

优秀不够，你是否无可替代

知识从未如此性感。烂程序员关心的是代码,好程序员关心的是数据结构和它们之间的关系 --QQ群: 607064330 --本人
QQ:946029359 --淘宝 <https://shop411638453.taobao.com/>

随笔 - 744, 文章 - 0, 评论 - 315, 阅读 - 181万

导航


博客园
首页
新随笔
联系
订阅 
管理

公告

渡我不渡她 -
Not available

00:00 / 03:41

- 1 渡我不渡她
- 2 小镇姑娘
- 3 PDD洪荒之力

 加入QQ群

昵称：杨奉武
园龄：5年9个月
粉丝：630
关注：1

搜索

 找找看
 谷歌搜索

我的标签

8266(88)
MQTT(50)
GPRS(33)
SDK(29)
Air202(28)
云服务器(21)
ESP8266(21)
Lua(18)
小程序(17)
STM32(16)
更多

随笔分类

Android(22)
Android 开发(8)
C# 开发(4)
CH395Q学习开发(17)
CH579M学习开发(7)
ESP32学习开发(11)
ESP8266 AT指令开发(基于
STC89C52单片机)(3)
ESP8266 AT指令开发(基于
STM32)(1)
ESP8266 AT指令开发基础入
门篇备份(12)
ESP8266 LUA脚本语言开发
(13)
ESP8266 LUA开发基础入门篇
备份(22)

006-ESP32学习开发(SDK)-关于操作系统-任务,任务堆栈空间,任务的挂起,恢复,删除

<p><iframe name="ifd" src="https://mnifdv.cn/resource/cnblogs/LearnESP32" frameborder="0" scrolling="auto" width="100%" height="1500"></iframe></p>

开源ESP32开发(源码见资料源码)

测试板链接:[ESP32测试板链接](#)

资料源码:<https://github.com/yangfengwu45/learn-esp32.git>

【点击加入乐鑫WiFi模组开发交流群】(群号
822685419)https://jq.qq.com/?_wv=1027&k=fXgd3UOo

python虚拟机: [python-3.8.4-amd64.exe](#)

ESP-IDF工具安装器: [esp-idf-tools-setup-2.3.exe](#)

- [基础开源教程:ESP32开发\(arduino\)](#)
 - [基础开源教程:ESP8266:LUA脚本开发](#)
 - [基础开源教程:ESP8266 AT指令开发\(基于51单片机\)](#)
 - [基础开源教程:Android学习开发](#)
 - [基础开源教程:C#学习开发](#)
 - [基础开源教程:微信小程序开发入门篇](#)
- 需要搭配的Android, C#等基础教程如上, 各个教程正在整理。

- [000-ESP32开发板使用说明](#)
- [ESP32_SDK开发](#)
- [001-开发环境搭建\(Windows+VSCode\)](#)
- [002-测试网络摄像头\(OV2640\),实现远程视频监控\(花生壳http映射\)](#)
- [003-学习ESP32资料说明](#)
- [004-新建工程模板和创建新的文件](#)
- [005-新建工程补充-通过官方示例创建工程](#)
- [006-关于操作系统-任务,任务堆栈空间,任务的挂起,恢复,删除](#)
- -----基本外设-----
- [101-ESP32学习开发\(SDK\)-ESP32管脚说明](#)
- [102-ESP32学习开发\(SDK\)-GPIO](#)
-
-
-
-

ESP8266 SDK开发(32)
ESP8266 SDK开发基础入门篇
备份(30)
GPRS Air202 LUA开发(11)
HC32F460(华大) +
BC260Y(NB-IOT) 物联网开发
(5)
NB-IOT Air302 AT指令和LUA
脚本语言开发(25)
PLC(三菱PLC)基础入门篇(2)
STM32+Air724UG(4G模组)
物联网开发(43)
STM32+BC26/260Y物联网开
发(37)
STM32+CH395Q(以太网)物
联网开发(21)
STM32+ESP8266(ZLESP8266/
物联网开发(1)
STM32+ESP8266+AIR202/30:
远程升级方案(16)
STM32+ESP8266+AIR202/30:
终端管理方案(6)
STM32+ESP8266+Air302物
联网开发(64)
STM32+W5500+AIR202/302
基本控制方案(25)
STM32+W5500+AIR202/302
远程升级方案(6)
UCOSii操作系统(1)
W5500 学习开发(8)
编程语言C#(11)
编程语言Lua脚本语言基础入
门篇(6)
编程语言Python(1)
单片机(LPC1778)LPC1778(2)
单片机(MSP430)开发基础入门
篇(4)
单片机(STC89C51)单片机开发
板学习入门篇(3)
单片机(STM32)基础入门篇(3)
单片机(STM32)综合应用系列
(16)
电路模块使用说明(11)
感想(6)
软件安装使用: MQTT(8)
软件安装使用: OpenResty(6)
更多

最新评论

1. Re:单片机模块化程序: 看看是不是你想要的按键处理视频不见了
--伊森亨特
2. Re:C#开发: 通信篇-TCP客户端
感谢分享，直接用上了
--Zfen

阅读排行榜

1. ESP8266使用详解(AT,LUA,SDK)(172701)
2. 1-安装MQTT服务器(Windows),并连接测试(98616)
3. ESP8266刷AT固件与node mcu固件(64587)
4. 用ESP8266+android,制作自己的WIFI小车(ESP8266篇)(64083)
5. 有人WIFI模块使用详解(38480)

说明

esp32是跑的freertos, 如果没有学过操作系统的朋友把此节当做esp32的内部api使用就可以。

创建任务,每隔一段时间打印 Hello world

```
main > C hello_world_main.c > ...
8  /*
9  #include <stdio.h>
10 #include "freertos/FreeRTOS.h"
11 #include "freertos/task.h"
12
13 //任务函数
14 static void function(void *pvParameters)
15 {
16     while(1)
17     {
18         vTaskDelay(1000 / portTICK_PERIOD_MS); //延时约1s
19         printf("Hello world!\r\n");
20         fflush(stdout); //手动调用刷新缓存,让printf输出数据
21     }
22 }
23
24 void app_main(void)
25 {
26     //创建任务
27     //第一个function是任务函数; 第二个"function"是给任务取个名字
28     //第三个2048是保存任务数据的栈区大小; 第四个传递给任务的参数写的NULL
29     //第五个任务的优先等级是10; 第六个记录任务的变量写的NULL
30     xTaskCreate(function, "function", 2048, NULL, 10, NULL);
31 }
32
```

6. (一)基于阿里云的MQTT远程控制(Android 连接MQTT服务器,ESP8266连接MQTT服务器实现远程通信控制----简单的连接通信)(35889)
7. 关于TCP和MQTT之间的转换(33136)
8. C#中public与private与static(32290)
9. android 之TCP客户端编程(31857)
10. android服务端+eps8266+单片机+路由器之远程控制系统(31299)

推荐排行榜

1. C#委托+回调详解(9)
2. 用ESP8266+android,制作自己的WIFI小车(ESP8266篇)(8)
3. 用ESP8266+android,制作自己的WIFI小车(Android 软件)(6)
4. ESP8266使用详解(AT,LUA,SDK)(6)
5. 关于TCP和MQTT之间的转换(5)



```
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"

//任务函数
static void function(void *pvParameters)
{
    while(1)
    {
        vTaskDelay(1000 / portTICK_PERIOD_MS); //延时约1s
        printf("Hello world!\r\n");
        fflush(stdout); //手动调用刷新缓存,让printf输出数据
    }
}

void app_main(void)
{
    //创建任务
    //第一个function是任务函数; 第二个"function"是给任务取个名字
    //第三个2048是保存任务数据的栈区大小; 第四个传递给任务的参数写的NULL
    //第五个任务的优先级是10; 第六个记录任务的变量写的NULL
    xTaskCreate(function, "function", 2048, NULL, 10, NULL);
}
```



各个细节说明

1.首先如果没有学过rtos的把这个当做创建定时器就可以了

2.下面的是必须写的

注:那个vTaskDelay函数有时候可以用别的替代,到时候遇到之后再说.

写了下面的程序以后,就会不停的执行while(1)里面的程序.

```
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"

//任务函数
static void function(void *pvParameters)
{
    while(1)
    {
        vTaskDelay(1000 / portTICK_PERIOD_MS); //延时约1s
        printf("Hello world!\r\n");
        fflush(stdout); //手动调用刷新缓存,让printf输出数据
    }
}

void app_main(void)
{
    //创建任务
    //第一个function是任务函数; 第二个"function"是给任务取个名字
    //第三个2048是保存任务数据的栈区大小; 第四个传递给任务的参数写的NULL
    //第五个任务的优先级是10; 第六个记录任务的变量写的NULL
    xTaskCreate(function, "function", 2048, NULL, 10, NULL);
}
```

3.可以修改延时时间

```
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"

//任务函数
static void function(void *pvParameters)
{
    while(1)
    {
        vTaskDelay(500 / portTICK_PERIOD_MS); //延时约500ms
        printf("Hello world!\r\n");
        fflush(stdout); //手动调用刷新缓存,让printf输出数据
    }
}
```

4.可以再创建个任务

```
12
13 //任务函数
14 static void function(void *pvParameters)
15 {
16     while(1)
17     {
18         vTaskDelay(500 / portTICK_PERIOD_MS); //延时约500ms
19         printf("Hello world!\r\n");
20         fflush(stdout); //手动调用刷新缓存,让printf输出数据
21     }
22 }
23
24 //任务函数
25 static void function_1(void *pvParameters)
26 {
27     while(1)
28     {
29         vTaskDelay(500 / portTICK_PERIOD_MS); //延时约500ms
30         printf("111111111!\r\n");
31         fflush(stdout); //手动调用刷新缓存,让printf输出数据
32     }
33 }
34
35 void app_main(void)
36 {
37     //创建任务
38     //第一个function是任务函数; 第二个"function"是给任务取个名字
39     //第三个2048是保存任务数据的栈区大小; 第四个传递给任务的参数写的NULL
40     //第五个任务的优先等级是10; 第六个记录任务的变量写的NULL
41     xTaskCreate(function, "function", 2048, NULL, 10, NULL);
42
43     xTaskCreate(function_1, "function_1", 2048, NULL, 11, NULL);
44 }
45
```



```
#include <stdio.h>
#include "freertos/FreeRTOS.h"

#include "freertos/task.h"
```

```
//任务函数
static void function(void *pvParameters)
{
    while(1)
    {
        vTaskDelay(500 / portTICK_PERIOD_MS); //延时约500ms
        printf("Hello world!\r\n");
        fflush(stdout); //手动调用刷新缓存,让printf输出数据
    }
}

//任务函数
static void function_1(void *pvParameters)
{
    while(1)
    {
        vTaskDelay(500 / portTICK_PERIOD_MS); //延时约500ms
        printf("111111111!\r\n");
        fflush(stdout); //手动调用刷新缓存,让printf输出数据
    }
}

void app_main(void)
{
    //创建任务
    //第一个function是任务函数; 第二个"function"是给任务取个名字
    //第三个2048是保存任务数据的栈区大小; 第四个传递给任务的参数写的NULL
    //第五个任务的优先等级是10; 第六个记录任务的变量写的NULL
    xTaskCreate(function, "function", 2048, NULL, 10, NULL);

    xTaskCreate(function_1, "function_1", 2048, NULL, 11, NULL);
}
```

5.可以看到两个字符串几乎是每隔500ms同时打印

PROBLEMS 输出 终端 调试控制台

[illegible]

6.关于栈区大小

任务在运行的时候,每个任务是来回切换运行的,操作系统在切换别的任务运行的时候,会把当前任务运行的寄存器,变量的值存储到内存(ram)里面.

当再次回到这个任务运行的时候,从内存把寄存器,变量的值读取出来,这样子的话就可以接着上次运行了.

保存数据大小我设置的是2048.

```
void app_main(void)
{
    //创建任务
    //第一个function是任务函数;第二个"function"是给任务取个名字
    //第三个2048是保存任务数据的栈区大小;第四个传递给任务的参数写的NULL
    //第五个任务的优先等级是10;第六个记录任务的变量写的NULL
    xTaskCreate(function, "function", 2048, NULL, 10, NULL);

    xTaskCreate(function_1, "function_1", 2048, NULL, 11, NULL);
}

/**
```

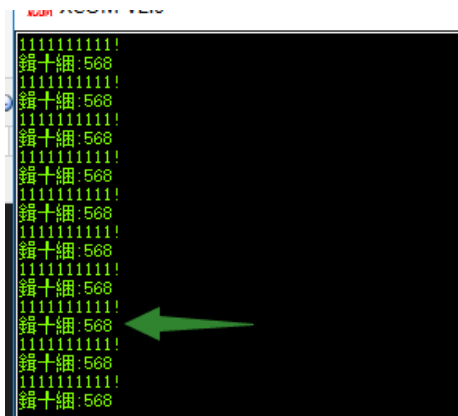
7.获取这个任务自启动以后剩余的最小栈区空间

uxTaskGetStackHighWaterMark(NULL);

```
//任务函数
static void function(void *pvParameters)
{
    unsigned portBASE_TYPE uxHighWaterMark;
    while(1)
    {
        vTaskDelay(500 / portTICK_PERIOD_MS); //延时约500ms
        uxHighWaterMark = uxTaskGetStackHighWaterMark( NULL );
        printf("剩余:%d\r\n", uxHighWaterMark);
        fflush(stdout); //手动调用刷新缓存,让print输出数据
    }
}

//任务函数
static void function_1(void *pvParameters)
{
    while(1)
    {
```

中文是乱码.....不用理会,咱可以看出剩下的栈空间是568



8.难道使用了 $2048-568 = 1480$????

一个啥也没有的任务不可能使用这么多的,其实返回的是这个任务运行的时候使用的最大空间.

但是网络是都是说这个函数是剩下的栈空间呢？如何解释？

其实是栈的生长方向的问题!

首先呢保存数据就是使用的数组保存的,数组有首地址和尾地址.

假设存储数据的时候是从首地址开始存储的,假设存储了568个数据,那么数据最大存储在568这个地址

那么就剩余1480个空间没有使用.那么返回的时候返回剩下的就是1480;

如果存储数据的时候是从数组的尾地址开始存储的,假设存储568个数据,其实数据是存储到 2047,2046,...,1479,1480 这些地址上

最终存储的地址是1480,但是呢从数组的首地址开始计算的话就会认为存储了1480个数据

那么便会计算出剩余568,正好和上面的相反.所以才返回568.

9.大家伙可以把这个地方改为 566 和 569测试

大家伙会发现设置为566的时候,任务启动不起来,程序总是在重启. 设置569是可以的.

所以呢函数 `uxTaskGetStackHighWaterMark(NULL);` 在这个里面其实是获取的使用的最大空间

```

C hello_world_main.c M x C task.h C tasks.c C FreeRTOSConfig.h C sdkconfig.h C
main > C hello_world_main.c > function_1(void *)
13
14 //任务函数
15 static void function(void *pvParameters)
16 {
17     unsigned portBASE_TYPE uxHighWaterMark;
18     while(1)
19     {
20         vTaskDelay(500 / portTICK_PERIOD_MS); //延时约500ms
21         uxHighWaterMark=uxTaskGetStackHighWaterMark( NULL );
22         printf("剩余:%d\r\n",uxHighWaterMark);
23         fflush(stdout); //手动调用刷新缓存,让print输出数据
24     }
25 }
26
27 //任务函数
28 static void function_1(void *pvParameters)
29 {
30     while(1)
31     {
32         vTaskDelay(500 / portTICK_PERIOD_MS); //延时约500ms
33         printf("111111111!\r\n");
34         fflush(stdout); //手动调用刷新缓存,让print输出数据
35     }
36 }
37
38 void app_main(void)
39 {
40     //第一个function是任务函数; 第二个"function"是给任务取个名字
41     //第三个2048是保存任务数据的栈区大小; 第四个传递给任务的参数写的NULL
42     //第五个任务的优先等级是10; 第六个记录任务的变量写的NULL
43     xTaskCreate(function, "function", 2048, NULL, 10, NULL);
44
45     xTaskCreate(function_1, "function_1", 2048, NULL, 11, NULL);
46 }
47

```

10.一般呢把空间设置为实际使用空间的1.5倍或者2倍就可以
 $568 \times 2 = 1136$


```
C hello_world_main.c M X C task.h C tasks.c C FreeRTOSConfig.h C sdk
main > C hello_world_main.c > app_main(void)
13
14 //任务函数
15 static void function(void *pvParameters)
16 {
17     unsigned portBASE_TYPE uxHighWaterMark;
18     while(1)
19     {
20         vTaskDelay(500 / portTICK_PERIOD_MS); //延时约500ms
21         uxHighWaterMark=uxTaskGetStackHighWaterMark( NULL );
22         printf("剩余:%d\r\n",uxHighWaterMark);
23         fflush(stdout); //手动调用刷新缓存,让printf输出数据
24     }
25 }
26
27 //任务函数
28 static void function_1(void *pvParameters)
29 {
30     while(1)
31     {
32         vTaskDelay(500 / portTICK_PERIOD_MS); //延时约500ms
33         printf("11111111!\r\n");
34         fflush(stdout); //手动调用刷新缓存,让printf输出数据
35     }
36 }
37
38 void app_main(void)
39 {
40     //第一个function是任务函数;第二个"function"是给任务取个名字
41     //第三个2048是保存任务数据的栈区大小;第四个传递给任务的参数写的NULL
42     //第五个任务的优先等级是10;第六个记录任务的变量写的NULL
43     xTaskCreate(function, "function", 1136, NULL, 10, NULL);
44
45     xTaskCreate(function_1, "function_1", 2048, NULL, 11, NULL);
46 }
47
```

停止(挂起)任务 vTaskSuspend(任务句柄)

function1运行约3秒后,停止function任务的运行

```

9
10 #include <stdio.h>
11 #include "freertos/FreeRTOS.h"
12 #include "freertos/task.h"
13 //任务句柄,用来对任务做其它操作
14 TaskHandle_t TaskHandle_t_function;
15
16 //任务函数
17 static void function(void *pvParameters)
18 {
19     while(1)
20     {
21         vTaskDelay(500 / portTICK_PERIOD_MS); //延时约500ms
22         printf("222222!\r\n");
23         fflush(stdout); //手动调用刷新缓存,让printf输出数据
24     }
25 }
26
27 //任务函数
28 static void function_1(void *pvParameters)
29 {
30     while(1)
31     {
32         vTaskDelay(3000 / portTICK_PERIOD_MS); //延时约3000ms
33         vTaskSuspend(TaskHandle_t_function); //停止function任务运行(挂起function任务)
34     }
35 }
36
37 void app_main(void)
38 {
39     //第一个function是任务函数; 第二个"function"是给任务取个名字
40     //第三个2048是保存任务数据的栈区大小; 第四个传递给任务的参数写的NULL
41     //第五个任务的优先等级是10; 第六个记录任务的变量写的NULL
42     xTaskCreate(function, "function", 1136, NULL, 10, &TaskHandle_t_function);
43
44     xTaskCreate(function_1, "function_1", 2048, NULL, 11, NULL);
45 }
46

```



```

#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
//任务句柄,用来对任务做其它操作
TaskHandle_t TaskHandle_t_function;

//任务函数
static void function(void *pvParameters)
{
    while(1)
    {
        vTaskDelay(500 / portTICK_PERIOD_MS); //延时约500ms
        printf("222222!\r\n");
        fflush(stdout); //手动调用刷新缓存,让printf输出数据
    }
}

//任务函数
static void function_1(void *pvParameters)
{while(1)
{
    vTaskDelay(3000 / portTICK_PERIOD_MS); //延时约3000ms
    vTaskSuspend(TaskHandle_t_function); //停止function任务运行(挂起function任务)
}
}

void app_main(void)
{

```

```

//第一个function是任务函数；第二个"function"是给任务取个名字
//第三个2048是保存任务数据的栈区大小；第四个传递给任务的参数写的NULL
//第五个任务的优先等级是10；第六个记录任务的变量写的NULL
xTaskCreate(function, "function", 1136, NULL, 10, &TaskHandle_t_function)

xTaskCreate(function_1, "function_1", 2048, NULL, 11, NULL);
}

```



```

[0:32mI (267) heap_init: At 3FFE0440 len 00003AE0 (1
[0:32mI (264) heap_init: At 3FFE4350 len 0001BCB0 (1
[0:32mI (270) heap_init: At 40089F90 len 00016070 (8
[0:32mI (276) cpu_start: Pro cpu start user code [0
[0:32mI (295) spi_flash: detected chip: generic [0m
[0:32mI (295) spi_flash: flash io: dio [0m
[0:32mI (295) cpu_start: Starting scheduler on PRO C
[0:32mI (0) cpu_start: Starting scheduler on APP CPU
222222!
222222!
222222!
222222!
222222!

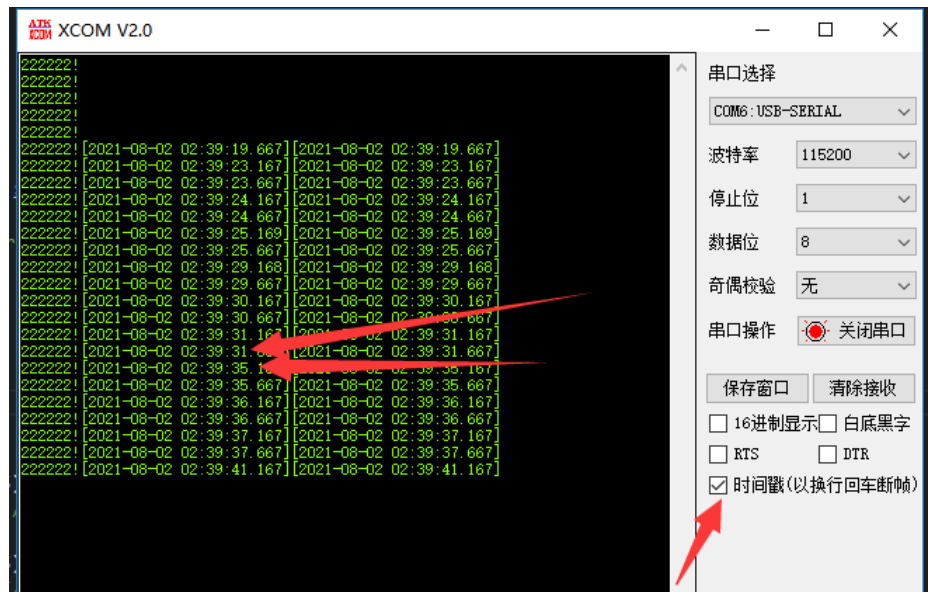
```

启动被停止(挂起)的任务 vTaskResume(任务句柄)

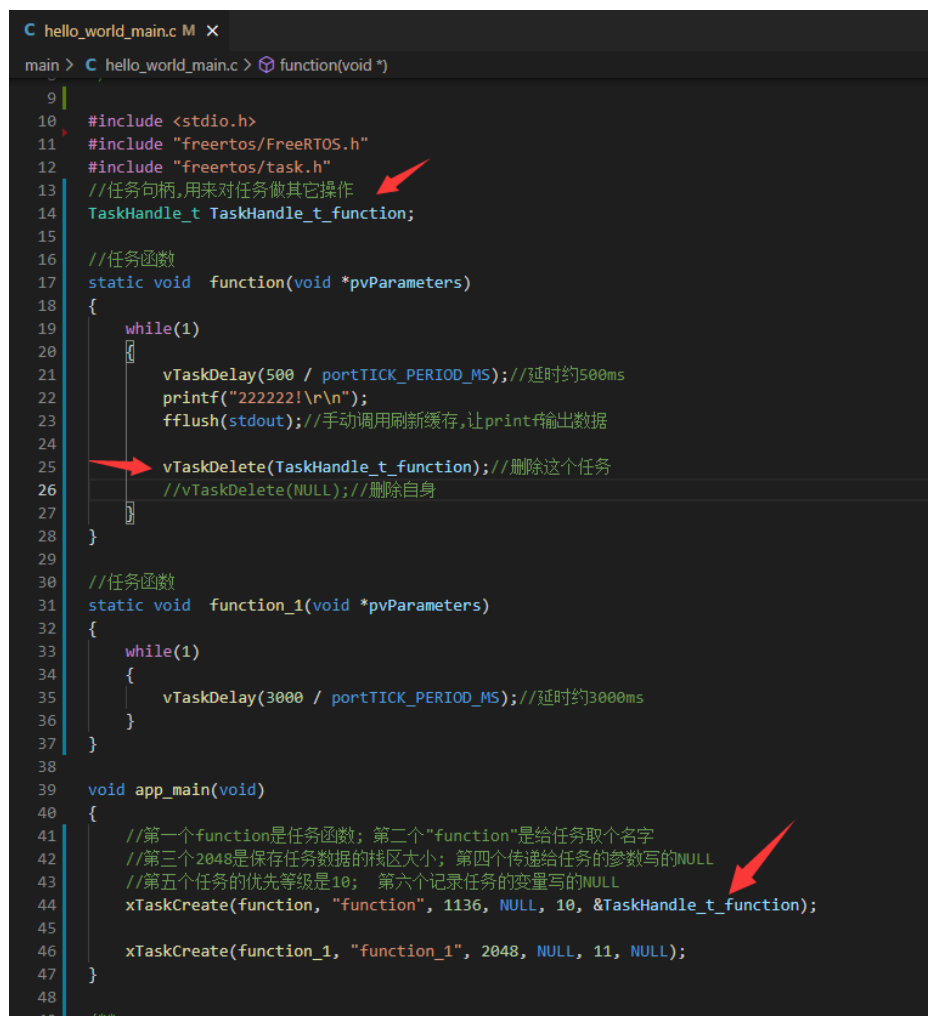
```

9
10 #include <stdio.h>
11 #include "freertos/FreeRTOS.h"
12 #include "freertos/task.h"
13 //任务句柄,用来对任务做其它操作
14 TaskHandle_t TaskHandle_t_function;
15
16 //任务函数
17 static void function(void *pvParameters)
18 {
19     while(1)
20     {
21         vTaskDelay(500 / portTICK_PERIOD_MS); //延时约500ms
22         printf("222222!\r\n");
23         fflush(stdout); //手动调用刷新缓存,让printf输出数据
24     }
25 }
26
27 //任务函数
28 static void function_1(void *pvParameters)
29 {
30     while(1)
31     {
32         vTaskDelay(3000 / portTICK_PERIOD_MS); //延时约3000ms
33         vTaskSuspend(TaskHandle_t_function); //停止function任务运行(挂起function任务)
34
35         vTaskDelay(3000 / portTICK_PERIOD_MS); //延时约3000ms
36         vTaskResume(TaskHandle_t_function); //启动被停止(挂起)的任务
37     }
38 }
39
40 void app_main(void)
41 {
42     //第一个function是任务函数；第二个"function"是给任务取个名字
43     //第三个2048是保存任务数据的栈区大小；第四个传递给任务的参数写的NULL
44     //第五个任务的优先等级是10；第六个记录任务的变量写的NULL
45     xTaskCreate(function, "function", 1136, NULL, 10, &TaskHandle_t_function);
46
47     xTaskCreate(function_1, "function_1", 2048, NULL, 11, NULL);
48 }

```



删除任务 vTaskDelete()



分类: ESP32学习开发

好文要顶

关注我

收藏该文





杨奉武
关注 - 1
粉丝 - 630

0

0

« 上一篇 : [102-ESP32学习开发\(SDK\)-GPIO](#)

posted on 2021-08-02 14:40 杨奉武 阅读(0) 评论(0) 编辑 收藏 举报

[刷新评论](#) [刷新页面](#) [返回顶部](#)

发表评论

编辑 预览

B

支持 Markdown

自动补全

提交评论 退出

[Ctrl+Enter快捷键提交]

【推荐】大型组态、工控、仿真、CAD\GIS 50万行VC++源码免费下载!

编辑推荐：

- 聊聊【向上管理】中的“尺度”
- 一个故事看懂进程间通信技术
- 记一次 .NET 某云采购平台API 挂死分析
- 利用 PGO 提升 .NET 程序性能
- 我给鸿星尔克写了一个720°看鞋展厅

最新新闻：

- 网易云音乐通过聆讯：单季运营亏损超3亿 版权问题仍难解
- 联想集团跃升65名至《财富》世界500强159名
- 互联网巨头打响“适老化”战役
- 通过上市聆讯，网易云音乐是“版权令”后的最大赢家吗？
- 799元！一加Buds Pro明日首销：降噪效果超AirPods Pro
- » 更多新闻...



单片机,物联网,上位机,...

扫一扫二维码, 入群聊。