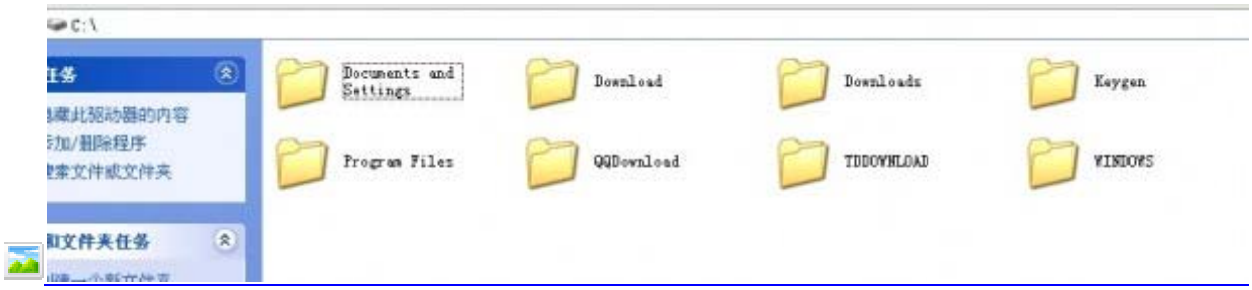
原创--IAR for AVR 入门学习笔记
AVR 单片机的编译软件五花八门，用宋丹丹的话就是：那是相当的多
汇编语言的开发平台就不说了（俺不大会，呵呵，说不出什么道道来）。
简单列举几个高级语言的开发平台：
WINAVR（GNU GCC AVR）；
ICC AVR
CodeVison AVR
IAR for AVR
BASIC AVR
FastAVR
BASCOM
其中用得最多的是完全免费的 WINAVR。我一直都是用的这个。
最专业，最好的，对 AVR 支持最全面的是 IAR。
但同时 IAR 也是最贵的一款开发软件（听说升级也要收费，真黑啊）。
呵呵，不过不怕，我们可以破解之。
下面就详细介绍如何破解 IAR。
这里安装破解的是 5.11B 版本的。全名 IAREmbeddedWorkbenchforAtmelAVR,v.5.11B
1、先到网上下载 5.11B 文件，大概 133M。解压后，有如下文件：

a1.jpg (29.87 KB)
2009-12-20 00:28

2、到网上下载破解文件 解压后生成文件 Keygen 包，将它复制到 C 盘根目录下，如下图

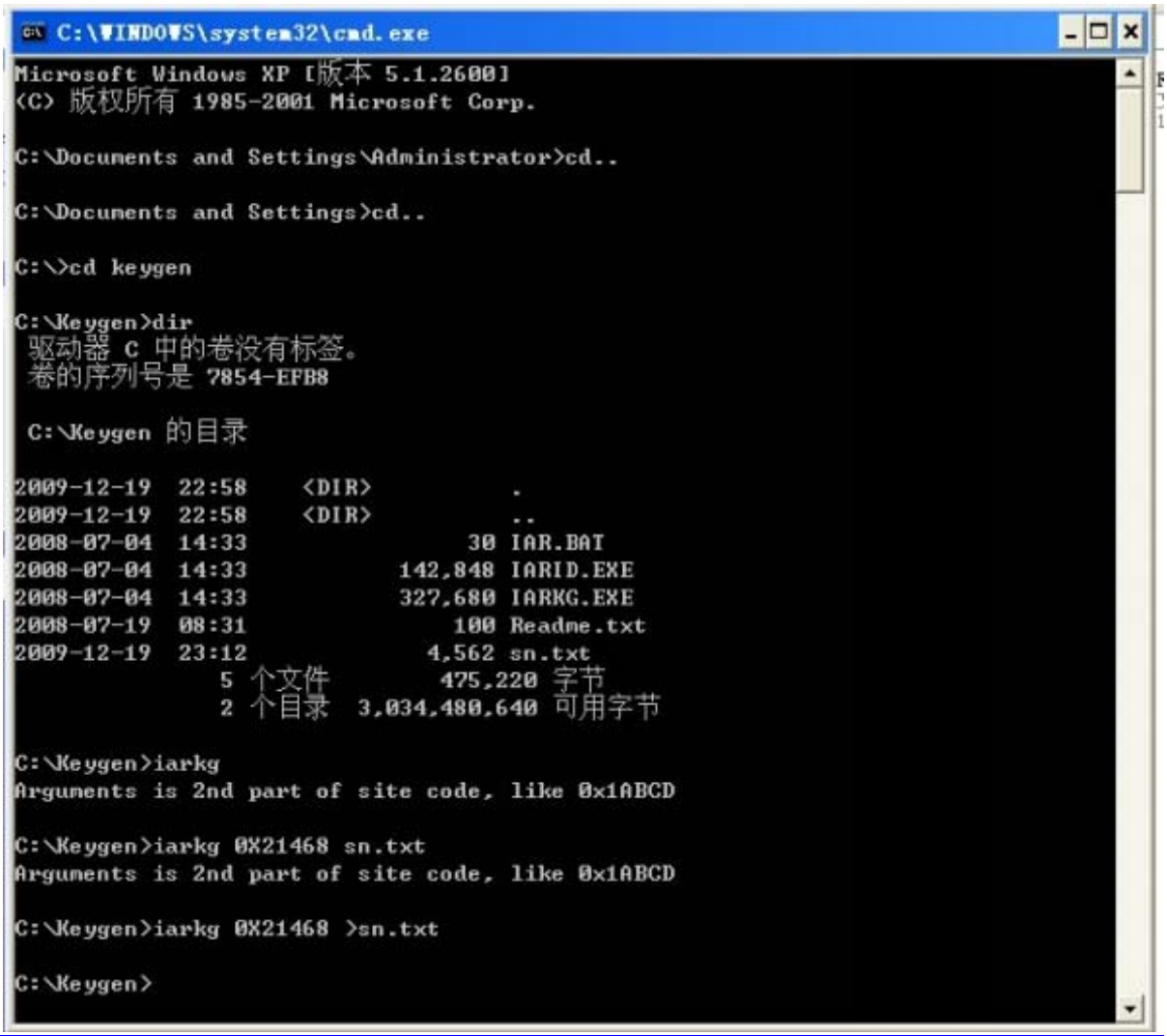


a2.jpg (63.75 KB)
2009-12-20 00:28

3、打开文件包 keygen,双击文件 IARID.exe，出现本电脑 ID 号，如图，记下来

a3.jpg (42.68 KB)
2009-12-20 00:28

4、从电脑的“程序---运行”输入“CMD”回车，按照下图操作，得到 sn.txt 文件;
注意输入计算机 ID 号的时候，所有字符全部大写，包括"0X"中的"X"也要大写，



a4.jpg (161.06 KB)
2009-12-20 00:28

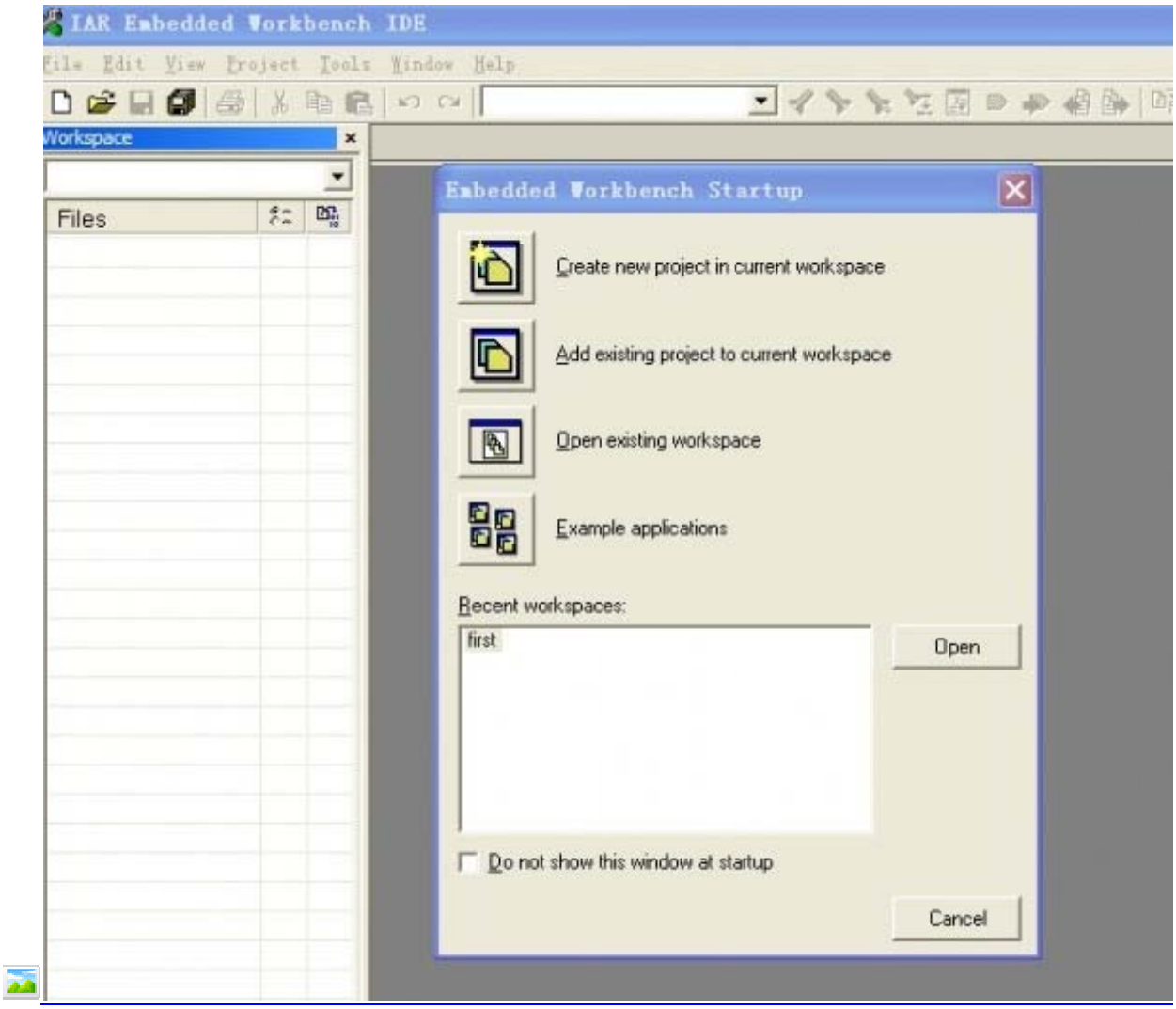
5、然后运行安装文件中的 autorun.exe,开始安装 IAR,当出现要求输入注册号的时候，请用记事本打开刚刚生成的 sn.txt 文件，找到 "EWAVR" version "2.25_WIN"对应的号码段，先输入序列号，NEXT 后，再复制 key:后面的一串字符，注意只复制#之前的那一部分，包括#也要复制。再点击 NEXT，如果出现提示错误，请重复步骤 4，重新生成 sn.txt 文件，再用新号填入试试，直到成功。很容易成功的！
6、下面就跟常规软件的安装方法一样了，静静等待安装完成！

要特别注意：1、获得的电脑 ID 号一定要大写！包括 16 进制符号：0X.....;
2、生成的 sn.txt 文件中，不是每个系列都能用，每次生的文件中，只有一个使能用于 AVR 的，就是那个"EWAVR" version "2.25_WIN"对应的号码段，其他就不要试了！

今天就写这么多吧，明天写一下怎么写一个入门程序！
IAR for AVR 编程的第一个程序
1、安装完成 IAR for AVR5.11B 后，就可以开始第一个入门程序的学习了。IAR 软件安装完成后，不会自动在桌面上建立一个快捷启动图标。我们可以按照以下顺序打开改软件“开始-程序-IAR System-IAR Embedded Workbench for Atmel AVR V5-IAR Embedded Workbench”。当然，在“IAR Embedded Workbench”图标上单击右键建立一个桌面快捷方式是很有必要的。

2、按照上述步骤，打开 IAR for AVR 的 IDE 开发环境，出现如下图的启动界面，在启动界面里，我们可以选择如下操作：在当前工作区新建一个工程项目；

在当前工作区添加一个已经存在的工程项目；
打开一个工作区；
查看应用实例。

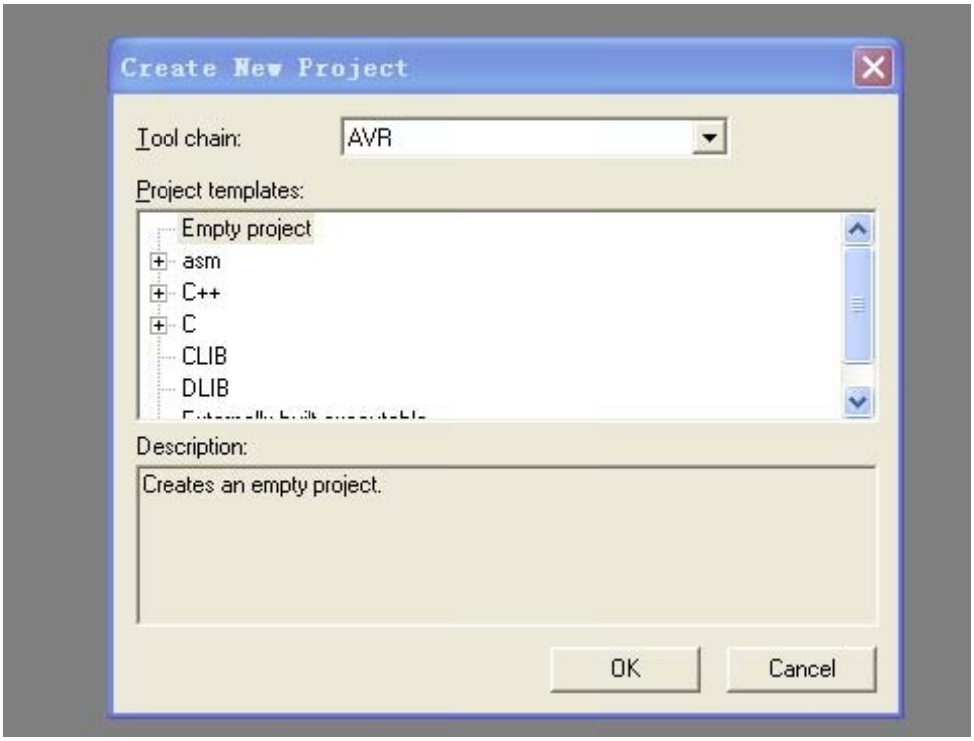


a5.jpg (119.84 KB)
2009-12-20 22:42

3、，在当前工作区新建一个项目文件，点击如下图的图标，

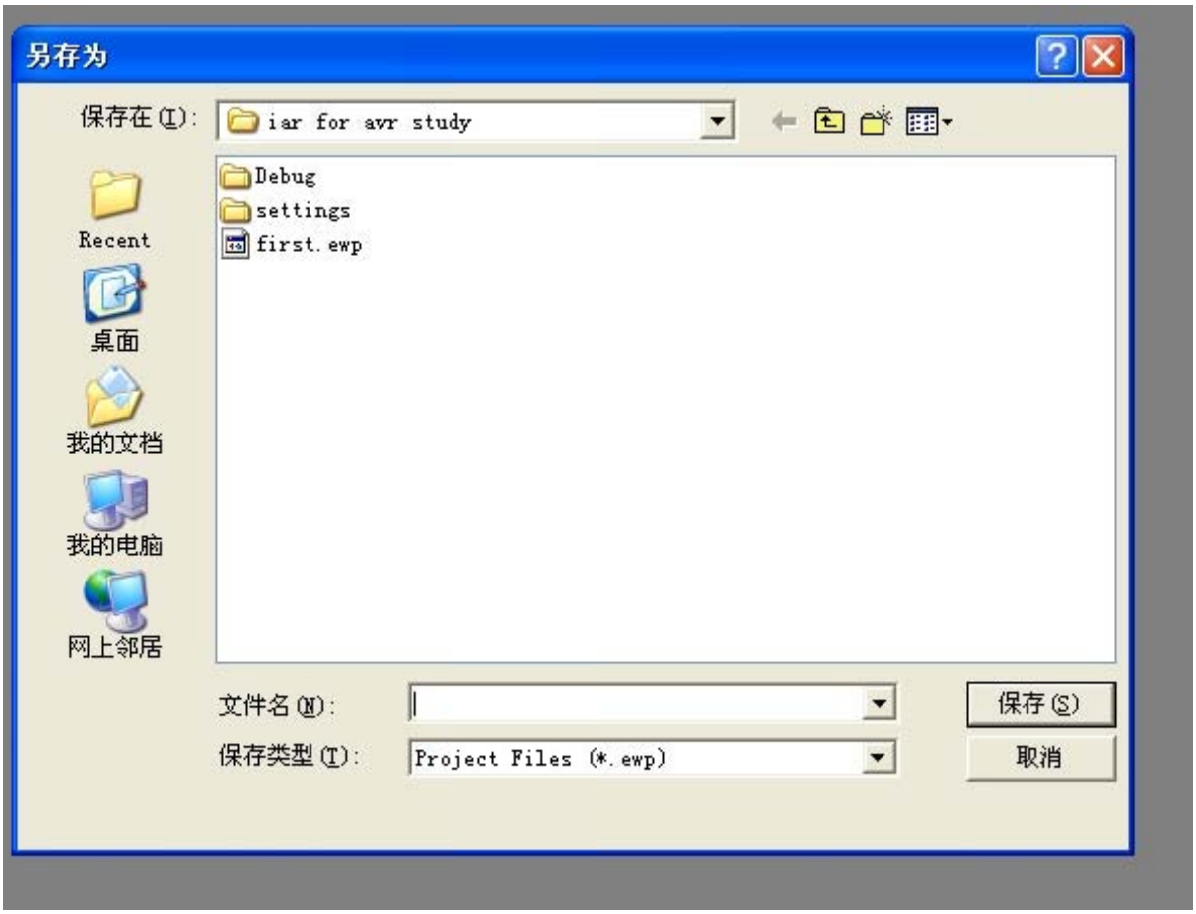
a6.jpg (18.13 KB)
2009-12-20 22:42

4、出现如下对话框，在“tool chain”中选择单片机类型：AVR。在“project templates”中选择空项目模板“empty project”。然后点击“OK”按钮，进入下一步操作



a7.jpg (48.25 KB)
2009-12-20 22:42

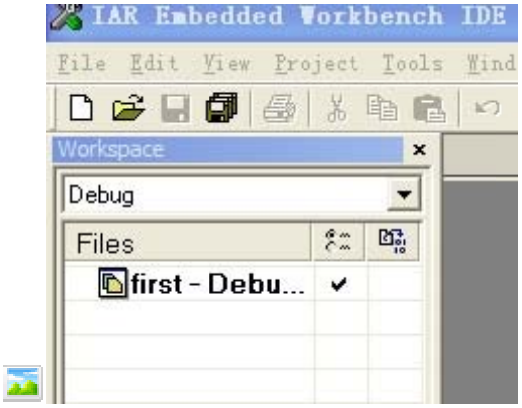
5、出现如下对话框，提示我们将该项目另存到那个文件夹中，选择我们即将要保存的文件夹，然后输入项目名称（注意文件格式是.ewp 格式），点击“保存”按钮，进入下一步操作



a8.jpg (75.16 KB)
2009-12-20 22:42

6、这时工程项目就出现在 IAR IDE 集成开发环境界面的左侧，如下图，一般在默认状态下，系统会生成两个项目文件

版本：调试版本“debug”和发行版本“release”。调试版本一般用以在项目文件开发阶段，进行程序的编译、仿真、调试等。发行版本则是程序开发完成后，用于向外发布项目文件所用。

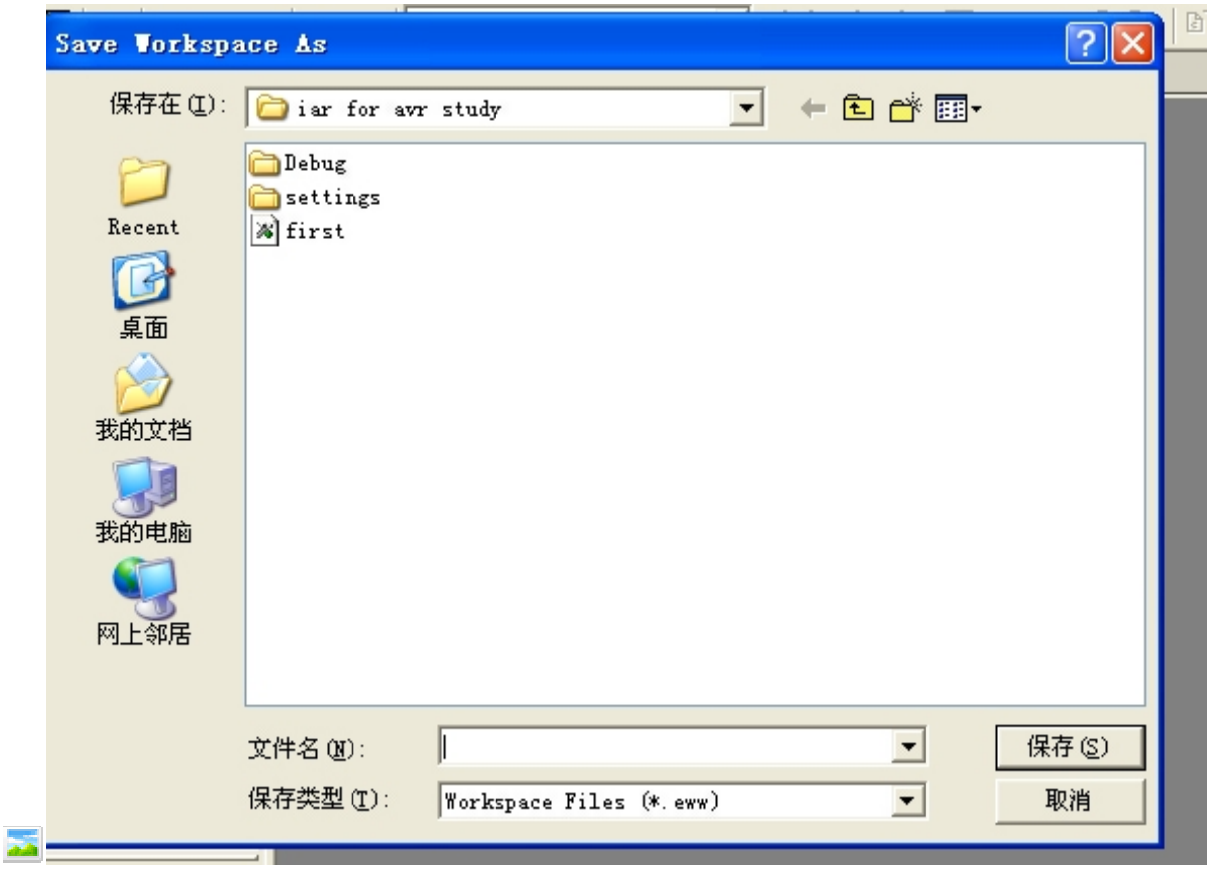


a9.jpg (33.54 KB)
2009-12-20 22:42

7、关于调试版本“debug”和发行版本“release”的选择如下图所示。在下拉箭头处进行两种版本的切换。

a10.jpg (4.92 KB)
2009-12-20 22:42

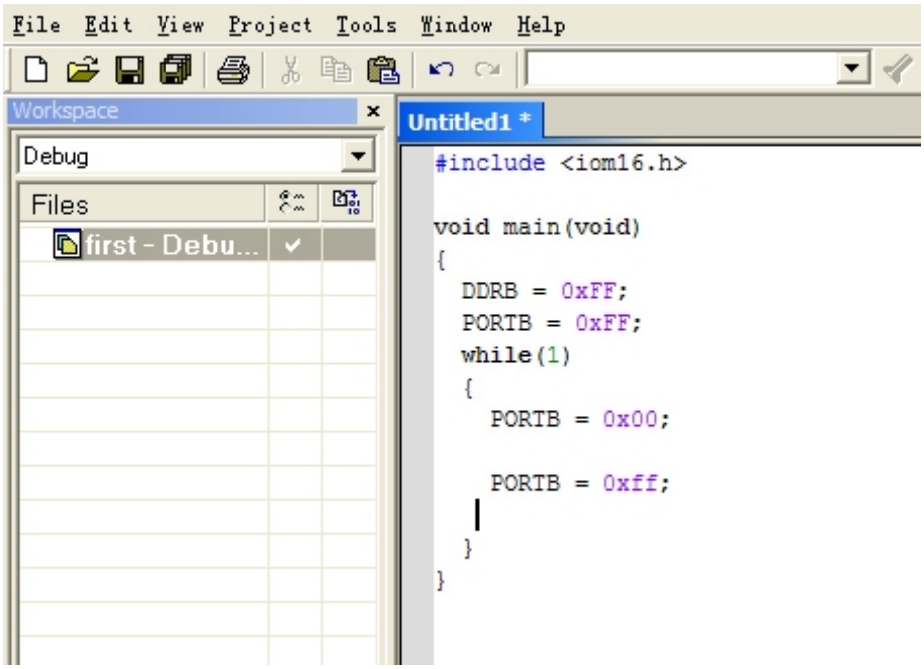
8、接下来把刚才创建的项目保存到一个工作区中，依次点击 file-save workspace 后，出现如图所示的保存界面，输入工作区的名称（注意文件格式是.eww 格式），选择存放路径，然后点击“保存”按钮进行保存。



a11.jpg (76.88 KB)
2009-12-20 22:42

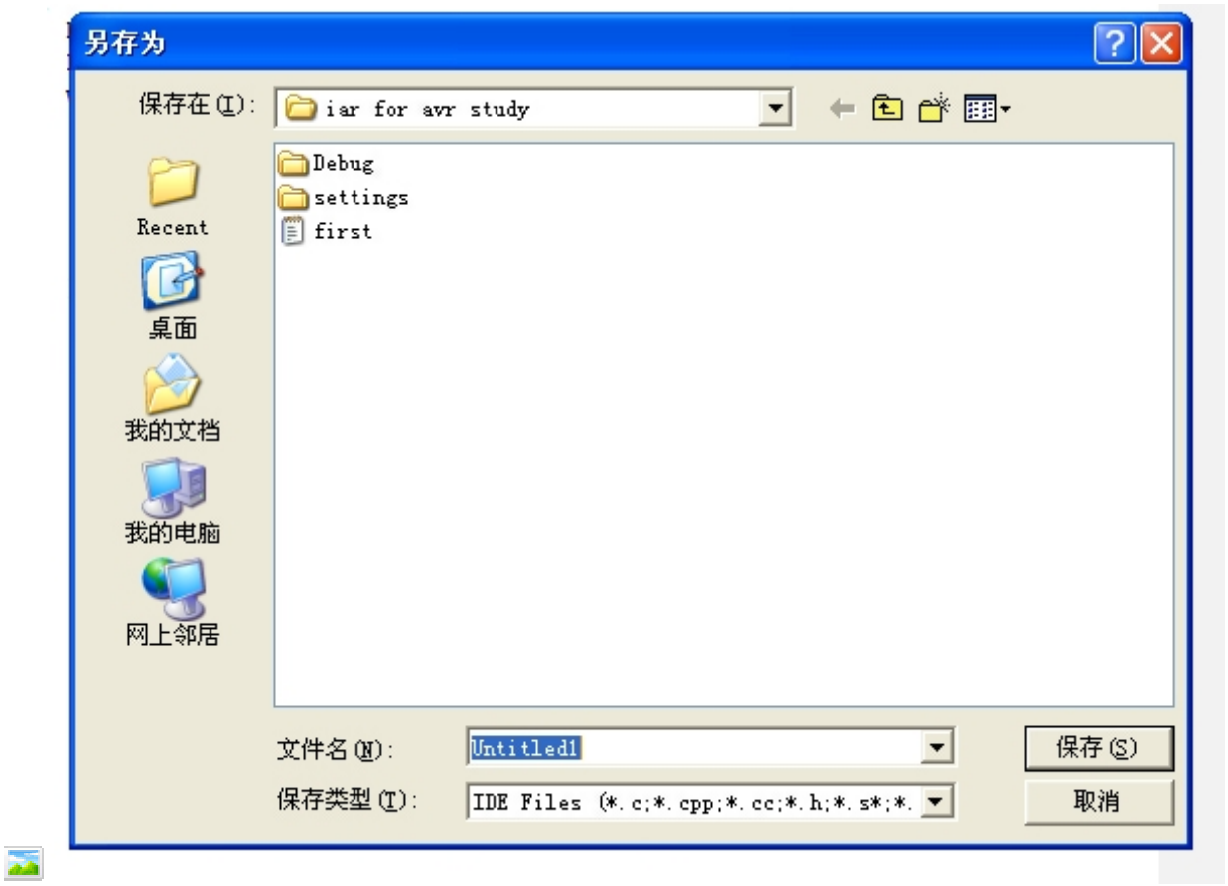
9、接下来可以编写源程序文件（也可直接添加程序文件到项目中），依次单击 new-file,编辑区就出现了名为 untitled1 的文本文件编辑窗口，输入以下程序代码：如下图所示

```
#include<iom16.h>
void main(void)
{
DDRB = 0xFF;
PORTB = 0xFF;
while(1)
{
PORTB = 0x00;
PORTB = 0xff;
}
}
```



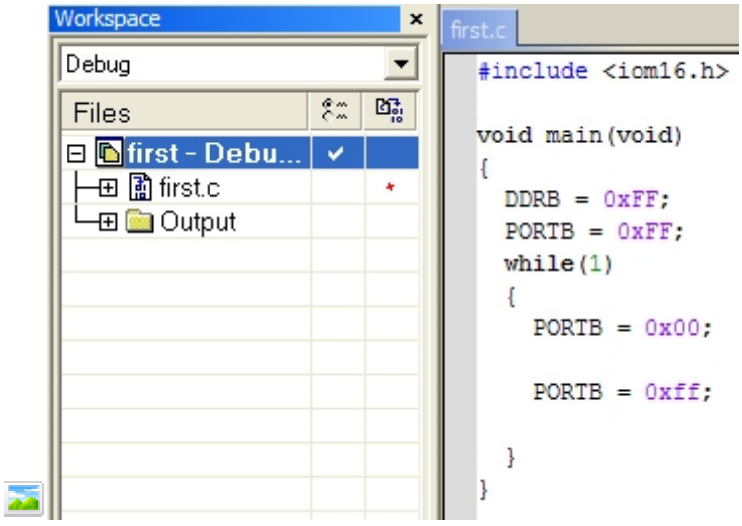
a12.jpg (68.62 KB)
2009-12-20 22:42

10、程序代码输入完毕后，需要保存该代码文件，依次点击 file-save,出现如下图所示的界面，然后输入要保存的文件名，保存在我们上面建立的文件夹中，这是可以看到，程序文件名称变为我们保存的文件名了。



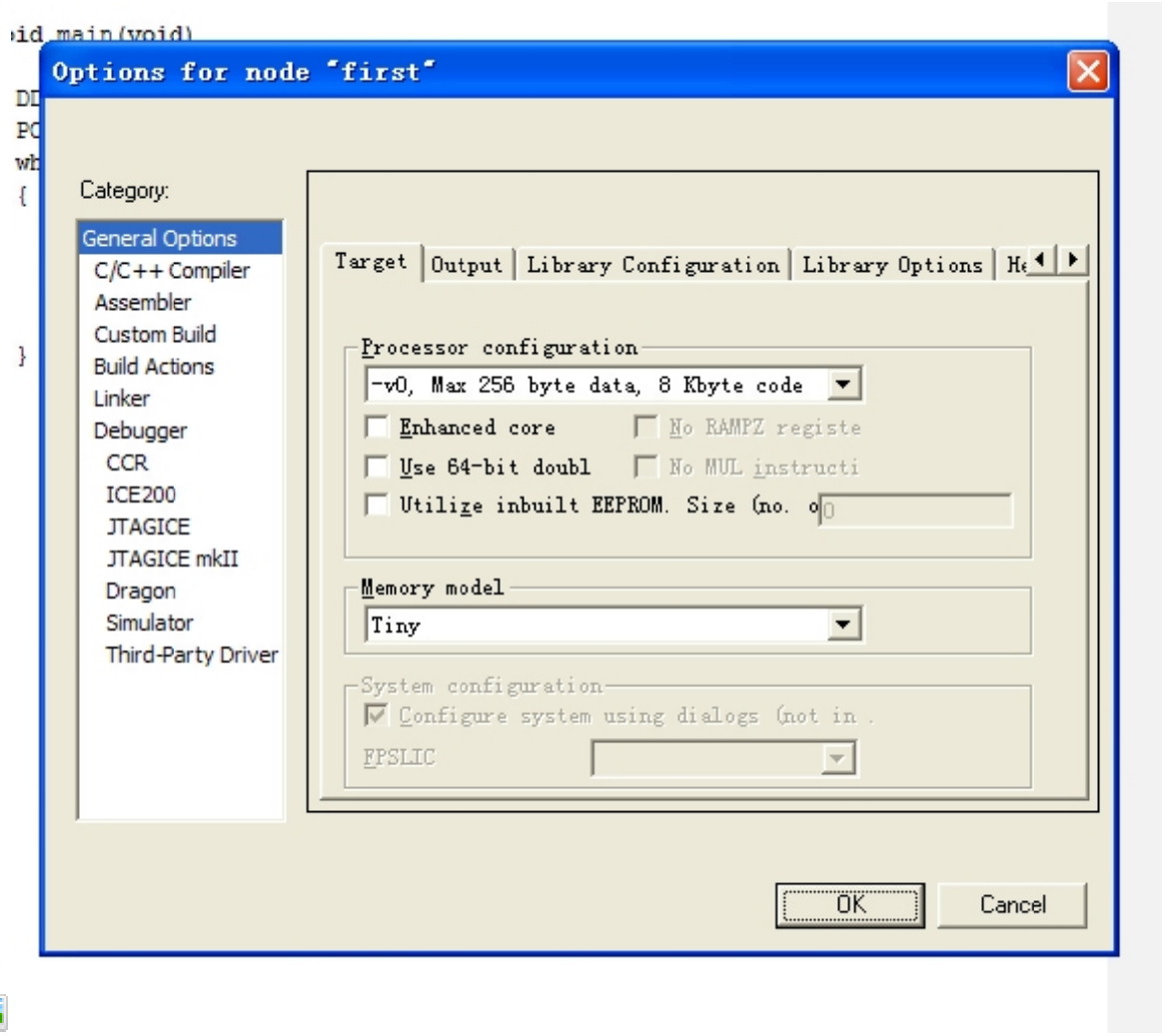
a13.jpg (79.29 KB)
2009-12-20 22:42

11、接下来需要将这个程序文件添加到项目文件中，在 workspace 框中，选择项目文件名，然后点击鼠标右键，在出现的下拉菜单中依次选择 add-add“xxx.c”（xxx.c 代表我们刚才编写的程序文件），这样便将程序文件添加到项目中了，这时 workspace 框，出现了变化，如下图，多出了一个 first.c 文件和 output 文件夹。first.c 就是我们刚才编写的程序文件，而 output 文件夹则是用来防止编译后的结果文件的。



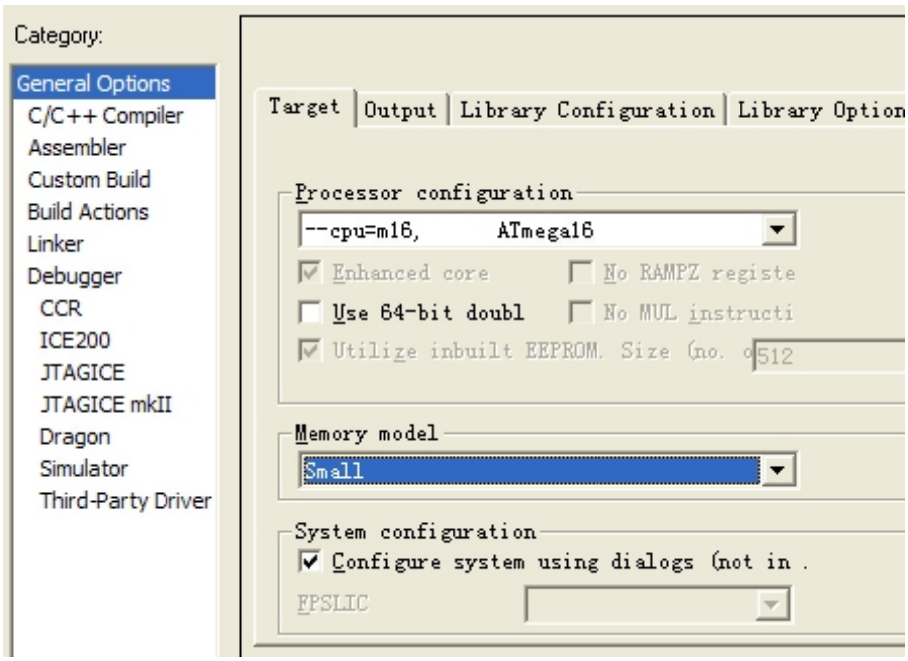
a14.jpg (54.94 KB)
2009-12-20 22:42

12、项目、工作区，程序文件都建好了，接下来该对项目文件进行编译、连接、调试方面的设置了。在 workspace 框中选择项目名称，右键点击，在弹出的下拉菜单中选择”options“选项，如下图，首先在 category 选项框下面选择 general options 选项，



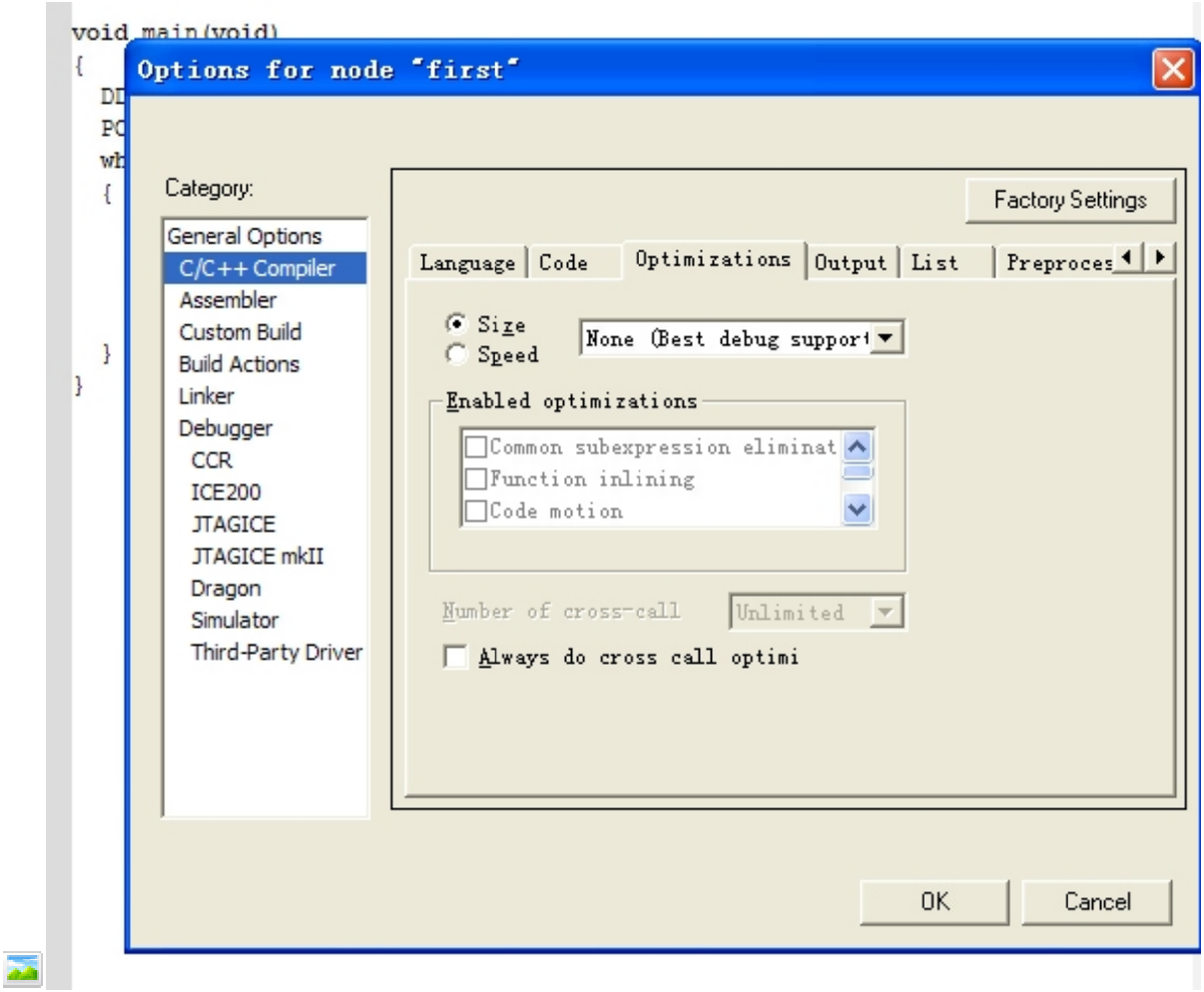
a15.jpg (127.44 KB)
2009-12-20 22:42

13、按照下图选择 CPU 型号和存储器模式



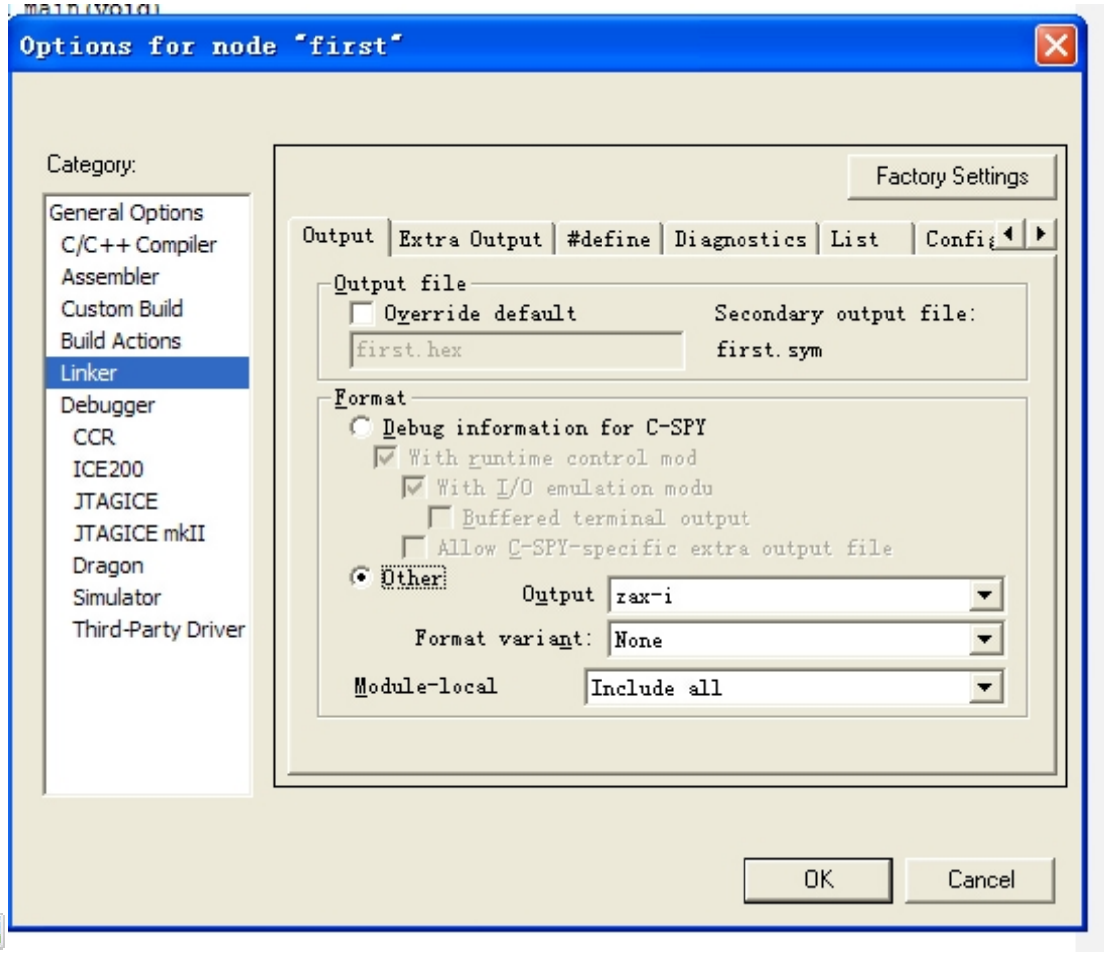
a16.jpg (91.5 KB)
2009-12-20 22:42

16、接着进行编译器的选择，如下图，优化选项设为 none



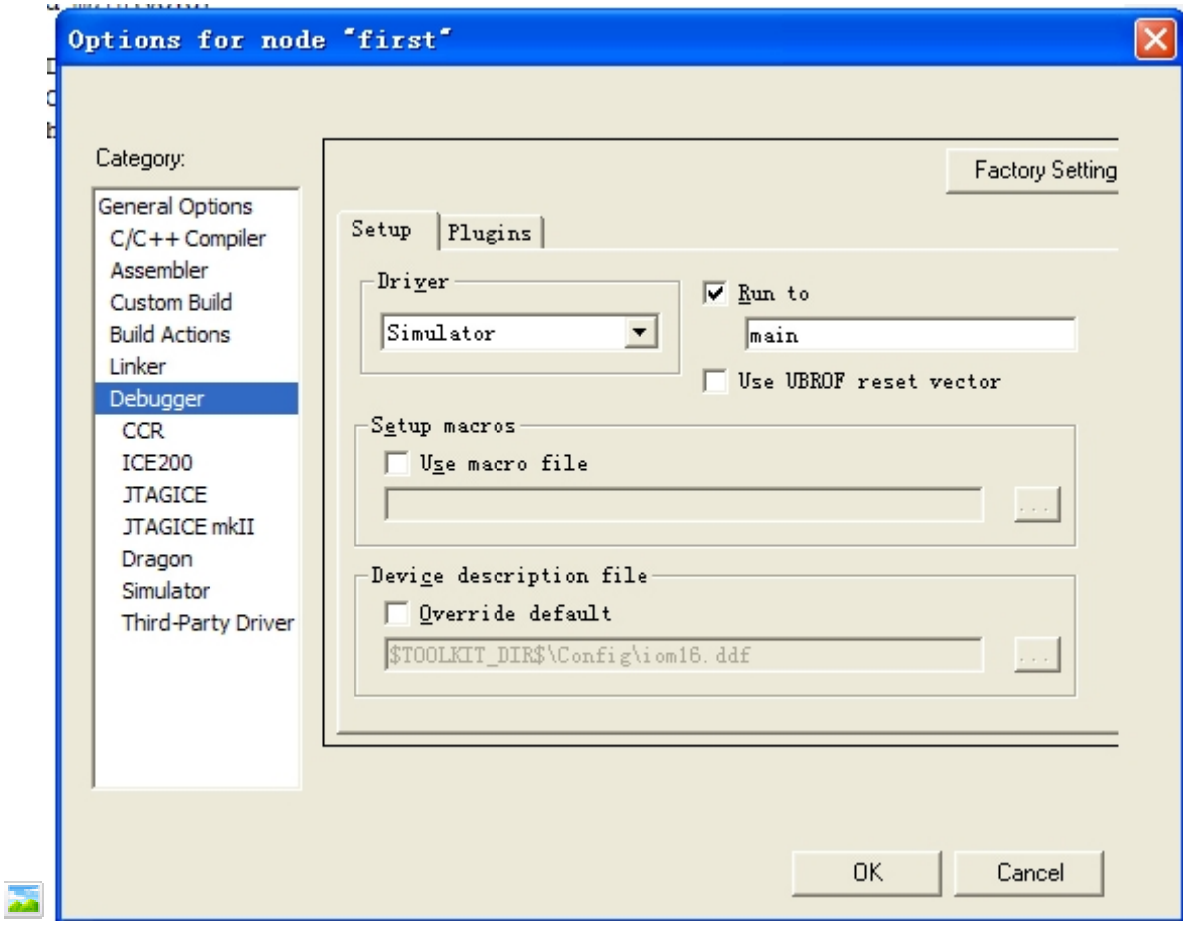
a17.jpg (116.79 KB)
2009-12-20 22:42

17、编译连接选项，如下图，编译的目的肯定是要输出可烧写的 hex 文件的，按照图中选项设置



a18.jpg (128.62 KB)
2009-12-20 22:42

18、调试选项设置，可根据实际情况选择仿真类型：软件仿真或者硬件仿真器仿真



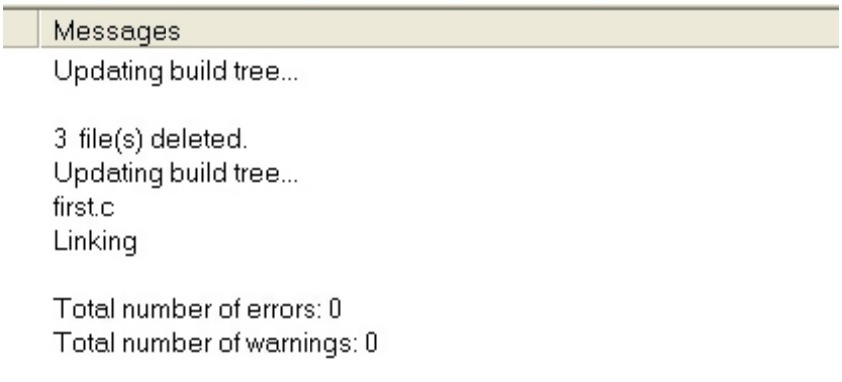
a19.jpg (106 KB)
2009-12-20 22:42

项目文件的建立和配置就是这些了，今天到这里吧，明天我们再来了解一下模拟仿真的步骤和操作事项

三、模拟仿真

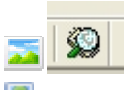
1、前面我们介绍了项目的创建和相关的配置，项目建立并配置后，就可以对该项目进行编译了，

2、选中项目文件，右键点击，在弹出的下拉菜单中选择“rebuild all”，在编译器的下面会出现调试信息，如下图，编译成功后，会提示没有错误和警告，如果有错误的话，会提示错误类型和所在位置。



a1.jpg (25.4 KB)
2009-12-21 23:14

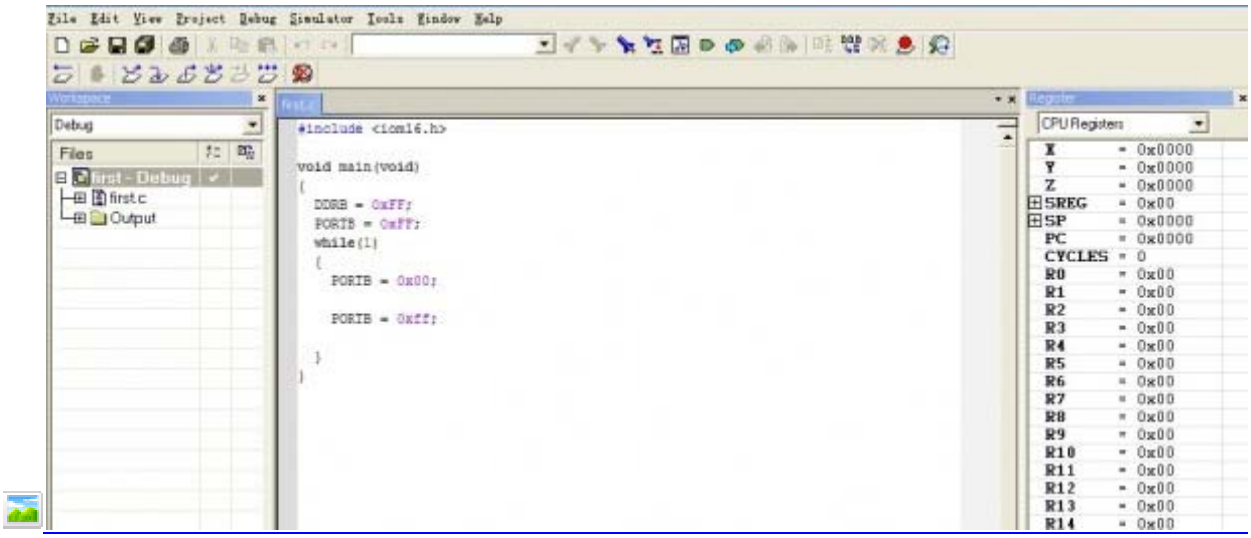
2、开始模拟仿真，点击 project 菜单下的 debug 就可以进入软件模拟仿真环境，当然也可以单击快捷方式进入模拟仿真。快捷方式如下图



a2.jpg (2.23 KB)

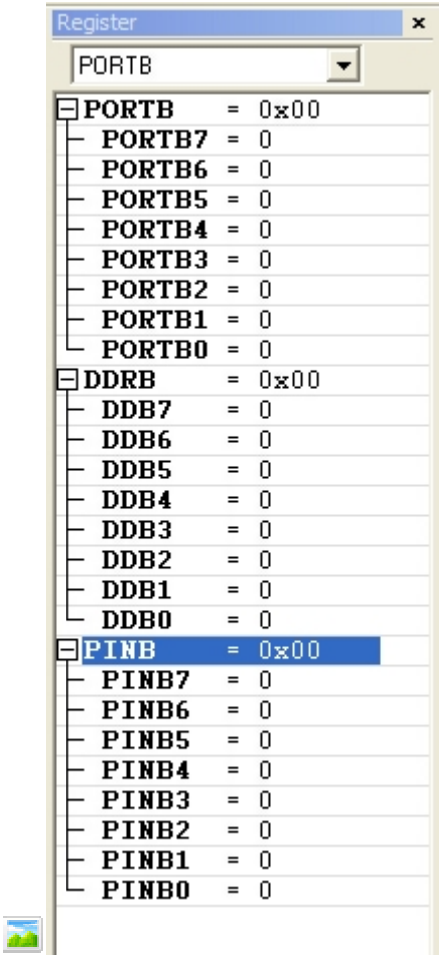
2009-12-21 23:14

3、进入模拟仿真环境后，如下图，选择 view 菜单下的 register 选项，即可看到各寄存器的情况。



a3.jpg (163.18 KB)
2009-12-21 23:14

4、在弹出的 register 对话框中，可以通过下拉箭头来选择查看不同的寄存器，例如我们这个例子主要用到了端口 B，我们就可以选择 PORTB 寄存器，如下图，可以通过点击“+”或者“-”符号来选择将该寄存器展开或者综合。

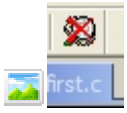


a4.jpg (68.28 KB)
2009-12-21 23:14

5、我们可以通过如下图的快件方式来选择程序运行方式：单步、跳过、全速、复位，退出仿真等

a5.jpg (23.6 KB)
2009-12-21 23:14

6、通过单步运行方式可以看到 PORTB 端口寄存器的内容随着程序的执行发生变化，说明端口的输出值在变化，这与程序的功能是一致的，
7、仿真调试结束后，可以选择 debug 菜单下的 stop debug 按钮退出仿真界面，当然也可以通过点击如下图所示的快捷按钮退出仿真环境。



a6.jpg (3.12 KB)
2009-12-21 23:14

今天就到这里，明天我们来学习一下中断程序的编写
四、中断程序的编写
1、在 IAR 中中断处理函数的形式为：

```
#pragma vector =  中断向量
_interrupt void  函数名(void)
{
/******中断服务函数的程序代码********/
}
```

例如：定时器 0 的溢出中断

```
#pragma vector = TIMER0_OVF_VECT
_interrupt void timer0(void)
{
/******中断服务函数的程序代码********/
}
```

2、不同型号的 AVR 芯片，其中断个数和中断向量各不相同。以下是 ATmega16 的中断向量表。从下面的中断向量表可以看出，所谓的中断向量，其实只是一个符号，利用#define 语句，将中断入口地址与中断向量符号联系了起来，这样用助记符来表示中断，便于记忆

```
#define RESET_vect (0x00)
#define INT0_vect (0x04)
#define INT1_vect (0x08)
#define TIMER2_COMP_vect (0x0C)
#define TIMER2_OVF_vect (0x10)
#define TIMER1_CAPT_vect (0x14)
#define TIMER1_COMPA_vect (0x18)
#define TIMER1_COMPB_vect (0x1C)
#define TIMER1_OVF_vect (0x20)
#define TIMER0_OVF_vect (0x24)
#define SPI_STC_vect (0x28)
#define USART_RXC_vect (0x2C)
```

#define USART_UDRE_vect (0x30)
#define USART_TXC_vect (0x34)
#define ADC_vect (0x38)
#define EE_RDY_vect (0x3C)
#define ANA_COMP_vect (0x40)
#define TWI_vect (0x44)
#define INT2_vect (0x48)
#define TIMER0_COMP_vect (0x4C)
#define SPM_RDY_vect (0x50)

3、中断函数不能进行参数传递，所以中断函数的输入和输出参数全部为 void，即没有参数。

4、中断函数不能被程序中的任何函数调用，只能是有中断发生时，系统自动调用中断函数。

5、中断程序的编写就是这样了，但是作为一篇帖子，这些内容显得有些单薄。下面再来个例子吧：外部中断 INT0/INT1 的嵌套中断例子

本例子实现外部中断 0 和外部中断 1 的嵌套，具体实现以下功能：程序开始，端口 PROTB 输出 0XFF，INT1 中断触发后，PORTB 输出 0X0F，并一直在 INT1 中断程序中循环，在进入 INT1 中断后，打开全局中断，这样当 INT0 中断发生后，便产生了在 INT1 中断中嵌套发生 INT0 中断。进入 INT0 中断后，PORTB 输出 0X00。

程序文件如下：

```
#include
unsigned int counter;
void delay_ms(unsigned int cnt)
{
unsigned int i,j;
for(i=0;i<CNT;I++)< font>
{
For(j=0;j<2000;j++);
}
}
Void main(void)
{
PORTB=0XFF;
DDRB=0XFF;
MCUCR=0X0A;
GICR=0XC0;
SREG=0X80;
While（1）
{
}
}
#pragma vector = INT0_VECT
_interrupt void int0(void)
{
PORTB=0X00;
Delay_ms(2000);
}
#pragma vector = INT1_VECT
_interrupt void int1(void)
{
SREG=0X80;
PORTB=0X0F;
```

```
For(counter=0;counter<6000;counter++)
{
Delay_ms(2000);
}
}
```

中断函数的介绍就是这些了，下一篇我们来学习一下如何对 AVR 单片机内部的 EEPROM 进行操作。

五、使用库函数读写内部 EEPROM

1、IAR for AVR 软件提供了 2 个读写内部 EEPROM 的库函数：

```
#define _EEPUT(ADR,VAL)
(*((unsigned char
_eeprom*)ADR)=VAL)
#define _EEPUT(VAR,ADR)
(VAR =*((unsigned char
_eeprom*)ADR))
```

可以看出，这两个函数的作用分别是：将数据 VAL 写入到内部 EEPROM 地址 ADR 处；

从内部 EEPROM 的地址 ADR 处读出数据 VAR。

2、这两个库函数包含在头文件 inavr.h 文件中。所以如果要使用这两个库函数，程序文件中必须要包含这个头文件。

3、下面以一个简单示例来说明这两个函数的用法。

程序中实现了将一个数据写入到内部 EEPROM 的指定位置，然后再将该位置的数据读出。

```
#include
#include
Void main(void)
{
Unsigned char val=0;

_EEPUT(120,200);
//将数据 200 写入内部 EEPROM 的地址 120 位置处；

_EEPUT(val,120);
//从内部 EEPROM 的地址 120 位置处读取数据，并将数据赋值给
//变量 val
}
```

呵呵，IAR for AVR 的学习就是这样了，对于一种编程软件，

1、我们首先要学会怎么建立项目文件，怎么向项目文件中添加各种程序文件

2、然后就是项目文件的相关设置，主要是芯片类型，晶振频率，编译和连接、输出的烧录文件以及仿真文件的设置

3、然后就是如何编译项目文件，以及如何进行简单的软件模拟仿真

4、还有就是要了解该软件的各种库文件。