们提取的 FreeRTOS 的核心代码，该文件夹下的 具体内容作用，这里就不再赘述。

##### 4、拷贝 FreeRTOSConfig.h 文件到 user 文件夹

***FreeRTOSConfig.h*** 文件是 FreeRTOS 的工程配置文件，因为 FreeRTOS 是***可以裁剪的 实时操作内核***，应用于不同的处理器平台，用户可以通过修改这个 FreeRTOS 内核的配置 头文件来裁剪 FreeRTOS 的功能，所以我们把它拷贝一份放在 user 这个文件夹下面。

打开 FreeRTOSv9.0.0 源码，在“***FreeRTOSv9.0.0\FreeRTOS\Demo***”文件夹下面找到 “ ***CORTEX\_STM32F103\_Keil*** ” 这 个 文 件 夹 ， 双 击 打 开 ， 在 其 根 目 录 下 找 到 这 个 “FreeRTOSConfig.h”文件，然后拷贝到我们工程的 user 文件夹下即可，



##### 5、添加 FreeRTOS 源码到工程组文件夹

在上一步我们只是将 FreeRTOS 的源码放到了本地工程目录下，还没有添加到开发环 境里面的组文件夹里面，FreeRTOS 也就没有移植到我们的工程中去。

新建 FreeRTOS/src 和 FreeRTOS/port 组，接下来我们在开发环境里面新建 FreeRTOS/src 和 FreeRTOS/port 两个组文件夹，

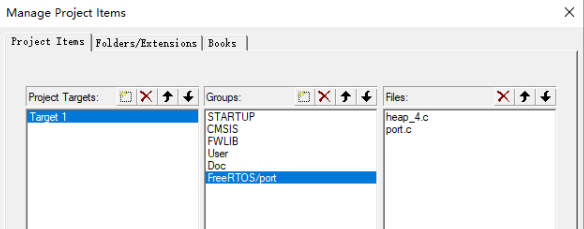
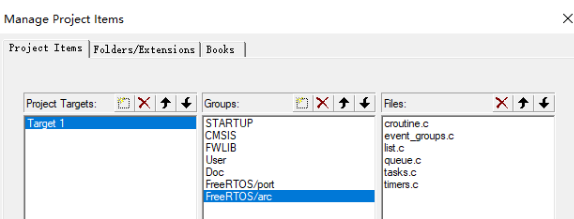
***FreeRTOS/src*** 用于存放 src 文件夹的内容，就是内核

***FreeRTOS/port*** 用于存放 port\MemMang 文件夹 （内存管理）与 port\RVDS\ARM\_CM？文件夹（具体什么架构CPU）的内容，“？”表示 3、4 或者 7，具体选择哪个得看你 使用的是哪个型号的 STM32 开发板。

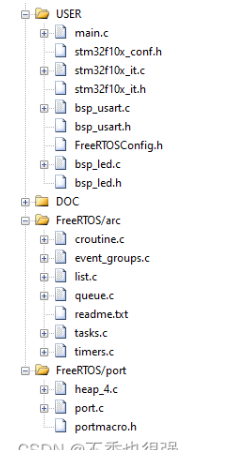
如果是F3系列就拷贝port\RVDS\ARM\_CM3，F4系列就拷贝port\RVDS\ARM\_CM4。

在 FreeRTOS/port 分组中添加 MemMang 文件夹中的文件只需选择其中一个即可，我们选择“***heap\_4.c***”，这是 FreeRTOS 的一个内存管理源码文件。

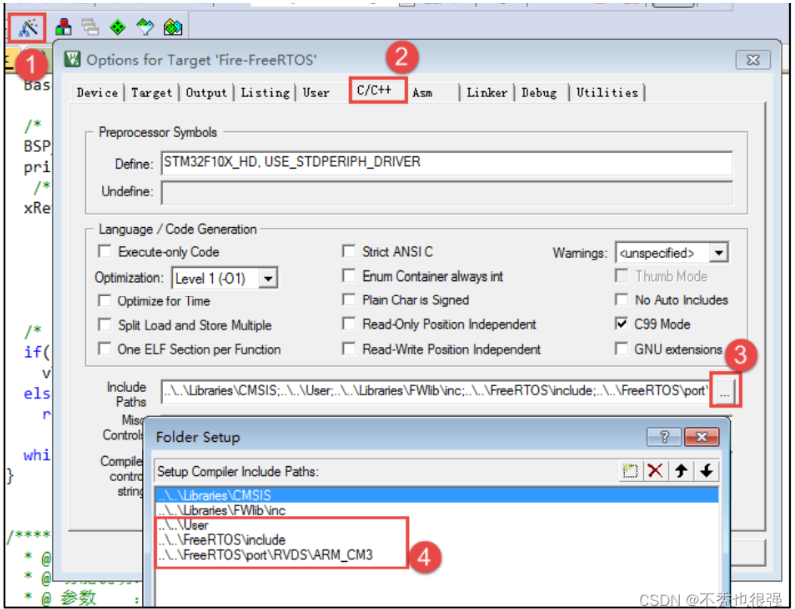
同时，需要根据自己的开 发板型号在 FreeRTOS\port\RVDS\ARM\_CM?中选择，“？”表示 3、4 或者 7，具体选择 哪个得看你使用的是哪个型号的 STM32 开发板。

然后在 user 分组中添加我们 FreeRTOS 的配置文件“FreeRTOSConfig.h”，因为这是 头文件（.h），所以需要在添加时选择文件类型为“All files (\*.\*)”，至此我们的 FreeRTOS 添加到工程中就已经完成，效果图如下



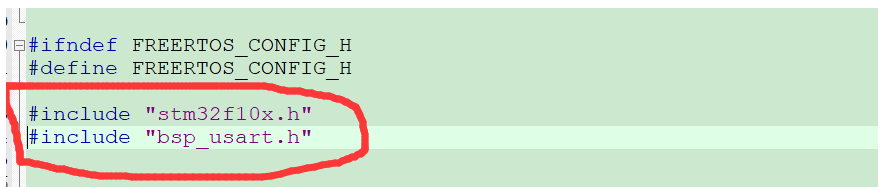
##### 6、指定 FreeRTOS 头文件的路径



这一步不过多解释，用过STM32的人都知道

##### 7、FreeRTOSConfig.h 文件修改

FreeRTOSConfig.h 头文件的内容修改的不多，具体是：修改与对应开发板的头文件 ， 如果是使用野火 STM32F1 的开发板，则包含 F1 的头文件#include "stm32f10x.h"，同理是 使用了其它系列的开发板，则包含与开发板对应的头文件即可，当然还需要包含我们的串 口的头文件“bsp\_usart.h”，因为在我们 FreeRTOSConfig.h 中实现了断言操作，需要打印 一些信息。其他根据需求修改即可，



##### 8、修改 stm32f10x\_it.

SysTick 中断服务函数是一个非常重要的函数，FreeRTOS 所有跟时间相关的事情都在 里面处理，SysTick 就是 FreeRTOS 的一个心跳时钟，驱动着 FreeRTOS 的运行，就像人的 心跳一样，假如没有心跳，我们就相当于“死了”，同样的，FreeRTOS 没有了心跳，那么 它就会卡死在某个地方，不能进行任务调度，不能运行任何的东西，因此我们需要实现一 个 FreeRTOS 的心跳时钟，FreeRTOS 帮我们实现了 SysTick 的启动的配置：在 port.c 文件 中已经实现 vPortSetupTimerInterrupt()函数，并且 FreeRTOS 通用的 SysTick 中断服务函数 也实现了：在 port.c 文件中已经实现 xPortSysTickHandler()函数，所以移植的时候只需要我 们在 stm32f10x\_it.c 文件中实现我们对应（STM32）平台上的 SysTick\_Handler()函数即可。 FreeRTOS 为开发者考虑得特别多，PendSV\_Handler()与 SVC\_Handler()这两个很重要的函 数都帮我们实现了，在 port.c 文件中已经实现 xPortPendSVHandler()与 vPortSVCHandler() 函数，防止我们自己实现不了，那么在 stm32f10x\_it.c 中就需要我们注释掉 PendSV\_Handler()与 SVC\_Handler()这两个函数了，具体看一下代码

|  |
| --- |
| /\* Includes ------------------------------------------------------------------\*/  #include "stm32f10x\_it.h"  #include "FreeRTOS.h" //FreeRTOS使用  #include "task.h"  /\*\* @addtogroup STM32F10x\_StdPeriph\_Template  \* @{  \*/  /\* Private typedef -----------------------------------------------------------\*/  /\* Private define ------------------------------------------------------------\*/  /\* Private macro -------------------------------------------------------------\*/  /\* Private variables ---------------------------------------------------------\*/  /\* Private function prototypes -----------------------------------------------\*/  /\* Private functions ---------------------------------------------------------\*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\* Cortex-M3 Processor Exceptions Handlers \*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\*\*  \* @brief This function handles NMI exception.  \* @param None  \* @retval None  \*/  void NMI\_Handler(void)  {  }    /\*\*  \* @brief This function handles Hard Fault exception.  \* @param None  \* @retval None  \*/  void HardFault\_Handler(void)  {  /\* Go to infinite loop when Hard Fault exception occurs \*/  while (1)  {  }  }  /\*\*  \* @brief This function handles Memory Manage exception.  \* @param None  \* @retval None  \*/  void MemManage\_Handler(void)  {  /\* Go to infinite loop when Memory Manage exception occurs \*/  while (1)  {  }  }  /\*\*  \* @brief This function handles Bus Fault exception.  \* @param None  \* @retval None  \*/  void BusFault\_Handler(void)  {  /\* Go to infinite loop when Bus Fault exception occurs \*/  while (1)  {  }  }  /\*\*  \* @brief This function handles Usage Fault exception.  \* @param None  \* @retval None  \*/  void UsageFault\_Handler(void)  {  /\* Go to infinite loop when Usage Fault exception occurs \*/  while (1)  {  }  }  /\*\*  \* @brief This function handles SVCall exception.  \* @param None  \* @retval None  \*/  //void SVC\_Handler(void)  //{  //}  /\*\*  \* @brief This function handles Debug Monitor exception.  \* @param None  \* @retval None  \*/  void DebugMon\_Handler(void)  {  }  extern void xPortSysTickHandler(void);  //systick中断服务函数  void SysTick\_Handler(void)  {  #if (INCLUDE\_xTaskGetSchedulerState == 1 )  if (xTaskGetSchedulerState() != taskSCHEDULER\_NOT\_STARTED)  {  #endif /\* INCLUDE\_xTaskGetSchedulerState \*/  xPortSysTickHandler();  #if (INCLUDE\_xTaskGetSchedulerState == 1 )  }  #endif /\* INCLUDE\_xTaskGetSchedulerState \*/  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\* STM32F10x Peripherals Interrupt Handlers \*/  /\* Add here the Interrupt Handler for the used peripheral(s) (PPP), for the \*/  /\* available peripheral interrupt handler's name please refer to the startup \*/  /\* file (startup\_stm32f10x\_xx.s). \*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\*\*  \* @brief This function handles PPP interrupt request.  \* @param None  \* @retval None  \*/  /\*void PPP\_IRQHandler(void)  {  }\*/ |

##### 9、修改main.c里面的代码

|  |
| --- |
| #include "FreeRTOS.h"  #include "task.h"  /\* 开发板硬件bsp头文件 \*/  #include "bsp\_led.h"  #include "bsp\_usart.h"      int main(void)  {  //里面什么都不做  } |

修改好后编译看看有没有错误

笔者这是没有任何问题的。

本篇内容参考野火的Freertos内核实现与应用开发指南，详细代码可以去野火官方下载中心下载查看

<https://doc.embedfire.com/products/link/zh/latest/index.html>

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## 8266系列资料

#### 参考链接

<http://aiclouds3.0-docs.aithinker.com/#/1.quick/background>

下载开发 SDK 资源包：

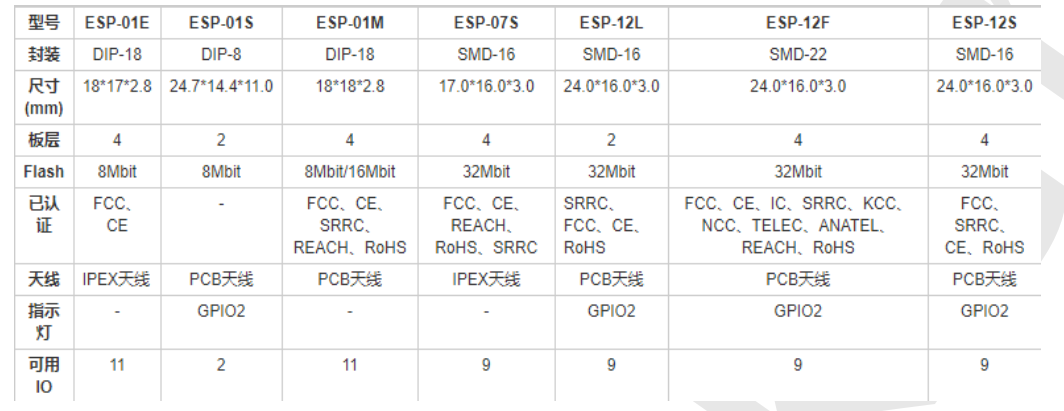
git clone --recursive <https://gitee.com/xuhongv/AiThinkerProjectForESP.git>

IDF\_PATH

|  |
| --- |
|  |

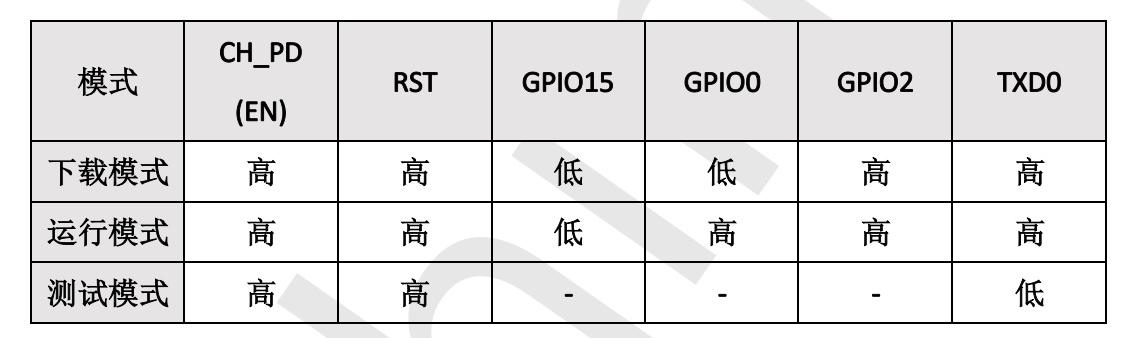
|  |
| --- |
|  |

#### 硬件资料



这里的32Mbit（32M位）就是4MB（4M字节），

#### 启动模式



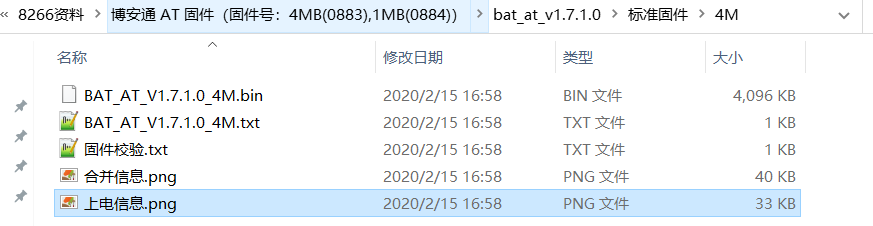
#### 上电信息

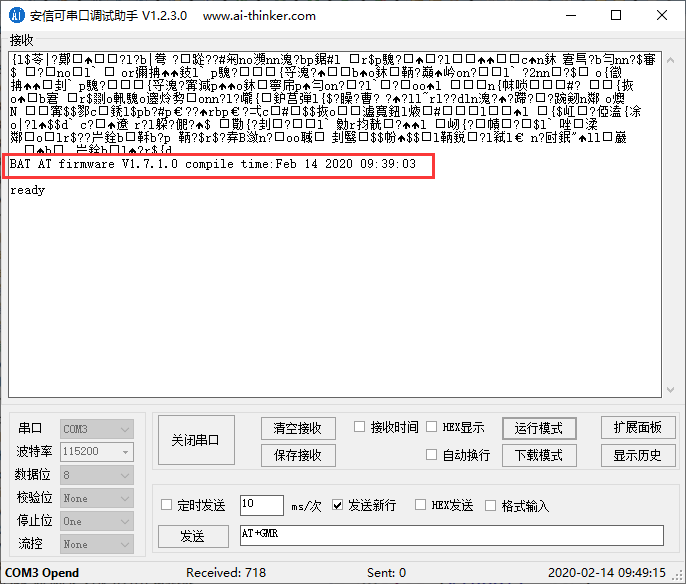
下面是购买后的出厂固件打印信息：

|  |
| --- |
| ################################################ 74880波特率 BootLoader 的程序  ets Jan 8 2013,rst cause:2, boot mode:(3,2)  load 0x40100000, len 6960, room 16  tail 0  chksum 0xa5  load 0x3ffe8008, len 24, room 8  tail 0  chksum 0x2c  load 0x3ffe8020, len 3196, room 8  tail 4  chksum 0xe7  csum 0xe7  ################################################ 115200波特率 系统启动完成  arch:ESP8266, 1  compile\_time:Ai-Thinker|B&T  wifi\_mac:4417933d4CCA  sdk\_version:v3.4-22-g967752e2  firmware\_version:2.2.0  compile\_time:Jul 31 2021 11:41:32  ready |

##### 别人的上电信息，不知道怎么来的

来自于





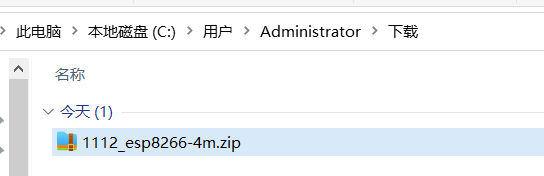
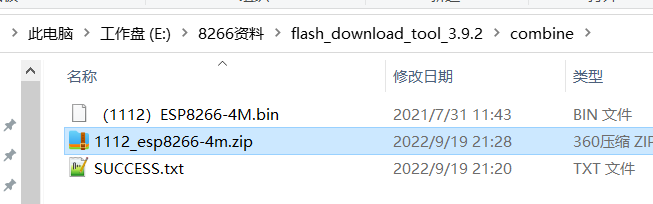
#### 官方固件下载流程

到官方的网站中下载固件，<https://docs.ai-thinker.com/%E5%9B%BA%E4%BB%B6%E6%B1%87%E6%80%BB>

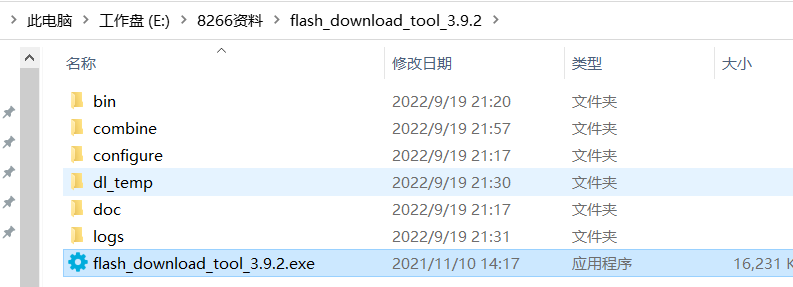


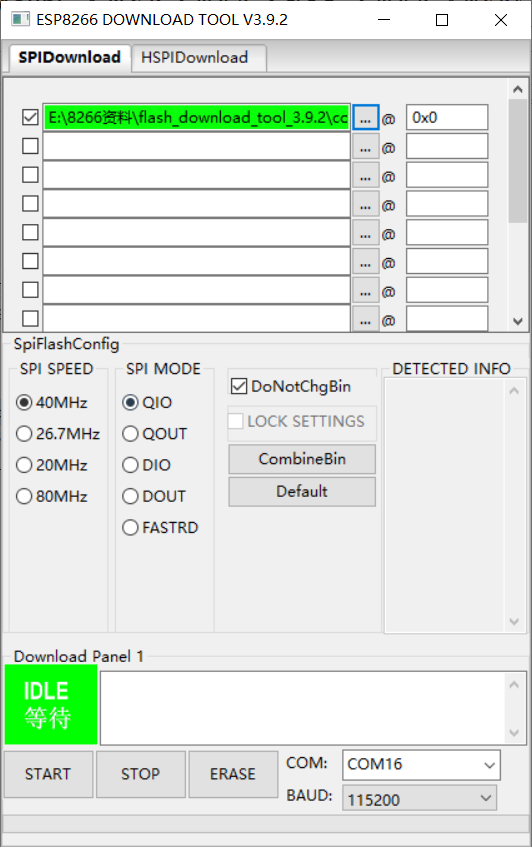
下面是出厂时候的固件，从打印日期可知，这里的固件版本为21.7.31号，就是上面的4号

|  |
| --- |
| arch:ESP8266, 1  compile\_time:Ai-Thinker|B&T  wifi\_mac:4417933d4CCA  sdk\_version:v3.4-22-g967752e2  firmware\_version:2.2.0  compile\_time:Jul 31 2021 11:41:32  ready |

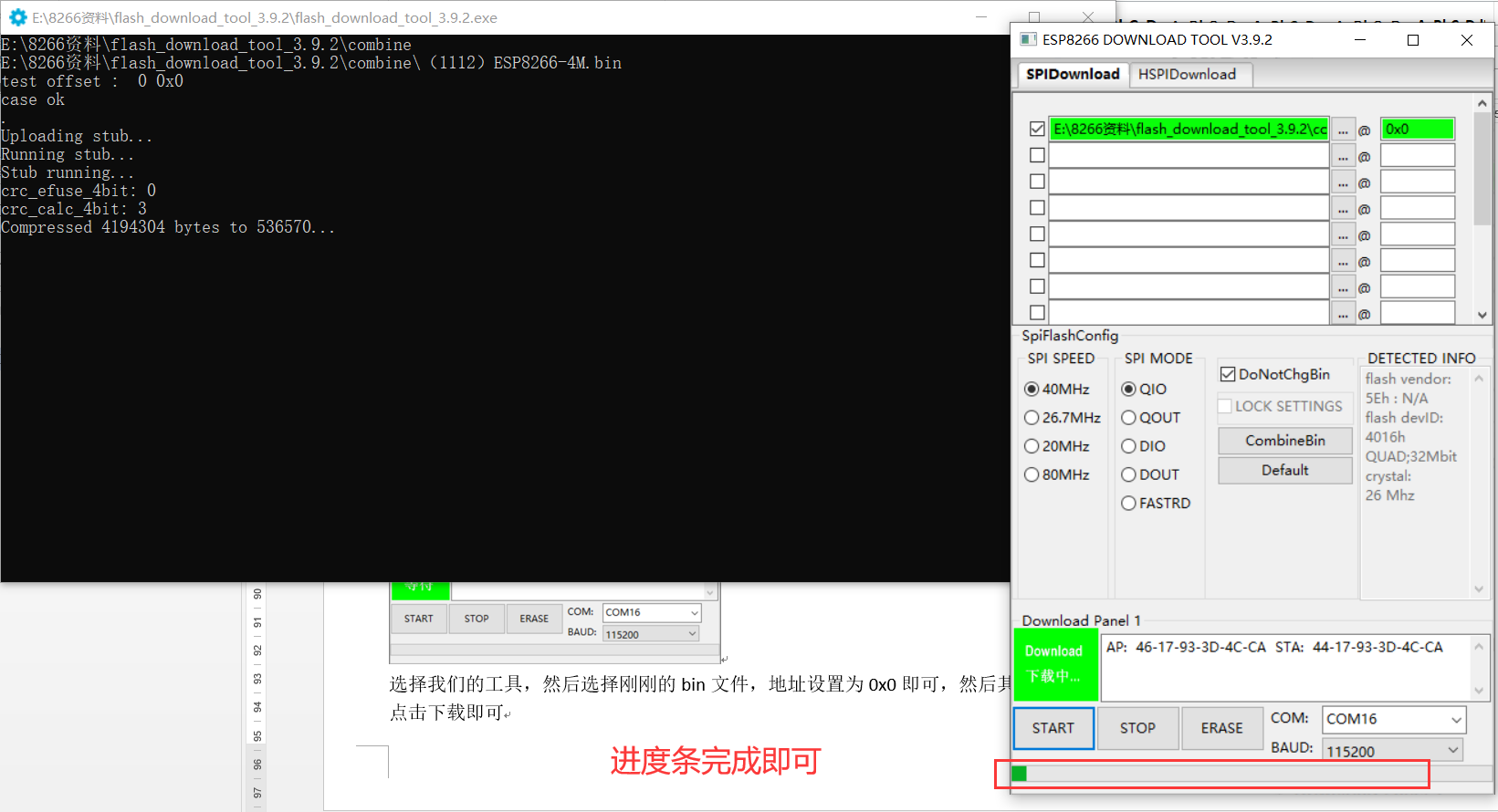
放到我们的下载工具目录中

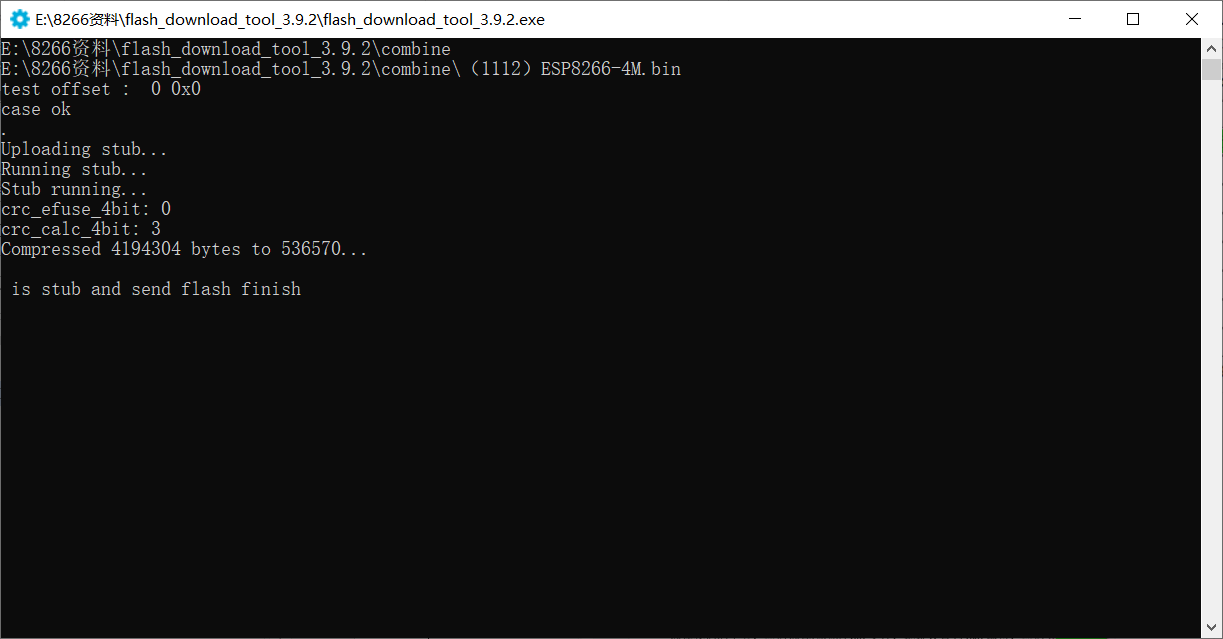




选择我们的工具，然后选择刚刚的bin文件，地址设置为0x0即可，然后其他都是默认，设备为下载模式

点击下载即可





如上已完成。

##### 验证

切换到运行模式，关闭下载工具。



可见与我们出厂时候一样的。

###### 使用说明

<https://docs.espressif.com/projects/esp-at/zh_CN/latest/esp32/AT_Command_Set/index.html>

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

#### IDE编译步骤

#### SDK下载和编译

<https://docs.ai-thinker.com/%E5%AE%98%E6%96%B9%E6%89%8B%E5%86%8C%E8%B5%84%E6%96%992>

官方应用手册

##### 12.ESP8266\_RTOS\_SDK V3.0 及之后版本

SDK代码<https://github.com/espressif/ESP8266_RTOS_SDK>

编程指南 <https://docs.espressif.com/projects/esp8266-rtos-sdk/en/latest/get-started/index.html>

##### IDE下载

<https://docs.ai-thinker.com/%E5%BC%80%E5%8F%91%E5%B7%A5%E5%85%B72>

开发工具

下载地址：<https://pan.baidu.com/s/1bk_mc-X9Aol9MgiU0gZA0A#list/path=%2F>

网盘链接提取码： shm3

　　更新时间：2020.10.16

　　更新说明：支持 ESP8266 NONOS 2.0和3.0

<https://blog.csdn.net/Boantong_/article/details/106229281>

安装方法

##### 编译一个 RTOS 操作系统代码

###### 3.1 下载开发 SDK 资源包：

此SDK是基于官方的RTOS master分支上有所修改，并增加了开发模板，详情见 README 文档； 因为SDK和开发环境是分开的，所以下载的地方随心所欲；

但是强调：下载路径务必不能带空格或中文字符！

下载时候务必要把子模块拉取下来，不能直接点击 download 下载，否则得到的是一个不完整的开发包！！

所以，请自行百度教程安装 git 版本管理的软件；

git clone --recursive <https://gitee.com/xuhongv/AiThinkerProjectForESP.git>

###### 3.2 导入安信可IDE

1、点击C/C++分支，选择 Existing Code as MakeFile Project 工程;

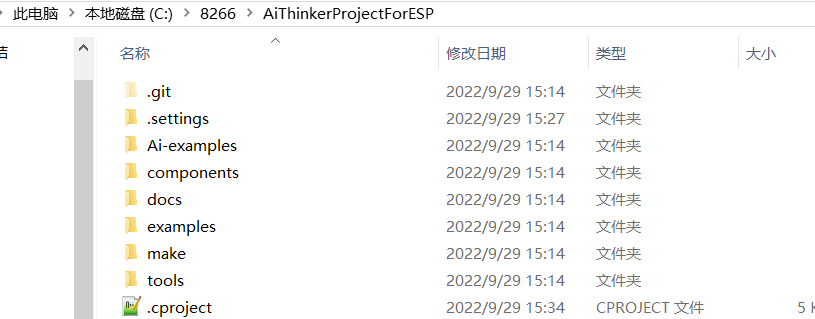
2、复制刚刚的下载的文件夹路径，import --> Cross Gcc , 并且去掉对应的 C++ 勾勾；

###### 3.3 配置环境步骤：

1、项目属性设置，鼠标选中项目名称右键点击，在右侧菜单中选择Properties

2、在 Properties --> C/C++ Build --> Build directory 选择编译的工程路径，比如 examples\get-started\project\_template （也可以选择其他的，比如hello world）工程。

3、添加IDF环境变量在 Properties --> C/C++ Build --> Environment 点击Add ，路径为刚刚的下载的文件夹路径，变量名字为 IDF\_PATH;就是下载SDK包的路径，比如C:\8266\AiThinkerProjectForESP



4、控制台输出只保留 Console ，其他一律关闭！

###### 3.4 编译步骤：

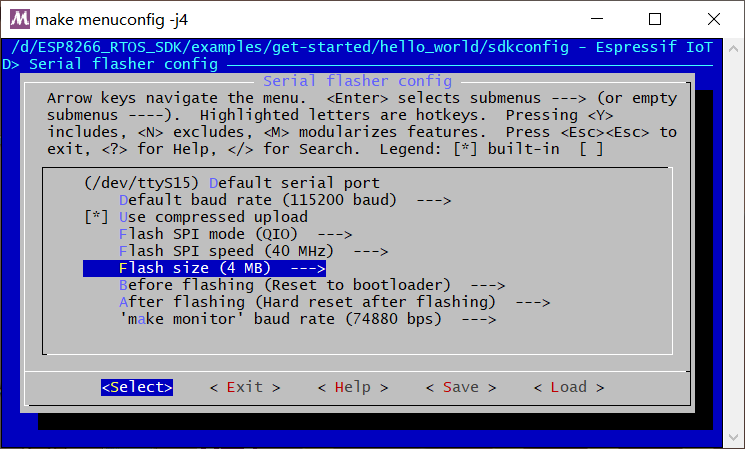
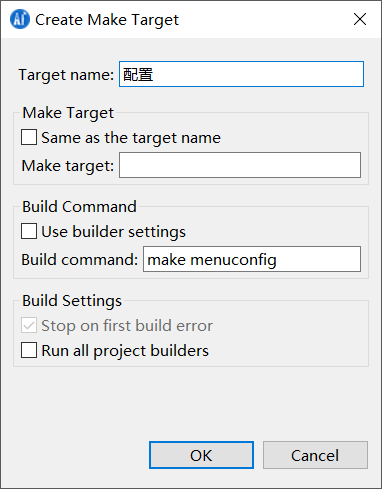
使用下面两个命令都是可以的，建议使用第二个

***make menuconfig*** 设置，同样地，我们可以利用快捷键去快速设置；

***mintty.exe -e make menuconfig***

***注意：***

***1、如果出现生成配置时间长，可以尝试多等待一下，长达5分钟***



具体的配资可以餐卡如上

构建menuconfig菜单，选中项目名称，在右键菜单中选择 Make Targets --> Create；或者快捷键 Alt + F9 （或者设置成其他快捷键）;

在弹出的对话框中取消勾选Same as the target name 与 User builder settings 这2个选项，并且 在Build command 中输入mintty.exe -e make menuconfig。

配置完成后就可以编译，点击这个

然后，就可以编译工程了，点击 Build Project，编译过程需要的时间5分钟以上，取决电脑配置，耐心等待，当出现以下信息，说明编译成功！

###### 3.5 下载步骤：

使用ide进行下载

配置快捷键进行下载，其命令是 make flash ，如果指定端口进行烧录下载固件比如 com12 的话，

可以这样写： ***make flash ESPPORT=COM12*** ；

当编译完成之后，会出现如下的提示：表示我们可以使用如下命令进行下载导入到硬件中。

To flash all build output, run 'make flash' or:

python /c/8266/AiThinkerProjectForESP/components/esptool\_py/esptool/esptool.py --chip esp8266 --port /dev/ttyUSB0 --baud 115200 --before default\_reset --after hard\_reset write\_flash -z --flash\_mode dio --flash\_freq 40m --flash\_size 4MB 0x0 /c/8266/AiThinkerProjectForESP/examples/get-started/hello\_world/build/bootloader/bootloader.bin 0x10000 /c/8266/AiThinkerProjectForESP/examples/get-started/hello\_world/build/hello-world.bin 0x8000 /c/8266/AiThinkerProjectForESP/examples/get-started/hello\_world/build/partitions\_singleapp.bin

我们将这个命令重点拆开

|  |
| --- |
| python /c/8266/AiThinkerProjectForESP/components/esptool\_py/esptool/esptool.py  --chip esp8266  --port /dev/ttyUSB0  --baud 115200  --before default\_reset  --after hard\_reset  write\_flash -z  --flash\_mode dio  --flash\_freq 40m  --flash\_size 4MB  0x0 /c/8266/AiThinkerProjectForESP/examples/get-started/hello\_world/build/bootloader/bootloader.bin  0x10000 /c/8266/AiThinkerProjectForESP/examples/get-started/hello\_world/build/hello-world.bin  0x8000 /c/8266/AiThinkerProjectForESP/examples/get-started/hello\_world/build/partitions\_singleapp.bin |

如上可以看见具体下载的文件有哪些，

如下情况，表示能够正常的下载

|  |
| --- |
| 20:24:51 \*\*\*\* Build of configuration Default for project ESP8266\_RTOS\_SDK \*\*\*\*  **make flash ESPPORT=COM16**  WARNING: esp-idf git submodule components/json/cJSON may be out of date. Run 'git submodule update' in IDF\_PATH dir to update.  Toolchain path: /opt/esp8266/v5.2.0\_for\_3.0/bin/xtensa-lx106-elf-gcc  Toolchain version: esp-2020r3-49-gd5524c1  Compiler version: 8.4.0  Python requirements from D:/ESP8266\_RTOS\_SDK\requirements.txt are satisfied.  App "hello-world" version: v3.4-71-g7085cc76-dirty  Flashing binaries to serial port COM16 (app at offset 0x10000)...  esptool.py v2.4.0  Connecting....  Chip is ESP8266EX  Features: WiFi  MAC: 44:17:93:3d:4c:ca  Uploading stub...  Running stub...  Stub running...  Configuring flash size...  Compressed 11088 bytes to 7435...  Writing at 0x00000000... (100 %)  Wrote 11088 bytes (7435 compressed) at 0x00000000 in 0.7 seconds (effective 134.8 kbit/s)...  Hash of data verified.  Compressed 169072 bytes to 108608...  Writing at 0x00010000... (14 %)  Writing at 0x00014000... (28 %)  Writing at 0x00018000... (42 %)  Writing at 0x0001c000... (57 %)  Writing at 0x00020000... (71 %)  Writing at 0x00024000... (85 %)  Writing at 0x00028000... (100 %)  Wrote 169072 bytes (108608 compressed) at 0x00010000 in 9.6 seconds (effective 141.4 kbit/s)...  Hash of data verified.  Compressed 3072 bytes to 83...  Writing at 0x00008000... (100 %)  Wrote 3072 bytes (83 compressed) at 0x00008000 in 0.0 seconds (effective 1849.5 kbit/s)...  Hash of data verified.  Leaving...  Hard resetting via RTS pin... |

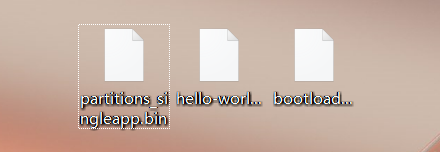
使用串口调试助手打印的结果

|  |
| --- |
| ets Jan 8 2013,rst cause:2, boot mode:(3,1)  load 0x40100000, len 7544, room 16  tail 8  chksum 0x2d  load 0x3ffe8408, len 24, room 0  tail 8  chksum 0x85  load 0x3ffe8420, len 3472, room 0  tail 0  chksum 0xaf  csum 0xaf  [0;32mI (43) boot: ESP-IDF v3.4-71-g7085cc76-dirty 2nd stage bootloader[0m  [0;32mI (43) boot: compile time 22:10:46[0m  [0;32mI (44) qio\_mode: Enabling default flash chip QIO[0m  [0;32mI (52) boot: SPI Speed : 40MHz[0m  [0;32mI (59) boot: SPI Mode : QIO[0m  [0;32mI (65) boot: SPI Flash Size : 16MB[0m  [0;32mI (71) boot: Partition Table:[0m  [0;32mI (77) boot: ## Label Usage Type ST Offset Length[0m  [0;32mI (88) boot: 0 nvs WiFi data 01 02 00009000 00006000[0m  [0;32mI (99) boot: 1 phy\_init RF data 01 01 0000f000 00001000[0m  [0;32mI (111) boot: 2 factory factory app 00 00 00010000 000f0000[0m  [0;32mI (122) boot: End of partition table[0m  [0;32mI (129) esp\_image: segment 0: paddr=0x00010010 vaddr=0x40210010 size=0x1cca8 (117928) map[0m  [0;32mI (182) esp\_image: segment 1: paddr=0x0002ccc0 vaddr=0x4022ccb8 size=0x070e8 ( 28904) map[0m  [0;32mI (192) esp\_image: segment 2: paddr=0x00033db0 vaddr=0x3ffe8000 size=0x00544 ( 1348) load[0m  [0;32mI (193) esp\_image: segment 3: paddr=0x000342fc vaddr=0x40100000 size=0x00080 ( 128) load[0m  [0;32mI (207) esp\_image: segment 4: paddr=0x00034384 vaddr=0x40100080 size=0x050c0 ( 20672) load[0m  [0;32mI (227) boot: Loaded app from partition at offset 0x10000[0m  Hello world!  This is ESP8266 chip with 1 CPU cores, WiFi, silicon revision 1, 16MB external flash  Restarting in 10 seconds...  Restarting in 9 seconds... |

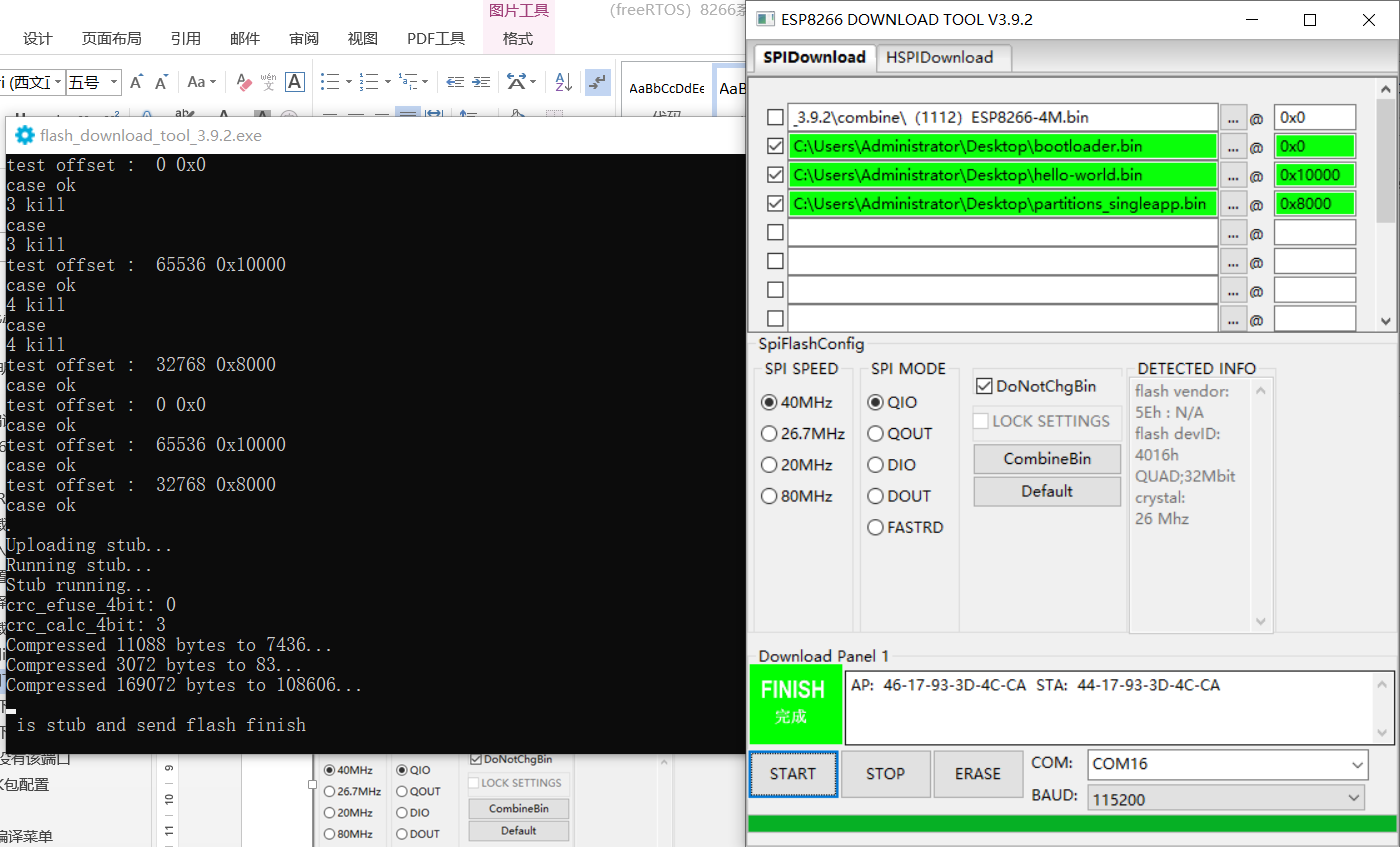
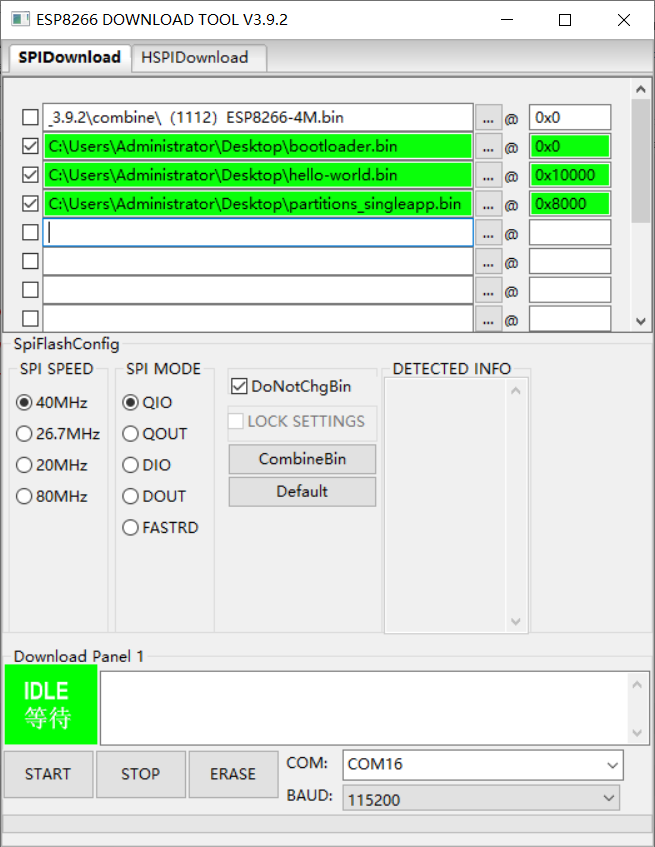
|  |
| --- |
|  |

使用下载工具进行下载

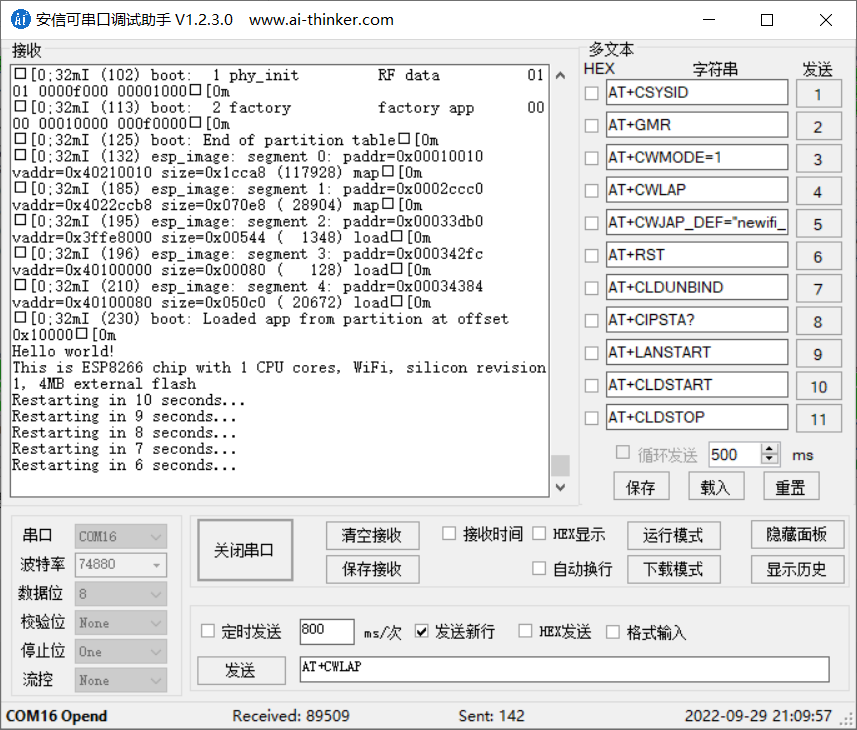
|  |
| --- |
| python /d/ESP8266\_RTOS\_SDK/components/esptool\_py/esptool/esptool.py --chip esp8266  --port /dev/ttyS15  --baud 115200  --before default\_reset  --after hard\_reset  write\_flash -z  --flash\_mode dio  --flash\_freq 40m  --flash\_size 16MB 这个地方配置错误了，应该是4MB  0x0 /d/ESP8266\_RTOS\_SDK/examples/get-started/hello\_world/build/bootloader/bootloader.bin  0x10000 /d/ESP8266\_RTOS\_SDK/examples/get-started/hello\_world/build/hello-world.bin  0x8000 /d/ESP8266\_RTOS\_SDK/examples/get-started/hello\_world/build/partitions\_singleapp.bin |



依次得到这三个文件



安装上面的方式下载，



下载后的效果是一样的。

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

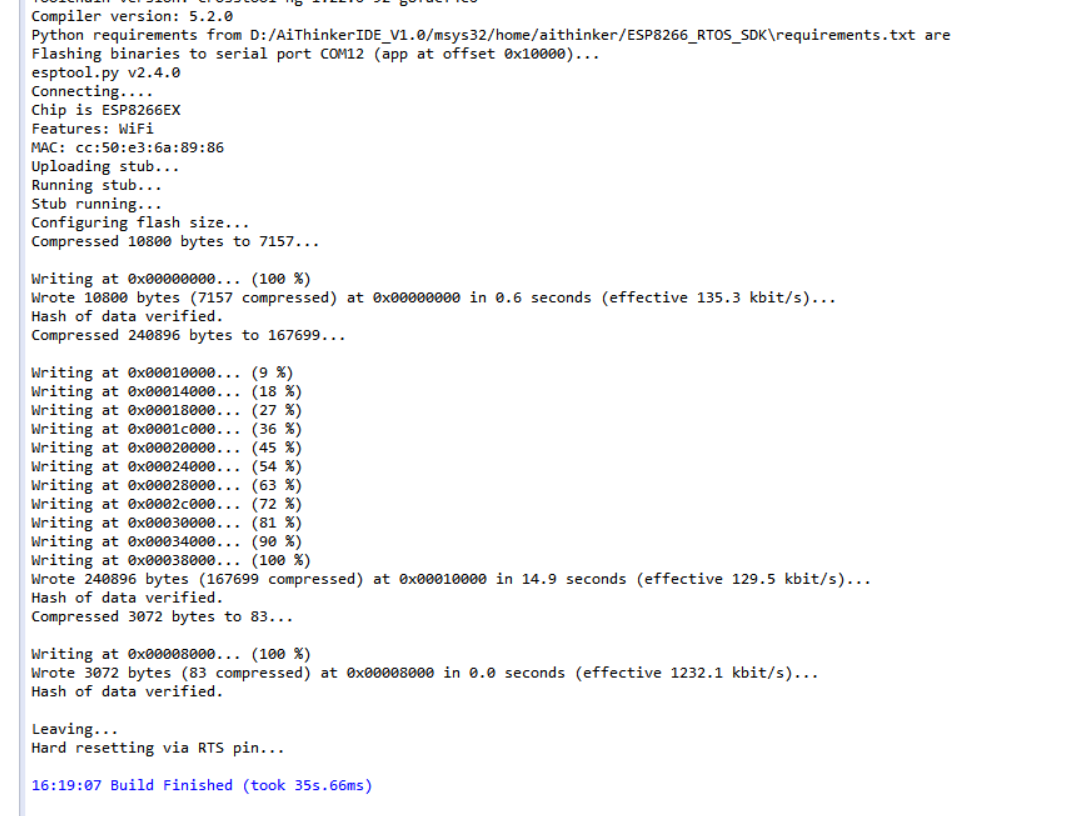
|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

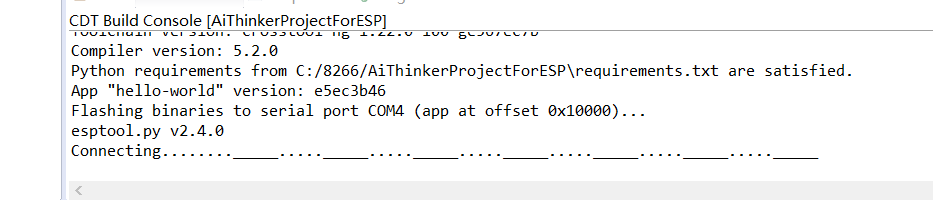
|  |
| --- |
|  |

1、下载正常



如果出现上面这样表示下载成功

2、下载时无法连接到MCU

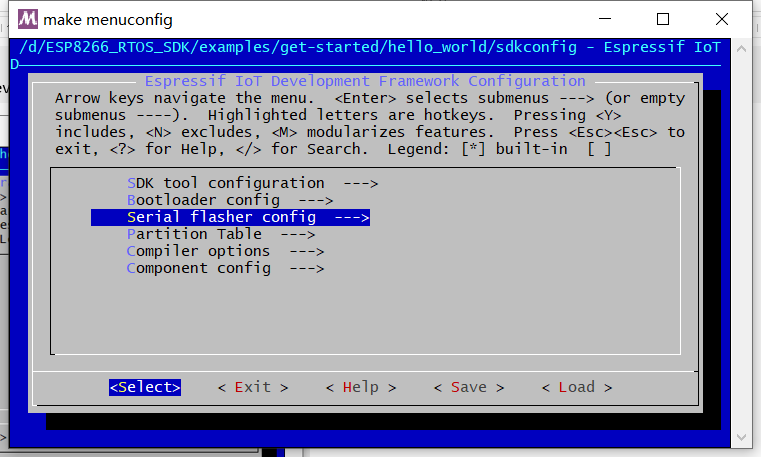


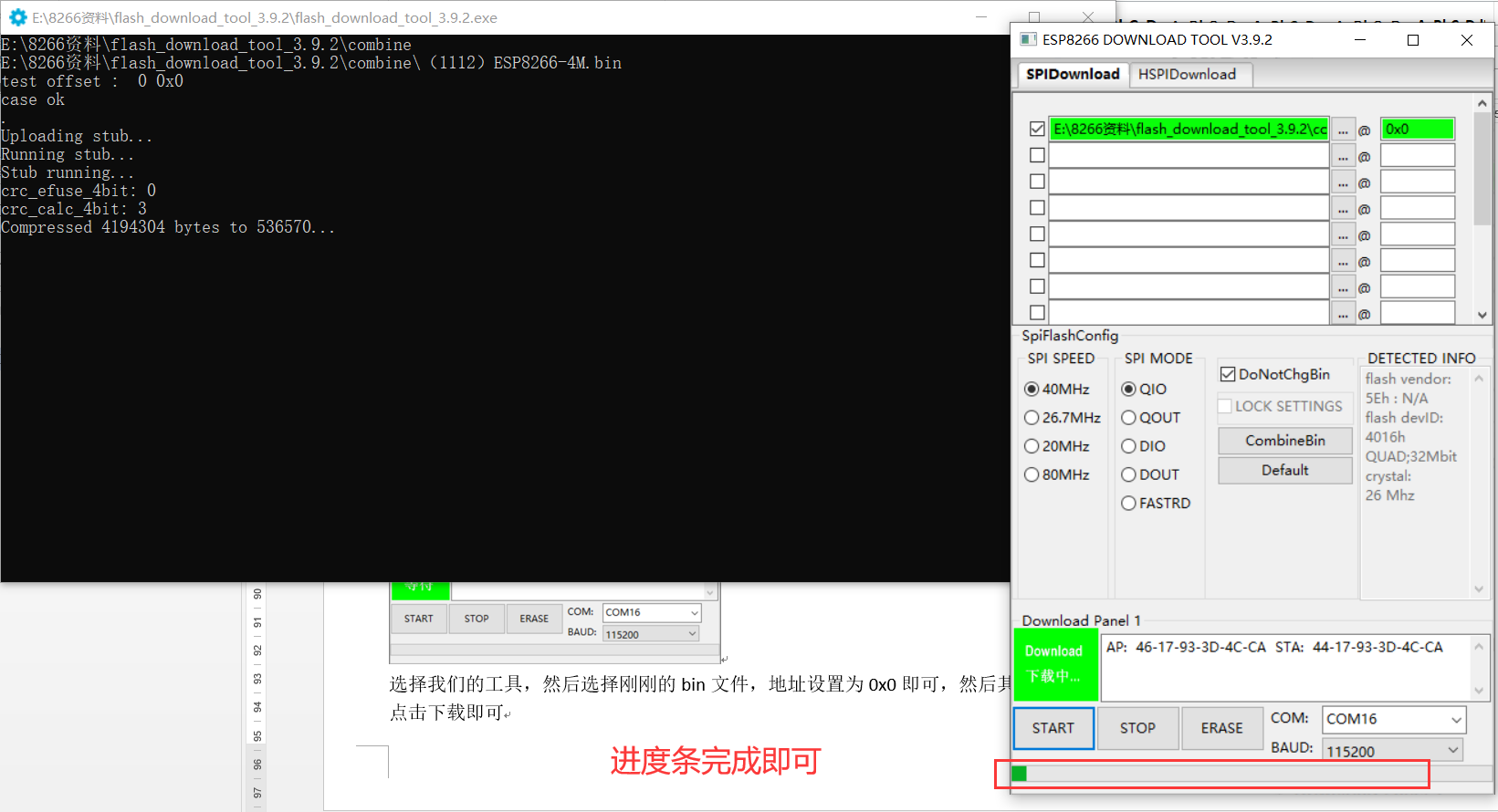
3、没有该端口

|  |
| --- |
| Flashing binaries to serial port COM5 (app at offset 0x10000)...  esptool.py v2.4.0  Traceback (most recent call last):  File "C:/8266/AiThinkerProjectForESP/components/esptool\_py/esptool/esptool.py", line 3034, in <module>  \_main()  File "C:/8266/AiThinkerProjectForESP/components/esptool\_py/esptool/esptool.py", line 3027, in \_main  main()  File "C:/8266/AiThinkerProjectForESP/components/esptool\_py/esptool/esptool.py", line 2735, in main  esp = chip\_class(args.port, initial\_baud, args.trace)  File "C:/8266/AiThinkerProjectForESP/components/esptool\_py/esptool/esptool.py", line 212, in \_\_init\_\_  self.\_port = serial.serial\_for\_url(port)  File "C:\8266\AiThinkerIDE\_V1.5.2\msys32\mingw32\lib\python2.7\site-packages/serial/\_\_init\_\_.py", line 88, in serial\_for\_url  instance.open()  File "C:\8266\AiThinkerIDE\_V1.5.2\msys32\mingw32\lib\python2.7\site-packages/serial/serialwin32.py", line 62, in open  raise SerialException("could not open port {!r}: {!r}".format(self.portstr, ctypes.WinError()))  serial.serialutil.SerialException: could not open port 'COM5': WindowsError(2, '\xcf\xb5\xcd\xb3\xd5\xd2\xb2\xbb\xb5\xbd\xd6\xb8\xb6\xa8\xb5\xc4\xce\xc4\xbc\xfe\xa1\xa3')  make: \*\*\* [/c/8266/AiThinkerProjectForESP/components/esptool\_py/Makefile.projbuild:76: flash] Error 1 |

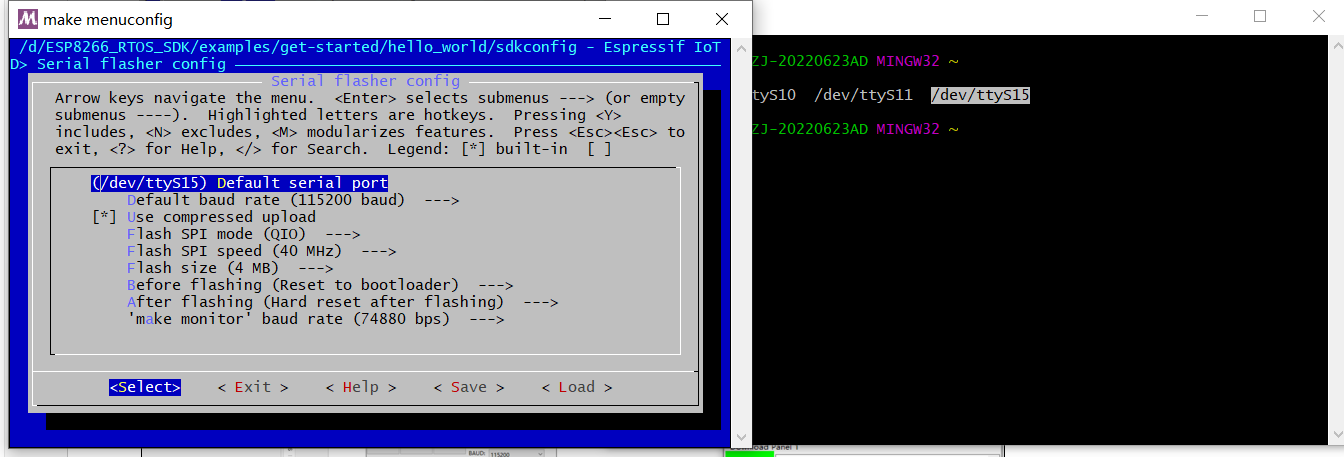
###### #--SDK包配置

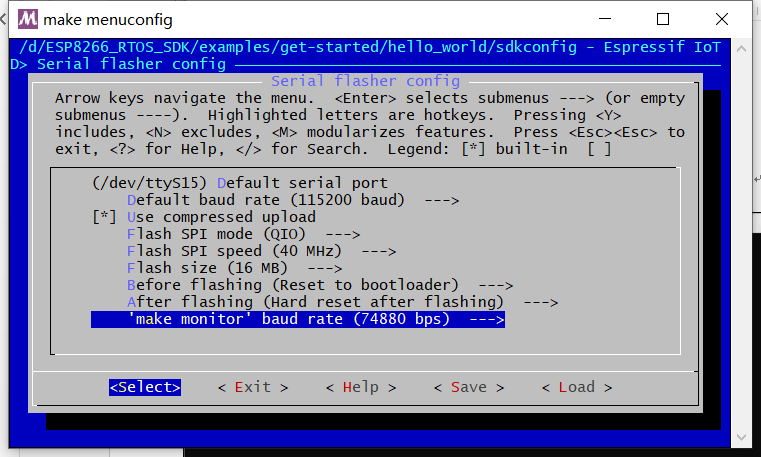
|  |
| --- |
| Administrator@XTZJ-20220623AD MINGW32 ~  # ls /dev/tty\*  /dev/tty /dev/ttyS10 /dev/ttyS11 /dev/ttyS15 |





进行配置





|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

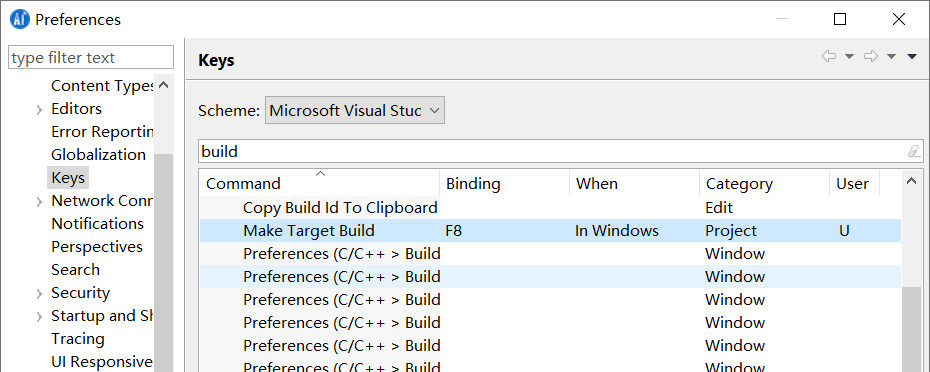
|  |
| --- |
|  |

|  |
| --- |
|  |

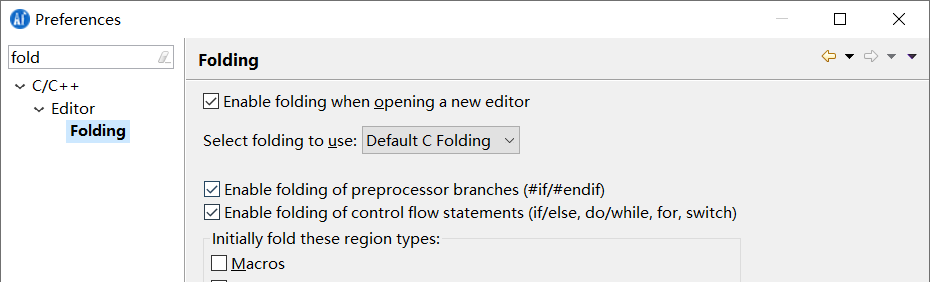
|  |
| --- |
|  |

## 常见的软件设置

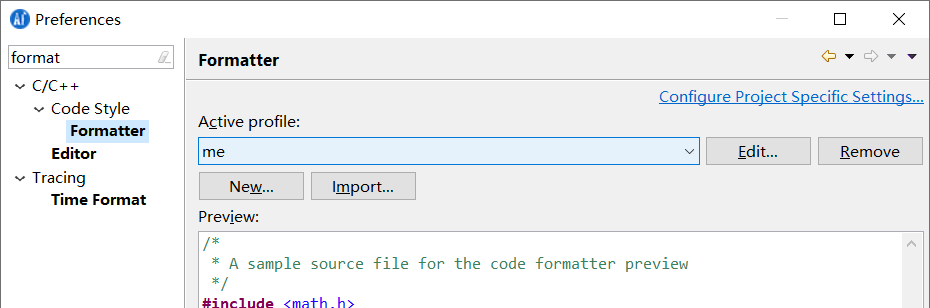
### 1、使用F8打开编译菜单



### 2、打开if\while等折叠



### 3、配置具体的格式化



4、

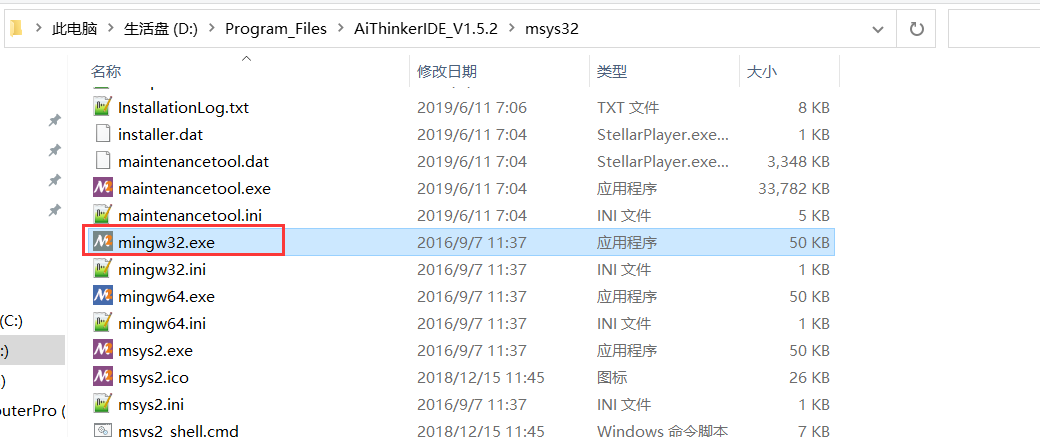
4、

4、

4、

**常见问题**

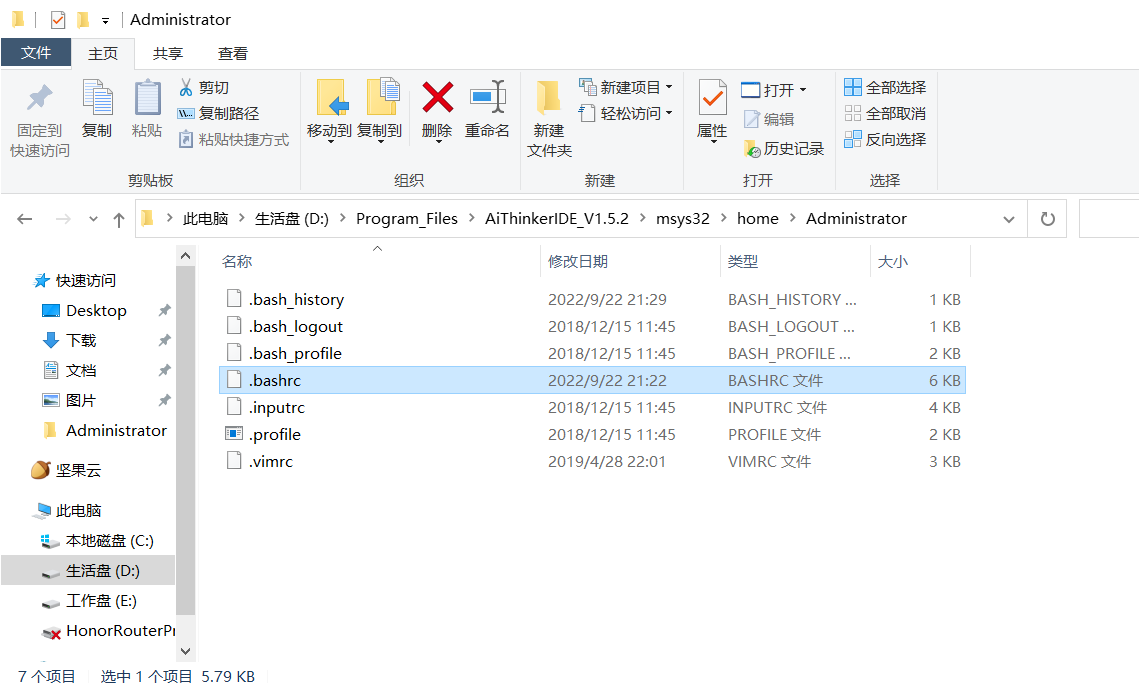
**1、这个软件只支持mingw32.exe，如果错误时也会出现提示**



**2、软件出现闪退，编译或者生成配置时闪退**

mintty.exe -e make menuconfig

出现编译时候闪退，通常情况下是，mintty.exe这个软件打开的终端用户没有设置IDF\_PATH路径



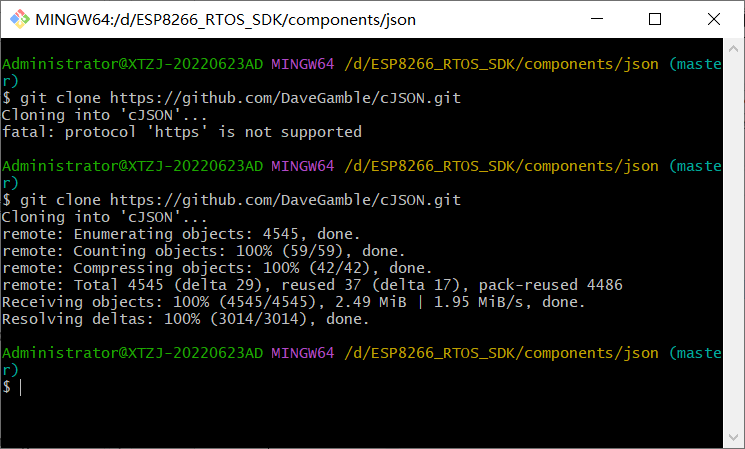
mintty.exe -e make menuconfig

mintty.exe –e 可以直接打开一个终端

**3、cjson克隆失败**

正克隆到 '/d/ESP8266\_RTOS\_SDK/components/json/cJSON'...

fatal: 无法访问 'https://github.com/DaveGamble/cJSON.git/'：SSL certificate problem: unable to get local issuer certificate



使用git上的bash可以下载

# git clone https://github.com/espressif/esp-lwip.git/

正克隆到 'esp-lwip'...

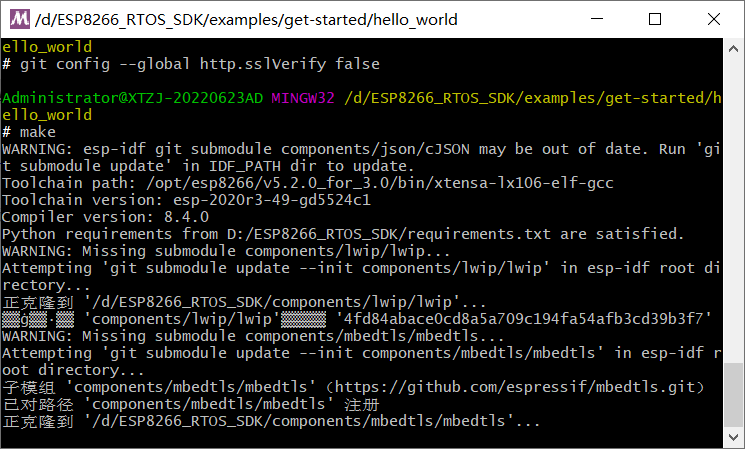
fatal: 无法访问 'https://github.com/espressif/esp-lwip.git/'：SSL certificate problem: unable to get local issuer certificate

根本原因是没有github的权限

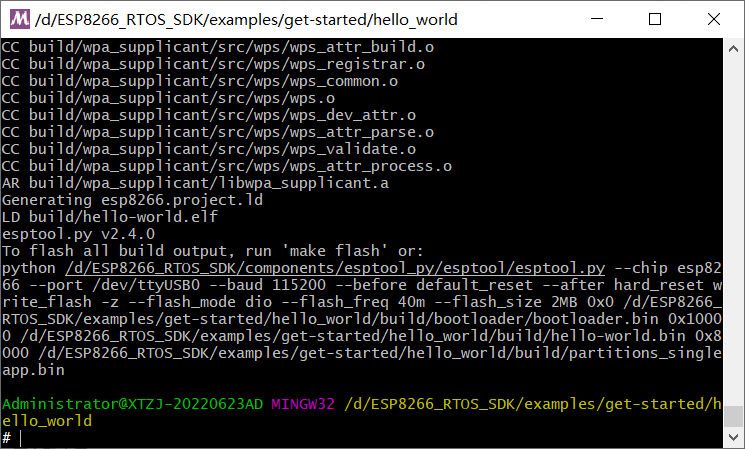
***SSL certificate problem: unable to get local issuer certificate***

<https://blog.csdn.net/qq_27674439/article/details/119327919> 问题解决连接

git config --global http.sslVerify false 使用这个命令即可



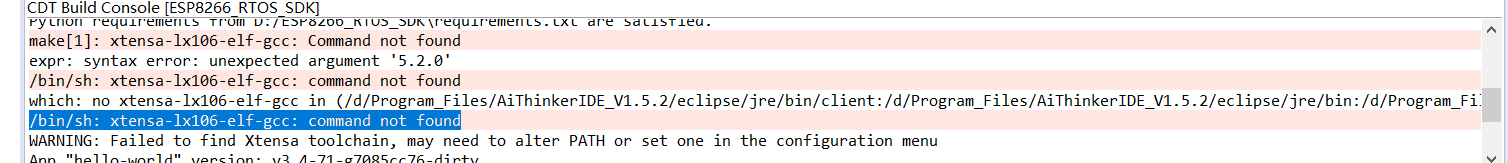
这里要稍微等待一下，等待完成后，即可编译完成。



如上就是编译完成的结果。

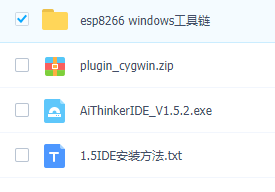
**4、/bin/sh: line 1: xtensa-lx106-elf-gcc: command not found**

/bin/sh: xtensa-lx106-elf-gcc: command not found

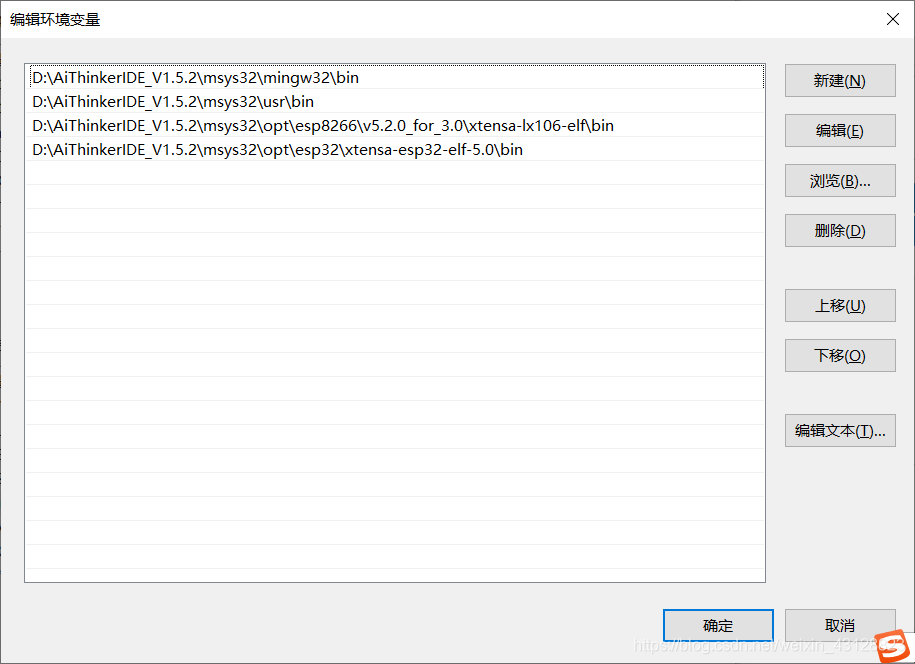


官方IDE下载链接：https://pan.baidu.com/s/1bk\_mc-X9Aol9MgiU0gZA0A 提取码：shm3

里面会有一个“esp8266 Windows工具链”



原来这东西叫工具链。将“v5.2.0\_for\_3.0.rar”解压，放入D:\AiThinkerIDE\_V1.5.2\msys32\opt\esp8266文件夹，如果原来有，则覆盖。同时，检查系统环境变量是否正确。我之前的环境变量就写错了。

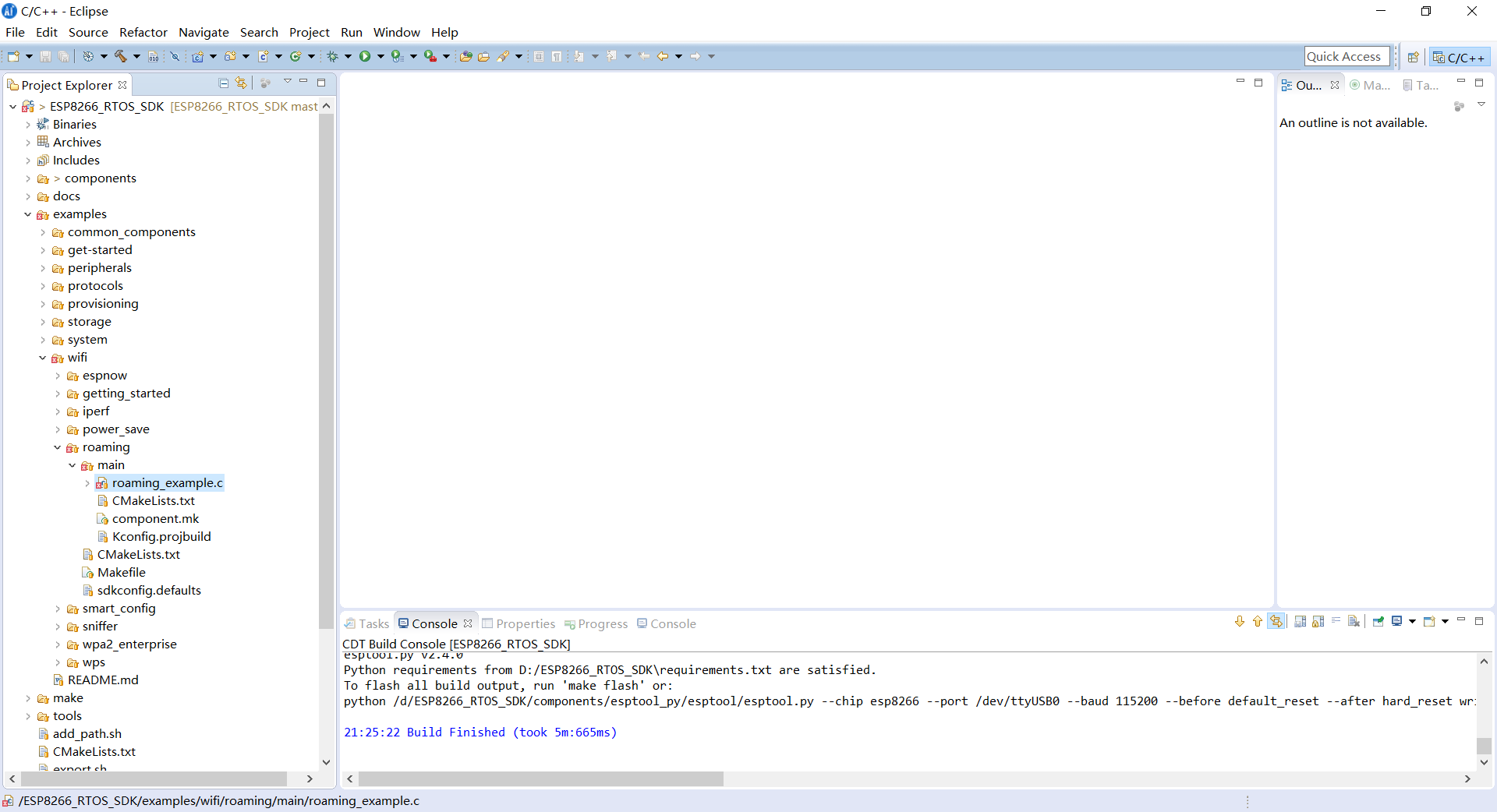


**原始的环境变量值**

|  |
| --- |
| D:/Program\_Files/AiThinkerIDE\_V1.5.2/eclipse/jre/bin/client;  D:/Program\_Files/AiThinkerIDE\_V1.5.2/eclipse/jre/bin;  D:/Program\_Files/AiThinkerIDE\_V1.5.2/eclipse/jre/lib/i386;  D:\Program\_Files\AiThinkerIDE\_V1.5.2\msys32\opt\esp32\xtensa-esp32-elf-5.0\bin;  D:\Program\_Files\AiThinkerIDE\_V1.5.2\msys32\opt\esp8266\v5.2.0\_for\_3.0\xtensa-lx106-elf\bin;  D:\Program\_Files\AiThinkerIDE\_V1.5.2\msys32\usr\bin;  D:\Program\_Files\AiThinkerIDE\_V1.5.2\msys32\mingw32\bin;  C:\Program Files (x86)\Intel\Intel(R) Management Engine Components\iCLS\;  C:\Program Files\Intel\Intel(R) Management Engine Components\iCLS\;  C:\Windows\system32;  C:\Windows;  C:\Windows\System32\Wbem;  C:\Windows\System32\WindowsPowerShell\v1.0\;  C:\Windows\System32\OpenSSH\;  C:\Program Files (x86)\Intel\Intel(R) Management Engine Components\DAL;  C:\Program Files\Intel\Intel(R) Management Engine Components\DAL;  C:\Program Files (x86)\Intel\Intel(R) Management Engine Components\IPT;  C:\Program Files\Intel\Intel(R) Management Engine Components\IPT;  C:\Program Files\dotnet\;  C:\Program Files (x86)\Notepad++\notepad++.exe;  C:\Program Files (x86)\NVIDIA Corporation\PhysX\Common;  C:\Program Files\Git\cmd;  C:\Program Files\Intel\WiFi\bin\;  C:\Program Files\Common Files\Intel\WirelessCommon\;  D:\Program\_Files\WinMerge;C:\Program Files\mosquitto\;  D:\Program\_Files\RT-ThreadStudio\repo\Extract\ToolChain\_Support\_Packages\ARM\GNU\_Tools\_for\_ARM\_Embedded\_Processors\10.2.1\bin;  C:\Users\Administrator\AppData\Local\Programs\Python\Python35\Scripts\;  C:\Users\Administrator\AppData\Local\Programs\Python\Python35\;  C:\Users\Administrator\AppData\Local\Microsoft\WindowsApps;;  D:\Program\_Files\Microsoft VS Code\bin;C:\Program Files\Intel\WiFi\bin\;  C:\Program Files\Common Files\Intel\WirelessCommon\;D:\Program\_Files\AiThinkerIDE\_V1.5.2\eclipse |

仔细对比看是否和官方的要求一致。

|  |
| --- |
| IDE安装路径\msys32\mingw32\bin  IDE安装路径\msys32\usr\bin  IDE安装路径\msys32\opt\esp8266\v5.2.0\_for\_3.0\xtensa-lx106-elf\bin  IDE安装路径\msys32\opt\esp32\xtensa-esp32-elf-5.0\bin  D:\Program\_Files\AiThinkerIDE\_V1.5.2\msys32\opt\esp32\xtensa-esp32-elf-5.0\bin;  D:\Program\_Files\AiThinkerIDE\_V1.5.2\msys32\opt\esp8266\v5.2.0\_for\_3.0\xtensa-lx106-elf\bin; 这条错误了  D:\Program\_Files\AiThinkerIDE\_V1.5.2\msys32\usr\bin;  D:\Program\_Files\AiThinkerIDE\_V1.5.2\msys32\mingw32\bin;  D:\Program\_Files\AiThinkerIDE\_V1.5.2\msys32\opt\esp8266\v5.2.0\_for\_3.0\bin; 追加 |



追加上面那条环境变量后，即可正常编译了。

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

#### 5、编译时候出现无法错误路径，注意这里的IDE和SDK安装路径不要出现空格等等

#### 6、下载时候出现如下，MCU型号错误

