

## 图文解说：如何将 STM32 的标准库编译成 lib 库

以前一直使用 STM32 的标准库，需要一步步地将代码加进去，将编译选项设置好，然后再编译整个工程。这个编译过程是一个相当慢的过程！完全编译大约需要一支烟的时间。每次建立工程都这么编译，是一个相当浪费时间和香烟的过程。

于是，我有了将库编译成 lib 文件的想法。本文就是我将 STM32F4 的标准库编译成 lib 文件并在工程中使用的过程。

适用对象：

- 1、熟悉库，不想再看库里边代码
- 2、有稳定的库，库文件更新不频繁
- 3、库文件多，每次编译时间长

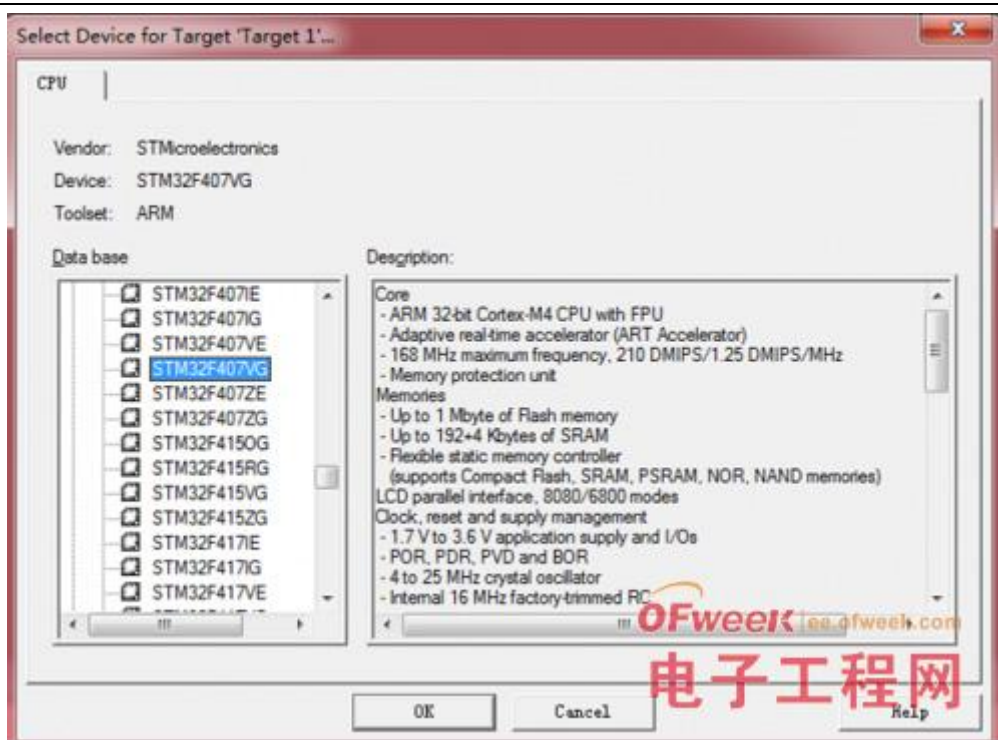
下面是我将 STM32F4 的标准库编译成 lib 并在工程中使用的过程：

### 1、建立创建 lib 的工程

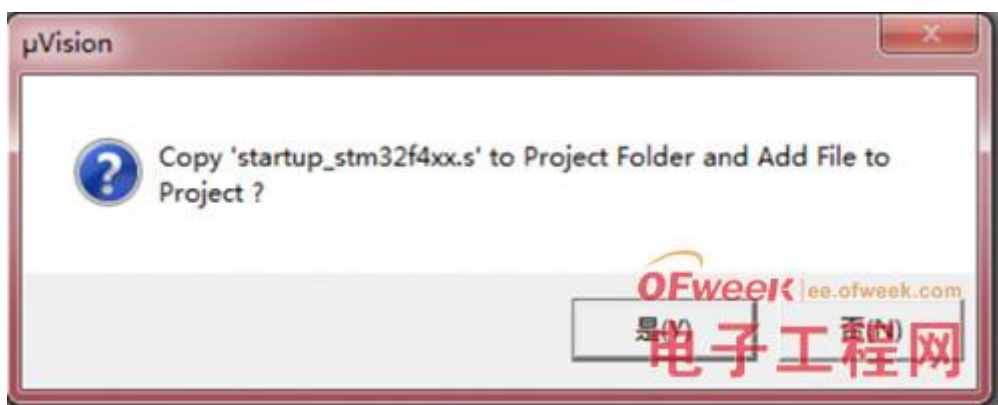
### 2、将库文件拷贝到工程目录：

将库里边目录\STM32F4xx\_StdPeriph\_lib  
v1.0.2\STM32F4xx\_StdPeriph\_Driver 下的 inc 和 src 两个文件夹拷贝到预创建工程的目录。我计划在目录 E:\学习\ARM\库\stm32f4-2 中创建库。于是我将两个文件夹拷贝到了这里。

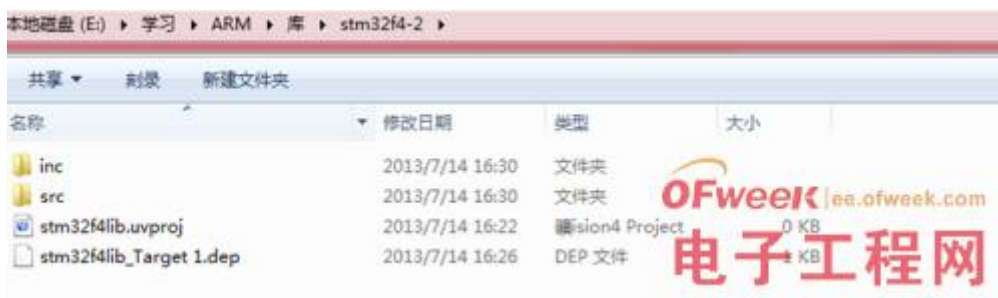
### 3、选择芯片：STM32F407VG



4、选择 NO。因为这不是可运行的程序，这里不需要加入启动文件。

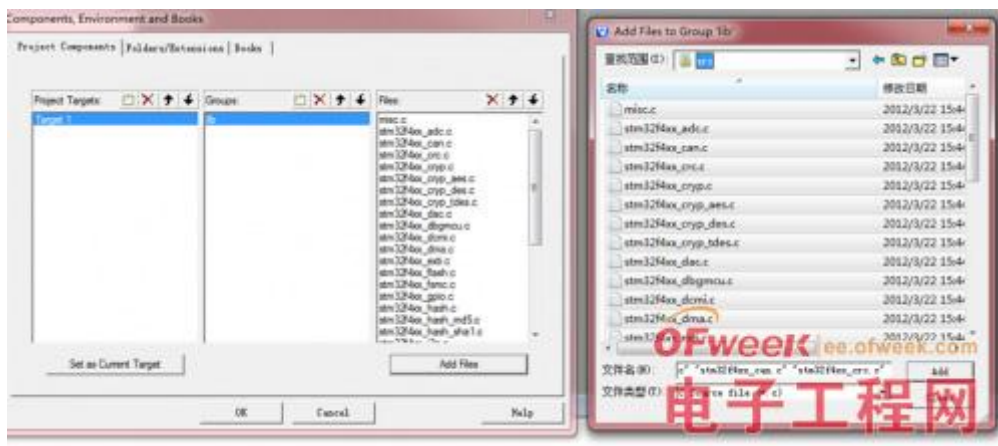


5、创建完工程后，工程结构如图：

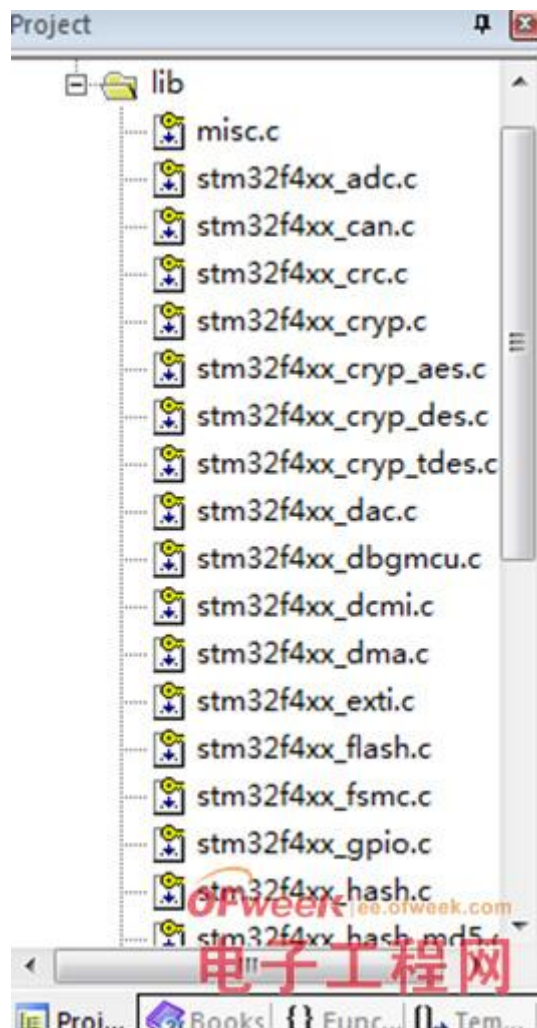


6、MDK 中点击工具栏上的  设置工程结构，并将库文件加入工程：

为了通用，我将库中所有的 C 文件都加入了工程



7、完成后 MDK 下的目录结构:



8、MDK 下设置输出选项:

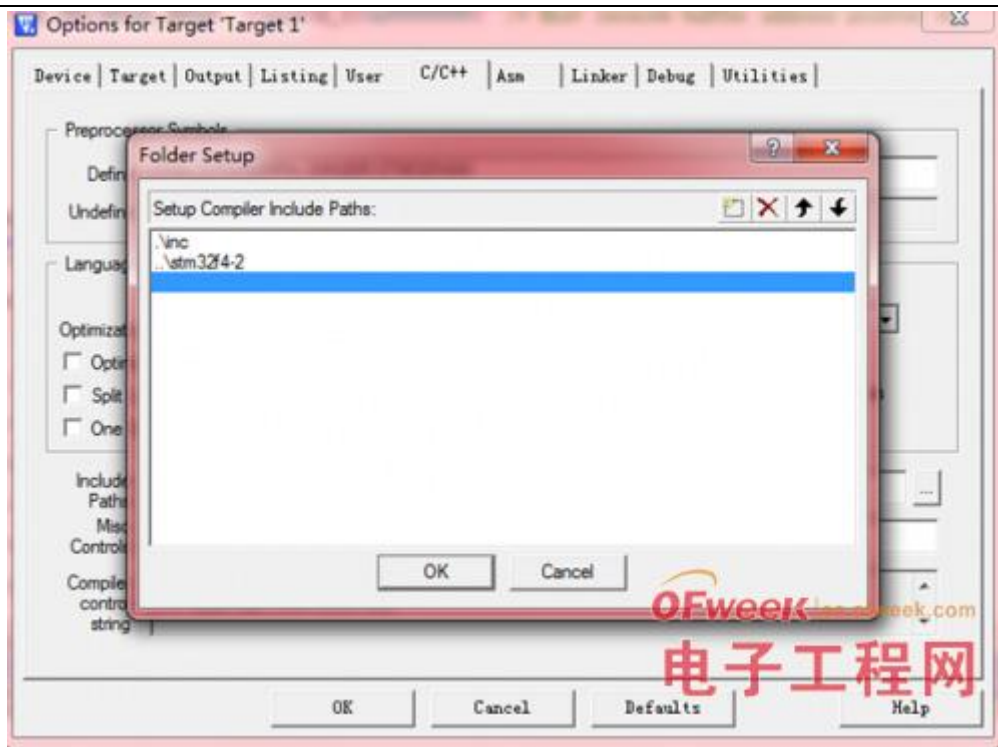
工程选项中设置输出，选择输出 lib 到目录 E:\学习\ARM\库\stm32f4-2\lib\:



## 9、设置 C 语言预编译宏和引用目录:

因为要使用 STM32F4 标准库，预编译选项设置：  
USE\_STDPERIPH\_DRIVER, STM32F4XX

将刚才拷贝的 inc 和工程根目录文件夹加入引用:



10、设置完成后：



11、将 stm32f4xx\_conf.h 文件拷贝到工程：

这个文件需要在标准库提供的示例工程中找：



我使用的是  
\\STM32F4-Discovery\_FW\_V1.1.0\\Project\\Peripheral\_Examples\\ADC3\_DMA\\stm32f4xx\_conf.h

这个文件引用了库文件中所有的头文件。因为不包含在库中，我将这个文件拷贝到 E:\\学习\\ARM\\库\\stm32f4-2 文件夹。

12、到此，工程设置完成。按 F7 编译，经过一支烟的时间即可生成库的 lib。

生成完成后，MDK 工程中：



```
Build Output
compiling stm32f4xx_adc.c...
compiling stm32f4xx_can.c...
compiling stm32f4xx_crc.c...
compiling stm32f4xx_cryp.c...
compiling stm32f4xx_cryp_aes.c...
compiling stm32f4xx_cryp_des.c...
compiling stm32f4xx_cryp_tdes.c...
compiling stm32f4xx_dac.c...
compiling stm32f4xx_dbgmcu.c...
compiling stm32f4xx_dcmi.c...
compiling stm32f4xx_dma.c...
compiling stm32f4xx_exti.c...
compiling stm32f4xx_flash.c...
compiling stm32f4xx_fsmc.c...
compiling stm32f4xx_gpio.c...
compiling stm32f4xx_hash.c...
compiling stm32f4xx_hash_md5.c...
compiling stm32f4xx_hash_shal.c...
compiling stm32f4xx_i2c.c...
compiling stm32f4xx_iwdg.c...
compiling stm32f4xx_pwr.c...
compiling stm32f4xx_rcc.c...
compiling stm32f4xx_rng.c...
compiling stm32f4xx_rtc.c...
compiling stm32f4xx_sdio.c...
compiling stm32f4xx_spi.c...
compiling stm32f4xx_syscfg.c...
compiling stm32f4xx_tim.c...
compiling stm32f4xx_usart.c...
compiling stm32f4xx_wwdg.c...
creating Library...
".\\lib\\stm32f4lib.lib" - 0 Error(s), 0 Warning(s).
```

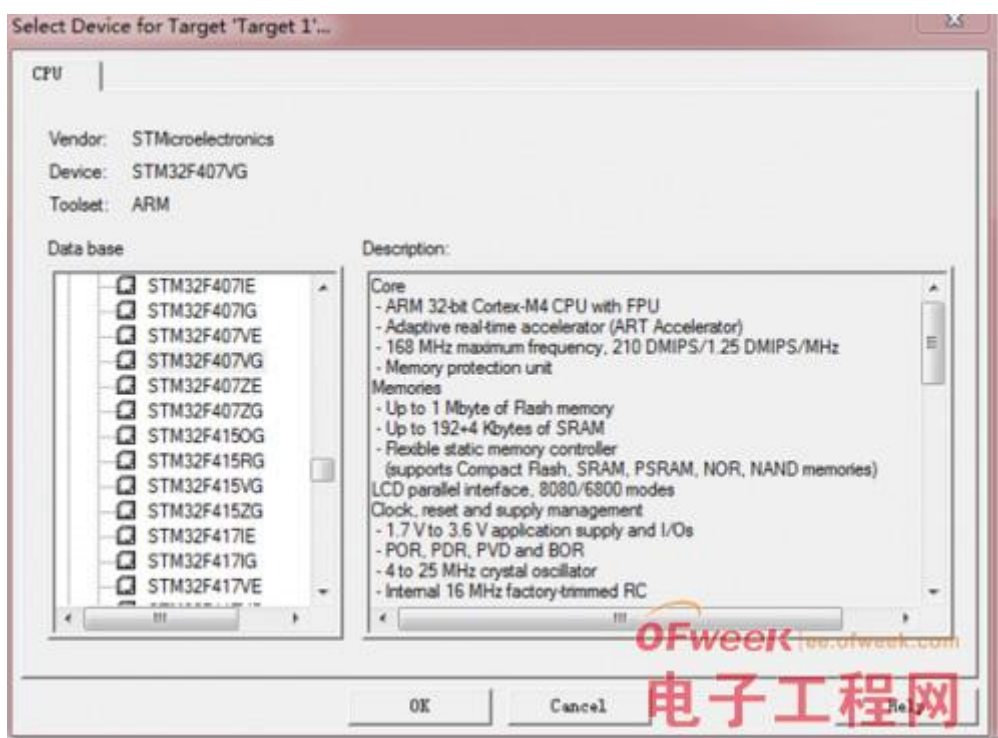
13、工程输出目录：

名称	修改日期	类型	大小
misc.crf	2013/7/14 16:51	CRF 文件	431 KB
misc.d	2013/7/14 16:51	D 文件	2 KB
misc.o	2013/7/14 16:51	O 文件	450 KB
stm32f4lib.lib	2013/7/14 16:53	Object File Library	14,131 KB
stm32f4xx_adc.crf	2013/7/14 16:51	CRF 文件	438 KB
stm32f4xx_adc.d	2013/7/14 16:51	D 文件	2 KB
stm32f4xx_adc.o	2013/7/14 16:51	O 文件	15 KB
stm32f4xx_can.crf	2013/7/14 16:51	CRF 文件	440 KB

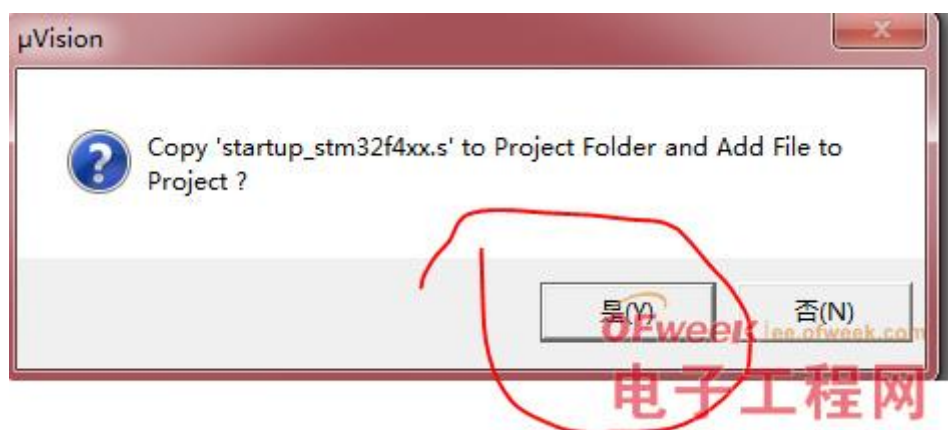
文件 stm32f4lib.lib 就是我们生成的 lib 文件

14、将库文件加入该工程


15、新建工程，我命名成 stm32f4use，处理器依旧选择 STM32F407VG。

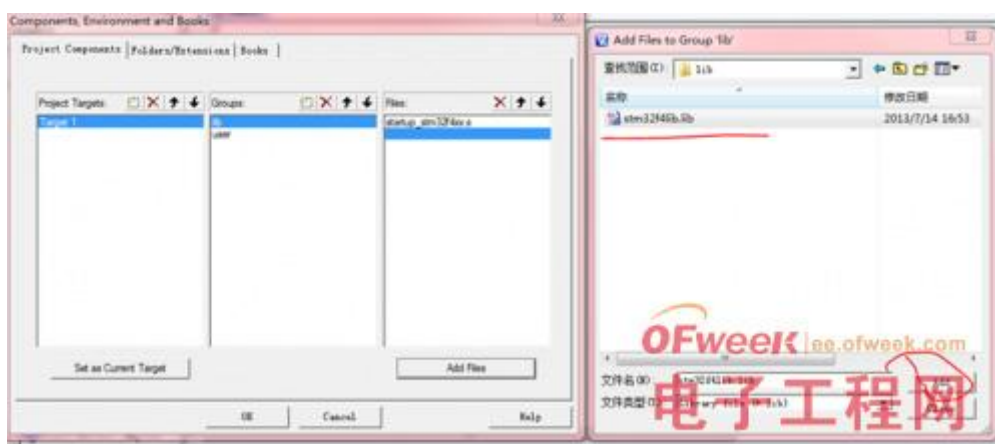


16、这是选择 yes，因为这是一个可执行的工程：



17、将标准库示例工程的  
\\STM32F4-Discovery\_FW\_V1.1.0\\Project\\Peripheral\_Examples\\ADC3\_DMA 中的  
system\_stm32f4xx.c 拷贝到工程目录 (E:\\学习\\ARM\\库\\stm32f4-2) 中

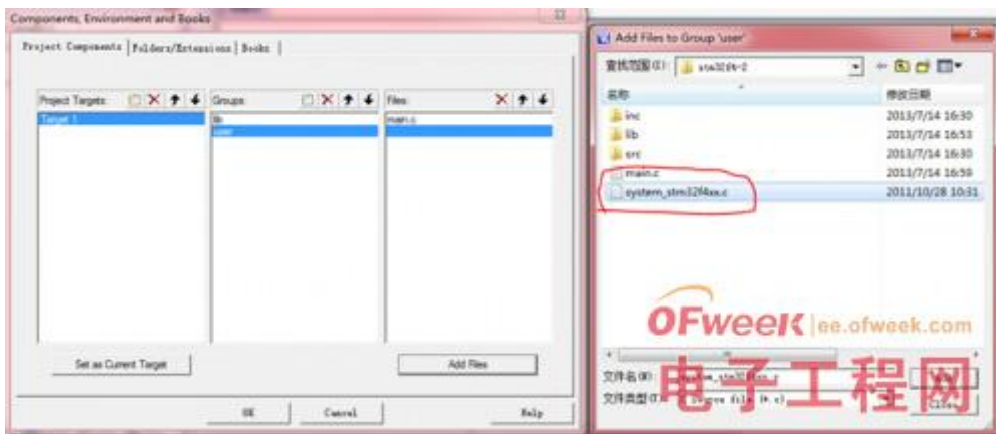
18、再次点击菜单上的  设置工程目录结构，将刚才生成的 lib 库加入到工程中：



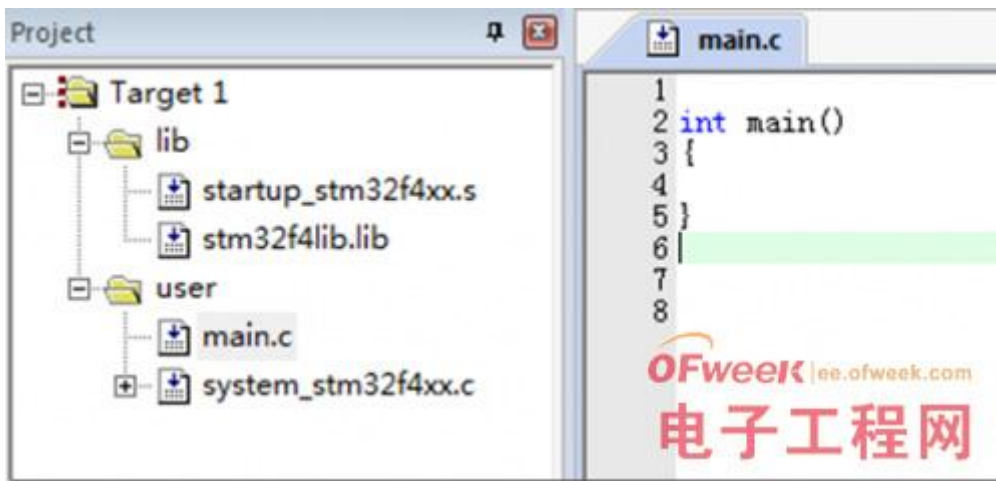
19、将示例工程  
\\STM32F4-Discovery\_FW\_V1.1.0\\Project\\Peripheral\_Examples\\ADC3\_DMA 中的  
system\_stm32f4xx.c 拷贝到工程目录。

20、将 main.c 和 system\_stm32f4xx.c 加入到工程





21、完成后的目录结构：

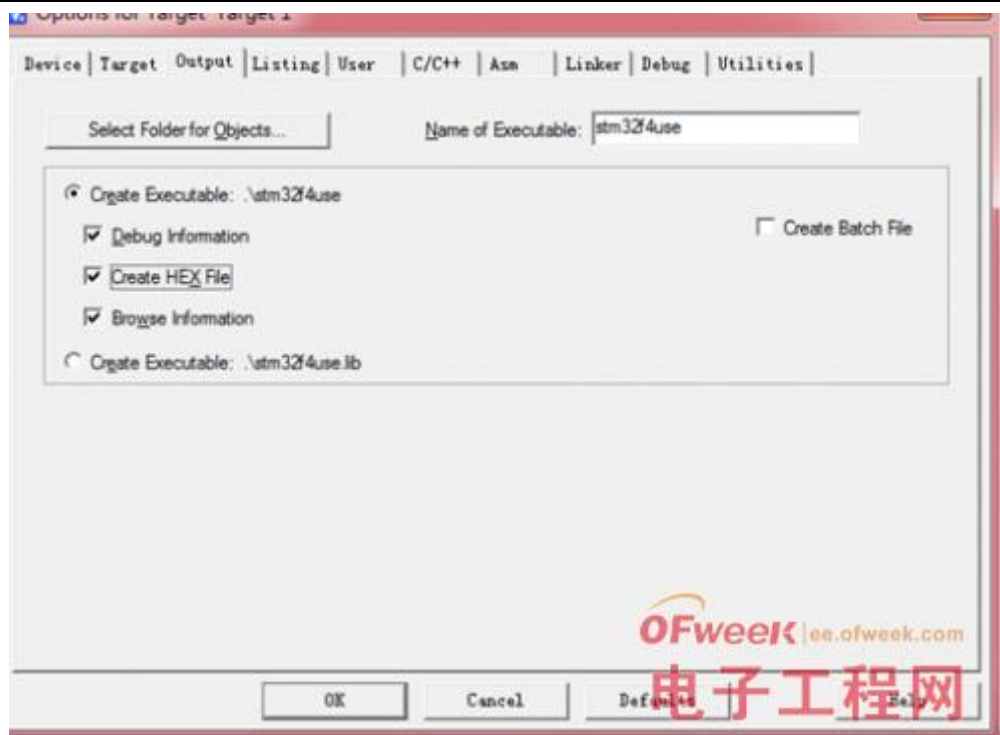


22、加入编译选项

与上边生成 lib 相似，预编译选项设置：USE\_STDPERIPH\_DRIVER, STM32F4XX

引用目录：.\inc;..\stm32f4-2 (这里的.\inc;..\ 文件夹就是刚才建立库时候的文件夹)

输出可执行文件：



### 23、添加几行简单的代码

//点亮一个 LED

```
#include <stm32f4xx.h>
```

```
#include "stm32f4xx_conf.h"
```

```
#include "stm32f4xx_tim.h"
```

```
int main()
```

```
{
```

```
GPIO_InitTypeDef GPIO_InitStructure;
```

```
RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD, ENABLE);
```

```
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
```

```
GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
```

```
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_12;
```

```
GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
```

```
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;
```

```
GPIO_Init(GPIOD, &GPIO_InitStructure);
```

```
GPIO_SetBits(GPIOD, GPIO_Pin_12);
```

```
while(1);
```

```
}
```

**24、工程设置完成**，按 F7 试试。现在编译速度快起来了，点个烟的时间就编译完了