STM32 开发环境搭建基础问题

安装好 keil4 之后搭建第一个环境报错

Keil 编译器报错: Error:L6218E:UndefinedsymbolSystemInit (startup_stm32f10x_hd.o)

该错误是:

这种错误是 startup_stm32f10x_hd.s 汇编文件里面定义了 SystemInit, 但是主程序没有定义该函数。所以解决方法有三种。

1: 屏蔽该 startup_stm32f10x_hd.s 汇编文件的三条语句

```
|.ceac|, code, REMDORDI
145
146; Reset handler
147 Reset_Handler PROC
                 EXPORT Reset_Handler
148
                                                   [WEAK]
                IMPORT __main
; IMPORT SystemInit
149
150
151
                ; LDR RO, =SystemInit
152
                 ; BLX
                          R0
                 LDR RO, =_main
153
                  BX
                         R0
                  ENDP
156
```

这种我不太喜欢, 万一出问题了呢,

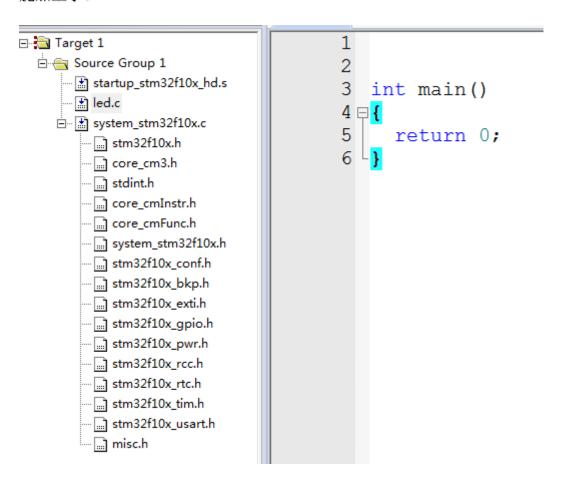
毕竟是官方给的完整版。

2: 在 main 函数增加一个 SystemInit 空函数

但是这样会影响软件代码的结构,我不是很喜欢

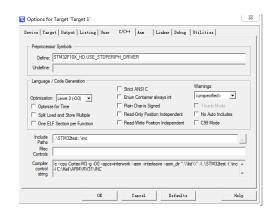
3:增加 system_stm32f10x.c,在工程里面增加该文件,但是增加该文件之后还要增加一系列的其他文件才能编译过,虽然麻烦,但是我还是喜欢这种。

要往工程当中增加一系列的 h 文件,但是这些 h 文件都是后面要用的,所以就干脆加上了。



这样就能正常编译通过了。

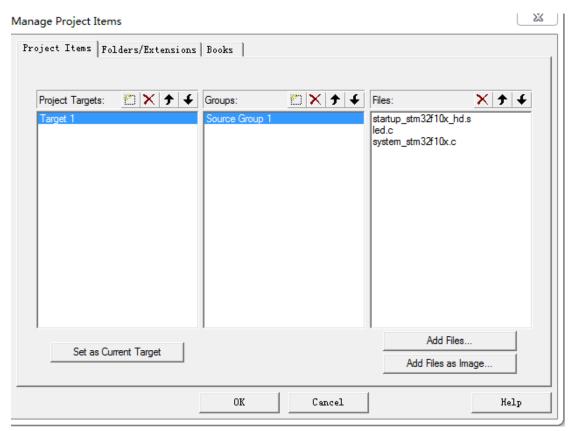
然后还有一点要注意就是要在编译器选项 C/C++中指定头文件路径,修改 define



API 函数接口文件添加

在建立一个工程之后,如果要用到其他 C 文件里面的 API 接口,那么就要用 keil

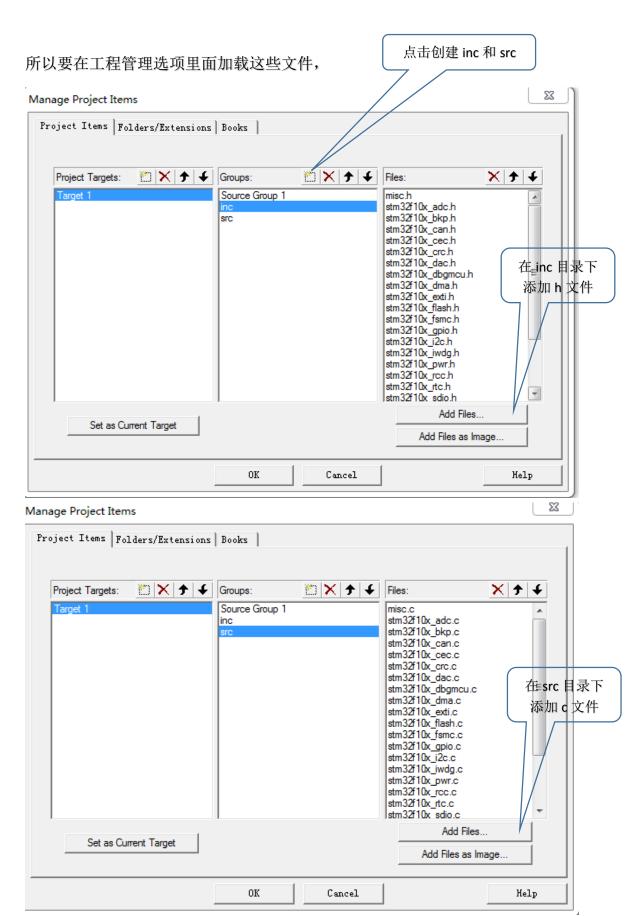




比如 STM32 要用到 SPI, I2C, PWM 这些模块的外设控制器,所以需要包含这些控制器的 API 库文件

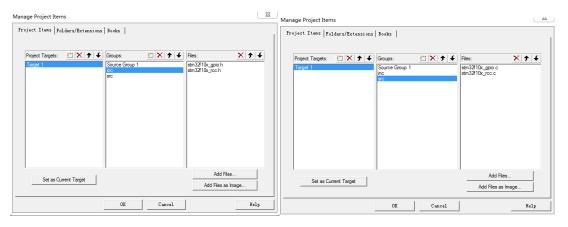
STM32test • inc			misc.c	stm32f10x_tim.c
到库中 ▼	共享 ▼ 新建文件实		stm32f10x_adc.c	stm32f10x_usart.c
3947	六季 · 奶油又什六		stm32f10x_bkp.c	stm32f10x_wwdg.c
	misc.h	stm32f10x_tim.h	stm32f10x_can.c	
	stm32f10x_adc.h	stm32f10x_usart.h	stm32f10x_cec.c stm32f10x_crc.c	
	stm32f10x_bkp.h stm32f10x_can.h	stm32f10x_wwdg.h		
	stm32f10x_can.in		stm32f10x_dac.c	
D:)	stm32f10x_crc.h		stm32f10x_dbgmcu.c	
	stm32f10x_dac.h		stm32f10x_dma.c	
	stm32f10x_dbgmcu.h		stm32f10x_exti.c	
	stm32f10x_dma.h		stm32f10x_flash.c	
	stm32f10x_exti.h		stm32f10x_fsmc.c	
	stm32f10x_fiash.h		stm32f10x_gpio.c	
rer (\\HI'	stm32f10x_gpio.h		stm32f10x_i2c.c	
	stm32f10x_i2c.h		stm32f10x_iwdg.c	
	stm32f10x_iwdg.h		stm32f10x_pwr.c	
	stm32f10x_pwr.h		stm32f10x_rcc.c	
	stm32f10x_rcc.h		stm32f10x_rtc.c	
	stm32f10x_rtc.h		stm32f10x_sdio.c	
	stm32f10x_spi.h		stm32f10x_spi.c	

有 h 文件就有 API 接口 实现的 C 文件



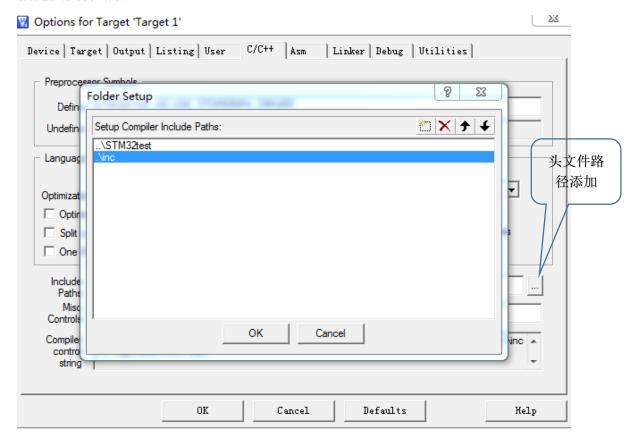
Src 和 inc 目录名字是自己取的,如果还要添加其他文件的 API 函数方法和上面一样我是把 STM32 所有外设控制器都加进来了,其实用不着。需要什么加什么就是了。

这里我们选择要用的 gpio 来操作 stm32, 其余的外设现在用不了



为什么要加入 RCC 头文件,因为我们知道操作外设是需要时钟的,GPIO 也是需要时钟。

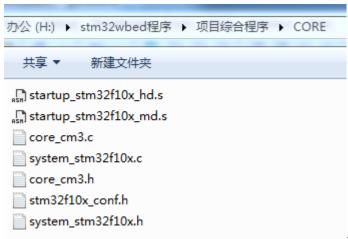
这里还有一个很烦人的地方,就是工程管理器加了头文件之后,那么还要给工程 指定头文件路径



就像这样,如果不加头文件路径,在使用该头文件里面的 api 函数的时候会编译报错。还要在路径中加上 stm32f10x.h,因为这里面定义了很多数据类型和地址。

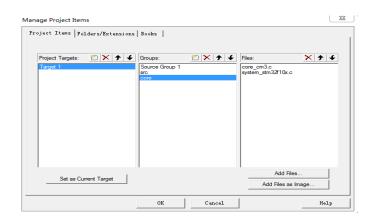
如果要定义 u32, u16, u8 这些别名的话,只有 stm32f10x.h 是不够的,要加入

core_cm3.c 和 core_cm3.h 这个文件



一般都是用一个 core 文件夹将

需要用的内核初始化文件放在一起,所以标准的一个 led 项目是必须要包含这一系列文件的。



ок

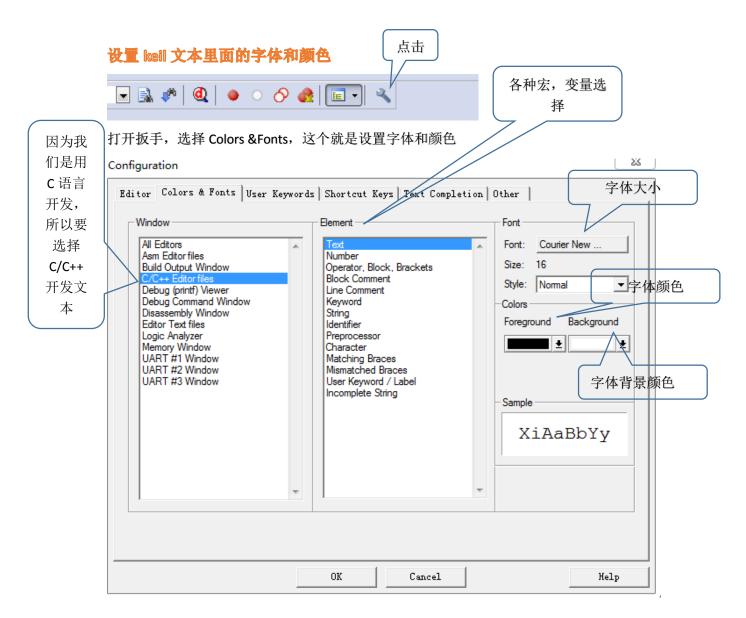
所以项目管理器是用来加 C 文件的目录的,不用加什么 h 文件的目录, h 文件的

Cancel

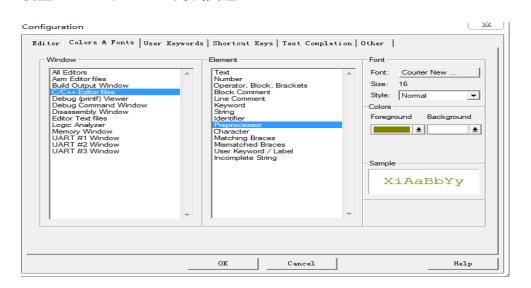
Defaults

Help

目录只需要在 C/C++里面的路径加载就可以了, 所以前面讲的项目管理很扯淡



设置 include 和 define 为浅黄色



反正设置字体大小,颜色和背景颜色都在这里。

上面讲的是设置系统默认关键字的颜色,如果是用户自定义的关键字,自己搞什么创造发明定义了一些关键字的别名,如果要用颜色表示怎么办呢?

没有定义前是这样

```
1
   #include<stdio.h>
 2
   #include "stm32f10x.h"
 3
 4
   typedef uint32 t zzc;
 5
            //没有颜色
   u16 zh;
 6
   zzc ah; //没有颜色
 7
 8
 9
   int main()
10 ₽ {
11
     return 0;
12 <sup>L</sup> }
```

老规矩打开扳手

面增加关键字的名字,然后确定,就可以看见自己定义的关键字变成红色了。

以上就是环境搭建的步骤。

搭建好的环境是这样的

```
☑ E:\Desktop\STM32test\STM32led.c - μVision4

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
                                                                      🖃 🗟 🥙 | @Q | 🍑 O 🔗 🚵 | 🛅 🔍
 □ 🚰 🛃 🗿 🐰 😘 🖺 🔊 🖭 🖛 ⇒ | 🏞 豫 豫 豫 [ 建 建 准 报 | 💆 w16
 🥩 🍱 🕮 🧼 📖 | Long | Target 1
                               🕟 🔊 🚹 🖶 🧇 🚳
Project
                            led.c stm32f10x.h
⊟- 🛅 Target 1
                                     1 #include<stdio.h>
  🖹 🍓 Source Group 1
                                     2 #include "stm32f10x.h"
     startup_stm32f10x_hd.s
    ⊞ led.c
                                     3
  🗏 🚔 src
                                     4
    stm32f10x_gpio.c
    ± stm32f10x_rcc.c
                                     5
  ⊟ <del>(</del> core
                                     6
    core_cm3.c
    system_stm32f10x.c
                                     7
                                     8
                                     9 int main()
                                   10 ₽ {
                                   11
                                   12
                                           return 0;
                                   13 \}
```