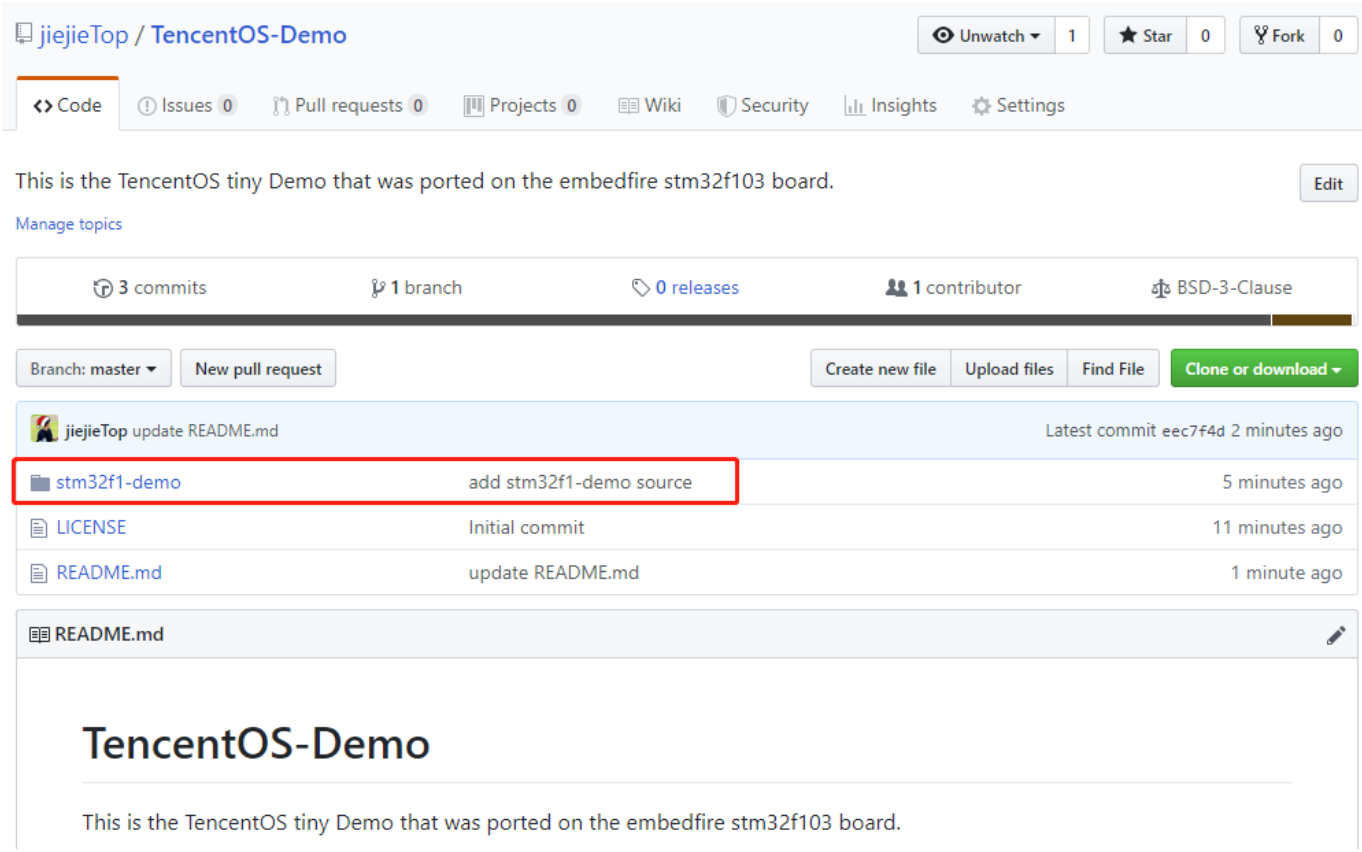
title: 【TencentOS tiny】 移植到STM32F103全教程 author: 杰杰 top: false cover: false toc: true mathjax: false date: 2019-10-12 22:57:04 img: coverImg: password: summary: tags: - TencentOS tiny - RTOS - 操作系统 - 物联网 categories: - 操作系统 - TencentOS tiny

移植前的准备工作

1. 获取STM32的裸机工程模板

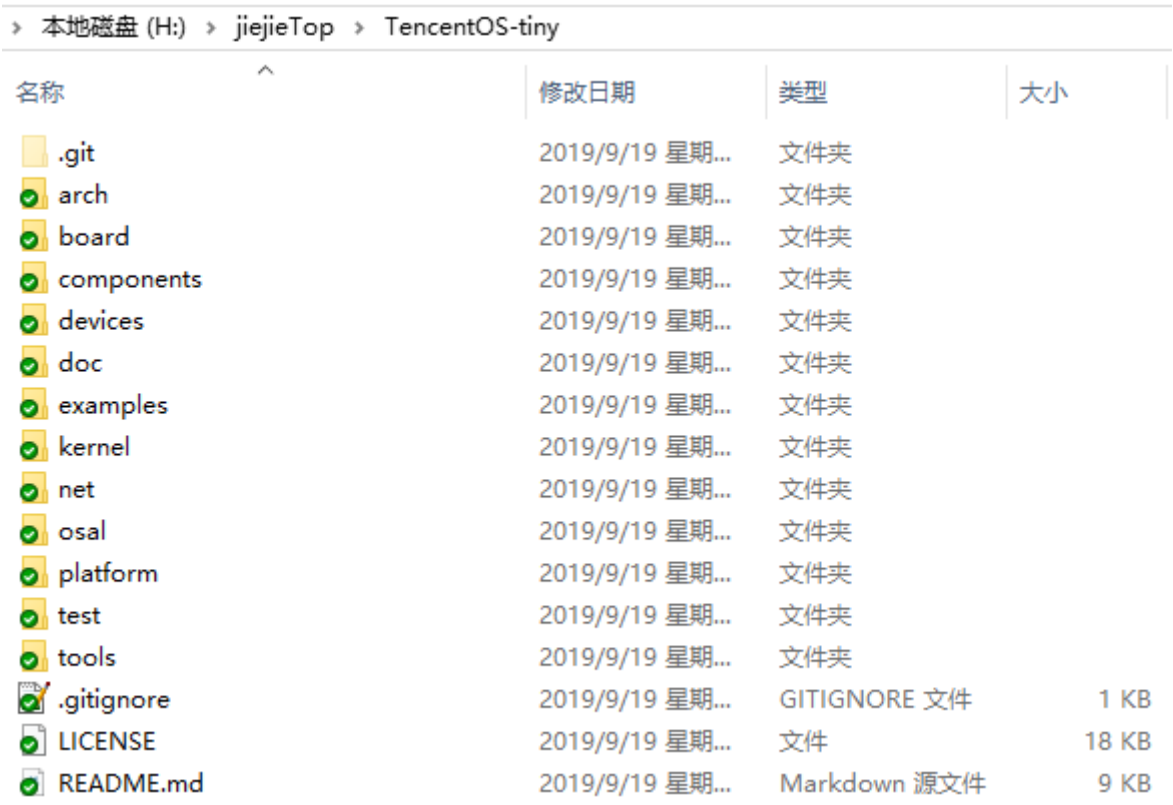
STM32的裸机工程模板直接使用野火STM32开发板配套的固件库例程即可。可以从我github上获取 https://github.com/jiejieTop/TencentOS-Demo



下载TencentOS tiny 源码

TencentOS tiny的源码可从TencentOS tiny **GitHub**仓库地址https://github.com/Tencent/TencentOS-tiny下载，如果GitHub下载慢，也可以通过腾讯工蜂开源仓下载，地址：https://git.code.tencent.com/Tencent_Open_Source/TencentOS-tiny ，大家在移植时并不需要把整个TencentOS tiny 源码放进工程文件中，否则工程的代码量太大。杰杰将在下文讲解如何将TencentOS tiny移植到工程中去，以及如何把TencentOS tiny源码中的核心部分单独提取出来，方便以后在不同的平台上移植。目前使用的是TencentOS tiny最新版本，由于TencentOS tiny在不断更新，如果以后TencentOS tiny

更新到更高的版本，则以最新的版本为准。



TencentOS tiny源码核心文件夹分析

打开TencentOS tiny源码文件，可以看见里面有12个文件夹，下面先来了解主要文件夹及其子文件夹的作用，然后将TencentOS tiny源码的核心文件提取出来，添加到工程根目录下的文件夹中，因为工程只需要有用的源码文件，而不是全部的TencentOS tiny源码，所以可以避免工程过于庞大。

一级目录	二 / 三级目录	说明（杰杰）
arch	arm	TencentOS tiny适配的IP核架构（含M核中断、调度、tick相关代码），对我们的移植很重要
arch	risc-v	TencentOS tiny适配的risc-v架构
board	TencentOS_tiny_EVB_MX	TencentOS tiny 定制开发板demo，包含AT适配框架、MQTT协议、安全组件等
component	connectivity / loraWAN	loRaWAN协议栈实现源码及适配层
	connectivity / Eclipse-Paho-MQTT	MQTT协议栈实现源码及适配层
	connectivity / TencentCloud_SDK	腾讯云C-SDK实现源码及适配层
	fs	文件系统实现源码
	security	mbedtls 安全协议源码
	utils	包含json相关源码
devices		TencentOS tiny适配的一些外设驱动（如串口wifi gprs驱动等）

一级目录	二 / 三级目录	说明（杰杰）
doc		TencentOS tiny相关技术文档及开发指南（建议多看这部分）
examples		TencentOS tiny提供的功能示例
kernel	core	TencentOS tiny内核源码（这部分是最重要的）
	hal	TencentOS tiny驱动抽象层
	pm	**TencentOS tiny低功耗模块源码**
net	at	TencentOS tiny为串口类通信模组提供的AT框架实现层
	lora_module_wrapper	TencentOS tiny为串口类LoraWAN模块提供的移植框架
	lwip	Lwip协议实现源码及适配层
	sal_module_wrapper	TencentOS tiny为串口类网络模块（wifi gprs）提供的socket移植框架
	tencent_firmware_module_wrapper	TencentOS tiny提供的腾讯定制模组移植框架
osal	cmsis_os	TencentOS tiny提供的cmsis os 适配
platform	hal	TencentOS tiny适配的部分芯片的驱动实现源码
	vendor_bsp	芯片厂家提供的原厂bsp固件库，如STM32的HAL库
test		存放TencentOS tiny提供的一些测试代码，含内核及上层模块示例及测试代码
tools		存放TencentOS tiny提供的工具，小程序，配置工具等

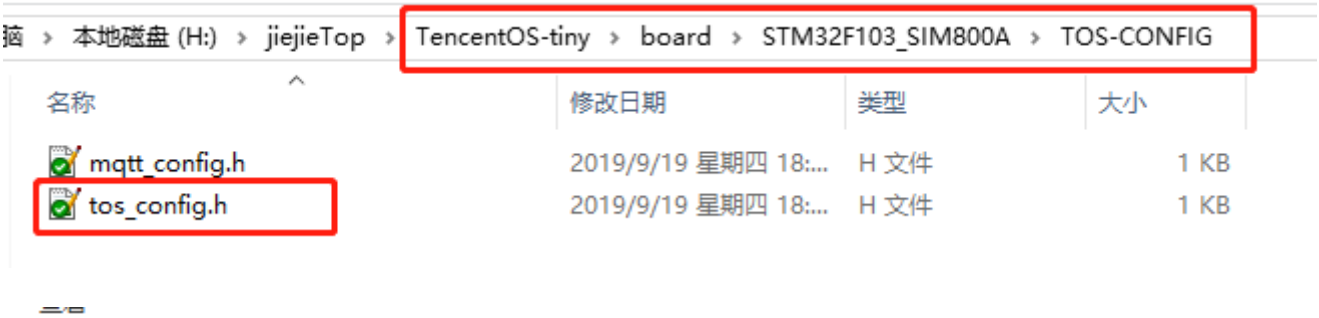
简单提一下我们的重点文件夹：

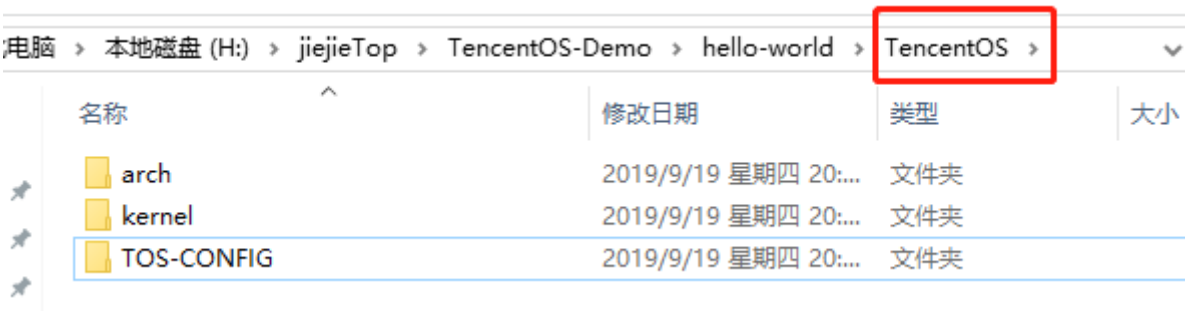
- **arch**: TencentOS tiny是软件，单片机是硬件，为了使TencentOS tiny运行在单片机上面，TencentOS tiny和单片机必须关联在一起，那么如何关联呢？还是要通过代码来关联，这部分关联的文件叫接口文件，通常由汇编语言和C语言联合编写。这些接口文件都是跟硬件密切相关的，不同的硬件接口文件是不一样的，但都大同小异。TencentOS tiny在arch\arm\arm-v6m目录中存放了cortex m0内核的单片机的接口文件，在arch\arm\arm-v7m目录中存放了cortex m3、m4和m7内核的单片机的接口文件，以及一些通用的接口文件，基于这些内核的mcu都可以使用里面的接口文件。
- **kernel**：kernel是TencentOS tiny内核核心源码，它的重要性我也不用多说，毕竟整个内核就是由这里面的文件组成，而其他文件夹都是基于内核的组件。

提取TencentOS tiny内核源码

将裸机工程源码重命名为hello-world，然后在裸机工程中新建一个TencentOS文件夹，接着将kernel文件夹、arch文件夹、添加到TencentOS文件夹下：

除了TencentOS tiny的核心文件外，还需要移植一下其他文件，如关于TencentOS tiny系统的配置文件。这是一些可以被用户修改的文件，所以会放在具体的工程文件中。board就是TencentOS tiny为一些常用开发板开发的demo文件夹，其内有各个工程的配置文件，选一个与移植芯片最相机的开发板，找到它的配置文件tos_config.h，比如我们可以选择：TencentOS-tiny\board\STM32F103_SIM800A\TOS-CONFIG路径下的配置文件，把它拷贝到我们工程中的TencentOS文件夹下，当然你也可以把整个TOS-CONFIG目录拷贝过去，把其他无关的配置删掉就好了。

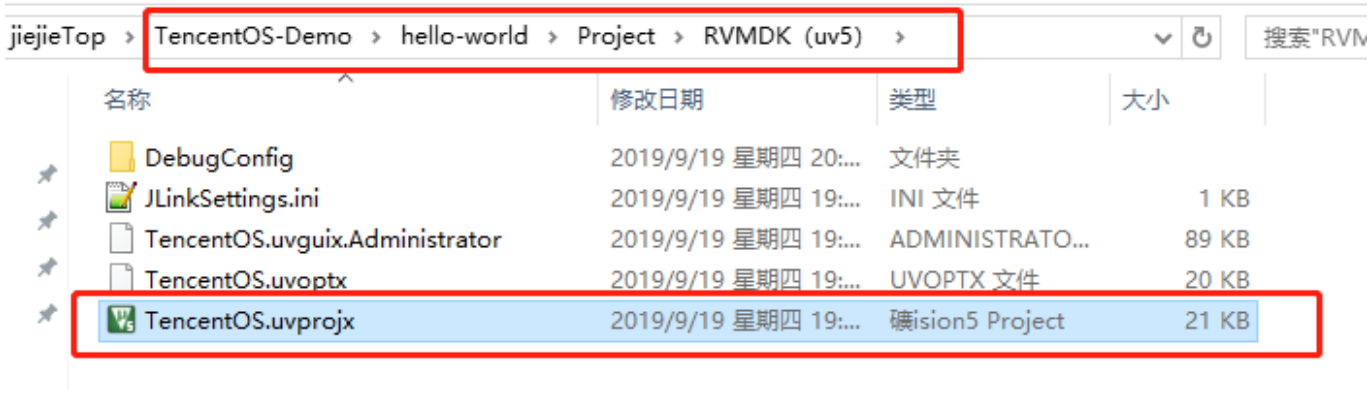




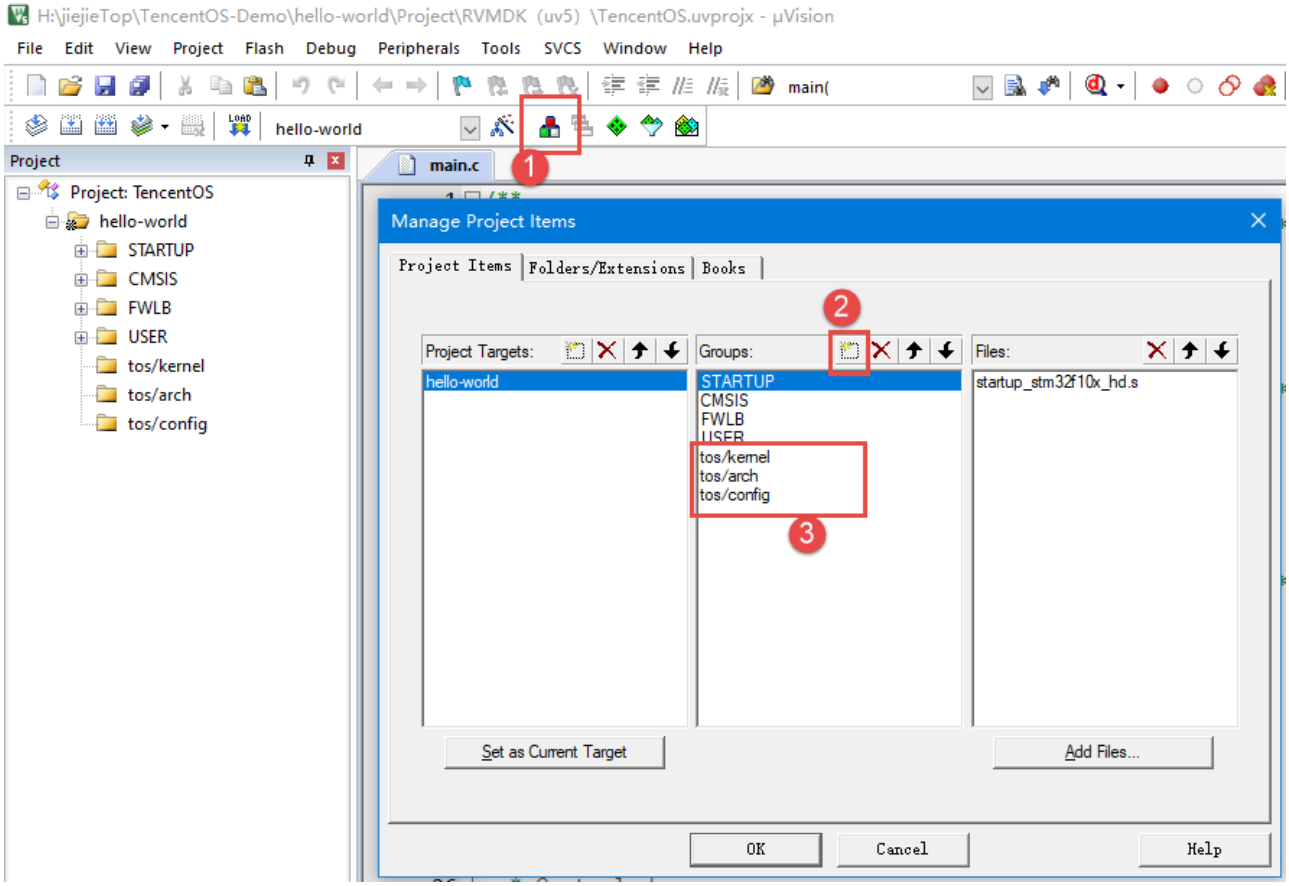
这个配置文件**很重要**，后续在移植工程时，我们需要对这个配置文件进行修改，这样子可以裁剪TencentOS tiny的功能，得到最适合的工程配置。

开始移植

打开TencentOS-Demo\hello-world\Project\RVMDK（uv5）路径下的TencentOS.uvprojx文件。

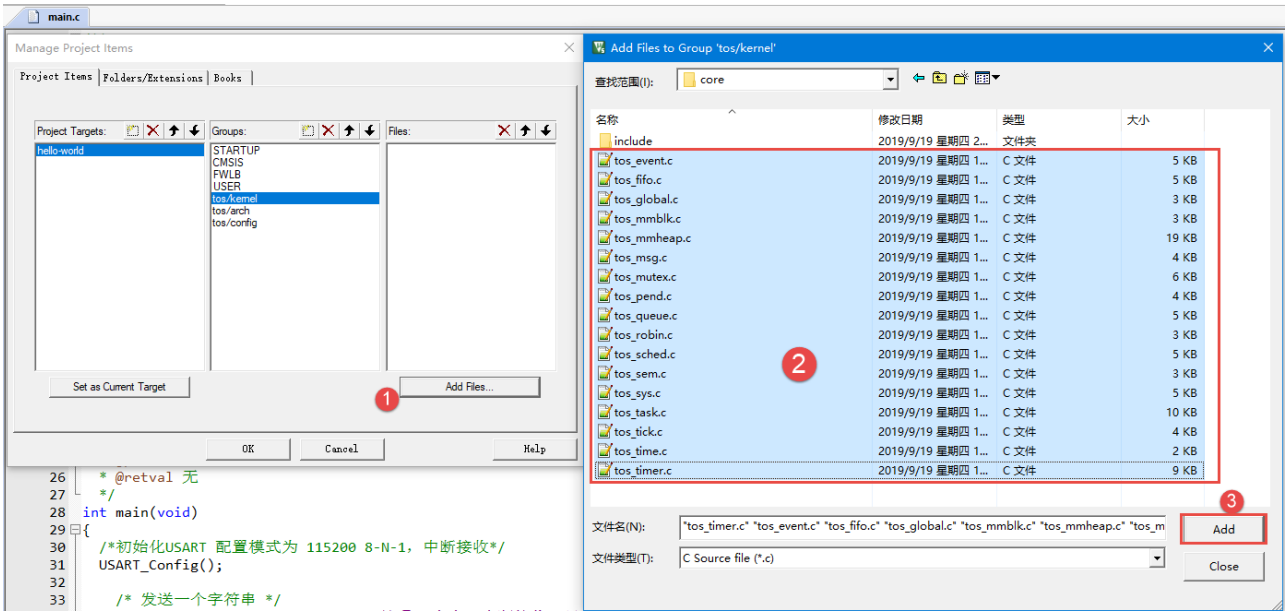


1. 根据下图的提示，新建3个工程分组，分别为tos/kernel、tos/arch、tos/config，这样可以见其名知其意，这些工程分组分别保存TencentOS tiny的内核源码、接口文件、以及配置文件。

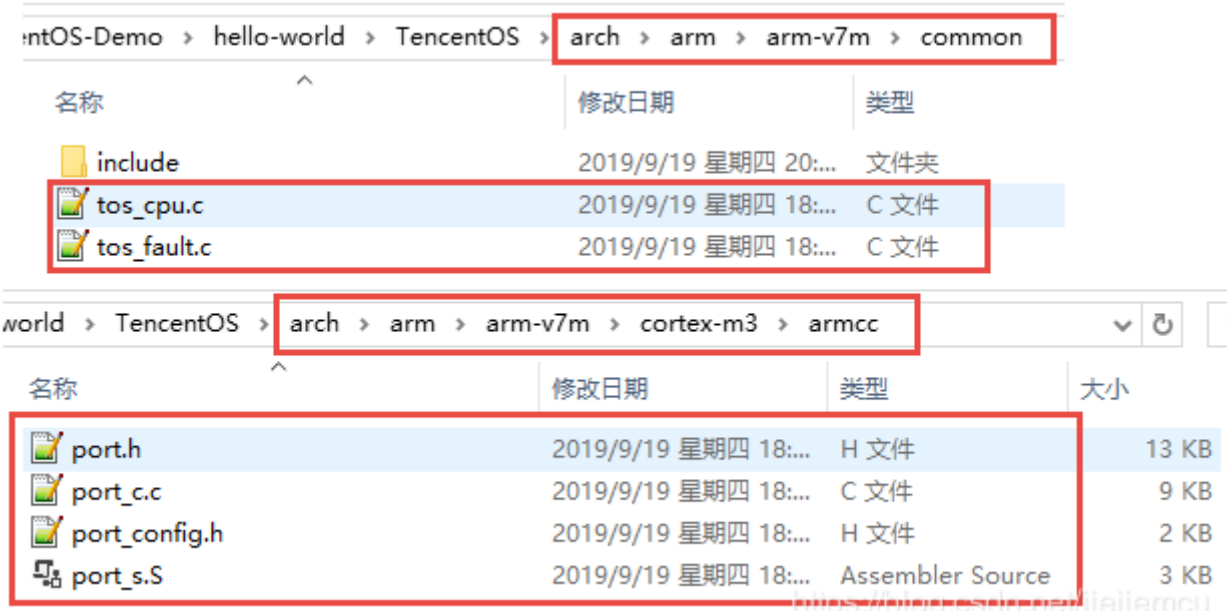


2. 根据下图将TencentOS-Demo\hello-world\TencentOS\kernel\core路径下的所有.c文件添加到tos/kernel工程分组中，也将\TencentOS-Demo\hello-world\TencentOS\kernel\pm目录下的所

有.c文件添加到tos/kernel工程分组中:

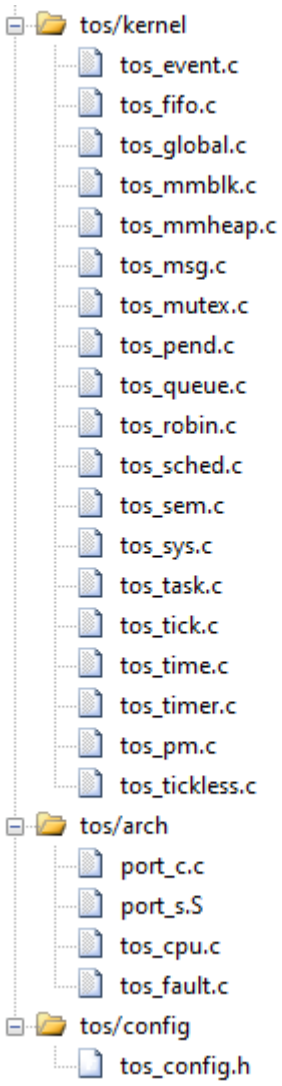


3. 同理将TencentOS-Demo\hello-world\TencentOS\arch\arm\arm-v7m\common路径下的tos_cpu.c、tos_fault.c添加到tos/arch工程分组下，也将TencentOS-Demo\hello-world\TencentOS\arch\arm\arm-v7m\cortex-m3\armcc路径下的 port_s.S、port_c.c文件添加到tos/arch工程分组下



4. 最后再将TencentOS-Demo\hello-world\TencentOS\TOS-CONFIG路径下的tos_config.h文件添加到tos/config工程分组中。

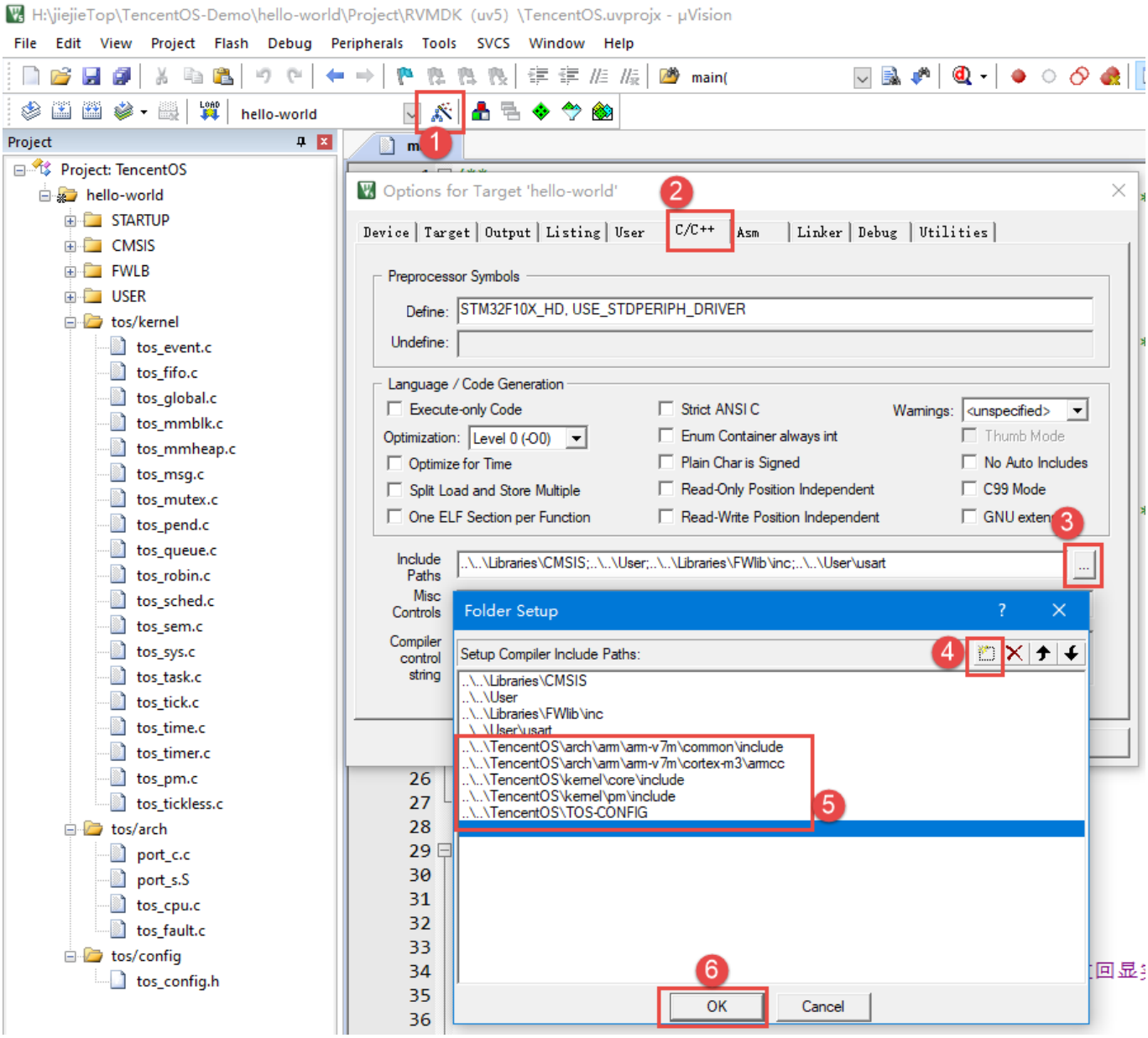
需要注意的是，在tos/arch分组中添加的port_s.S文件，需要在添加时选择文件类型为“All files (*.*)”，添加(*.h)文件类型的时候也需要选择文件类型为“All files (*.*)”



添加完成后的文件：

指定头文件路径

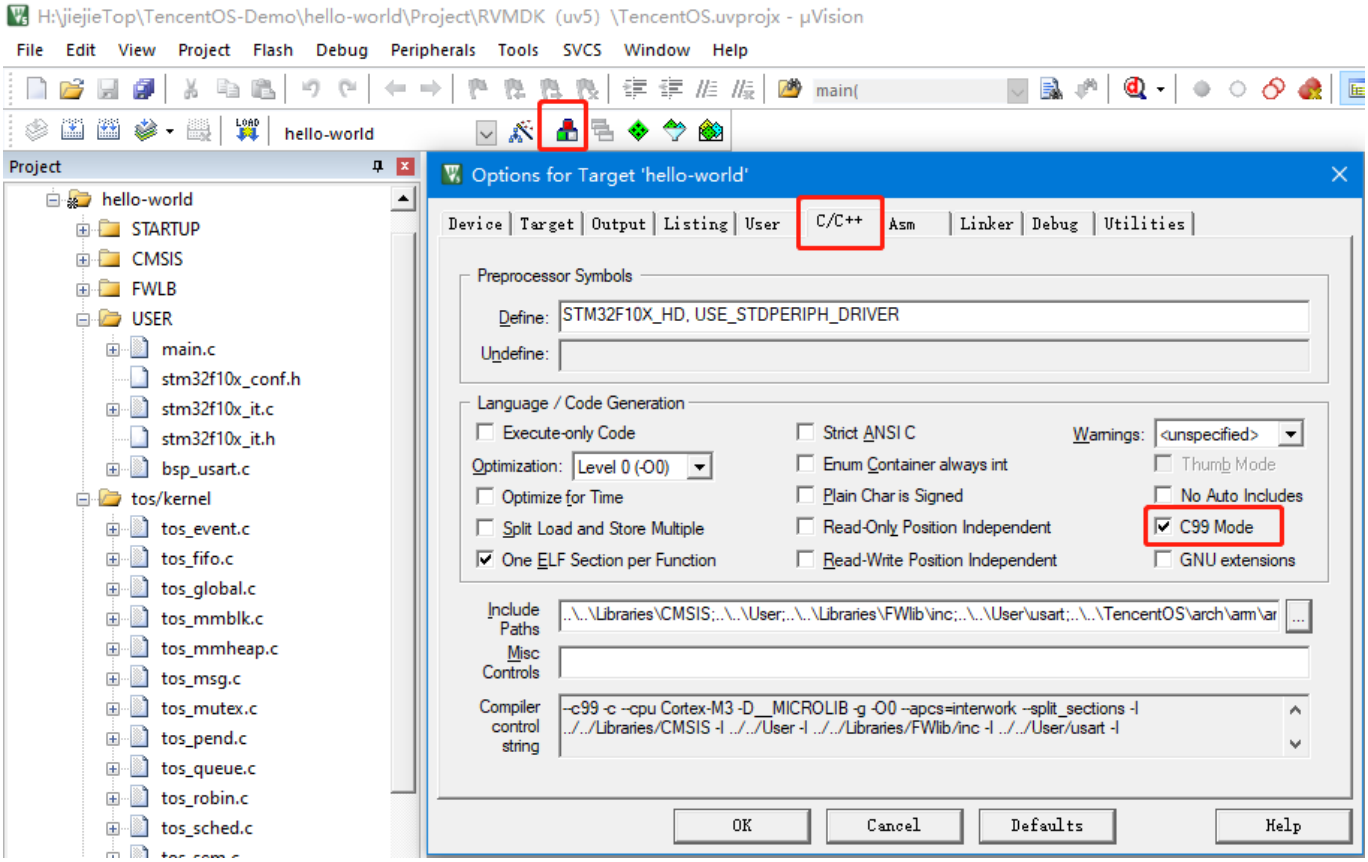
编译时需要为这些源文件指定头文件的路径，否则编译会报错。`TencentOS tiny`的源码中有很多头文件，必须将对应的路径添加到开发环境里。在添加`TencentOS tiny`源码时，一些其他的头文件夹也被复制到了工程目录中，所以这些文件夹的路径也要加到开发环境中。



这些头文件的路径分别是：

```
..\..\TencentOS\arch\arm\arm-v7m\common\include
..\..\TencentOS\arch\arm\arm-v7m\cortex-m3\armcc
..\..\TencentOS\kernel\core\include
..\..\TencentOS\kernel\pm\include
..\..\TencentOS\TOS-CONFIG
```

同时还要在配置中勾选支持C99模式：



尝试编译

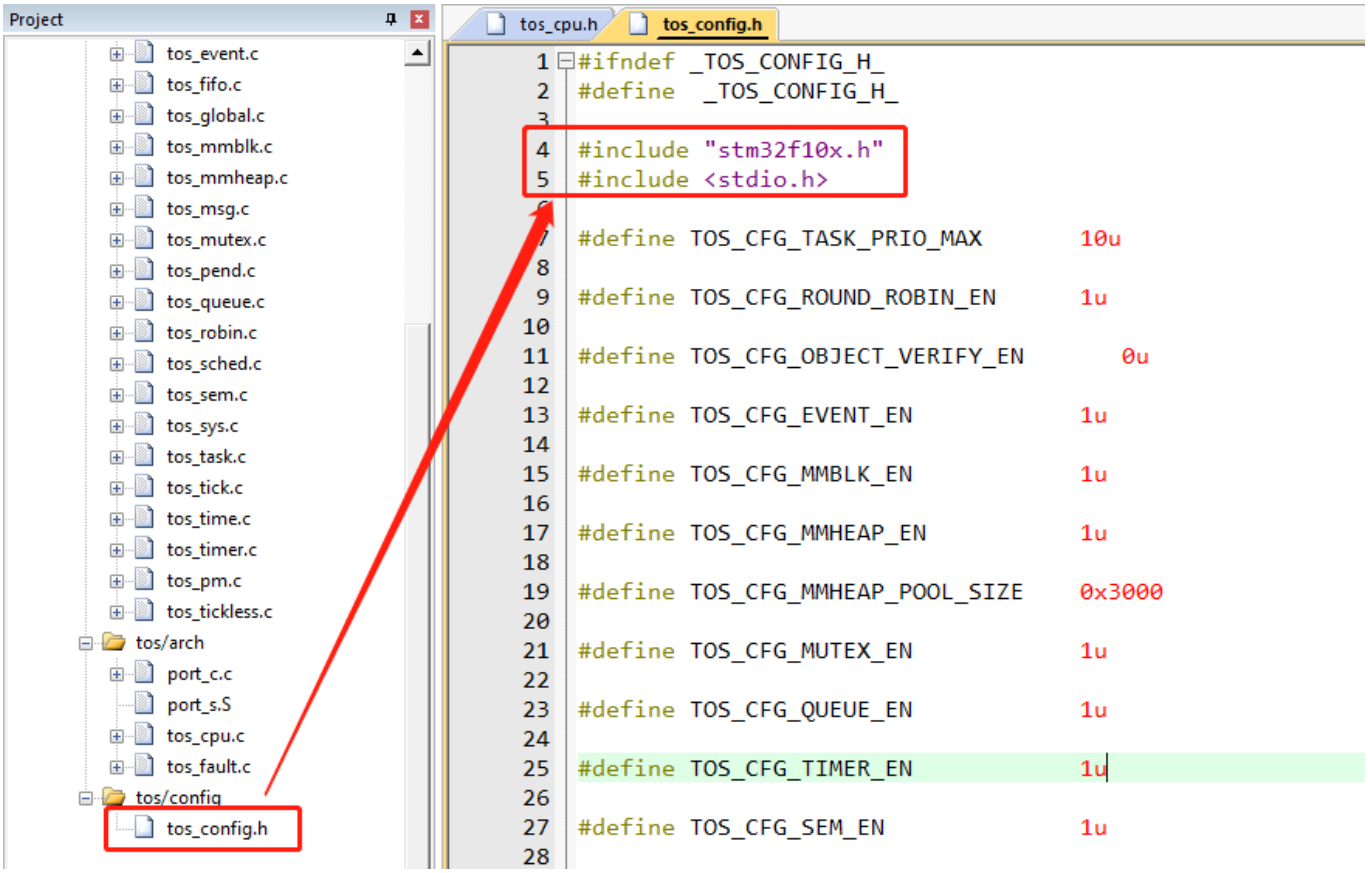
如果你走到这一步，那么可以尝试编译一下，不过我测试时编译是没通过的，原因是缺少了部分头文件：

```
Build Output
*** Using Compiler 'V5.06 update 6 (build 750)', folder: 'C:\Keil_v5\ARM\ARMCC\Bin'
Build target 'hello-world'
compiling tos_queue.c...
..\..\TencentOS\arch\arm\arm-v7m\common\include\tos_cpu.h(66): error:  #20: identifier "size_t" is undefined
                                                               size_t stk_size);
..\..\TencentOS\kernel\core\tos_queue.c: 0 warnings, 1 error
compiling tos_robin.c...
..\..\TencentOS\arch\arm\arm-v7m\common\include\tos_cpu.h(66): error:  #20: identifier "size_t" is undefined
                                                               size_t stk_size);
..\..\TencentOS\kernel\core\tos_robin.c: 0 warnings, 1 error
compiling tos_event.c...
..\..\TencentOS\arch\arm\arm-v7m\common\include\tos_cpu.h(66): error:  #20: identifier "size_t" is undefined
                                                               size_t stk_size);
..\..\TencentOS\kernel\core\tos_event.c: 0 warnings, 1 error
compiling tos_fifo.c...
..\..\TencentOS\arch\arm\arm-v7m\common\include\tos_cpu.h(66): error:  #20: identifier "size_t" is undefined
```

不过这不影响，我们在配置文件tos_config.h中修改一下就好，添加两句话

```
#include "stm32f10x.h"
#include <stdio.h> // 或者 #include <stddef.h>
```

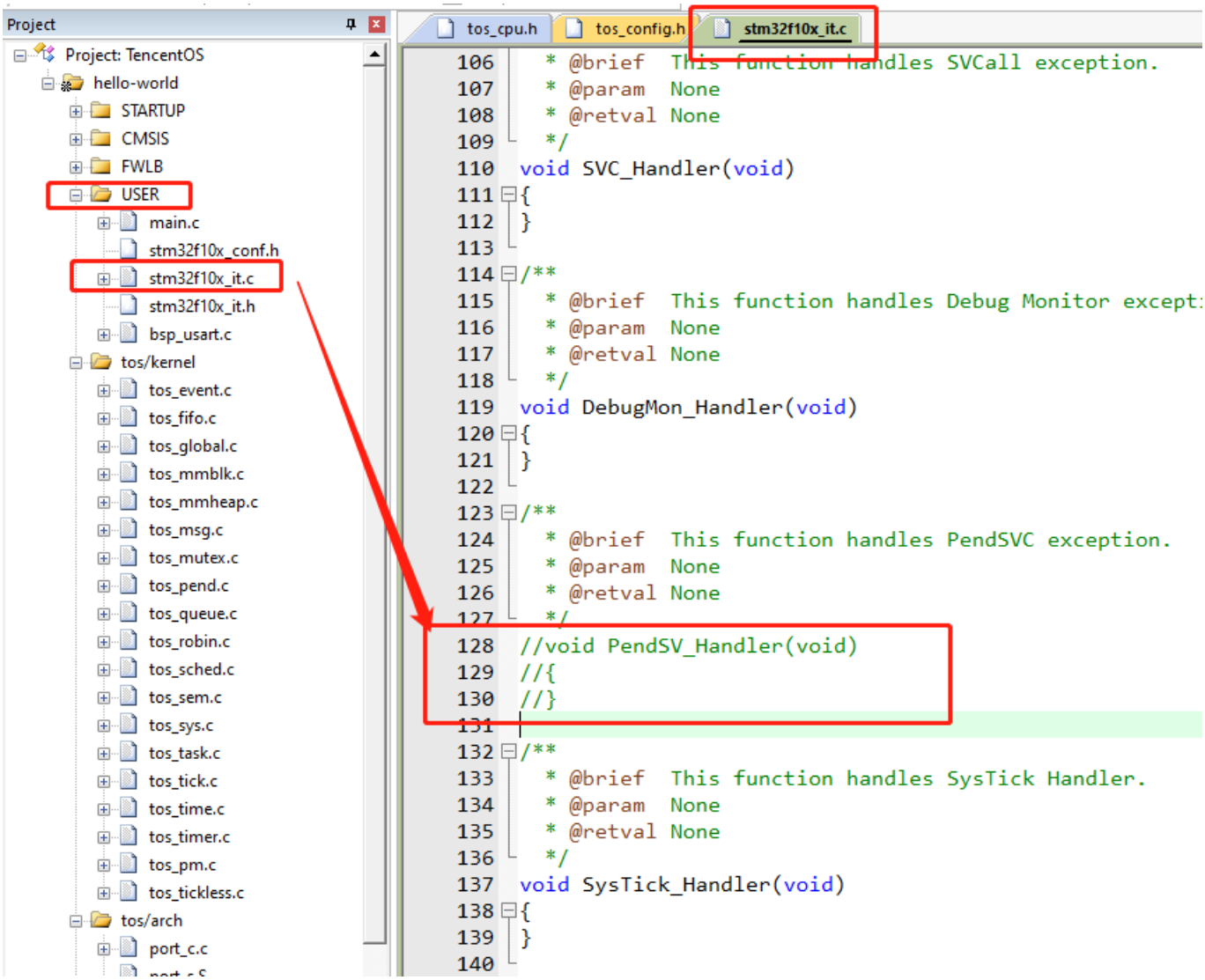
如下图：



修改中断函数

注释PendSV_Handler()函数

鉴于TencentOS tiny已经处理好PendSV与SysTick中断了，就不需要用户自己去处理，所以要在中断相关的源文件（stm32f10x_it.c文件）中**注释**（或者**删除**）PendSV_Handler()函数。



编写SysTick_Handler()函数

SysTick中断服务函数是一个非常重要的函数，TencentOS tiny所有跟时间相关的事情都在里面处理，SysTick就是TencentOS tiny的一个心跳时钟，驱动着TencentOS tiny的运行，就像人的心跳一样，假如没有心跳，我们就相当于“挂掉”，同样的，TencentOS tiny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TencentOS tiny的心跳时钟。代码如下：

注意：SysTick_Handler()中调用的都是TencentOS tiny中的函数，所以需要在stm32f10x_it.c文件中包含tos.h头文件。

```
#include "tos.h"

// SysTick_Handler()函数
void SysTick_Handler(void)
{
  if (tos_knl_is_running())
  {
    tos_knl_irq_enter();
    tos_tick_handler();
    tos_knl_irq_leave();
  }
}
```

编写main函数

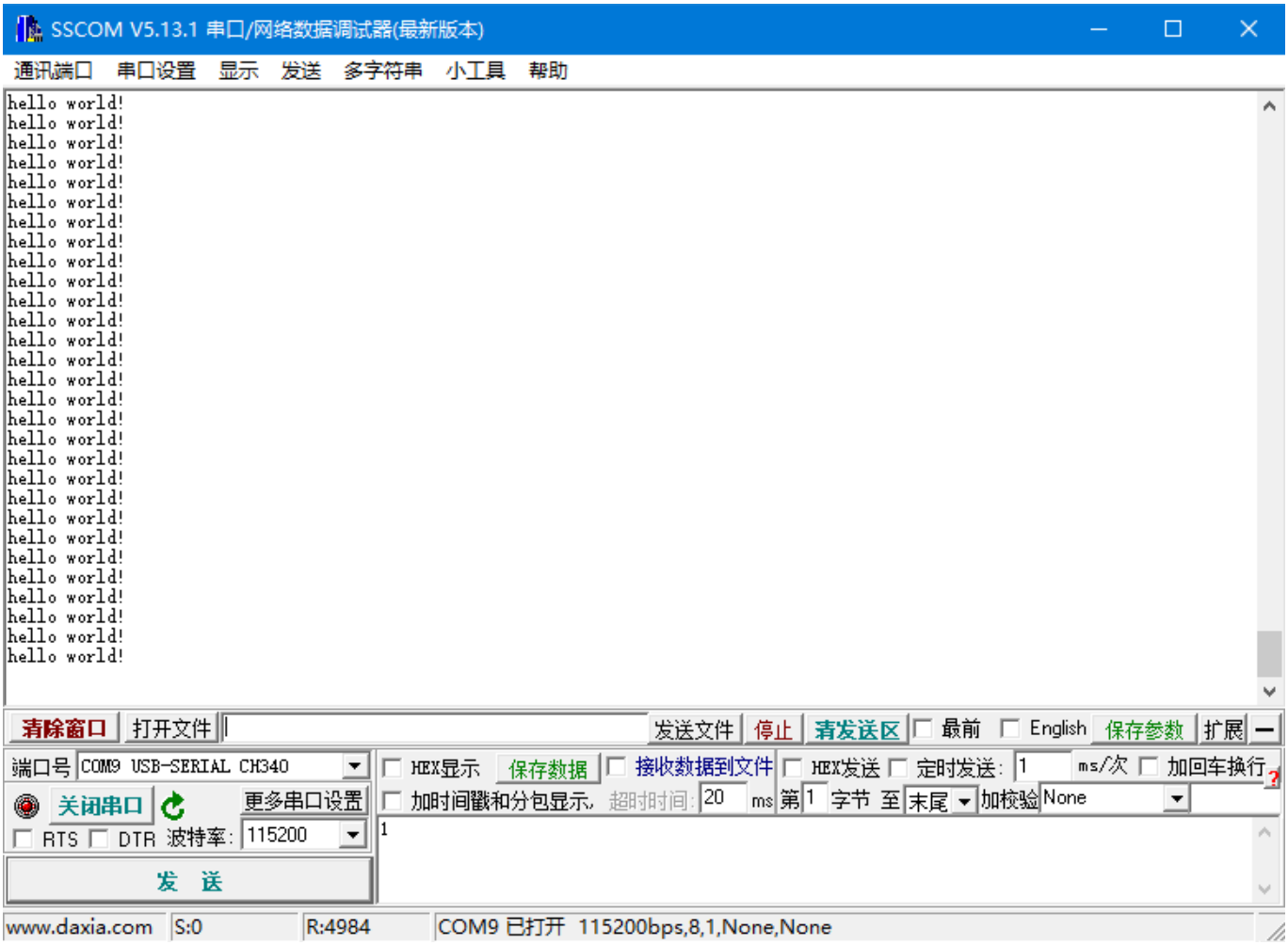
当你走到这一步，编译是不会出错了，此时我们已经完全移植好操作系统了，那么可以编写代码了，现在编写一个测试代码，在main.c文件中：

```
#include "stm32f10x.h"
#include "bsp_usart.h"
#include "tos.h"


k_task_t task;

k_stack_t task_stack[1024];

void test_task(void *Parameter)
{
    while(1)
    {
        printf("hello world!\r\n");
        tos_task_delay(1000);
    }
}

/**
  * @brief  主函数
  * @author 杰杰
  * @retval 无
  */
int main(void)
{
    k_err_t err;

    /*初始化USART 配置模式为 115200 8-N-1，中断接收*/
    USART_Config();

    printf("Welcome to TencentOS tiny\r\n");

    tos_knl_init(); // TOS Tiny kernel initialize

    err = tos_task_create(&task,
                          "task1",
                          test_task,
                          NULL,
                          2,
                          task_stack,
                          1024,
                          20);
    if(err != K_ERR_NONE)
        printf("TencentOS Create task fail! code : %d \r\n",err);
```

```
    tos_knl_start(); // Start TOS Tiny

}
```

下载

然后编译，下载到开发板上，就通过串口可以看到程序已经跑起来了：



end

至此，TencentOS tiny移植到stm32f1的过程全部完成！

喜欢就关注我吧！

相关代码可以在公众号后台获取。