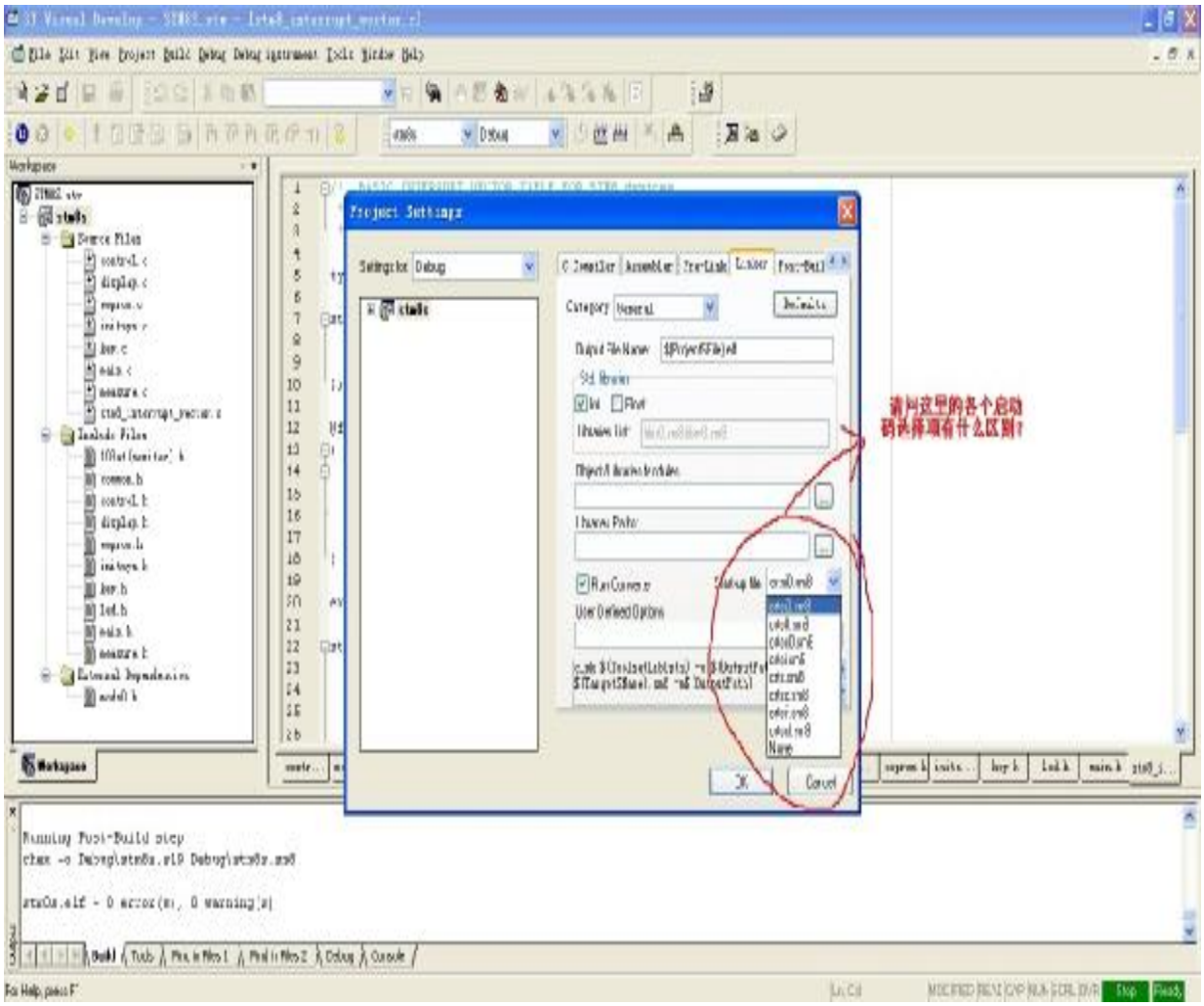
1、STM8 编译器支持两种存储器模式。函数指针和数据指针默认是@near指针（2个字节）
- stack short (mods0) 全局变量默认short range类型。 任何在long range范围的全局变量必须明确地用@near 来访问，除非通过指针访问。
- Stack Long (mods10) 全局变量默认为long range类型。任何在short range类型中的变量必须明确地用@tiny 来访问。
示例：

```
//-------------------------------------------------------------------------------
@near    uchar          TimeBaseCount=0;
uint     @near          ErrOpenTime;
//-------------------------------------------------------------------------------
void                CoolControl(void)
{
TimeBaseCount++;
if(TimeBaseCount){ErrOpenTime++;}
}
//-------------------------------------------------------------------------------
```

2、编译源代码时如何生成 HEX 格式？而不是 S19 格式!?
可以生成 HEX 格式的文件，做如下设置：setting->post_build，命令框内改成：chex -fi -o $(OutputPath)$(TargetSName).hex $(OutputPath)$(TargetSName).sm8，这样就能生产 hex 文件了。在项目目录中的 DEBUG 文件夹下就可以找到生成的 HEX 格式文件。
3、



Startup 文件，主要是将 C 环境下的一些重要的函数进行模块化。主要包含了以下几个方面：

BSS SECTION 初始化

ROM 到 RAM 复制（如果需要的话）

堆栈指针初始化

主函数指针或者其他程序进去点

外部中断的顺序设置（从 C 环境返回后）。很多用户为了特定的执行环境，要修改外部中断顺序

Crtsi.S 里没有对任何数据的初始化。

其余的都对分布在 RAM 中的静态变量进行了初始化，之间的区别在于变量初始化的范围和初始化表的范围（末尾带0的是程序小于64K 的启动文件）。

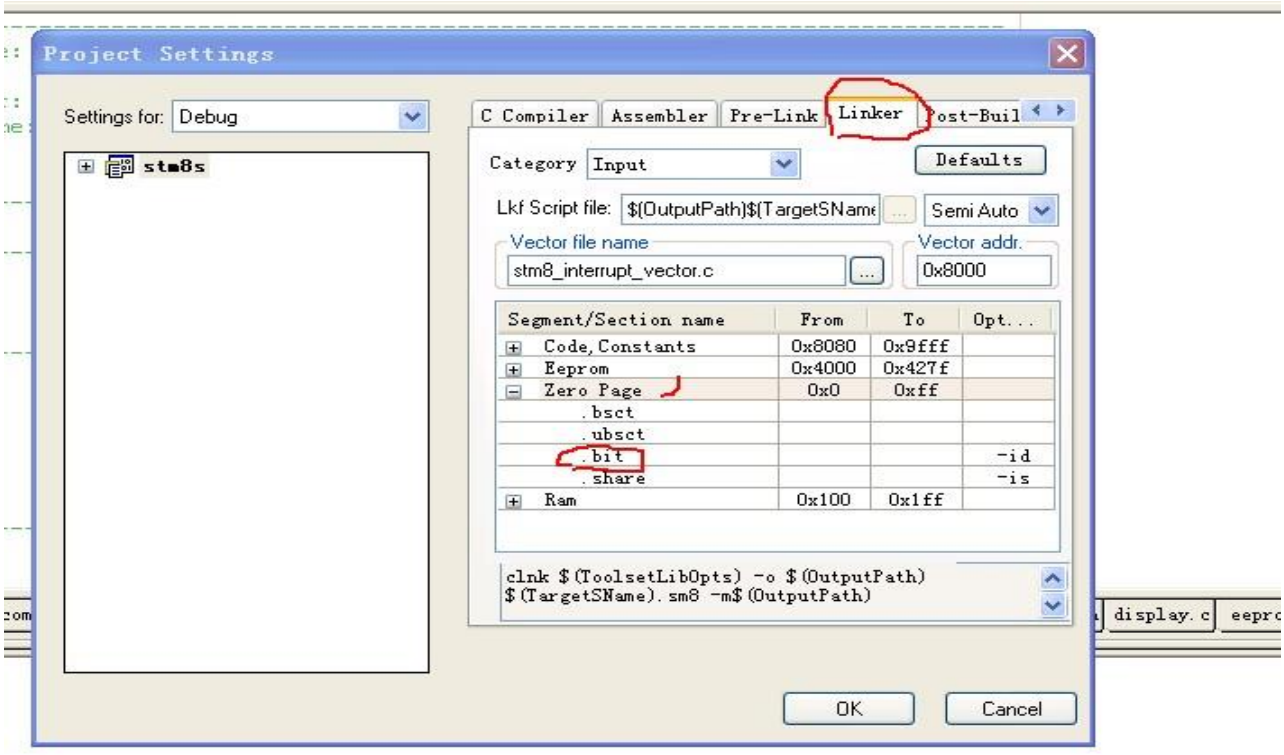
在 COSMIC 的 C Cross Compiler Uer`s Guide 文档中，有详细说明

Startup	Initialize	From Table in
crtsi(0).s	@near	@near
crtsx(0).s	@near and @far	@near
crtsif(0).s	@near	@far
crtsxf(0).s	@near and @far	@far

4、位变量:

4.1、定义示例：_Bool bitFlag;

4.2、位变量地址：默认的链接文件中，自动被定位到 RAM 的 Zero page，最大可定义256*8个位变量



5、STVD+COSMIC 下，芯片头文件必须手工包含所使用型号的头文件，源文件默认安装目录：C:\Program Files\STMicroelectronics\st_toolset\include,将对应 H 文件拷贝到当前项目路径中，并在 main.c 中通过 #include 语句将该 H 文件包含进来，但需要注意，很多时候，当前的这个 H 文件可能内部有第二层的头文件包含，

此时，还需要将该第二层的 H 文件拷贝到当前项目路径下。

6、STVD+COSMIC 下建立项目后，会自动生成 main.c 和 stm8_interrupt_vector.c 这两个文件，可以删除掉替换为自己的源代码文件，我的习惯时在当前项目目录中新建一个空文件夹，一般命名为 source，我会把所有的 C 文件和 H 文件都放到这个目录中，但超级郁闷的是，其它所有的文件都可以放到 source 文件夹下，但 stm8_interrupt_vector.c 却不可以，似乎 STVD 强制把它的路劲指向了项目第一层文件夹，当在第一层找不到该文件时，则编译不能通过，即使你通过 remove from workspace 和 add file to projest 重新包含也无济于事，STVD 会自动无视你的这些操作，并自动把路径强制指向当前项目文件夹第一层。