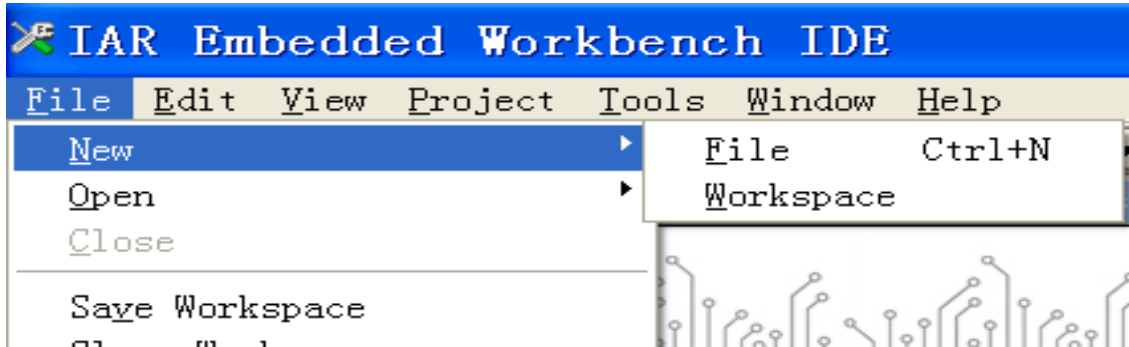


STM8教程

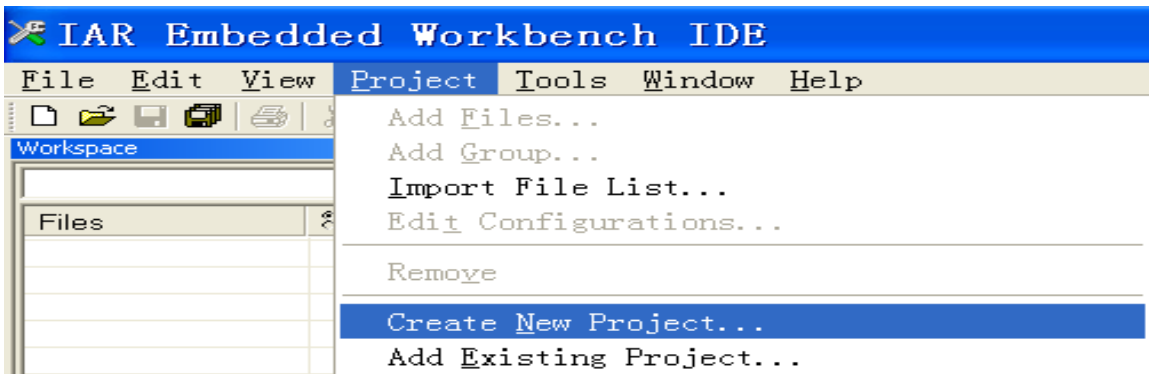
第一章：LED实验

作为入门的第一章，本章将如何新建工程跟简单的寄存器操作进行讲解
新建工程的方法如下：

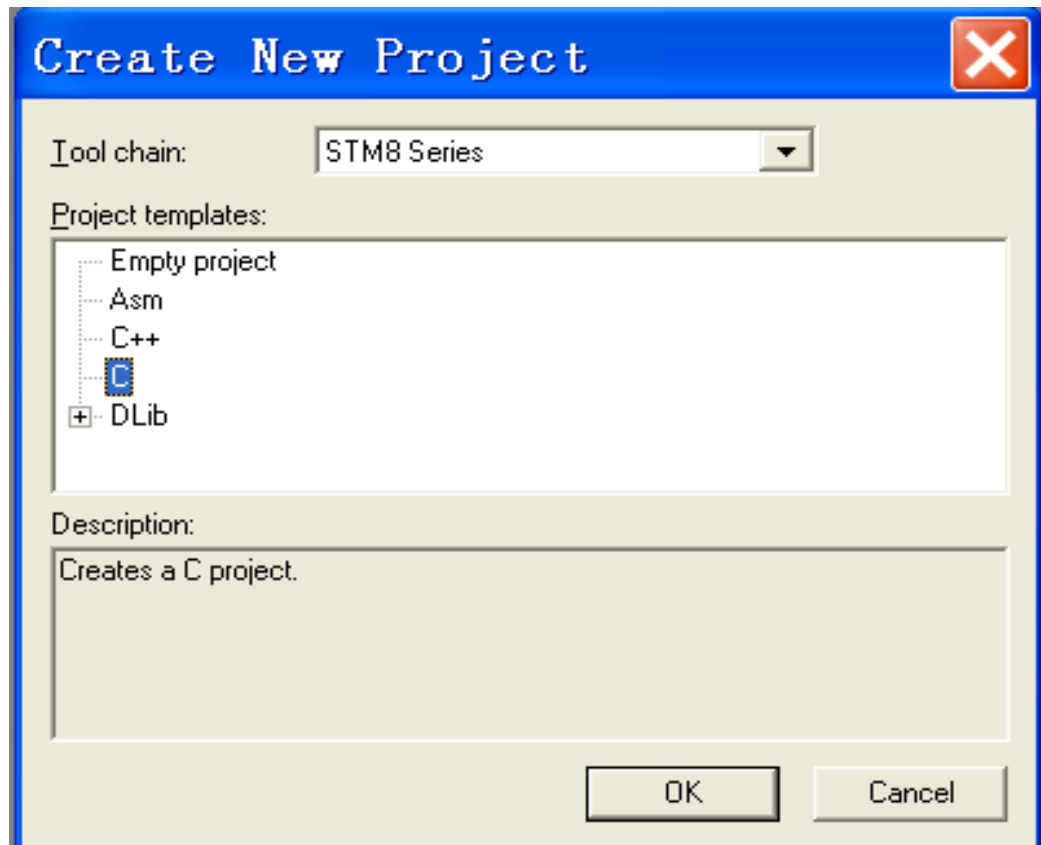
1、点击FILE，New，新建Workspace



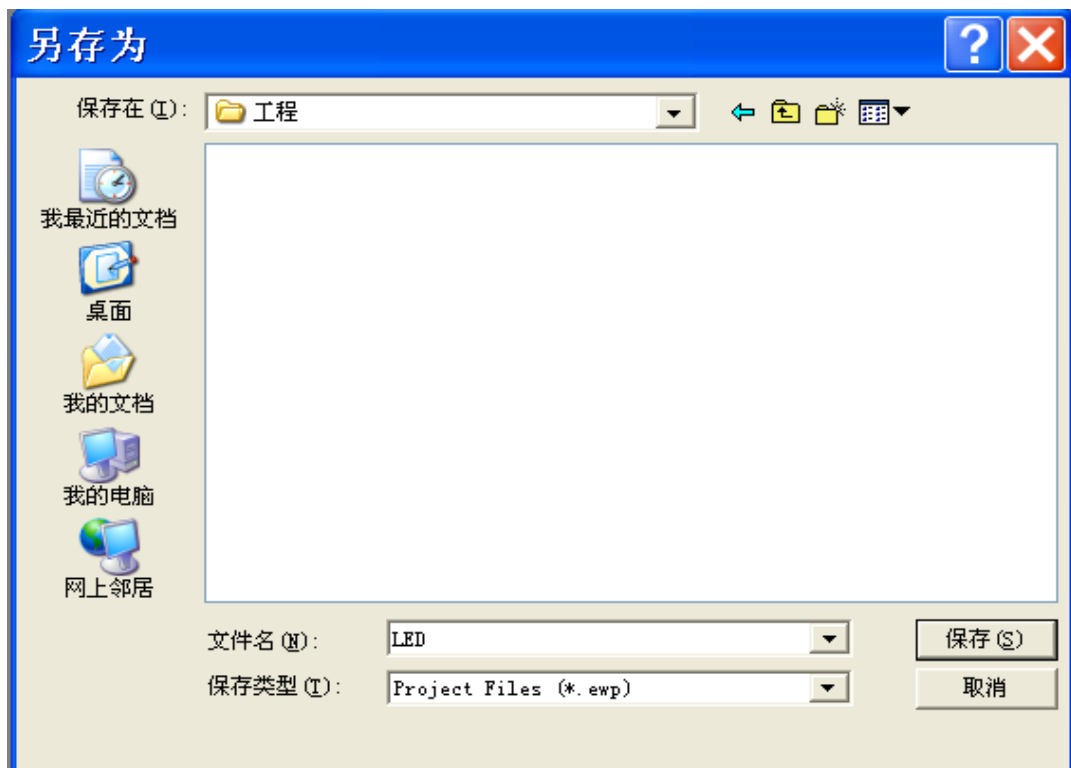
2、点击Project，Create New Project，创建新项目



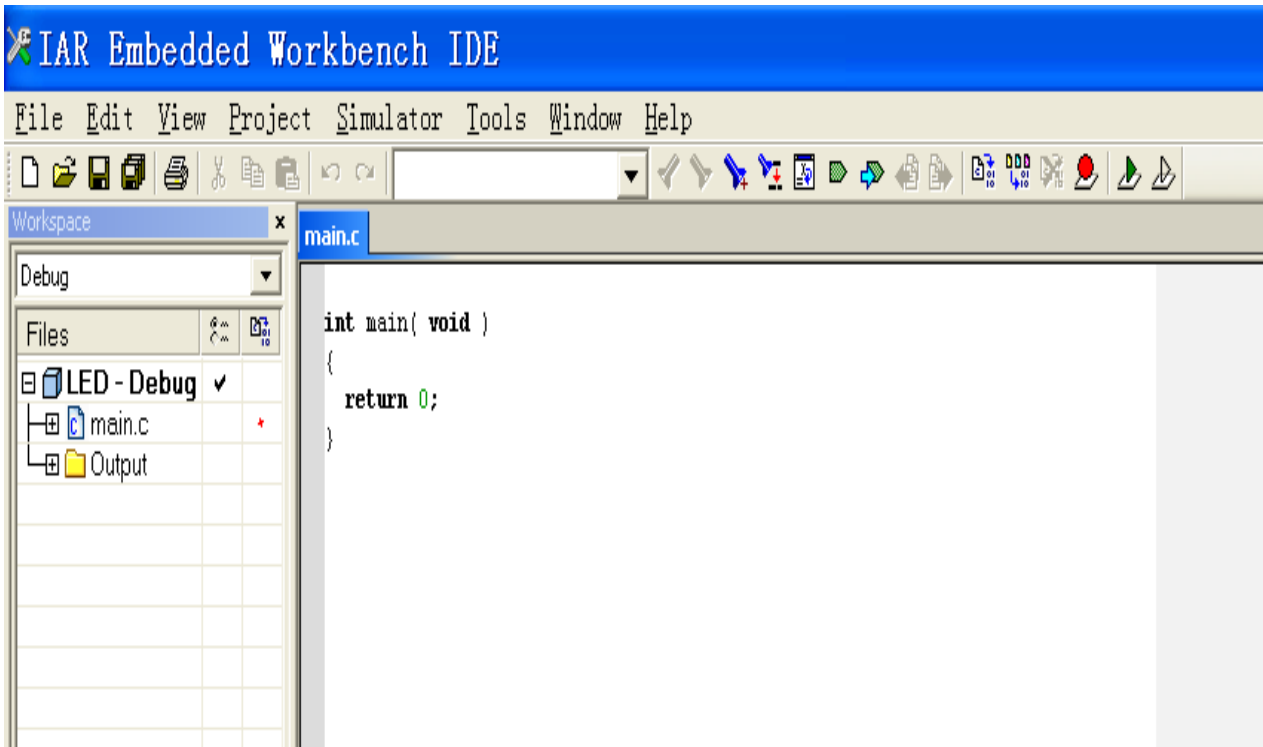
3、选择C，点确定



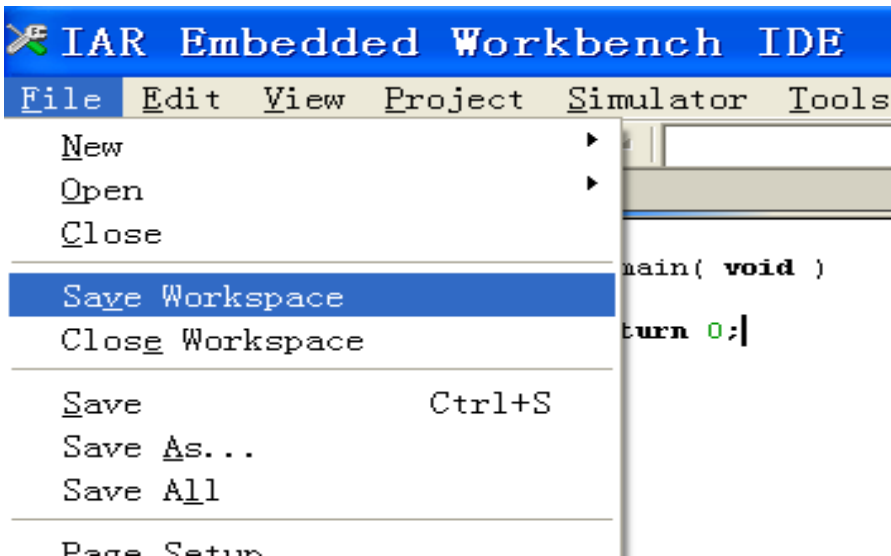
4、弹出另存为对话框，选择project保存的路径，并输入project的名字



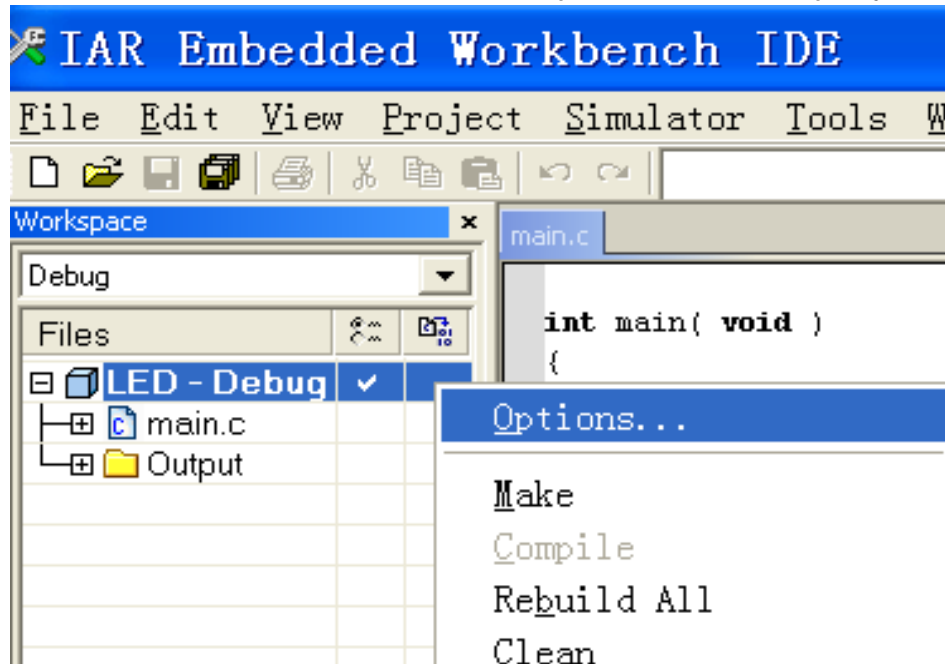
5、创建工程后的界面如下图所示



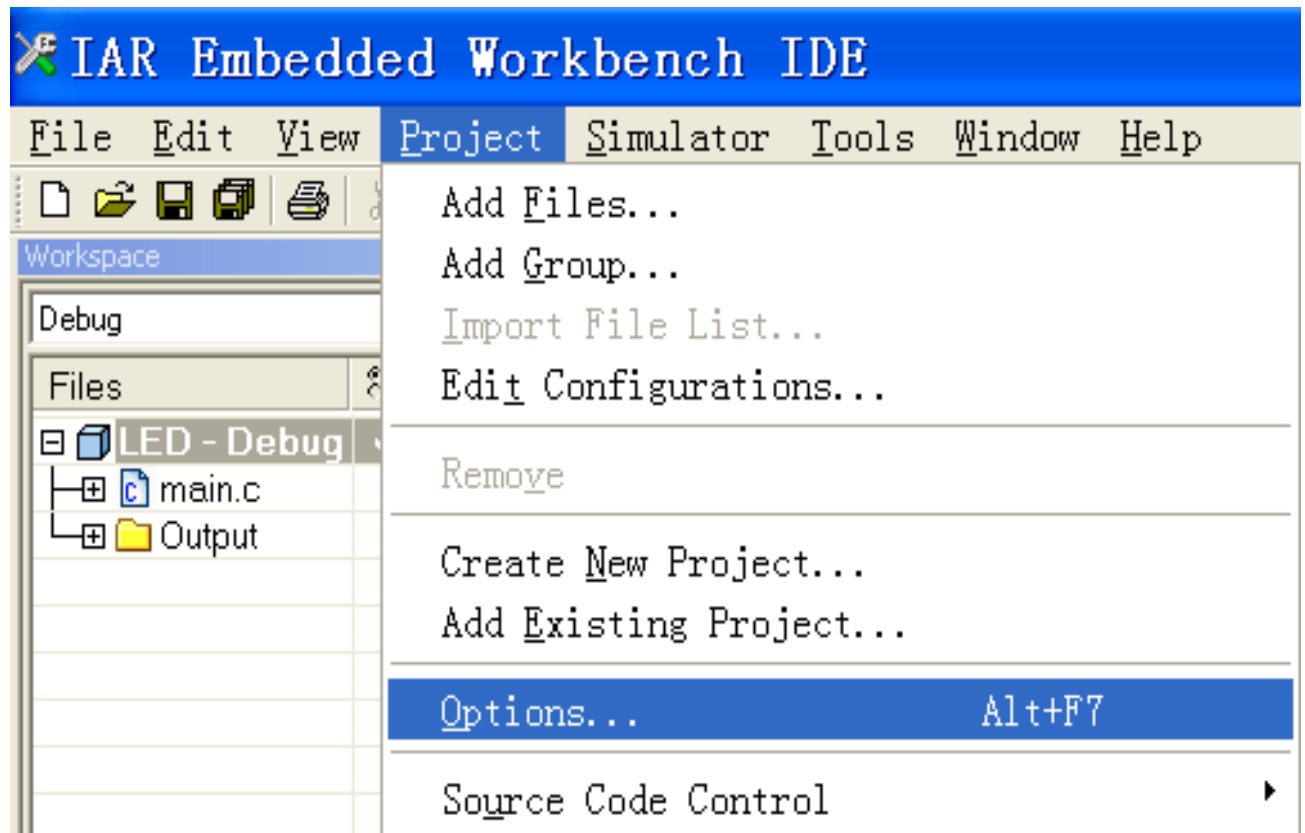
6、保存Workspace,指定要保存的路径，并输入Workspace的名字

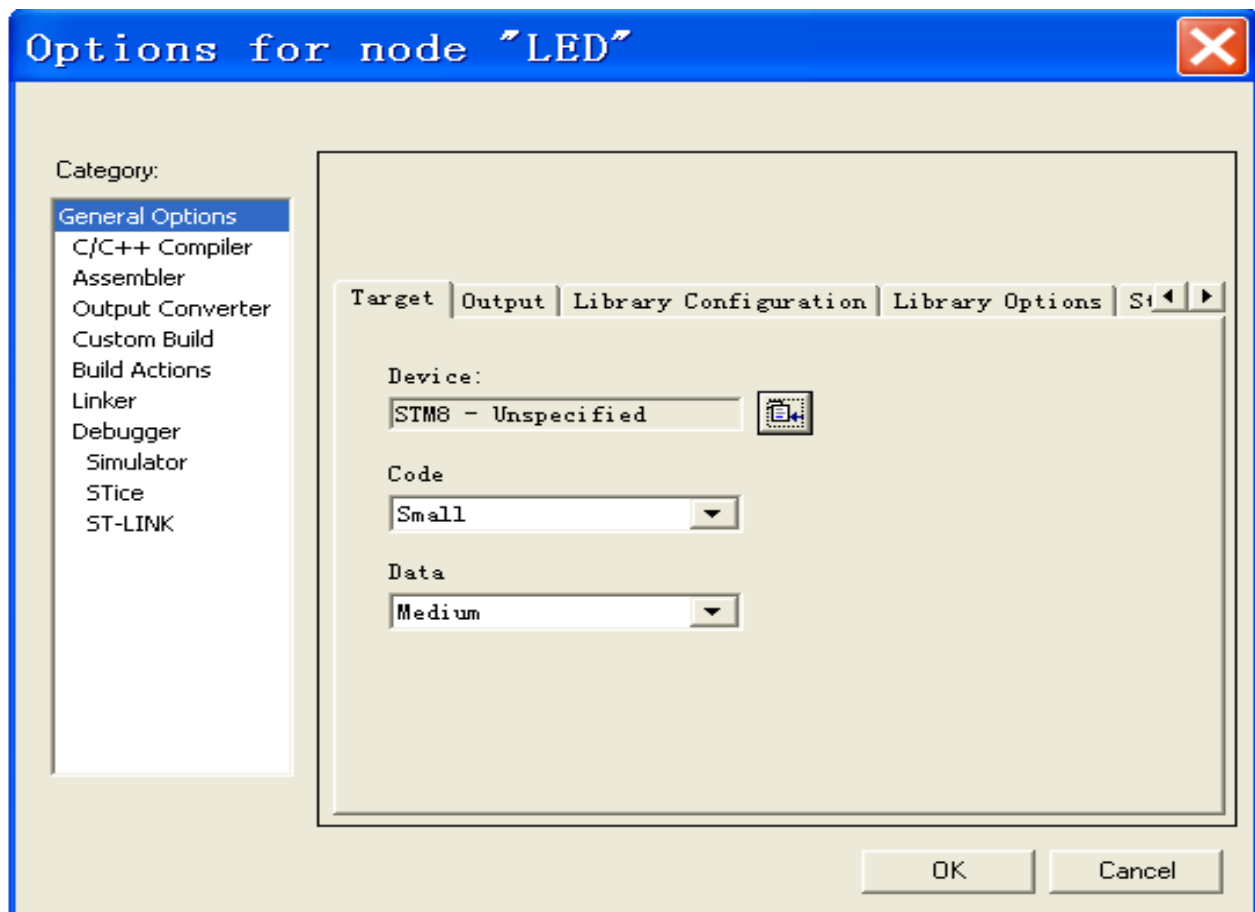


7、工程选项的有关配置：在Workspace窗口，选中project名，右键选择Options

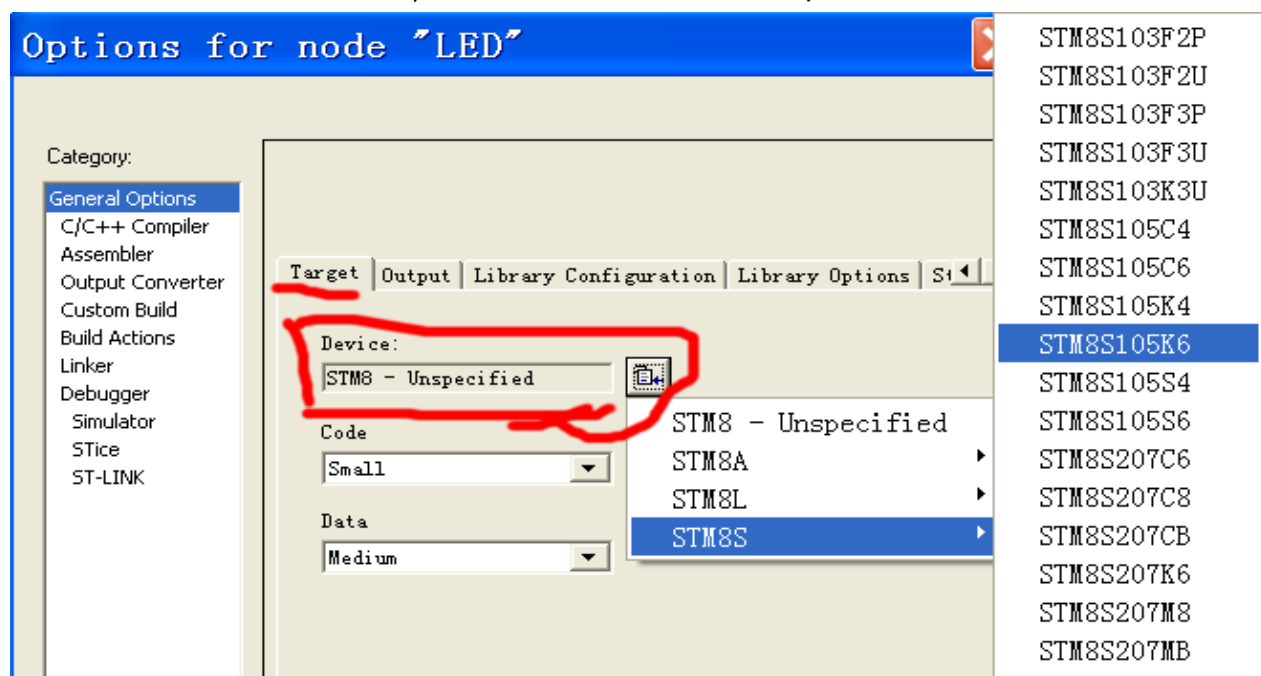


或者在工具菜单栏选择project->Options

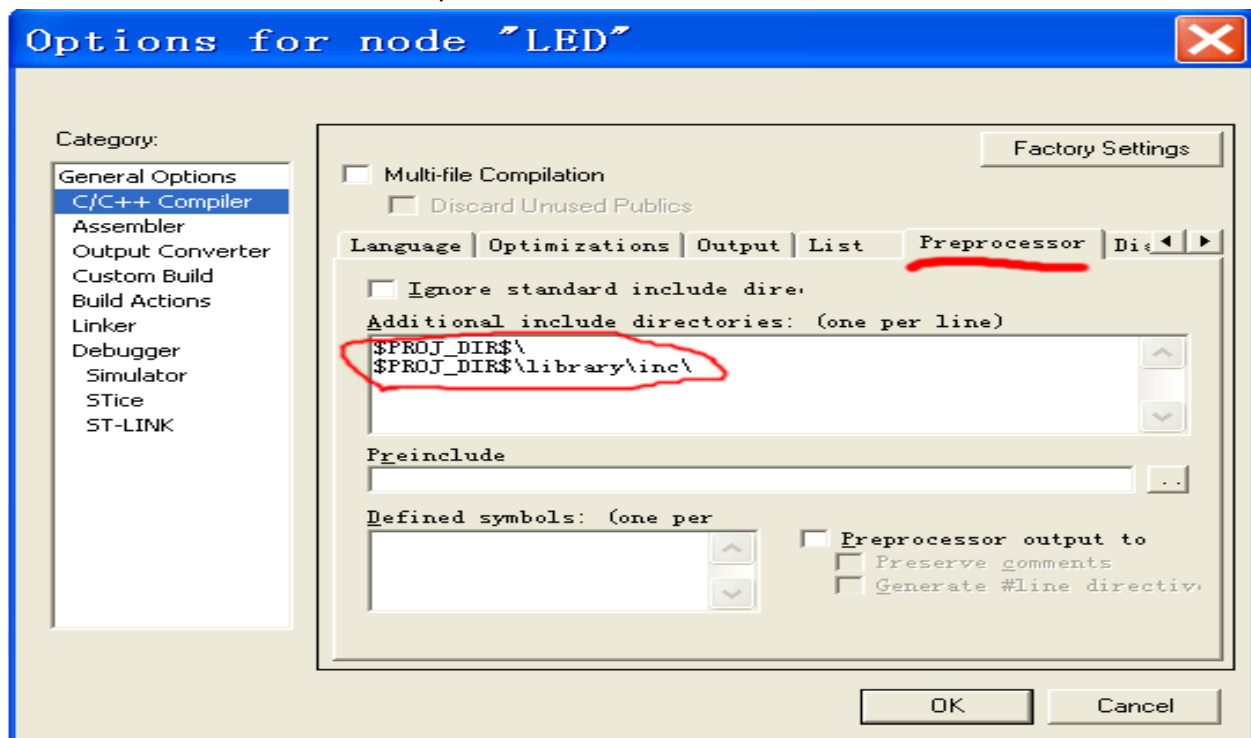




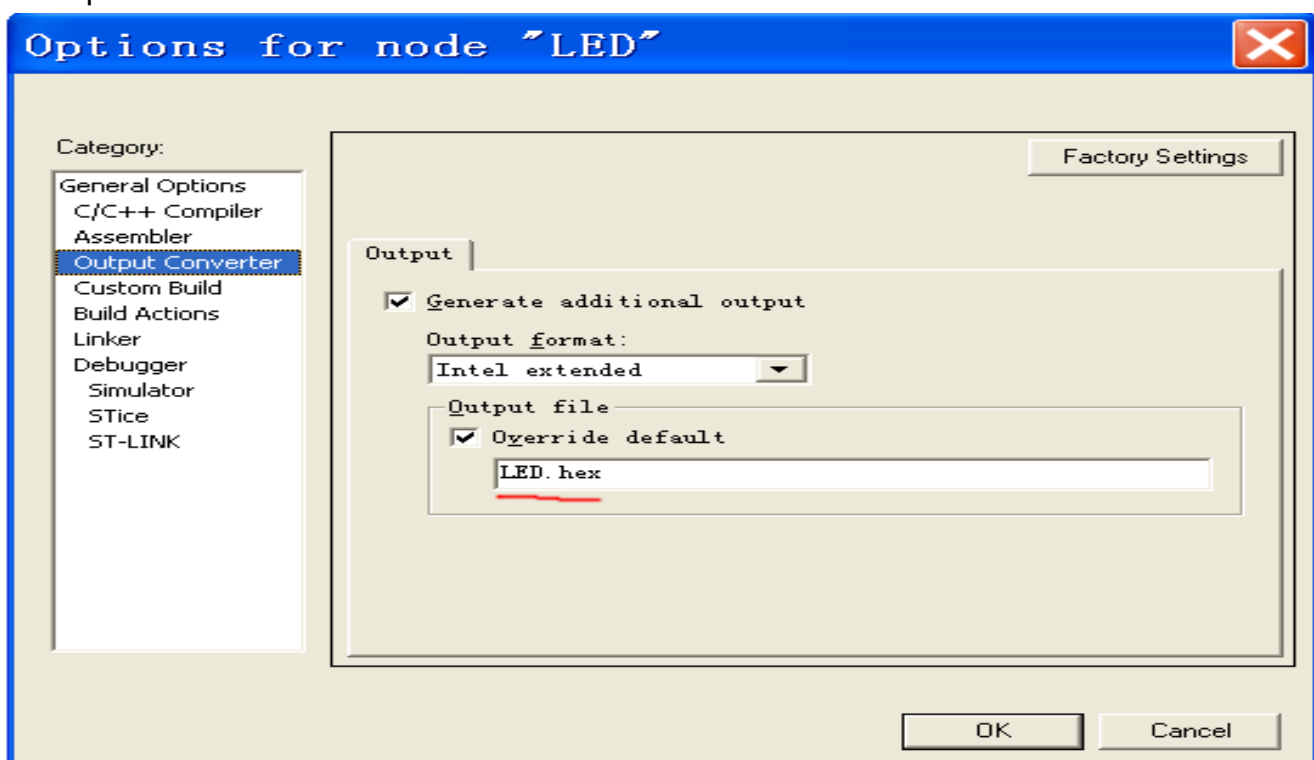
8、在Category中选择General Options,右边Target的Device选择设备型号,由于我们采用的是stm8s105k6t6,故选择相应的型号即可,其他的按默认设置



9、左边选择C/C++ Compiler,在C/C++ Compiler -> Preprocessor中的Additional栏中是设置*.h文件所在的位置,填入如下



10、左边选择Output Converter,在Output Converter -> Output中勾选Generate在Output file下的栏中输入生成hex的文件名



好了，工程建好了，接下来开始给力的时候咯，迫不及待吧，哈哈，别急纳
在这里讲几句唠叨的，在用某个单片机之前，先要看看数据手册对它的简单描述，
个人觉得这点很重要，在往后的学习中就会发现，并养成这个习惯的

唠叨部分：

查看stm8s105的数据手册，大概浏览时钟跟引脚的一些描述

4.5 时钟控制器

时钟控制器将来自不同振荡器的系统时钟(f_{MASTER})连接到内核和外设，它也为低功耗模式管理时钟的选通，并确保时钟的可靠性。

特点：

- 时钟分频：为了在速度和电流消耗之间找到一个最佳的平衡点，可以通过一个可编程的预分频器来调整CPU和外设的时钟频率。
- 安全的时钟切换：通过一个配置寄存器，可以在运行的时候安全地切换时钟源。新的时钟源准备好之前时钟信号不会被切换。这个设计能够保证无故障地切换时钟。
- 时钟管理：为了减少功耗，时钟控制器可以关闭内核、每个外设或存储器的时钟。
- 主时钟源：4个不同的时钟源可用来驱动主时钟
 - 1~16MHz高速外部晶振(HSE)
 - 最高至16MHz的高速外部时钟(HSE)
 - 16MHz高速内部RC振荡器(HSI)
 - 128kHz低速内部RC(LSI)
- 启动时钟：复位之后，单片机默认运行在内部2MHz时钟下(HSI/8)。一旦代码开始运行，应用程序就可以更改预分频比例和时钟源。
- 时钟安全系统(CSS)：这个功能可以用软件打开。一旦HSE时钟失效，CSS可以自动地将主时钟切换到内部RC(16MHz/8)，并且可以选择产生一个中断。
- 可配置的主时钟输出(CCO)：应用程序可以控制输出一个外部时钟。

如上大概知道这个芯片上电后的时钟默认为2MHz,并且可以配置不同的时候源，
大概先了解这些先，多了就难受了 😊

表4 符号和缩写说明

类型	I = 输入，O = 输出，S = 供电引脚	
电平	输入	CM = CMOS
	输出	HS = High sink 高吸收电流
输出速率	O1 = 慢速(最高到2MHz) O2 = 快速(最高到10MHz) O3 = 可配置成快速或慢速，复位后默认为慢速 O4 = 可配置成快速或慢速，复位后默认为快速	
端口和控制配置	输入	float = 浮置，wpu = 弱上拉
	输出	T = 真正的开漏结构，OD = 开漏结构，PP = 推挽

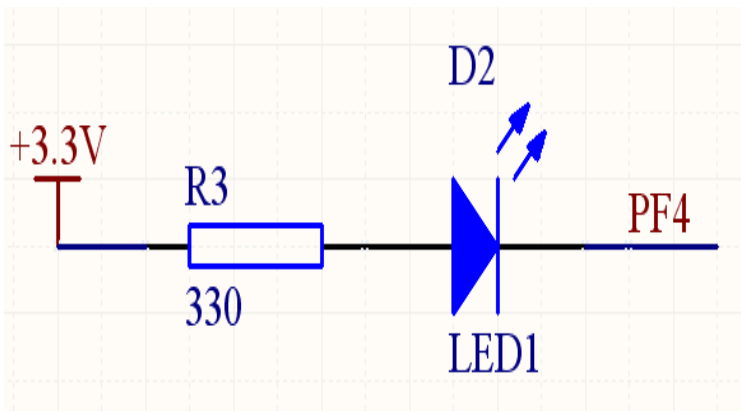
关于这个表，可以大概看出引脚作为输出时，可以配置不同的速率。输出输入的模式也有好几种。

哎呀，啥时候才实际操刀啊，好过过手隐啊 🤔

好吧。。。

大展拳脚：

板子的LED连接图：



实验内容：先点亮LED，然后再让它一闪一闪

11.9.3 端口 x 数据方向 (Px_DDR)

地址偏移值: 0x02

复位值: 0x00

7	6	5	4	3	2	1	0
DDR7	DDR6	DDR5	DDR4	DDR3	DDR2	DDR1	DDR0
rW	rW	rW	rW	rW	rW	rW	rW

位7:0	DDR[7:0]: 数据方向寄存器位 这些位可通过软件置1或置0, 选择引脚输入或输出 0: 输入模式 1: 输出模式
------	---

这是一个配置引脚的数据方向寄存器, 为输入或者输出。我们要控制LED灯亮, 故应配置为输出模式, 配置这个还是比较简单的。示例如下:

```
GPIOF->DDR |= BIT(4); //PF4 设置为输出模式
```

这里用了一个宏定义BIT, 在工程目录下有个头文件MyType.h。打开这个文件里面写了这个内容

```
#ifndef BIT
#define BIT(x)  (1 << (x))
#endif
```

这是一个宏定义, 把BIT(X)定义为(1<<(x)), 故在操作寄存器的某些时候带来方便。我个人比较喜欢BIT的写法跟(1<<x)的写法两者结合起来, 在某些时候, 如对速度有要求时, 直接对寄存器赋值, 如: GPIOF->DDR |= 0x10;

11.9.4 端口 x 控制寄存器 1 (Px_CR1)

地址偏移值: 0x03

复位值: 0x00

7	6	5	4	3	2	1	0
C17	C16	C15	C14	C13	C12	C11	C10
rW	rW	rW	rW	rW	rW	rW	rW

位7:0	C1[7:0]控制寄存器位 这些位可通过软件置1或置0, 用来在输入或输出模式下选择不同的功能。请参考表18 在输入模式时(DDR=0): 0: 浮空输入 1: 带上拉电阻输入 在输出模式时(DDR=1): 0: 模拟开漏输出(不是真正的开漏输出) 1: 推挽输出, 由CR2相应的位做输出摆率控制
------	---



这是一个具体配置输入/输出模式的寄存器，由于上面配置为输出，故只看输出模式的推挽输出跟开漏输出。有关于这些模式的详细介绍会在本章的末尾有详尽的细说。
这里配置为推挽输出 `GPIOF->CR1 |= BIT(4); //推挽输出`
要修改成某个引脚的配置模式时，只要修改GPIOF跟BIT中的第几个脚

11.9.5 端口 x 控制寄存器 2 (Px_CR2)

地址偏移值: 0x04

复位值: 0x00

7	6	5	4	3	2	1	0
C27	C26	C25	C24	C23	C22	C21	C20
RW	RW	RW	RW	RW	RW	RW	RW

位7:0	<p>C2[7:0]控制寄存器位</p> <p>相应的位通过软件置1或置0，用来在输入或输出模式下选择不同的功能。在输入模式下，由CR2相应的位使能中断。如果该引脚无中断功能，则对该引脚无影响。</p> <p>在输出模式下，置位将提高IO速度。此功能适用O3和O4输出类型。(参见引脚描述表)</p> <p><u>在输入模式时(DDR=0):</u>  </p> <p>0: 禁止外部中断</p> <p>1: 使能外部中断</p> <p>在输出模式时(DDR=1):</p> <p>0: 输出速度最大为2MHZ.</p> <p>1: 输出速度最大为10MHZ</p>
------	--

注意红色下划线，我们的LED用的是PF4引脚，查看引脚描述表可知，该引脚的输出类型为01，故无需配置此寄存器，至此，IO口初始化完了，接下来我们要灯亮，即输出低电平

11.9.1 端口 x 输出数据寄存器 (Px_ODR)

地址偏移值: 0x00
复位值: 0x00

7	6	5	4	3	2	1	0
ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
rW	rW	rW	rW	rW	rW	rW	rW

位7:0	<p>ODR[7:0]: 端口输出数据寄存器位</p> <p>在输出模式下，写入寄存器的数值通过锁存器加到相应的引脚上。读ODR寄存器，返回之前锁存的寄存器值。</p> <p>在输入模式下，写入ODR的值将被锁存到寄存器中，但不会改变引脚状态。ODR寄存器在复位后总是为0。位操作指令(BSET, BRST) 可以用来设置DR寄存器来驱动相应的引脚，但不会影响到其他引脚。</p>
------	---

这是IO端口输出数据的寄存器，比较简单。配置如下：

```
GPIOF->ODR &=~BIT(4); //PF4置0
```

整个实验的代码如下：

```
#include "stm8s.h"
#include "MyType.h"

void Delay(unsigned int ms);

int main( void )
{
    GPIOF->DDR |= BIT(4); //PF4 设置为输出模式
    GPIOF->CR1 |= BIT(4); //推挽输出

    GPIOF->ODR &=~BIT(4); //PF4置0
}
```

将代码下载进去然后观察实验：

第一个小实验就完成了，接下来让LED一闪一闪：

首先看看有关时钟方面的，stm8s105k6t6复位后默认时钟为内部的RC,2MHZ。内部RC最大可配置为16MHZ,为了以后方便，我们将时钟配置为内部RC,16MHZ.