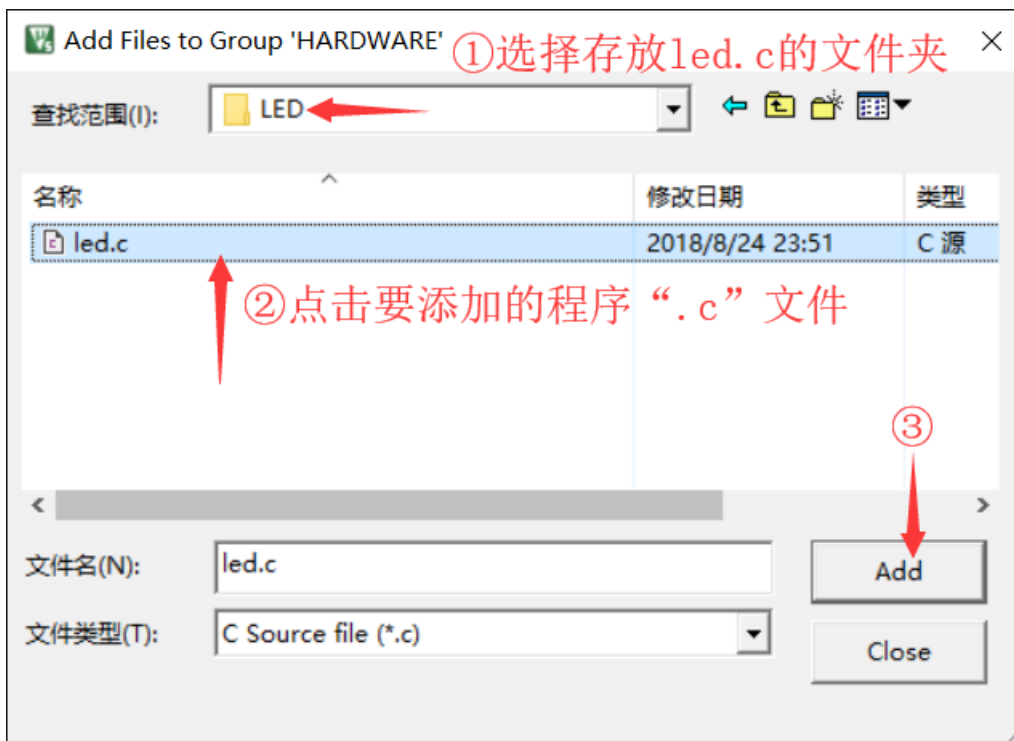
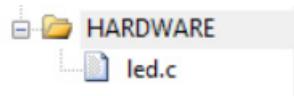


(P 2-4-63) 添加 led.c



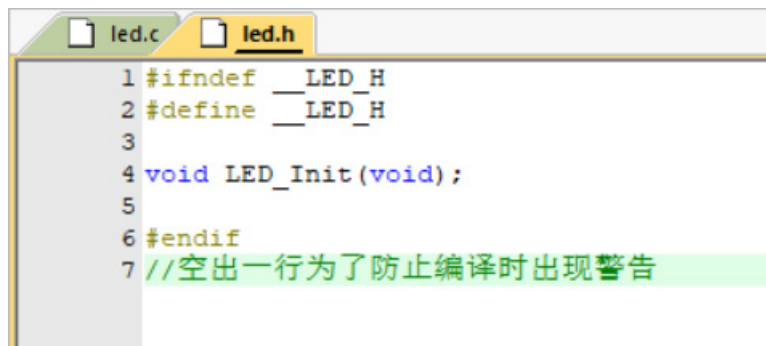
(P 2-4-64) 添加 led.c

点击“Add”添加结束后，窗口并不会自动关闭（避免要添加多个文件时重复操作带来的麻烦），所以我们点击右上角的“叉号”或是右下角的“Close”即可关闭窗口。再点击上一级窗口下方的“OK”，这时我们可以注意到右侧窗口中，“HARDWARE”左侧出现了其他文件夹前都有的“”，点击，就可以看到 HARDWARE 文件夹下出现了名为“led.c”的子文件。如下图。这样我们的配置工作就相当于结束了。



(P 2-4-65) led.c 子文件

(6) 接下来我们就要正式开始讲解代码内容了。我们来看书写好之后的代码文件是什么样的。首先先看“led.h”文件。



(P 2-4-66) led.h 文件

在 51 的学习中我们已经知道了“#ifndef”、“#define”、“#endif”的作用。不知道的话，我们利用下图简要解释一下。想要更深入了解的同学，自行百度即可。

提示：头文件中，使用 #ifndef #define #endif 条件编译，避免头文件内容重复定义。

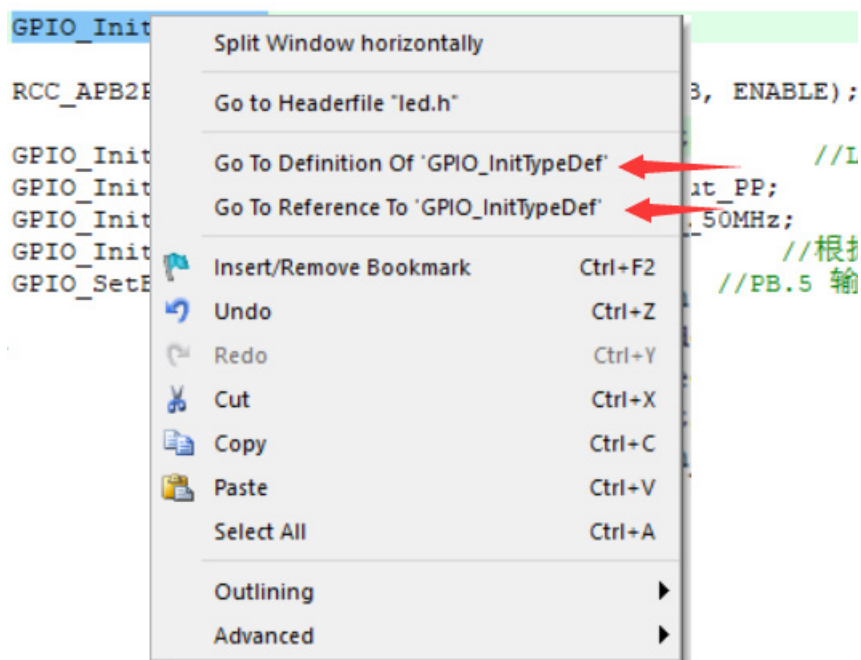
为了成功调用头文件，我们在“#ifndef”和“#define”后需要空一格，打上两个英文格式下的“_”（学习过 c 语言后我们知道，编程使用的符号都是全世界通用的英文格式），再键入我们头文件的名称，在本节中即是“LED”，再键入一次“_”，打上“H”，就成功地定义了我们的头文件。再下方的“void LED_Init(viod);”，是我们自己命名的 led 初始化函数（“LED_Init”也可以取成别的名字，只是这样方便程序员与阅读者理解，可以理解为默认的格式）。在头文件中定义了函数后，我们只要在其他文件的开头调用该头文件，就可以调用该头文件中包含的所有函数。切记，不要忘了在头文件声明函数体时，要在后方打上“;”。这一点我们在之前的学习过程中也应当了解过。

接下来我们看“led.c”中的代码。首先在开头调用我们刚才提到的头文件“led.h”，以及包涵我们即将要使用到的多种函数的系统头文件“stm32f10x.h”。接下来我们就要在“.c”文件中，敲写“LED_Init”函数体内的内容。

```
stm32f10x.h  main.c  led.c  led.h
1 #include "led.h"
2 #include "stm32f10x.h"
3
4 void LED_Init(void)
5 {
6
7     GPIO_InitTypeDef  GPIO_InitStructure;
8
9     RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);    //使能PB端口时钟
10
11     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;                //LED0-->PB.5  端口配置
12     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;         //推挽输出
13     GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;        //IO口速度为50MHz
14     GPIO_Init(GPIOB, &GPIO_InitStructure);                  //根据设定参数初始化GPIOB.5
15     GPIO_SetBits(GPIOB,GPIO_Pin_5);                          //PB.5 输出高
16
17 }
18
```

(P 2-4-67) led.c 中的代码

我们可以看到，在函数体内，我们先书写了一个名为“GPIO_InitStructure”的结构体“GPIO_InitTypeDef”。在这里教给大家一个小技巧，用鼠标放在想要知道来源的变量名或是函数体上，以“GPIO_InitTypeDef”为例，双击选中 GPIO_InitTypeDef，右键，在弹出的选项栏里我们可以看到这样两个选项，



(P 2-4-68) 弹出的选项

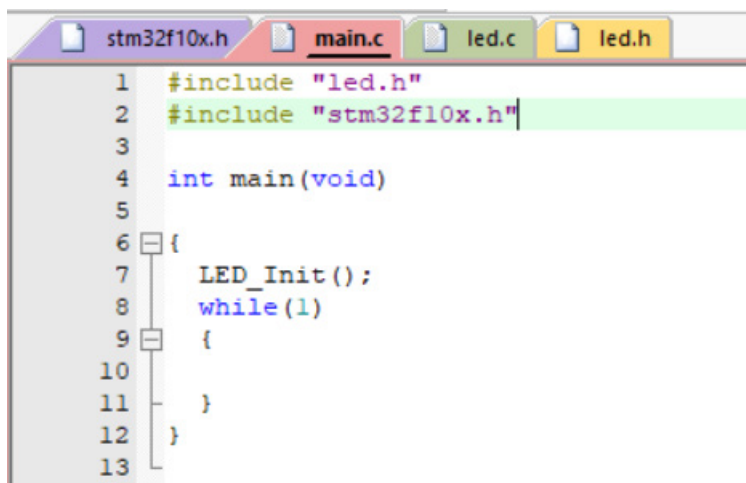
点击第一个选项我们可以查看该变量名的定义来自于哪，下一个选项可以看到该变量体映射到哪里也就是在哪里被使用。这一技巧就等待大家在实际操作中掌握了。再下一行，如同我们注释的一样，我们调用了“RCC_APB2PeriphClockCmd（）”函数来启动 GPIOB 的时钟使能，这样我们就可以使用 GPIOB 以下的所有端口。切记，如果不开启时钟使能的话，端口是无法正常使用的。再之后的五行，我们配置了要用的端口，在这里是 GPIOB 的第六号端口“GPIO_Pin_5”（在这里以此为例）；以及输入输出模式，在这里我们要输出电信号，所以我们选择通用推挽输出“GPIO_Mode_Out_PP”，其他的常用输出输入模式我们可以通过刚才的方法，双击“GPIO_Mode_Out_PP”，右键选择“Go to Definition”，便可以看到以下这些对应于不同模式的变量名，以及我们将要在接下来将要讲到的“寄存器版本”中，这些模式对应的寄存器数值。

```
GPIO_Mode_AIN = 0x0,  
GPIO_Mode_IN_FLOATING = 0x04,  
GPIO_Mode_IPD = 0x28,  
GPIO_Mode_IPU = 0x48,  
GPIO_Mode_Out_OD = 0x14,  
GPIO_Mode_Out_PP = 0x10,  
GPIO_Mode_AF_OD = 0x1C,  
GPIO_Mode_AF_PP = 0x18
```

(P 2-4-69) 变量

输出速度通常选择“GPIO_Speed_50MHz”即 50MHz 的输出速度。“GPIO_Init(GPIOB, &GPIO_InitStructure);”一行则是调用了“GPIO_Init（）”函数初始化了 GPIOB 内对应于“GPIO_InitStructure”结构体的变量。再下一行，我们调用了“GPIO_SetBits（）”函数，将 GPIOB_Pin_5 设置为输出高电平，即点亮 PB5 端口联结的小灯。另外说一句，如果想要输出低电平，则需要调用“GPIO_ResetBits（）”函数。

这样我们就完成了有关 led 的程序的编写。接下来要进入我们的主程序文件“main.c”，编写相关程序。其实是很简单的，与 led.c 相似，调用“led.h”与“stm32f10x.h”头文件，接下来写主函数“int main（void）”，在该函数内调用“LED_Init（）”，即运行程序时会先初始化我们所需的与 led 小灯有关的配置（下面的 while（1）死循环在这里用不到）。这样我们点亮一个小灯的程序就写好了。



```
1  #include "led.h"  
2  #include "stm32f10x.h"  
3  
4  int main(void)  
5  
6  {  
7      LED_Init();  
8      while(1)  
9      {  
10  
11      }  
12  }  
13
```

(P 2-4-70) 检验程序