

# KF8 系列

# ChipON IDE 用户使用手册

2016-12 月

## 目录

目录.....	2
1. 概述.....	5
2. 安装.....	6
3. 启动 CHIPON IDE 程序.....	10
4. 新建汇编项目.....	12
5. 新建汇编类型源文件代码与编译.....	16
6. 新建 C 项目.....	22
7. 新建 C 类型源文件代码与编译.....	26
8. 查看芯片寄存器信息.....	32
9. 芯片配置位配置.....	33
10. 芯片 BLOCKEE 配置.....	36
11. 下载 HEX 文件到单片机.....	39
12. 资源使用率.....	42
13. 回写校准值.....	43
14. 项目开发与调试.....	44
15. 固件升级.....	54
16. 硬件仿真（调试器）使用常见错误解决办法.....	55
17. IDE 使用技巧.....	58
17.1 切换工作空间.....	58
17.2 导出项目.....	59

17.3	导入项目.....	60
17.4	关键字高亮显示设置.....	61
17.5	代码补全.....	63
17.6	重构.....	64
17.7	显示行号.....	67
17.8	相关快捷键.....	68
17.9	历史文件比较.....	69
17.10	定制透视图.....	71
17.11	查找/替换.....	73
17.12	查找特定位置的字符串.....	74
17.13	返回到上一个操作位置.....	75
17.14	关闭工程.....	75
17.15	打开工程.....	76
17.16	最大化、最小化和关闭当前窗口.....	76
17.17	快速的注释代码.....	78
17.18	改变视图窗口的位置.....	78
17.19	改变编辑器字体.....	78
17.20	恢复初始视图布局.....	79
17.21	清理项目.....	80
17.22	快速定位错误信息位置.....	80
17.23	恢复删除文件.....	81
17.24	调试设置.....	83
17.25	代码跳转.....	84
17.26	管理配置.....	85
17.27	编译结果查看.....	86
17.28	使代码编译效率高.....	87

在使用本手册前,请您认真阅读以下使用许可协议。只有在同意以下使用许可协议的情况下,方能使用本手册中介绍的产品。

### 版权公告

未经上海芯旺微电子科技有限公司书面允许,任何公司、个人不得以任何形式复制本使用手册的全部或部分内容。

### 重要声明

上海芯旺微电子科技有限公司努力使本手册中提供的信息准确和适用,然而,产品及手册可能包括技术或印刷上的错误。上海芯旺微电子科技有限公司保留在不事先通知的情况下改变本使用手册全部或部分内容的权力。

## 1. 概述

ChipON IDE 为上海芯旺微电子有限公司出品的一款针对 KungFu 芯片开发使用的集成开发环境。

ChipON IDE 主要支持以下功能:

- ◆ 支持汇编项目
- ◆ 支持 C 语言项目
- ◆ 支持 ICSP 在线下载
- ◆ 支持多项目管理
- ◆ 支持汇编和 C 语言的智能录入
- ◆ 支持汇编和 C 语言代码的悬浮提示
- ◆ 支持芯片资源使用率实时查看
- ◆ 支持查看相关芯片的信息
- ◆ 支持 HEX 文件查看以及编辑
- ◆ 支持 C 语言中高亮显示汇编关键字

ChipON IDE 支持的芯片型号如下, 但更多型号支持工具内型号选择。

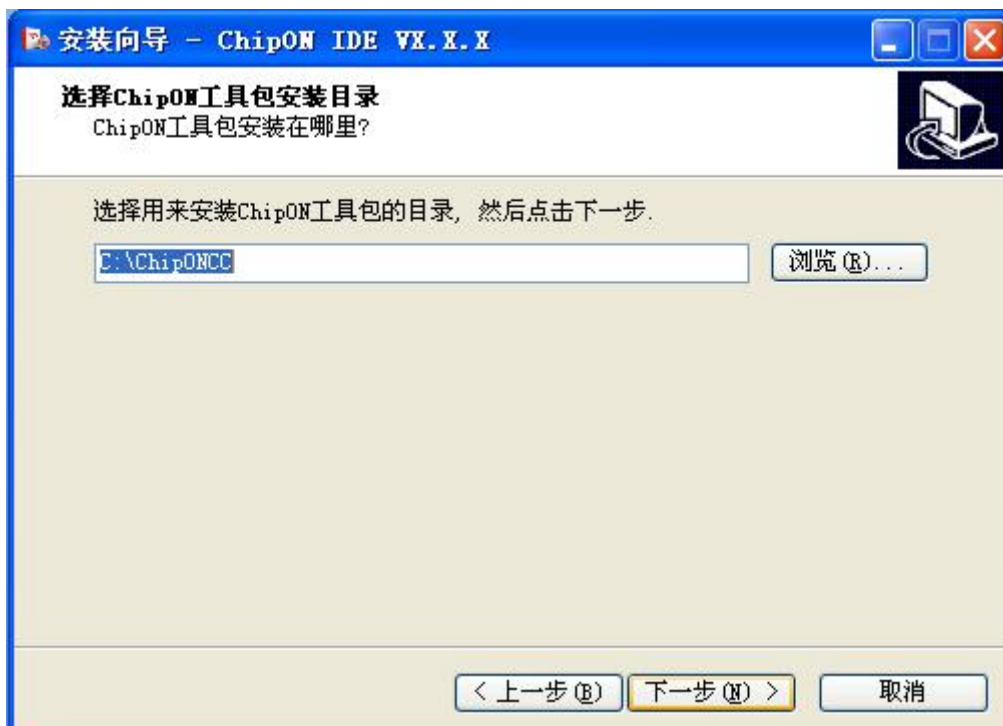
KF8F200	KF8F201	KF8F202	KF8F203
KF8F204	KF8F210	KF8F211	KF8F212
KF8F213	KF8F214	KF8F221	KF8F222
KF8F232	KF8F233	KF8F300	KF8F303
KF8F304	KF8F310	KF8F312	KF8F313
KF8F321	KF8F323	KF8F324	KF8F333
KF8F334	KF8F335	KF8F336	KF8S1005
KF8S1006	KF8S1010	KF8S1011	KF8S1022
KF8S1023	KF8S1024	KF8S1025	KF8S1100
KF8S1101	KF8S1001	KF8S1020	KF8V111
KF8V112	KF8V120	KF8V200	KF8V211
KF8V212	KF8V216	KF8V218	KF8V220
KF8TS2402	KF8TS2408	KF8TS2410	KF8TS2414
KF8TS2302	KF8TS2308	KF8TS2310	KF8TS2314
KF8F4156	KF8F4158	KF8V204	KF8V427
KF8TS2716	KF8S210	KF8S212	KF8S360
KF8F4155	KF8F3155	KF8F2156	KF8F3156
KF8TS2516	KF8V327	KF8V429	KF8S1007
KF8F1020	KF8TS2702	KF8TS2708	KF8TS2710
KF8TS2714	KF8V304	KF8V404	KF8V325

## 2. 安装

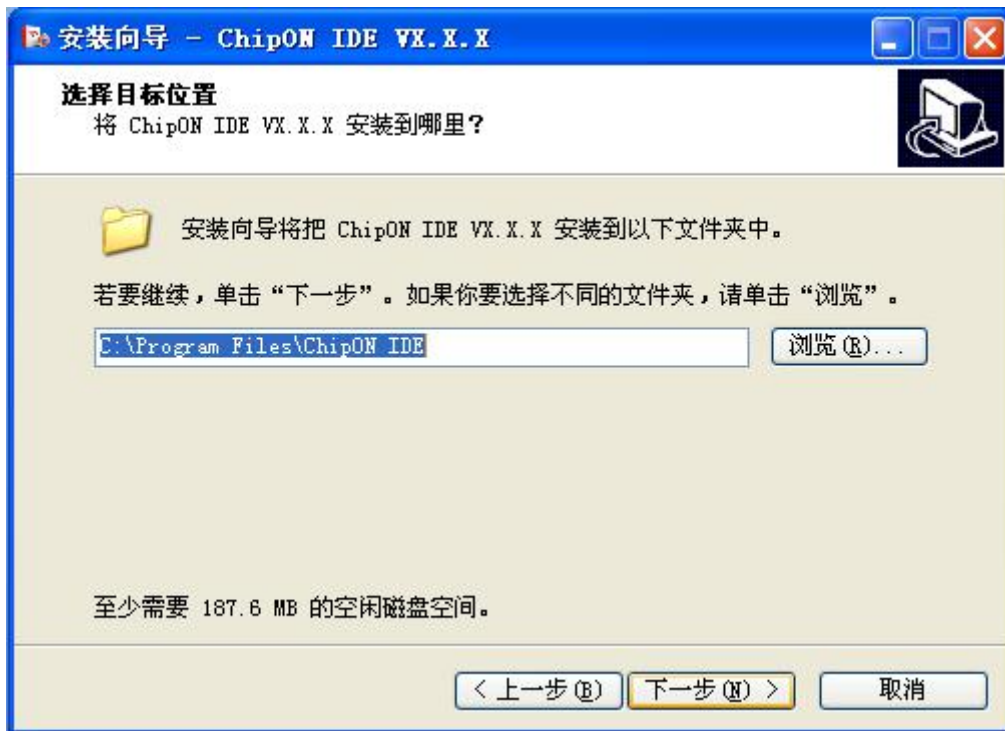
双击 ChipON IDE VX.X.X.exe 安装文件，进入安装向导界面，下一步：



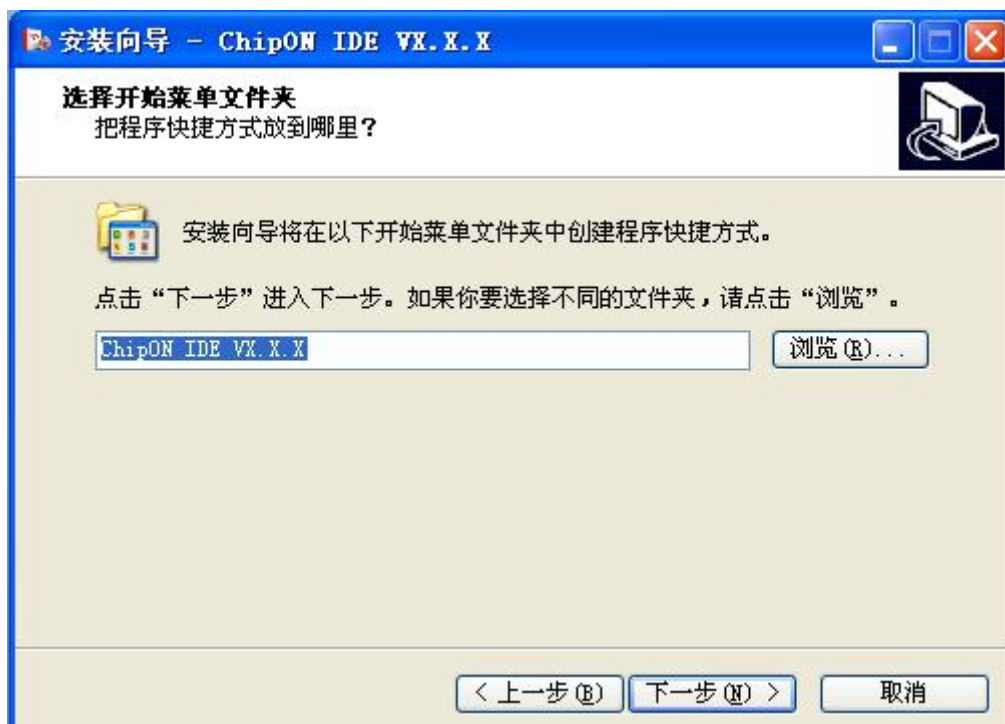
设置用来安装 ChipON 工具包的目录，下一步：



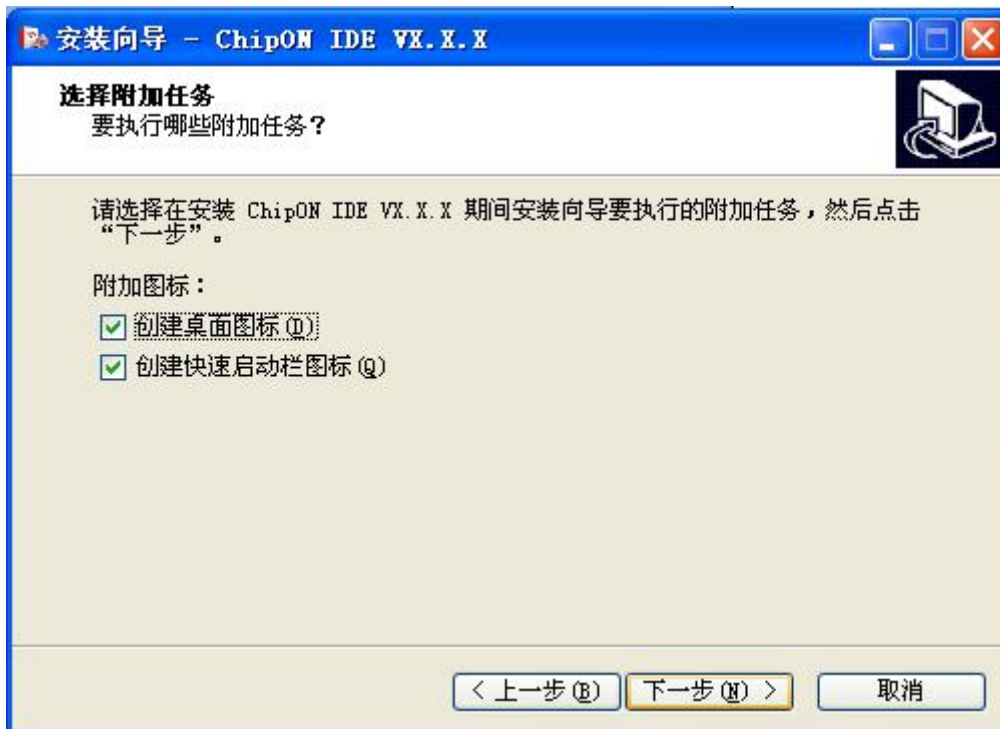
设置 ChipON IDE 的安装位置，下一步：



设置程序快捷方式存放的位置，下一步：



选择是否创建桌面和快速启动栏图标，下一步：

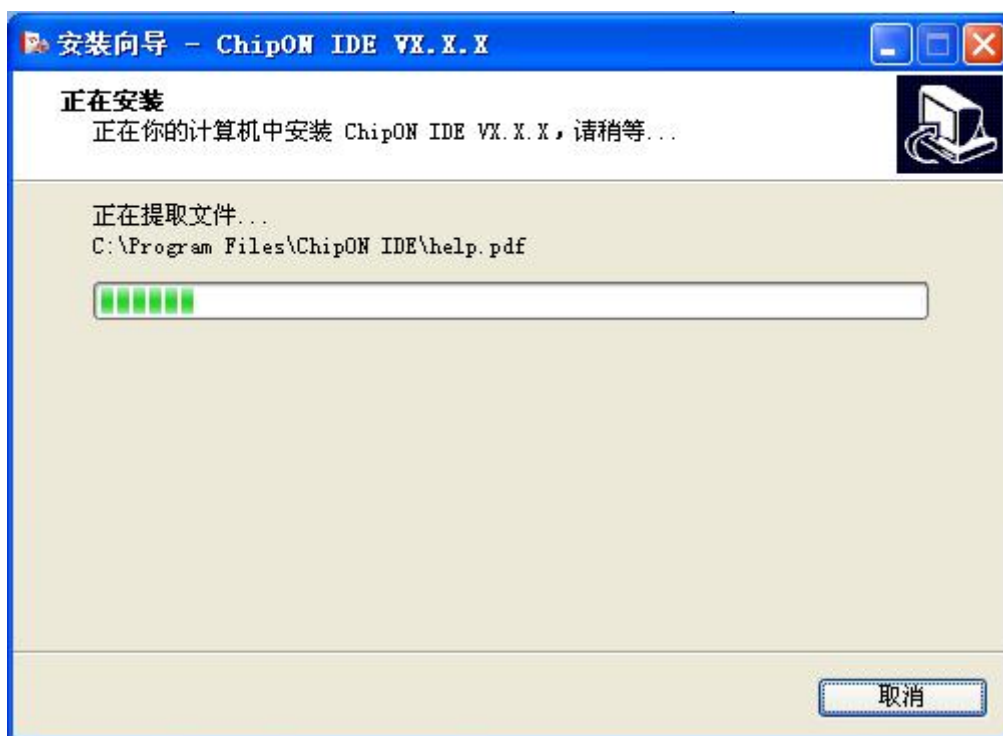


确认选择的安装位置，点击“安装”按钮：





正在进行安装, 请稍等:



安装完成后点击“完成”, 则完成安装过程, 开始运行 ChipOn IDE.



### 3. 启动 ChipON IDE 程序

在开始-->所有程序-->ChipON IDE VX. X. X-->ChipON IDE VX. X. X 找到 ChipON IDE, 点击则运行程序, 如图:



如果创建了桌面快捷方式的话, 可以直接点击启动, 如下图所示:

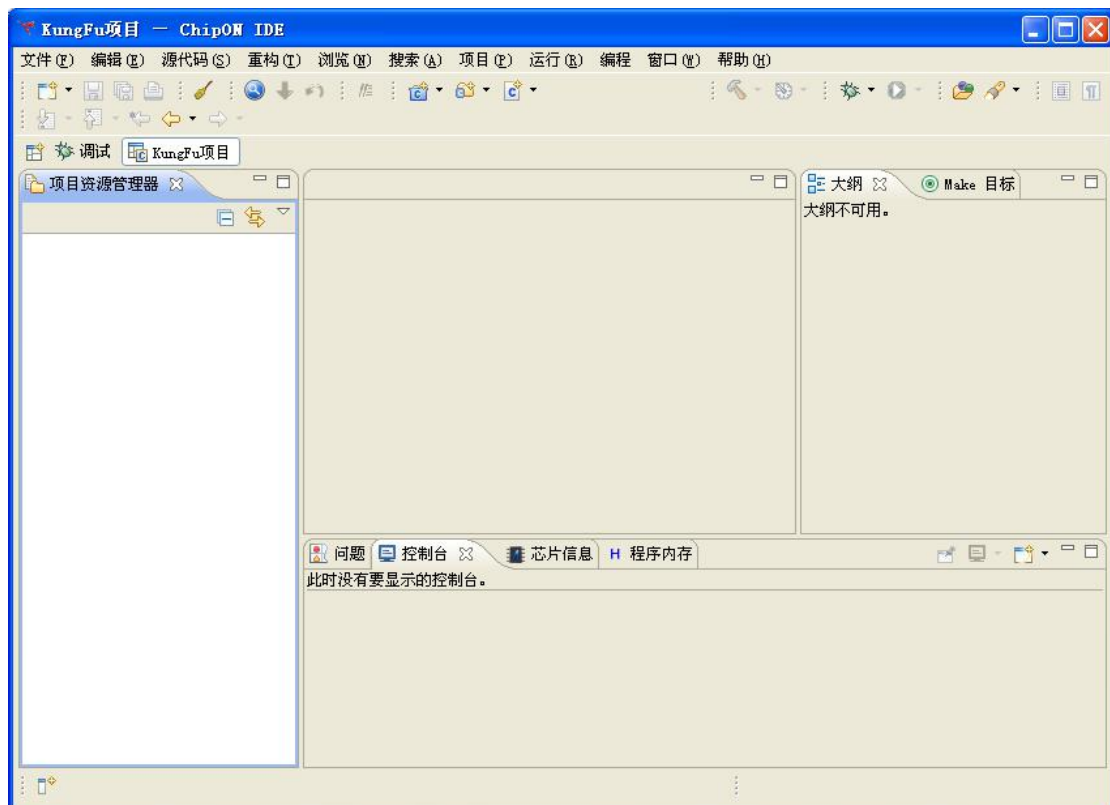


程序运行之前, 需要选择工作空间, 此工作空间是决定你创建项目存放的源文件路径, 可设置默认工作空间, 以后打开不在询问, 点击确认按钮, 进入开发界面:



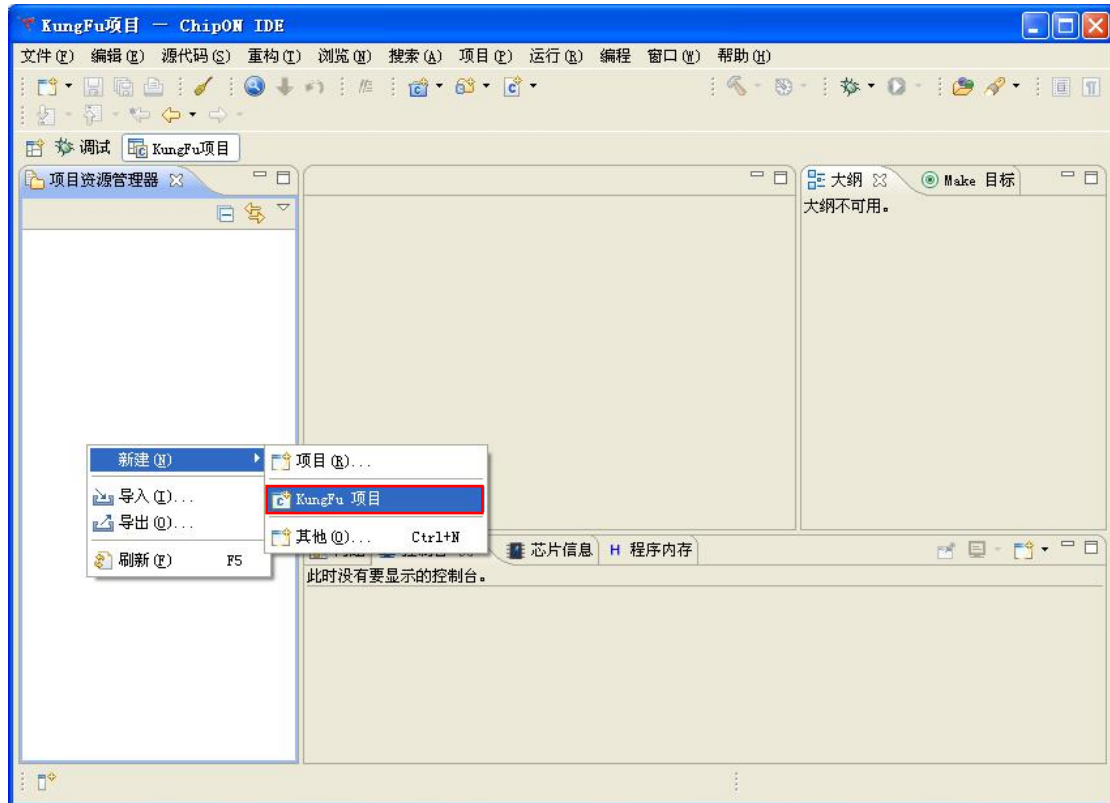
需要注意的是 ChipON IDE 采用空间的项目管理模式, 一个空间下可以存在单个项目, 也可以存在多个项目。可以建立过个工作空间, 但历史记录只保留最近的 5 个空间, 每一个空间要求为独立的, 即空间下不能建立新的工作空间。

开发主界面:



## 4. 新建汇编项目

在项目资源管理器空白处中点击右键，选择新建-->>KungFu 项目：



也可以通过单击文件-->>新建-->> KungFu 项目 或工具栏中的新建功能进行选择。

设置项目名称, 项目位置可使用缺省位置, 也可以自定义位置, **建议使用缺省位置, 即项目建立在当前的工作空间中;**

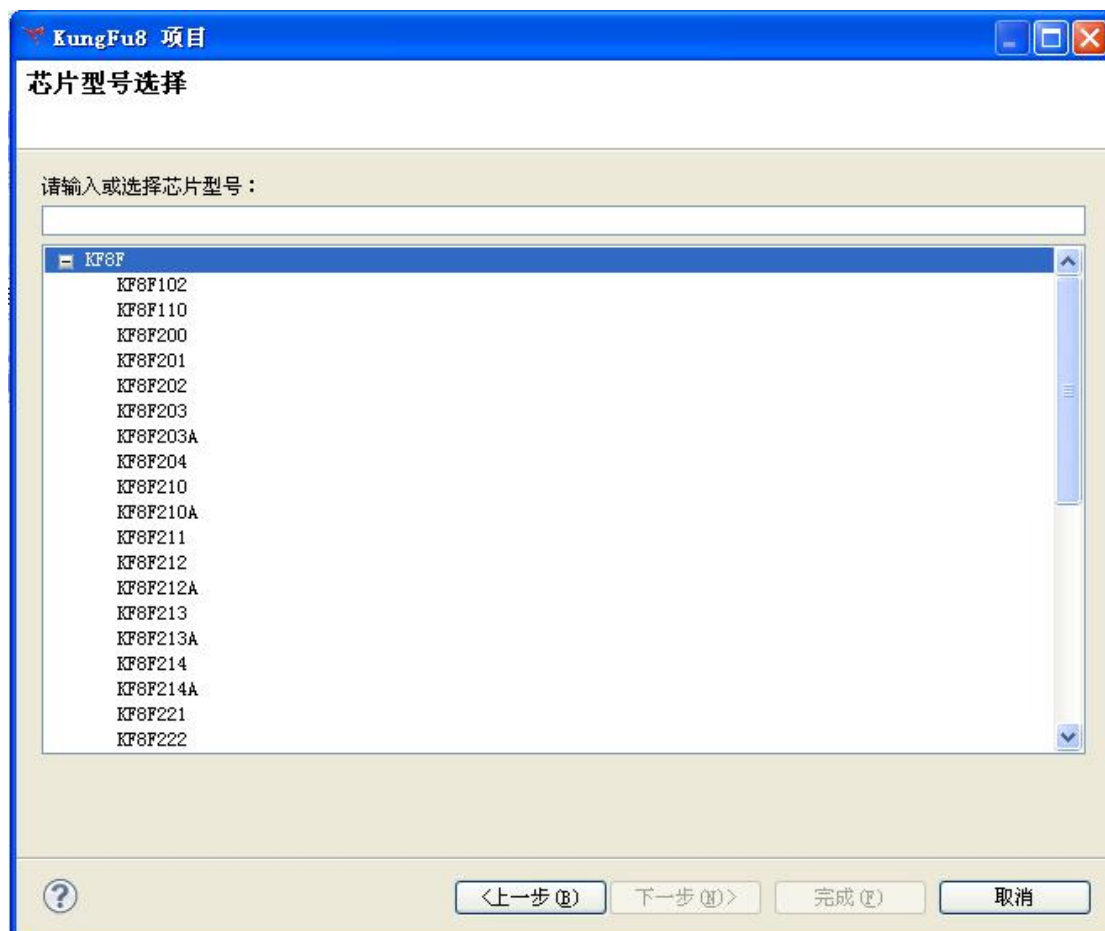
选择新建项目的类型与工具链, 在项目类型中选择“KF8-asm”, 然后下一步:



选择要部署的平台和配置，即调试模式 Debug 或正常运行模式 Release。一般按默认，不需修改。点击下一步：



选择项目所要用到的芯片型号，这里可以通过树列表从类型下选择型号，也可以通过输入进行快速过滤，如选择 KF8F312，在输入框内输入 312，并在下面列表过滤后的内容中选择对应的完整型号即可。

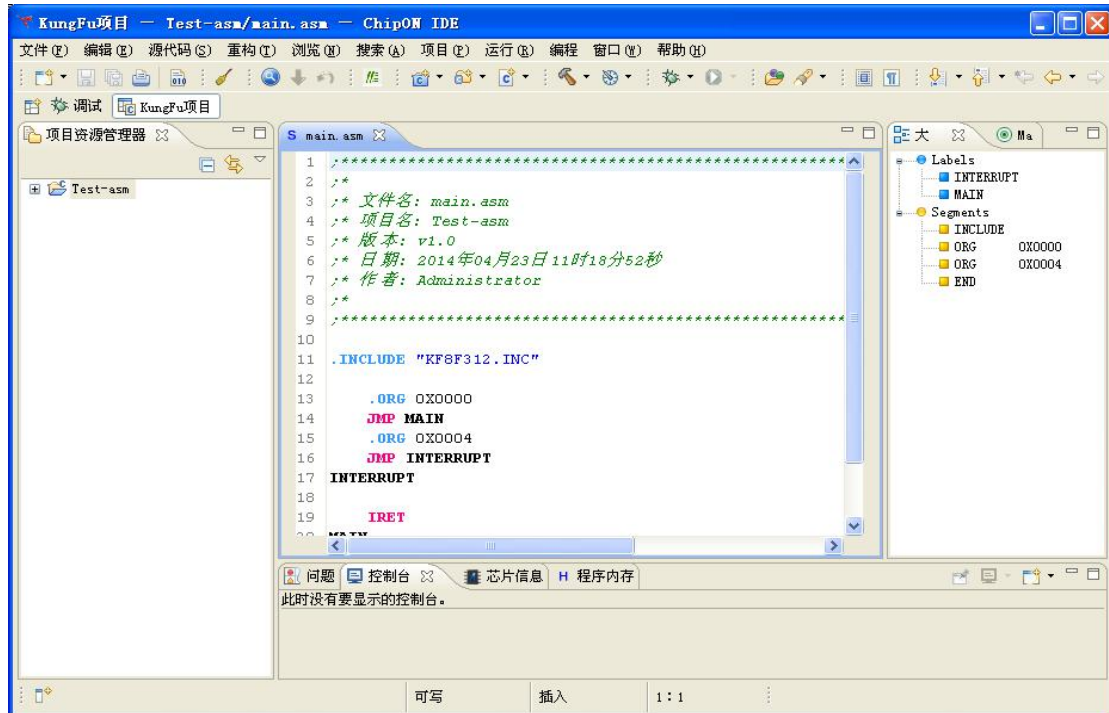


在选择型号后提供了调试电压值选择和辅助配置字的配置功能，可以选择在代码中设定，或者使用对话框进行配置。其中使用对话框配置情况下建立

“config\_set.asm”的文件，内容如：

```
“ __config 0x2007,0x06fd  
  .end ” 其中 0x2007 为配置字的映射地址，0x06FD 为配置字结果。
```

至此，一个最简单 test-asm 项目已创建完成。默认提供了选择型号的基本框架代码，包括 MAIN 入口和中断的入口表达。

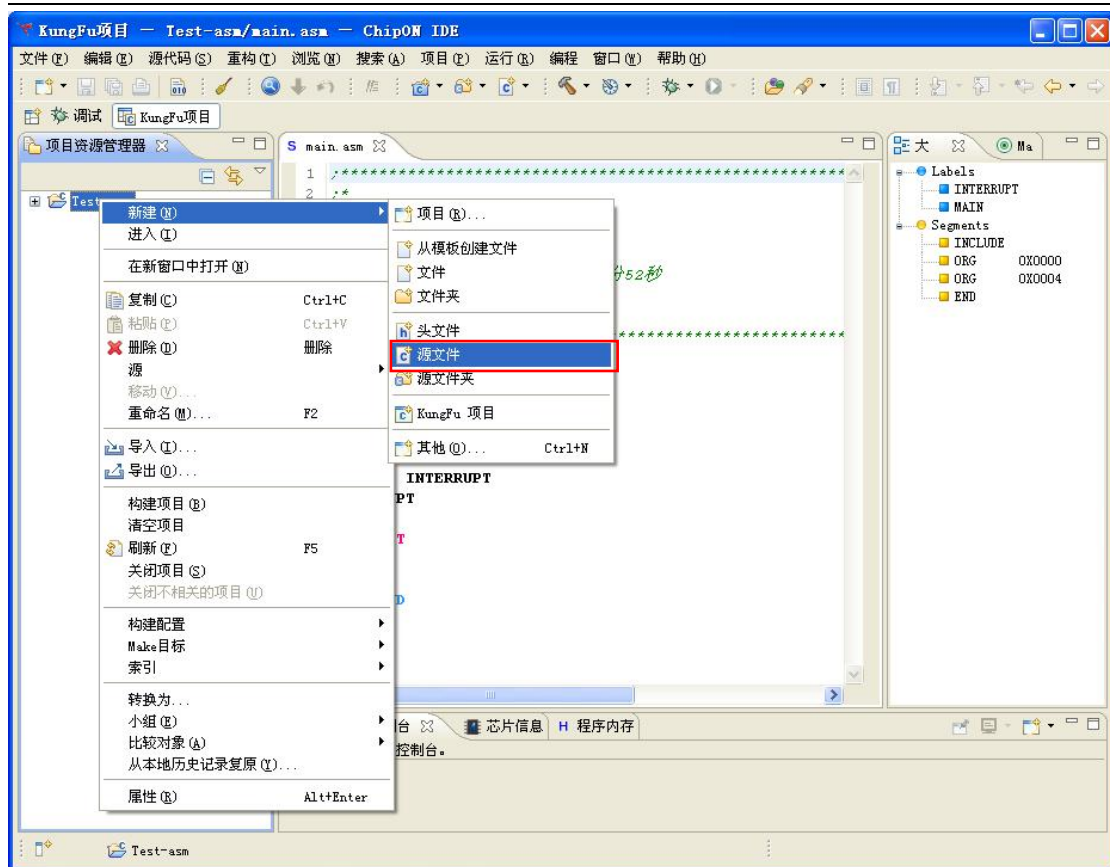


## 5. 新建汇编类型源文件代码与编译

### (1) 新建文件

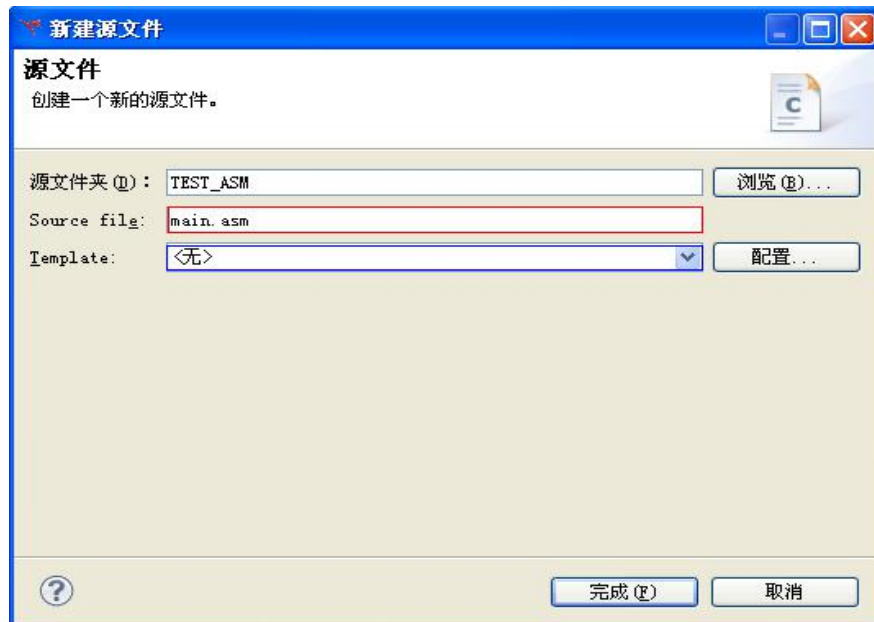
右键选择工程，新建-->>源文件：



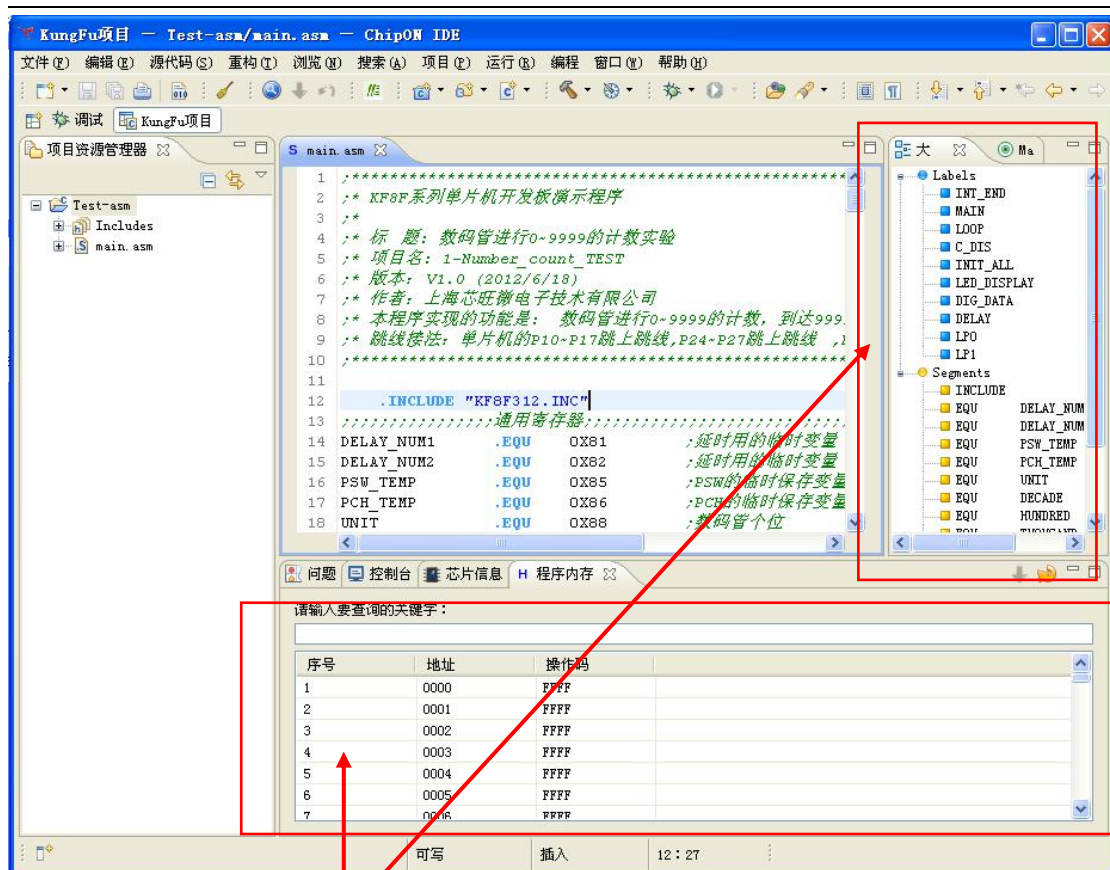


在新建源文件的页面中设置相应项:

注意: 红线圈住的地方应为源文件名字, 蓝线圈住的地方应选择为无。



在编辑器视图中, 向源文件中写入代码:



在编辑过程中,双击“芯片信息”视图中的对应寄存器名字,可直接将该寄存器名字添加到编辑器光标所在位置处。

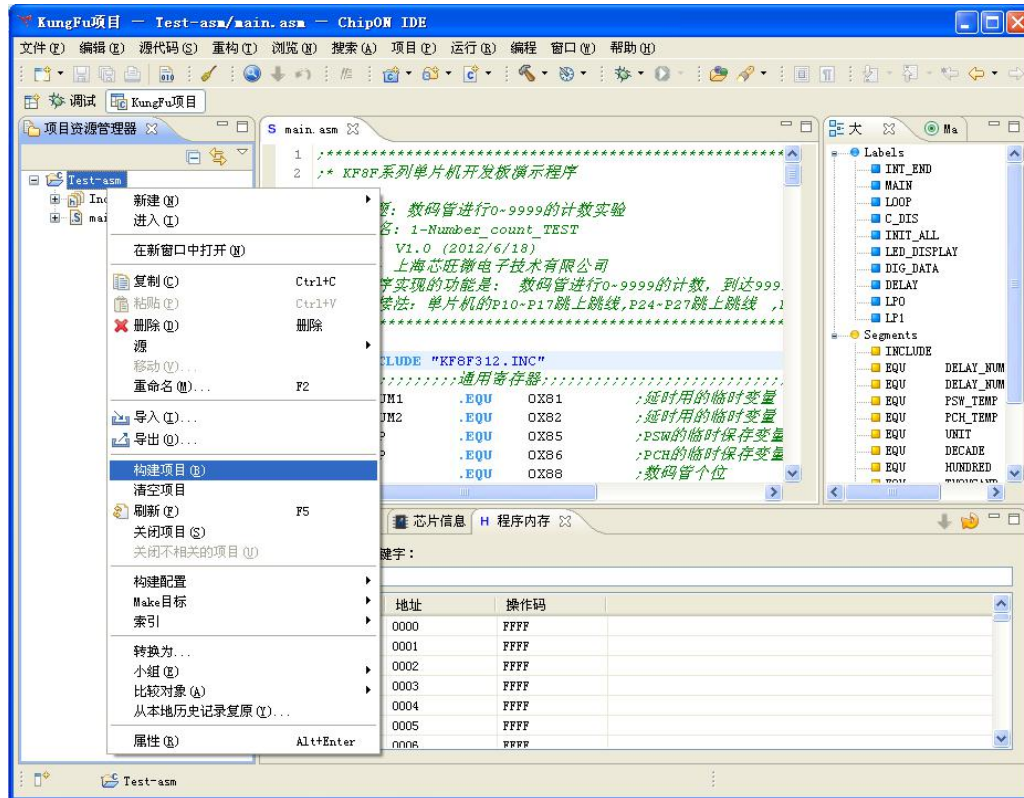
同时,通过“大纲”视图,可查看源代码中的宏定义等信息,可以方便代码的编写查找。

也可以建立头文件,汇编头文件的后缀为. INC ,格式可以参照芯片的头文件格式。

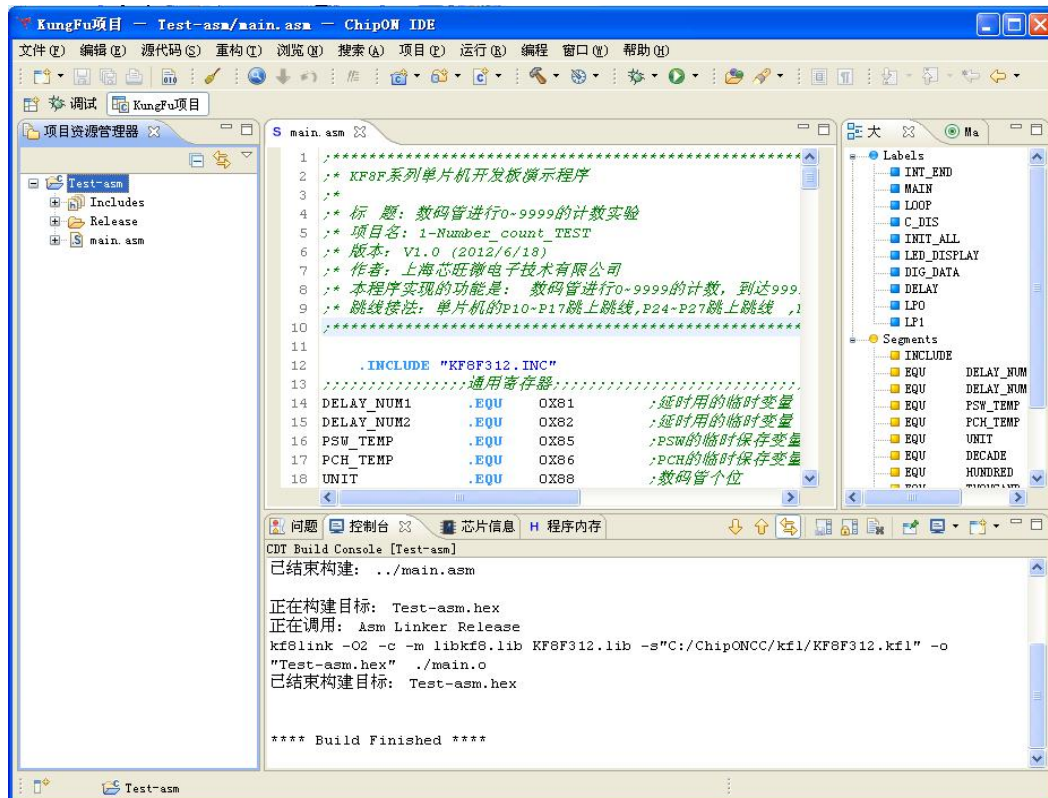
## (2) 编译源文件

**注意:** 首先应选中项目,选择项目可以通过在项目资源管理器或对应的代码文件。

方法 1 右键点击项目,然后点击构建:



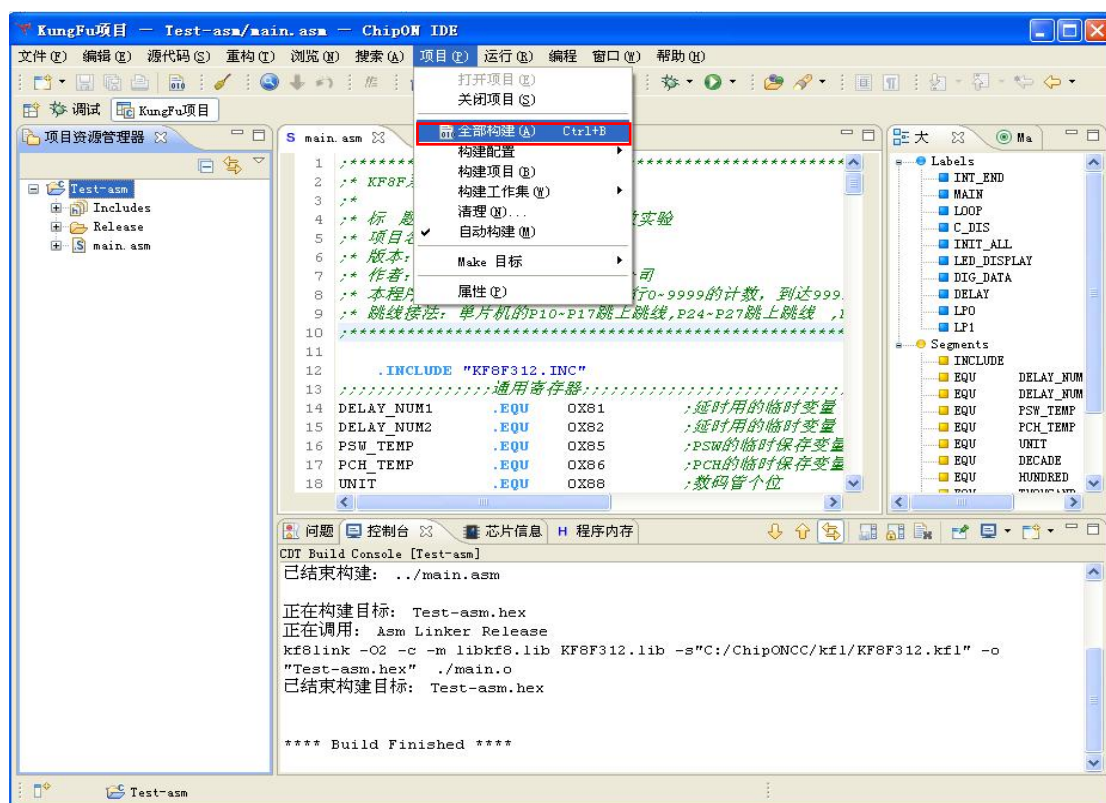
方法 2 通过快捷工具栏构建, 这里会按照选定的模式执行编译, 修改模式可以通过下拉三角选择。



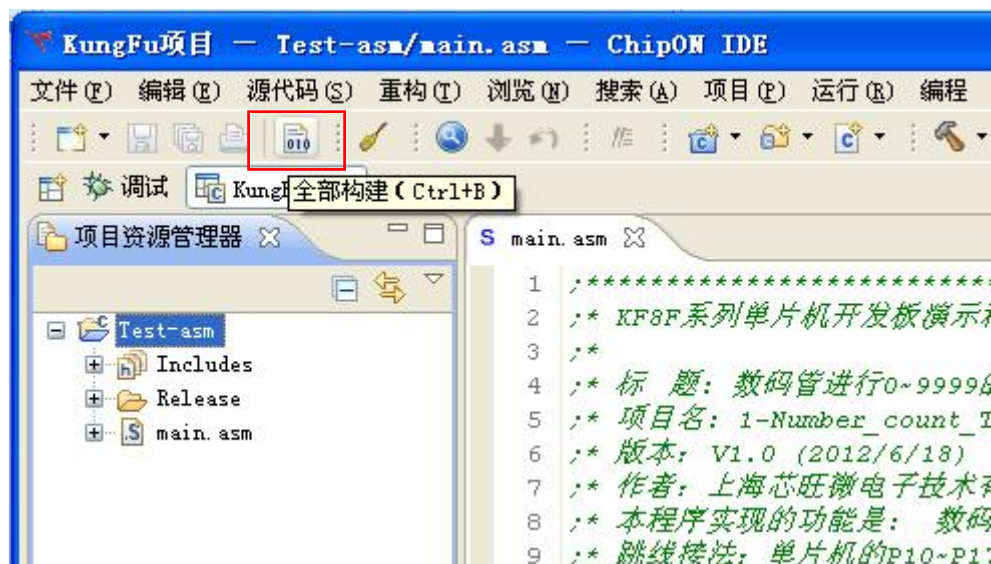
方法 3 点击项目下拉菜单中的构建全部, 或运用快捷键 CTRL+B, 此时工作



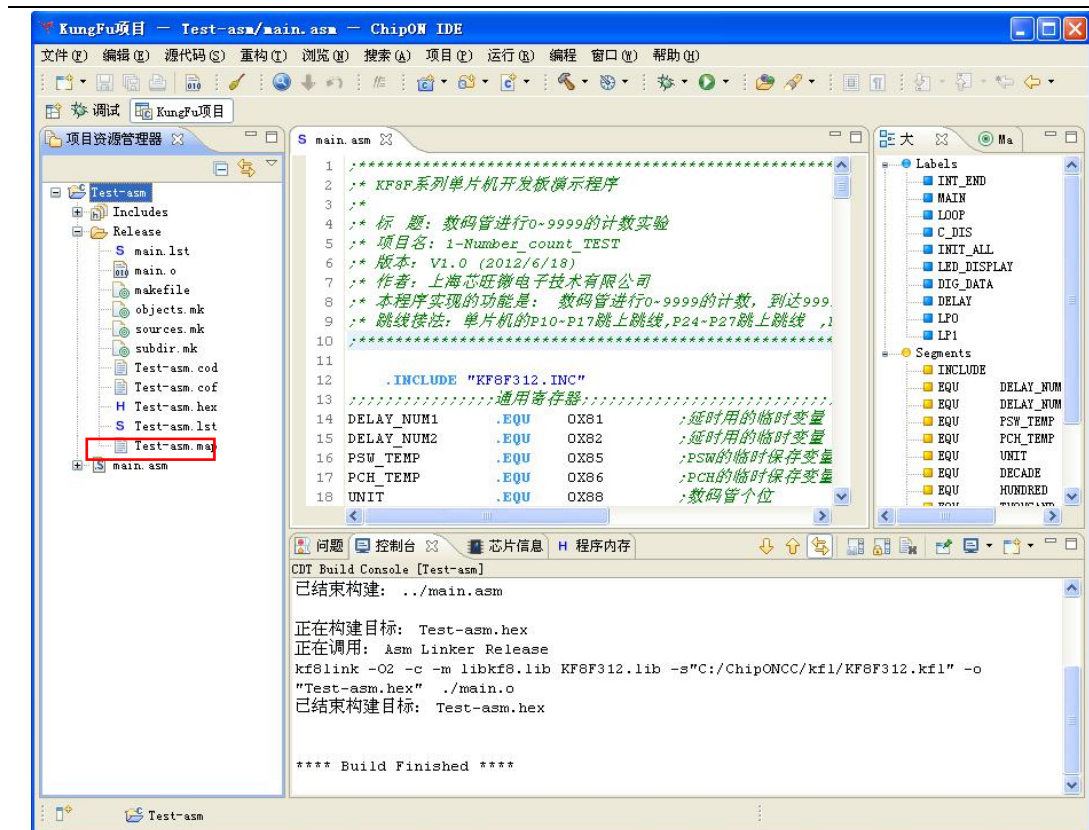
空间下的所有项目都被全部构建:



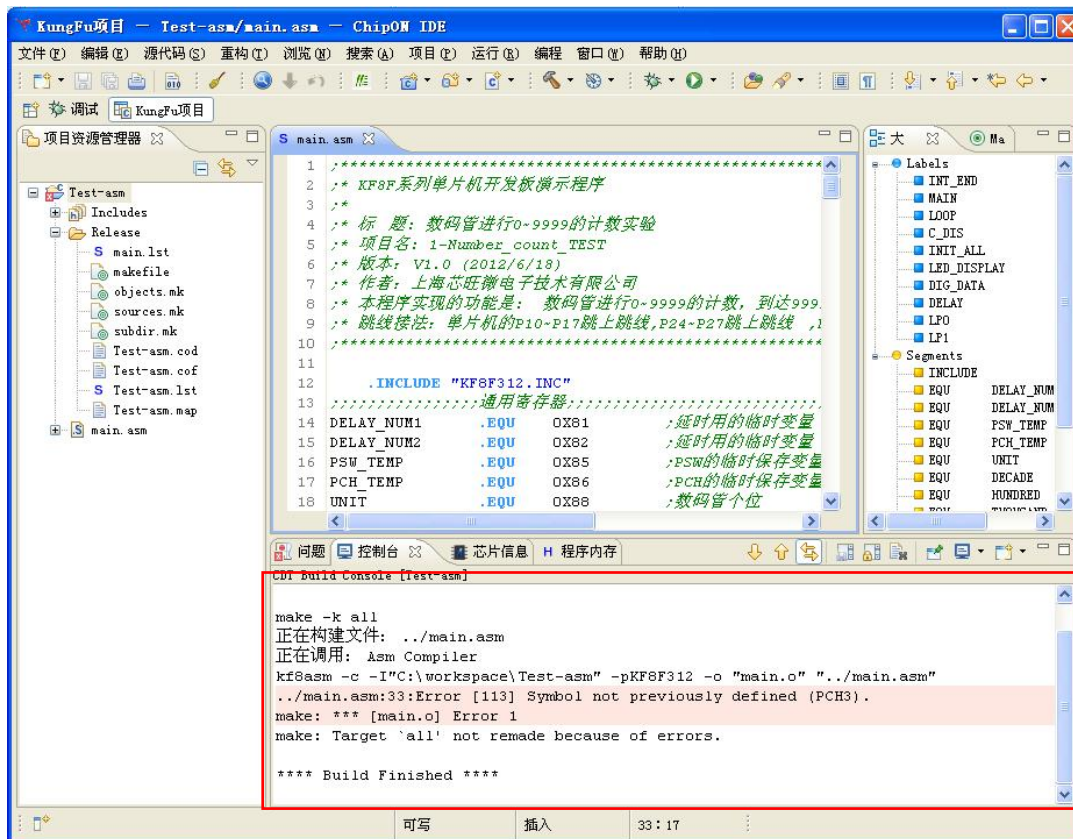
方法 4 点击快捷工具栏中的构建全部, 此时工作空间下的所有项目均被全部构建:



编译完成后, 在“Release”或“Debug”目录下, 生成有对应的 hex 文件, 并包括一个 hex 关联的 lst 文件以及芯片程序段和数据段使用情况的 map 文件。

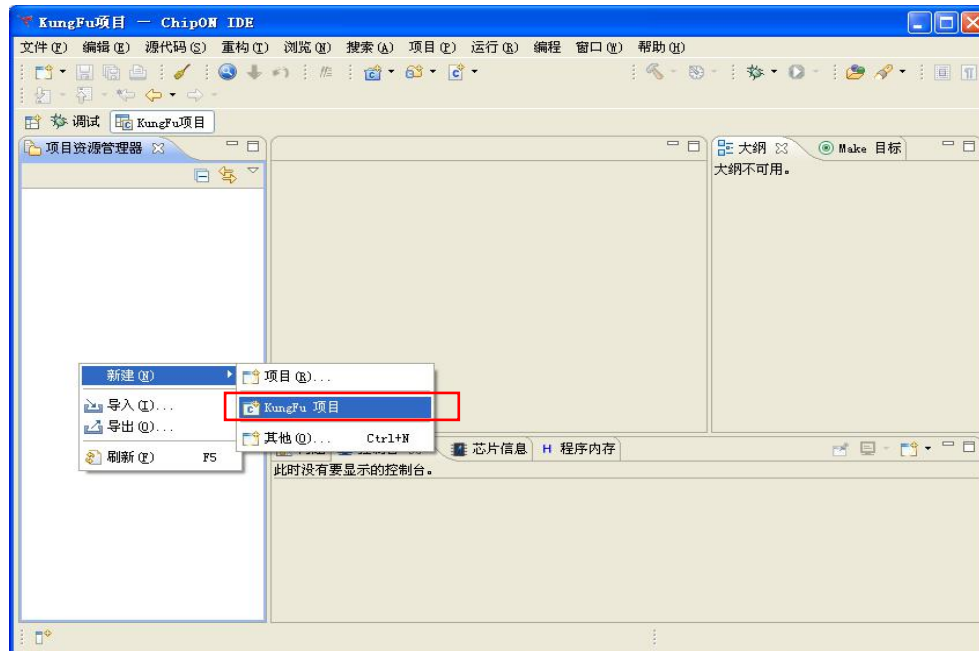


如果代码有错误,在控制台视图中将提示错误信息,但错误信息不局限于红色文件标志。



## 6. 新建 C 项目

在项目资源管理器空白处中点击右键，选择新建-->KungFu8 项目。也可以通过单击文件-->新建--> KungFu 项目 或工具栏中的新建功能进行选择。



设置项目名称，项目位置可使用缺省位置，也可以自定义位置，**建议使用缺省位置，即项目建立在当前的工作空间中；**

选择新建项目的类型与工具链，在项目类型中选择“KF8-c”，然后下一步：



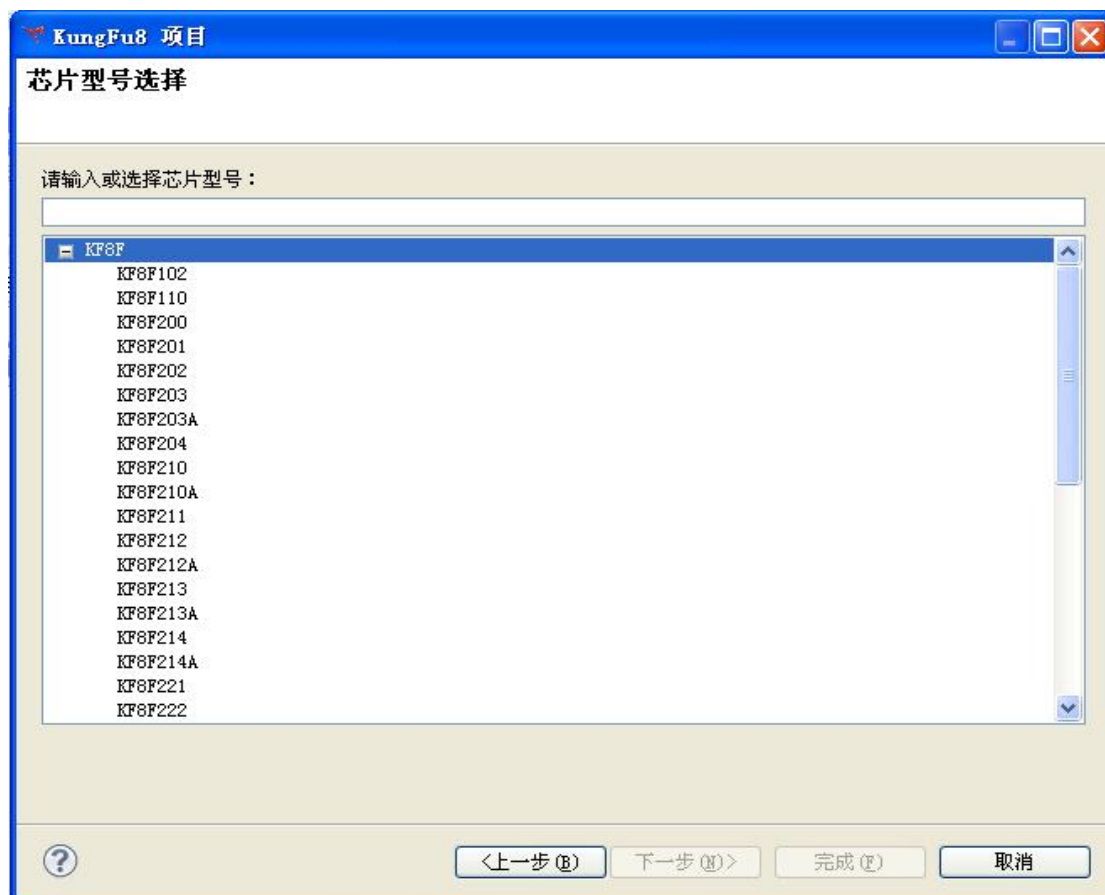


选择要部署的平台和配置，即调试模式 Debug 或正常运行模式 Release。一般按默认，不需修改。点击下一步：





选择项目所要用到的芯片型号，这里可以通过树列表从类型下选择型号，也可以通过输入进行快速过滤，如选择 KF8F312，在输入框内输入 312，并在下面列表过滤后的内容中选择对应的完整型号即可。



在选择型号后提供了调试电压值选择和辅助配置字的配置功能，可以选择在代码中设定，或者使用对话框进行配置。其中使用对话框配置情况下建立

“config\_set.c”的文件，内容如：

```
“ __sfr __at(0x2007) CONFIG =0x07d4;”
```

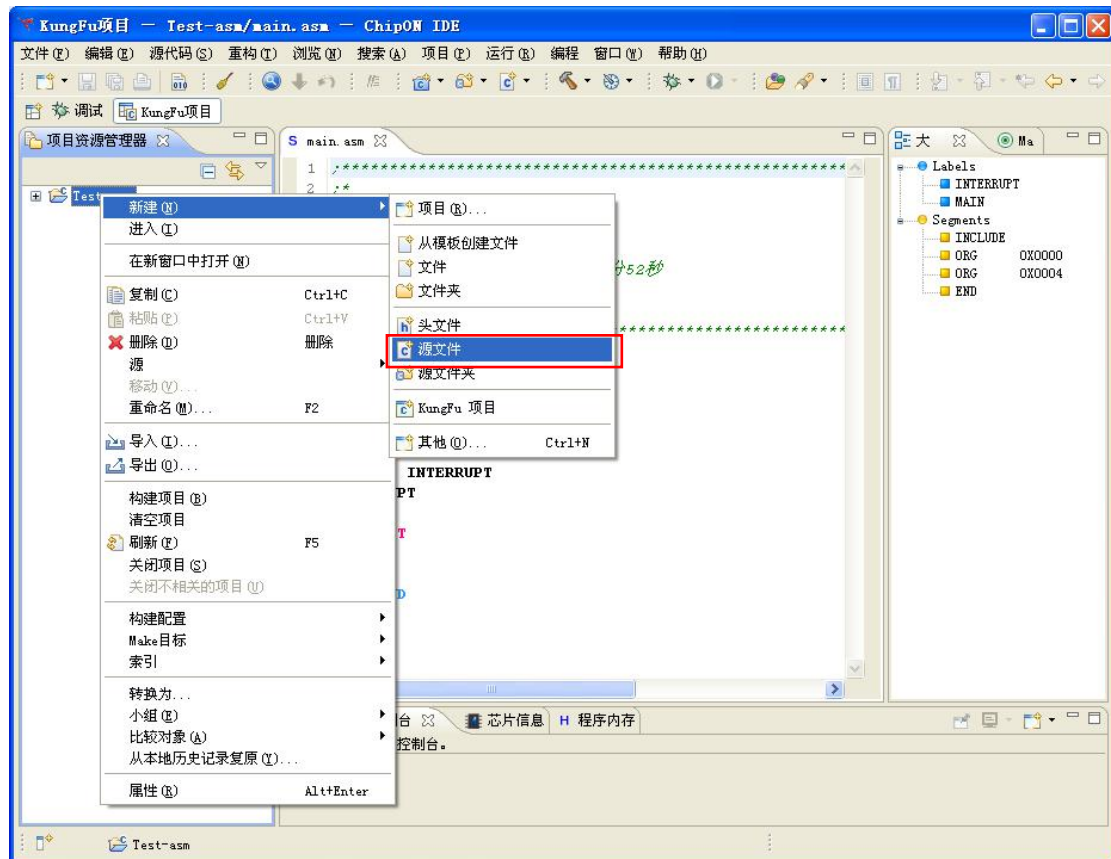
其中0x2007为配置字的映射地址，0x06FD为配置字结果。

至此，test-c 项目已创建完成, 提供了 main.c 的文件，建立了中断和 main 函数，可以在此基础上扩展开发项目代码。

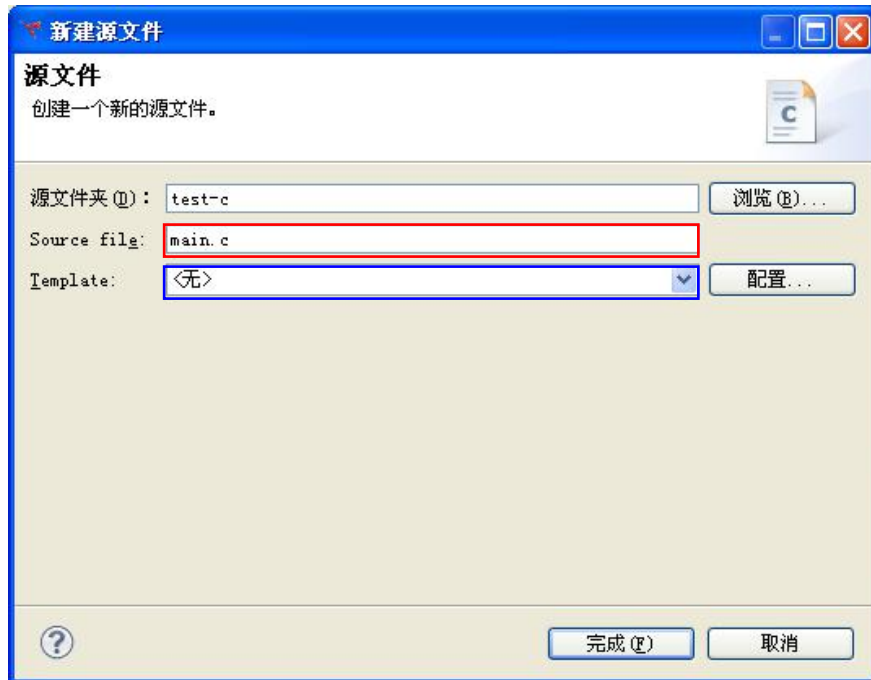
## 7. 新建 C 类型源文件代码与编译

### (1) 新建源文件

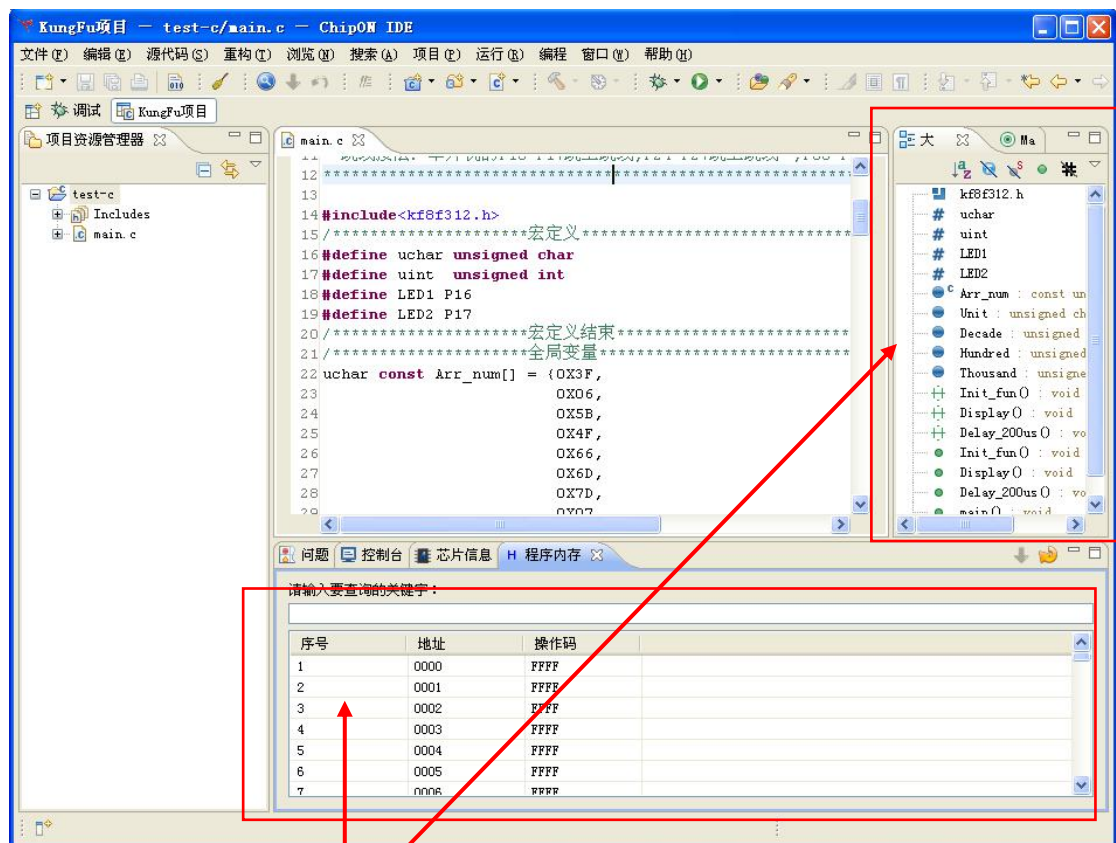
右键选择工程，新建-->源文件：



在新建源文件的页面中设置相应项。后缀文件应该为小写的 c，而不是大写。  
注意：红线圈住的地方应为源文件名字，蓝线圈住的地方应选择为无。



在编辑器视图中，向源文件中写入代码：



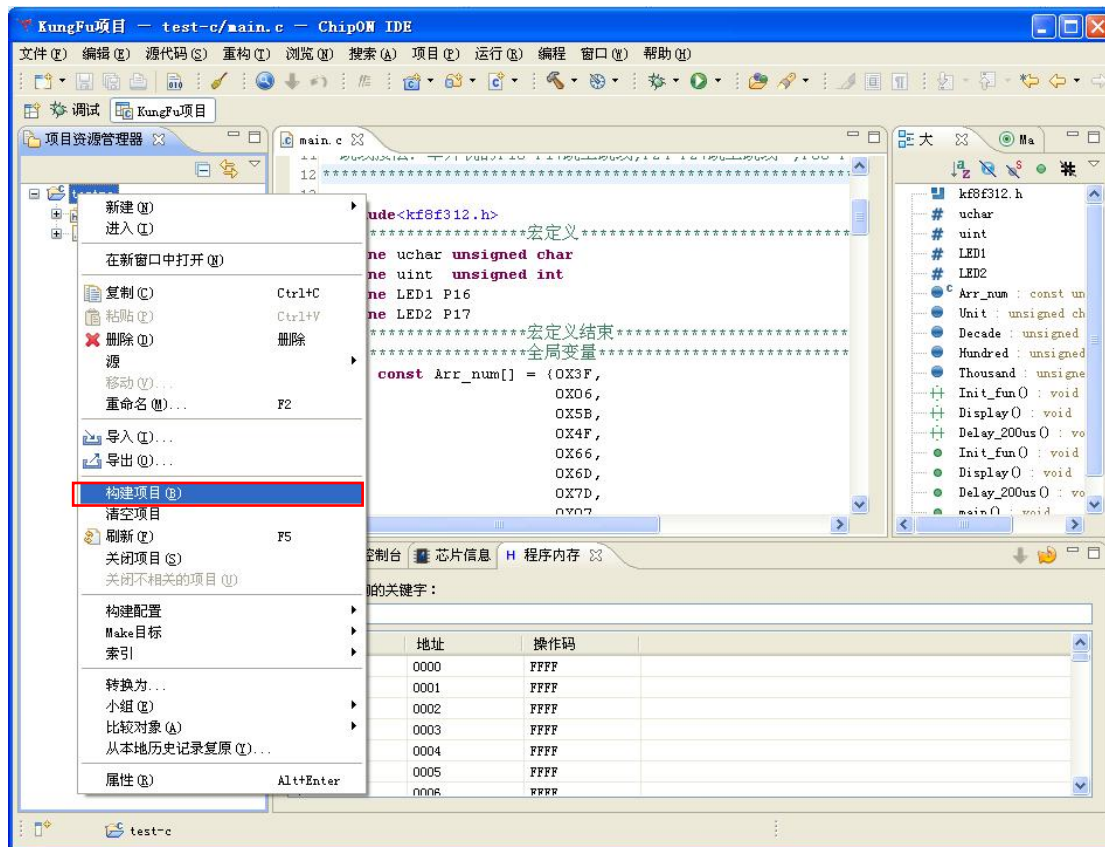
在编辑过程中，双击“芯片信息”视图中的对应寄存器名字，可直接将该寄存器名字添加到编辑器光标所在位置处。

通过“大纲”视图，可查看源代码中的宏定义等信息，可以方便代码的编写查找。

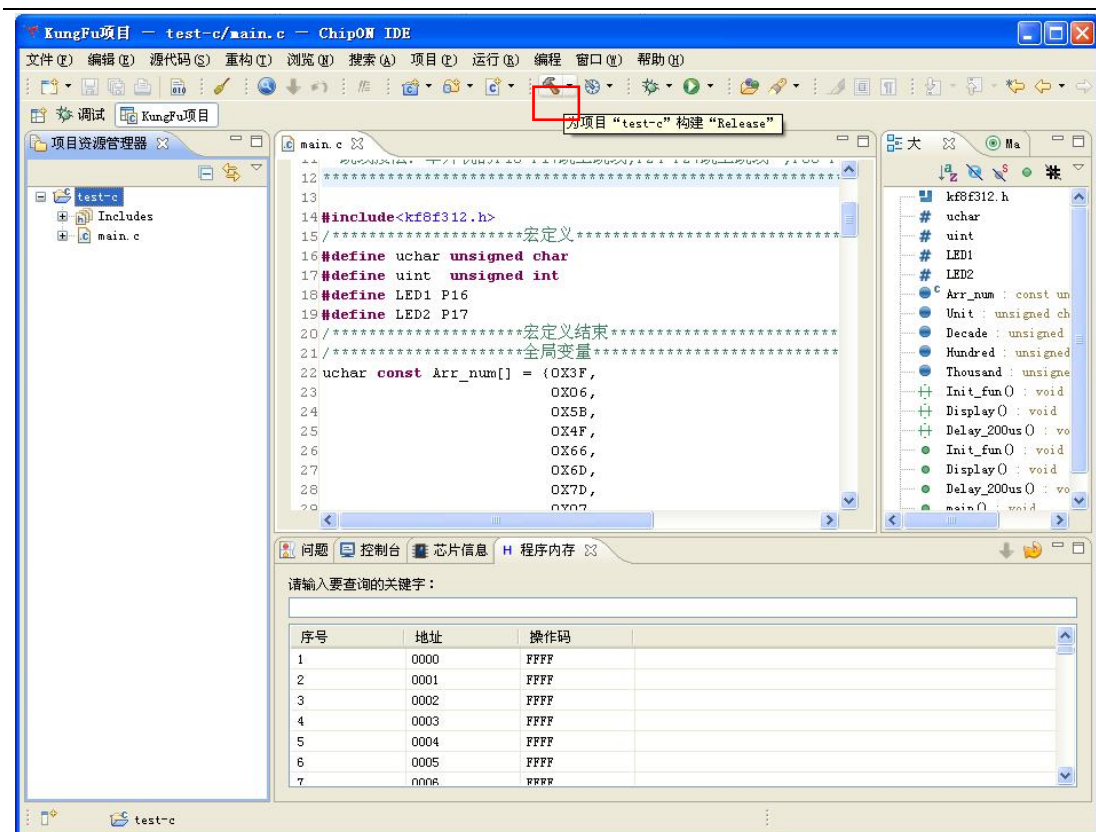
## (2) 编译源文件

注意：首先应选中项目，选择项目可以通过在项目资源管理器或对应的代码文件。

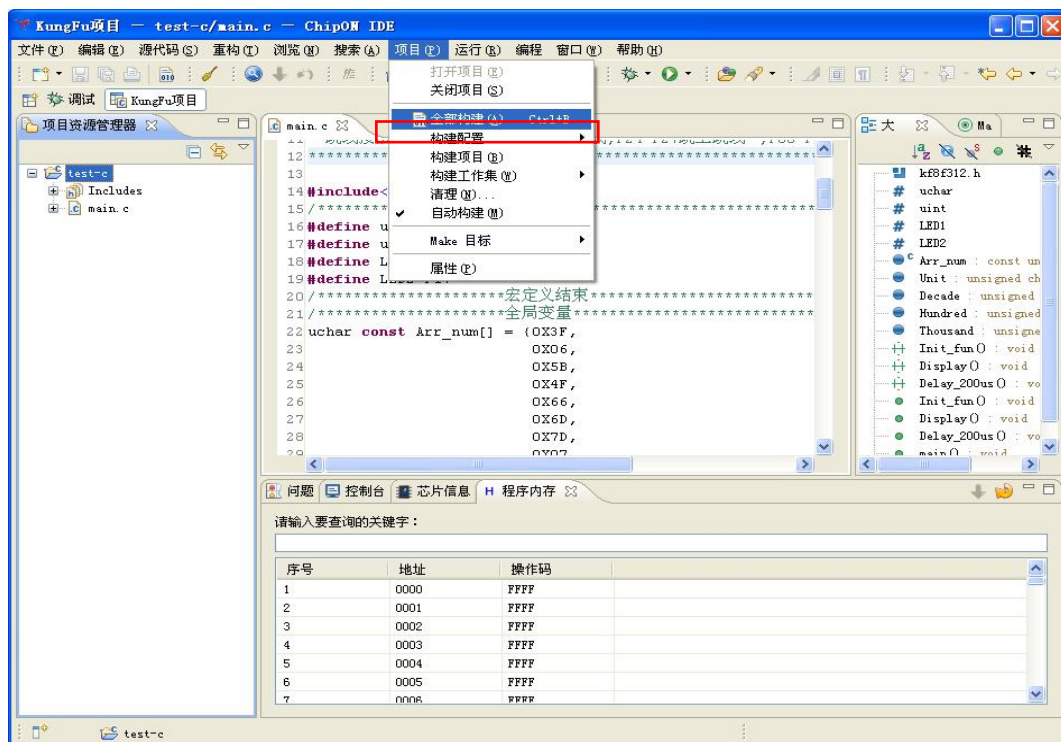
方法 1 右键点击项目，然后点击构建：



方法 2 通过快捷工具栏构建：

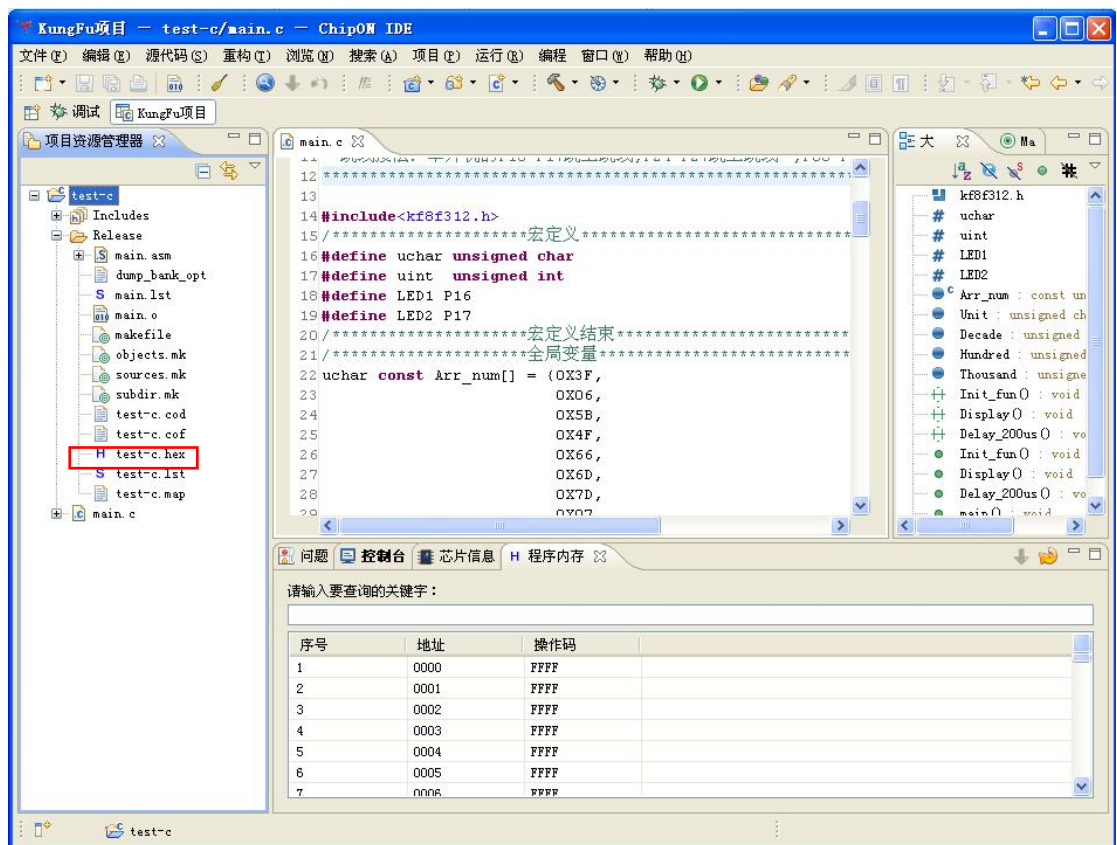
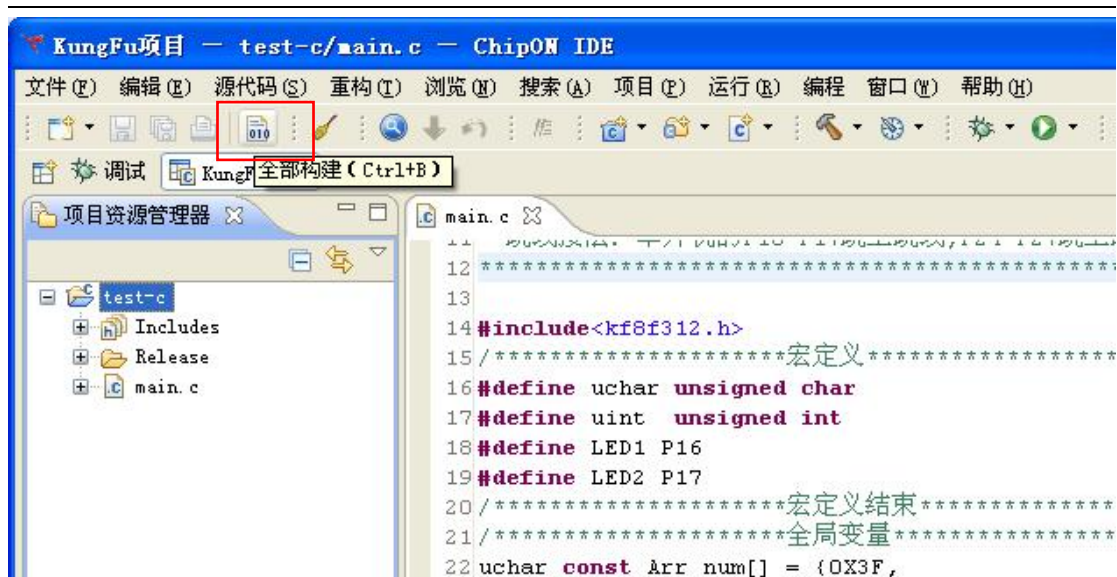


方法 3 点击项目下拉菜单中的构建全部，或运用快捷键 CTRL+B，此时工作空间下的所有项目都被全部构建：

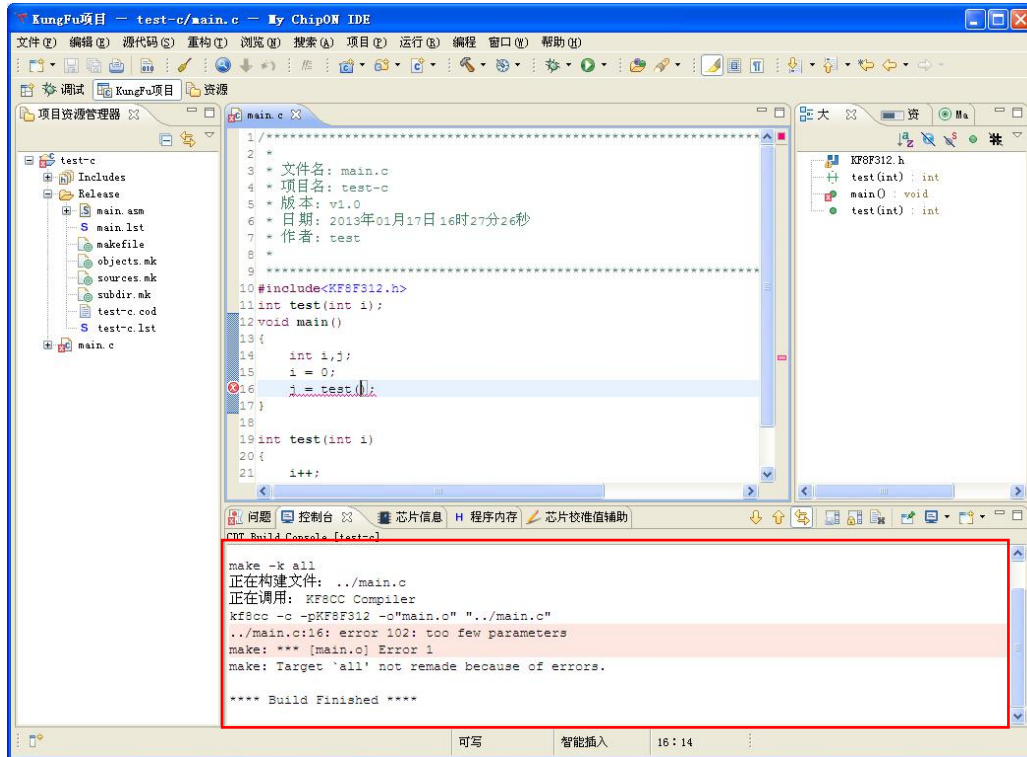


方法 4 点击快捷工具栏中的构建全部，此时工作空间下的所有项目均被全部构建：



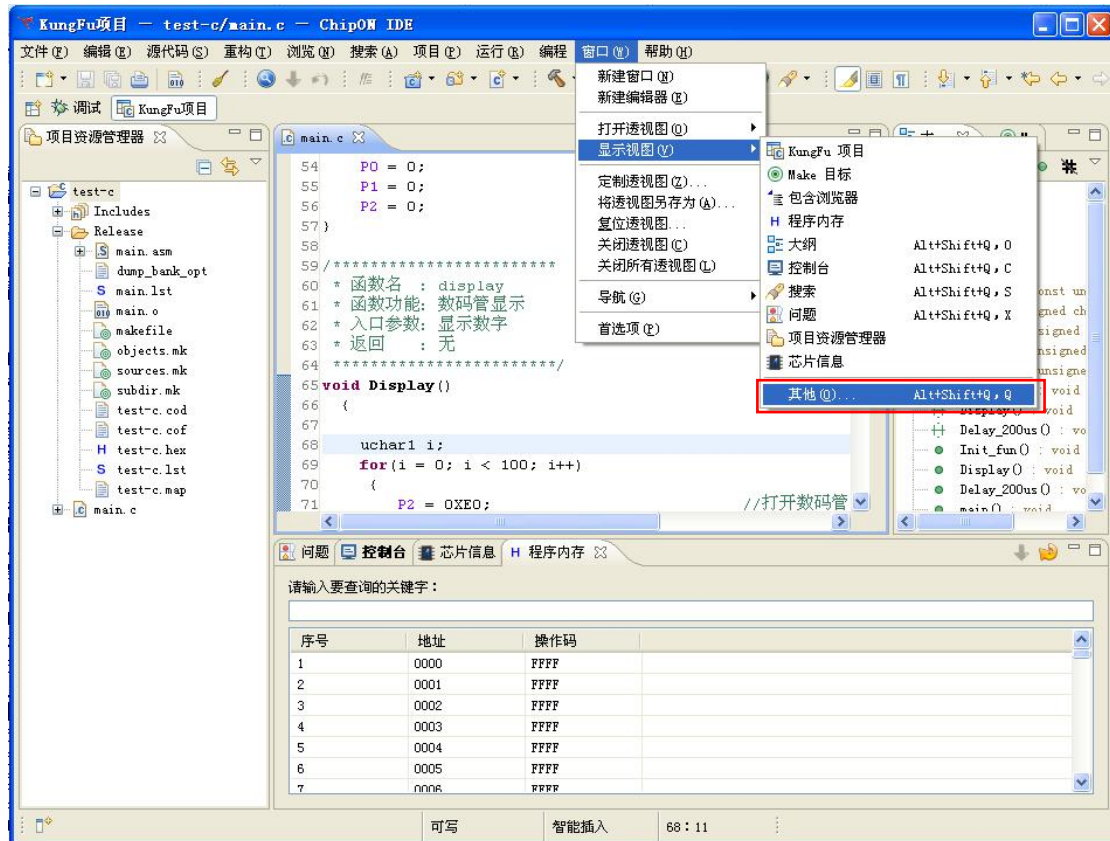


如果代码有错误，在控制台视图中将提示错误信息：



## 8. 查看芯片寄存器信息

选中项目，在窗口-->显示视图-->其他:

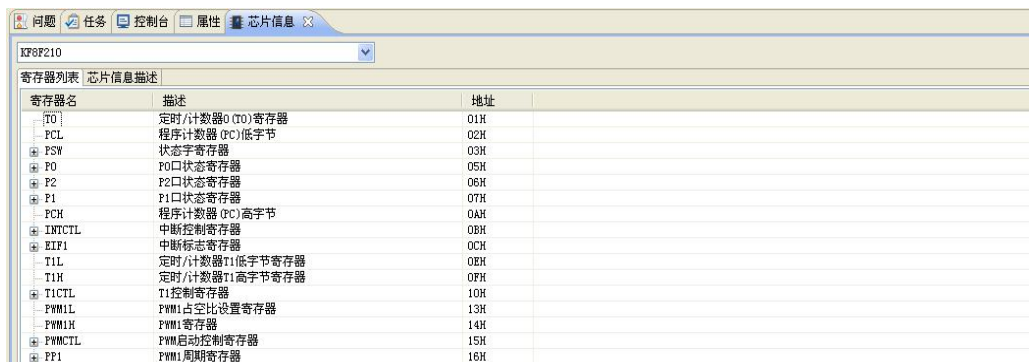


然后找到“ChipON 类别”，选择“芯片信息”，点击确定，打开视图:





在“芯片信息”视图中可以查看寄存器信息,在此显示的信息是该项目对应的芯片的寄存器列表和寄存器的位以及芯片信息描述,方便开发查询,更多寄存器信息以数据手册为准。

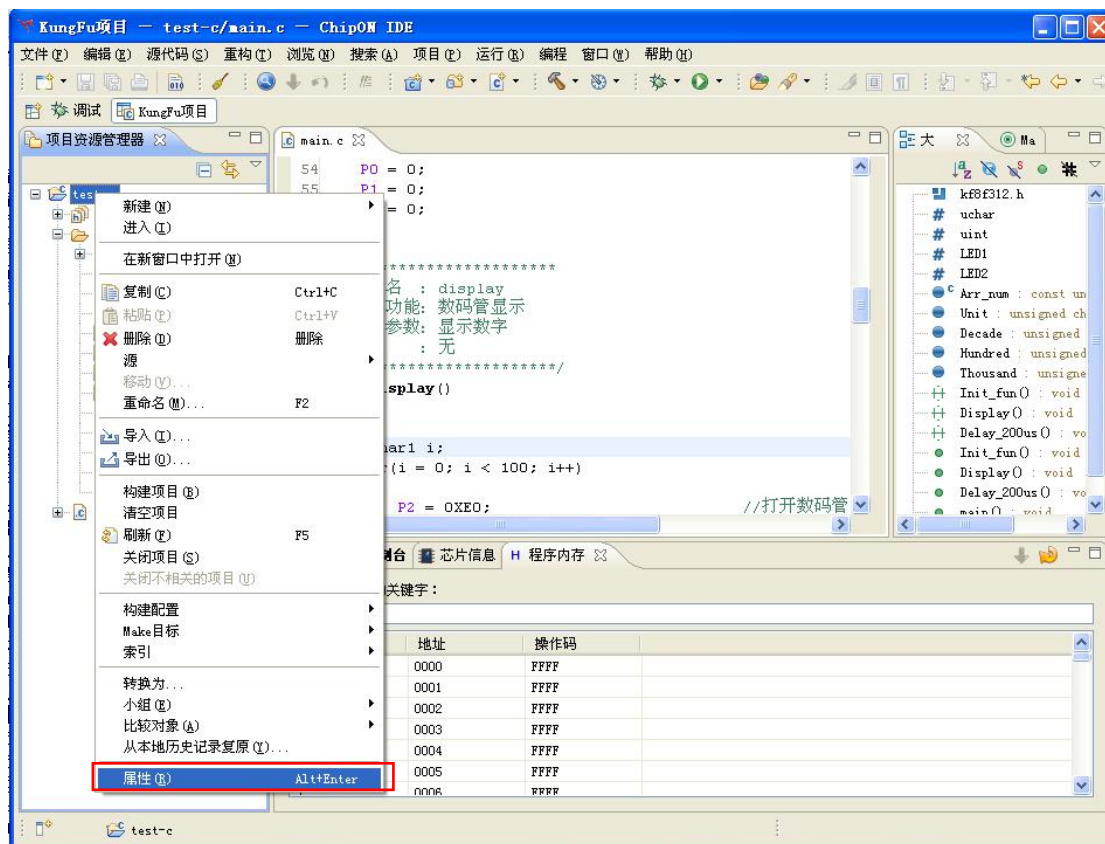


寄存器名	描述	地址
T0	定时/计数器0(T0)寄存器	01H
PCL	程序计数器(PC)低字节	02H
PSW	状态字寄存器	03H
P0	P0口状态寄存器	05H
P2	P2口状态寄存器	06H
P1	P1口状态寄存器	07H
PCH	程序计数器(PC)高字节	0AH
INTCTL	中断控制寄存器	0BH
EIF1	中断标志寄存器	0CH
T1L	定时/计数器T1低字节寄存器	0EH
T1H	定时/计数器T1高字节寄存器	0FH
T1CTL	T1控制寄存器	10H
PWM1L	PWM1占空比设置寄存器	13H
PWM1H	PWM1寄存器	14H
PWMCTL	PWM启动控制寄存器	15H
PP1	PWM1周期寄存器	16H

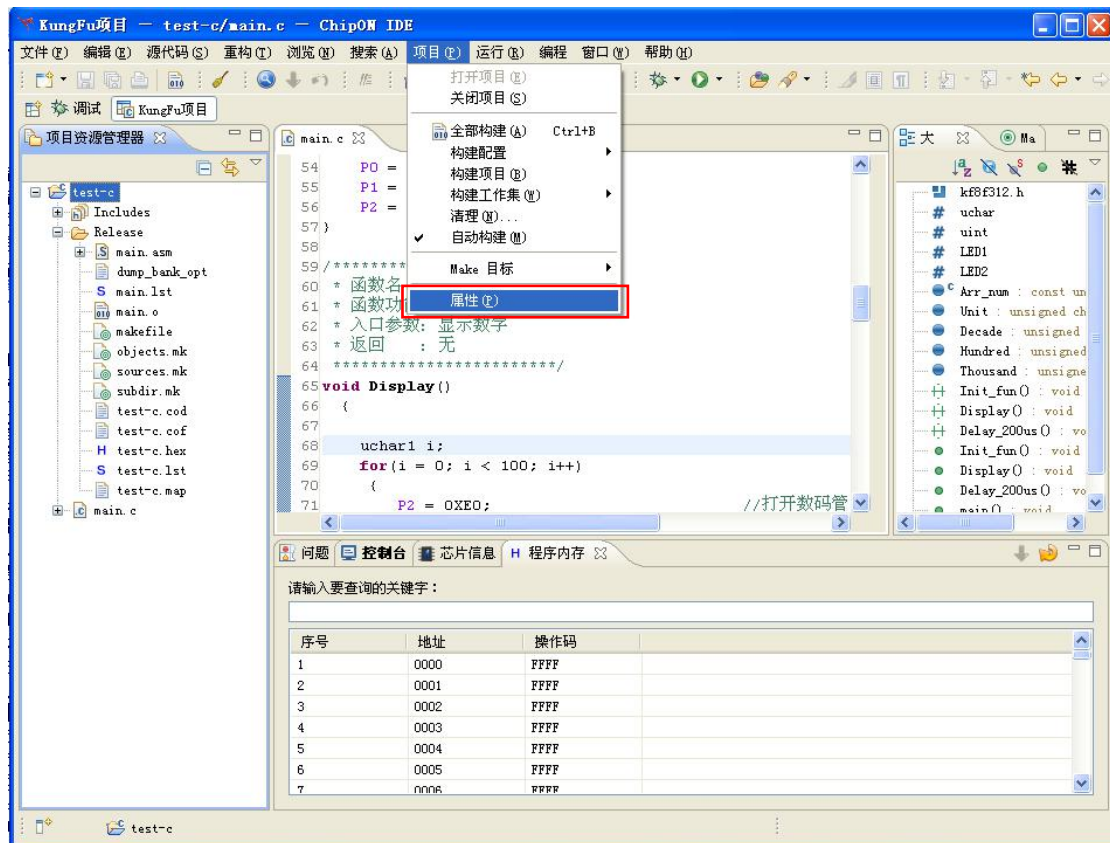
## 9. 芯片配置位配置

### (1) 打开配置页

方法 1 右键点击项目, 选择“属性”:

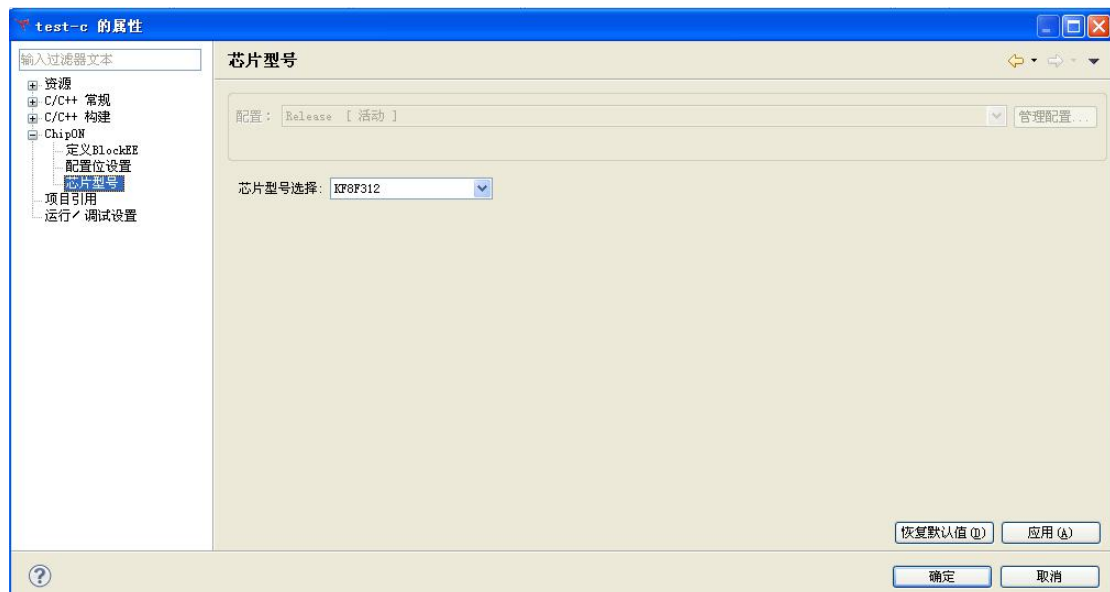


方法 2 选中项目后，通过菜单项 项目-->属性:



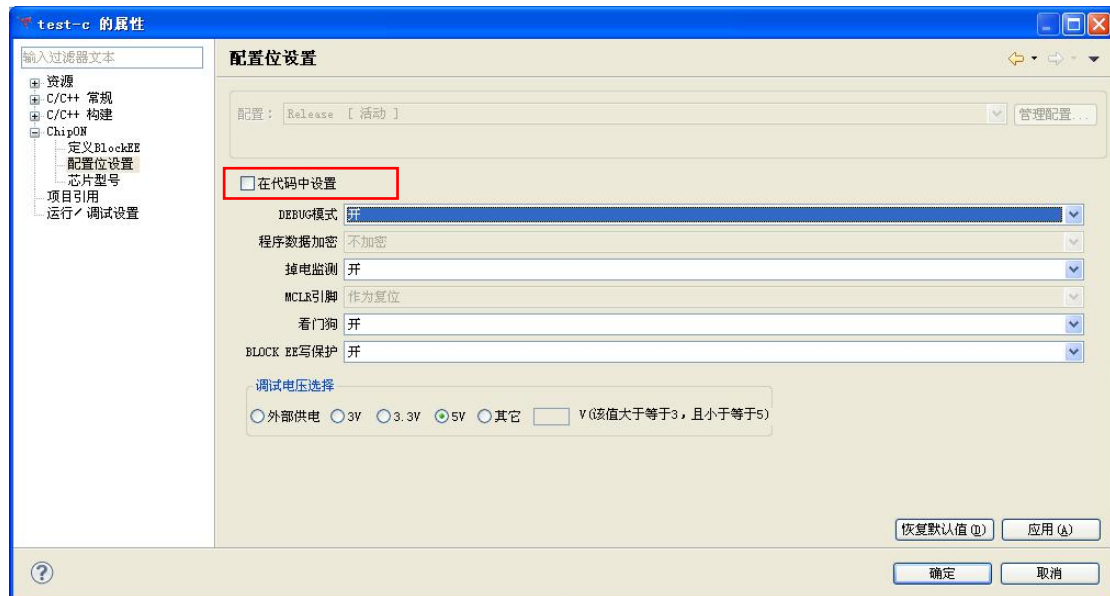
## (2) 设置芯片型号

选择 Chipon-->芯片型号，在此页面中更改芯片的型号，选择某一款芯片后点击“应用”按钮，然后“确定”即可。更改之后，整个项目所对应的芯片型号也随之更改。

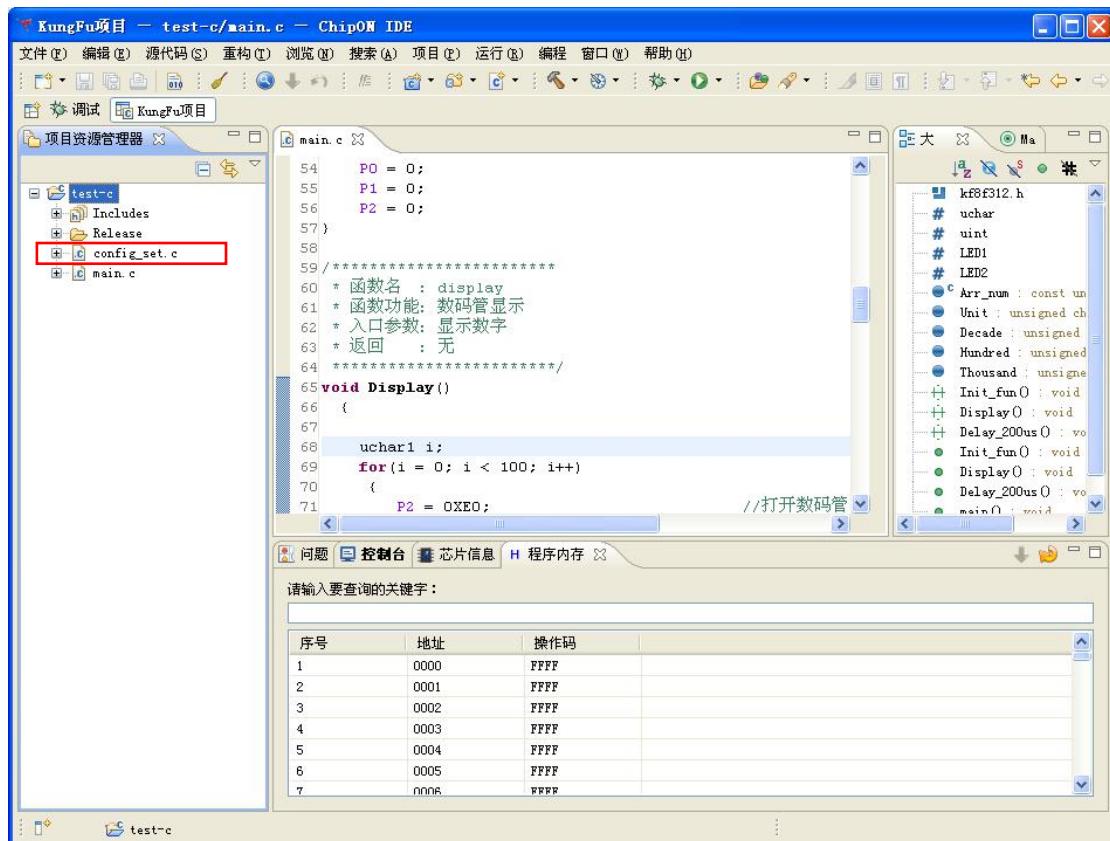


### (3) 配置位设置

选择 ChipOn-->配置位设置, 取消“在代码中设置”选项, 激活配置配设置, 根据需要设置所需项:



点击确定, 该项目完成配置位的设定, 同时在工程根目录下生成对应的配置字代码文件 config\_set.c 或 config\_set.asm。



当“在代码中设置”被选中，并应用到工程后，config\_set.c 或 config\_set.asm 文件将被删除。可以在资源使用率窗口下直接使用配置字的配置选项生成配置字，界面修改后需要点击“应用”按钮。

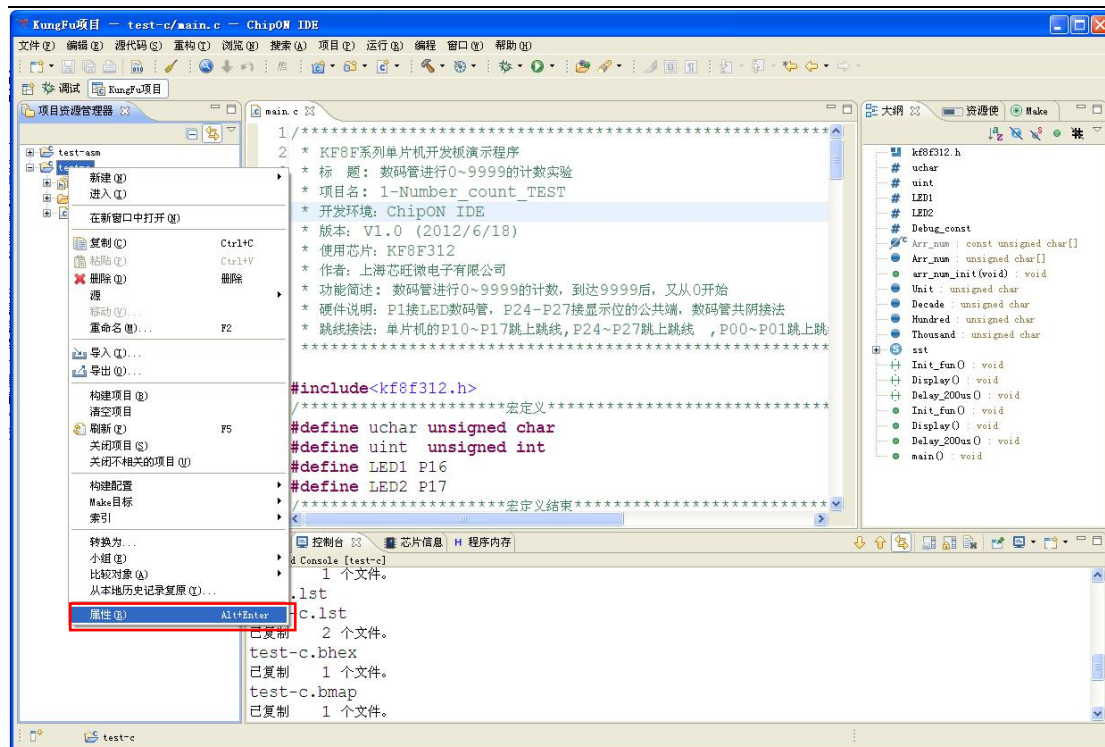


## 10. 芯片 BlockEE 配置

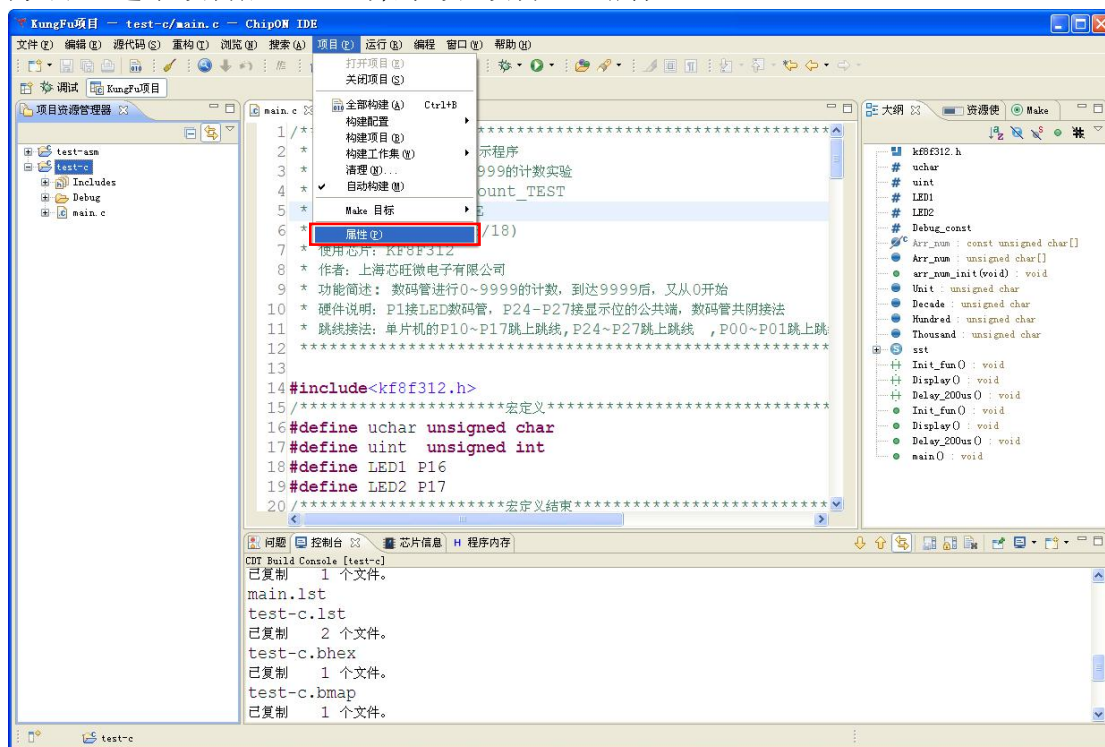
### (1) 打开配置页

方法 1 右键点击项目，选择“属性”：



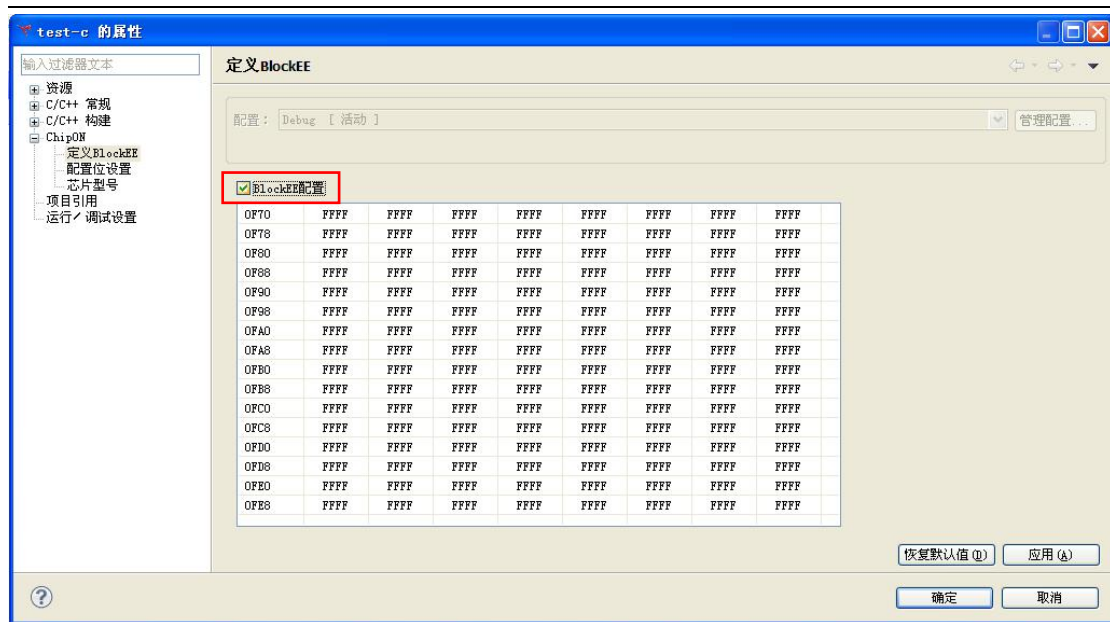


方法 2 选中项目后, 通过菜单项 项目-->属性:



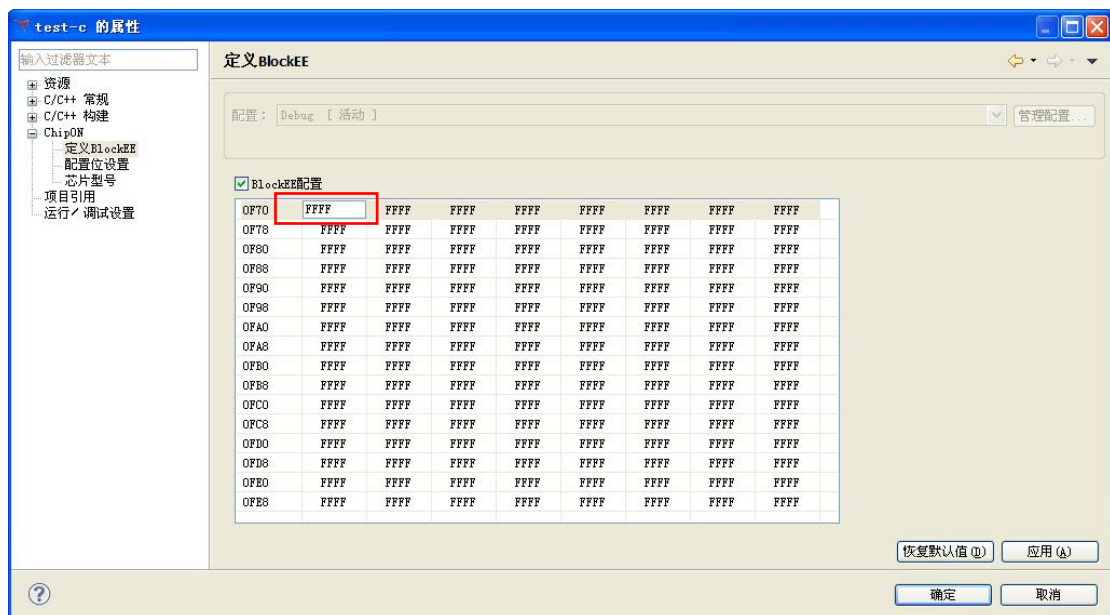
(2) 显示 BlockEE

选中“BlockEE 配置”选项, 在此页面中显示该芯片型号的 BlockEE 视图。

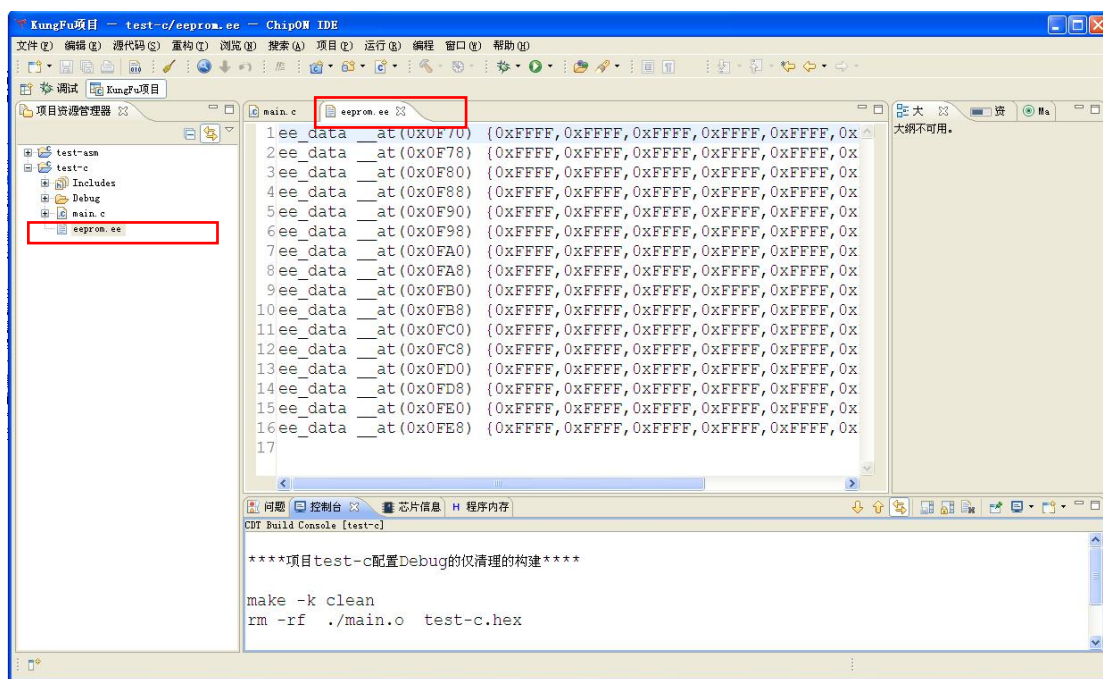


### (3) 修改 BlockEE 视图

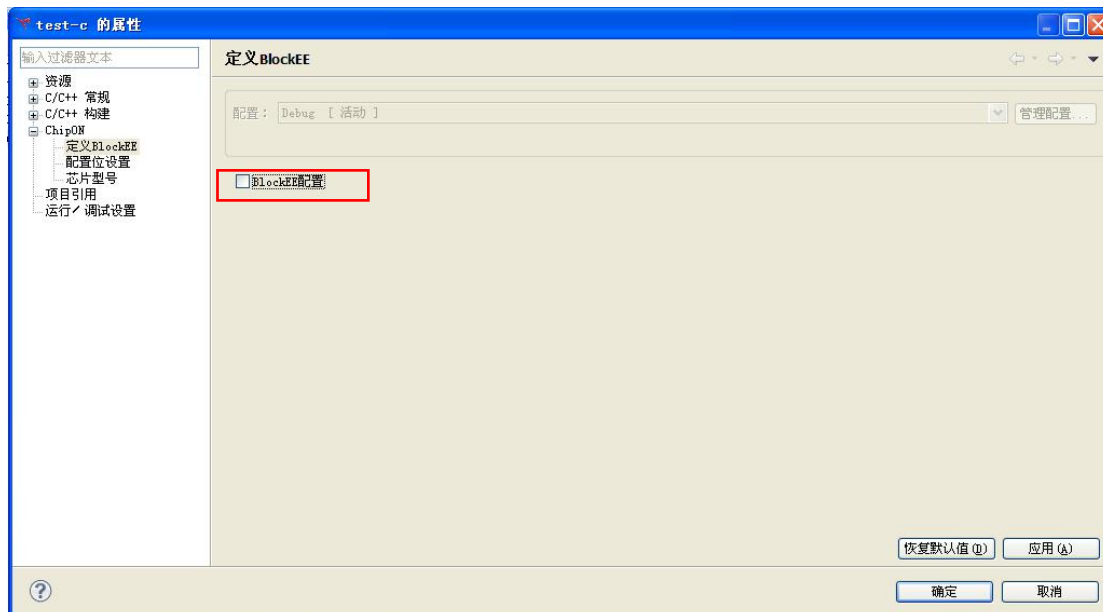
双击要修改的 BlockEE 视图中数据，数据修改完后点击“应用”按钮，然后“确定”即可。



点击确定，该项目完成 BlockEE 配置的设定，同时在工程根目录下产生 eeprom. ee 文件：



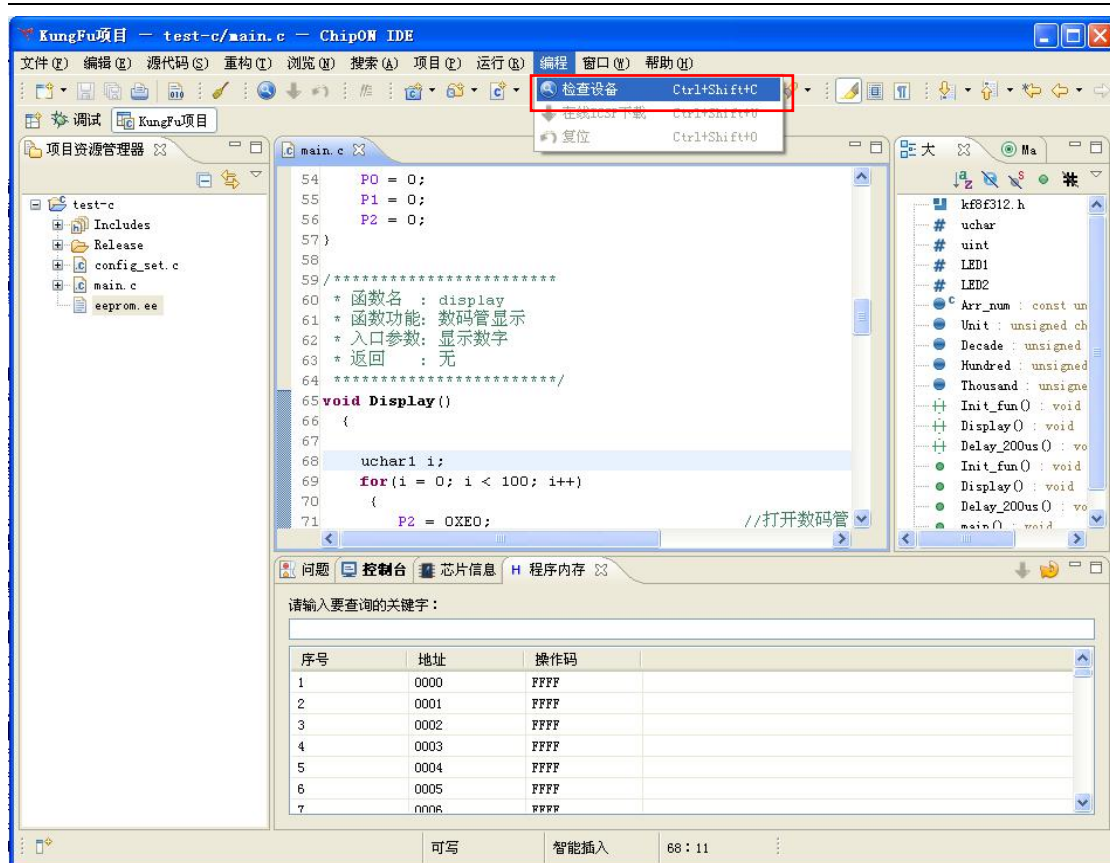
当“BlockEE 配置”选项被取消，并应用到工程后，eprom.ee 文件将被删除。



## 11. 下载 hex 文件到单片机

### (1) 检查设备

首次启动未识别到编程器时需要该操作，对硬件设备连接进行检查，菜单操作：编程-->检查设备：



或者点击快捷工具栏图标:

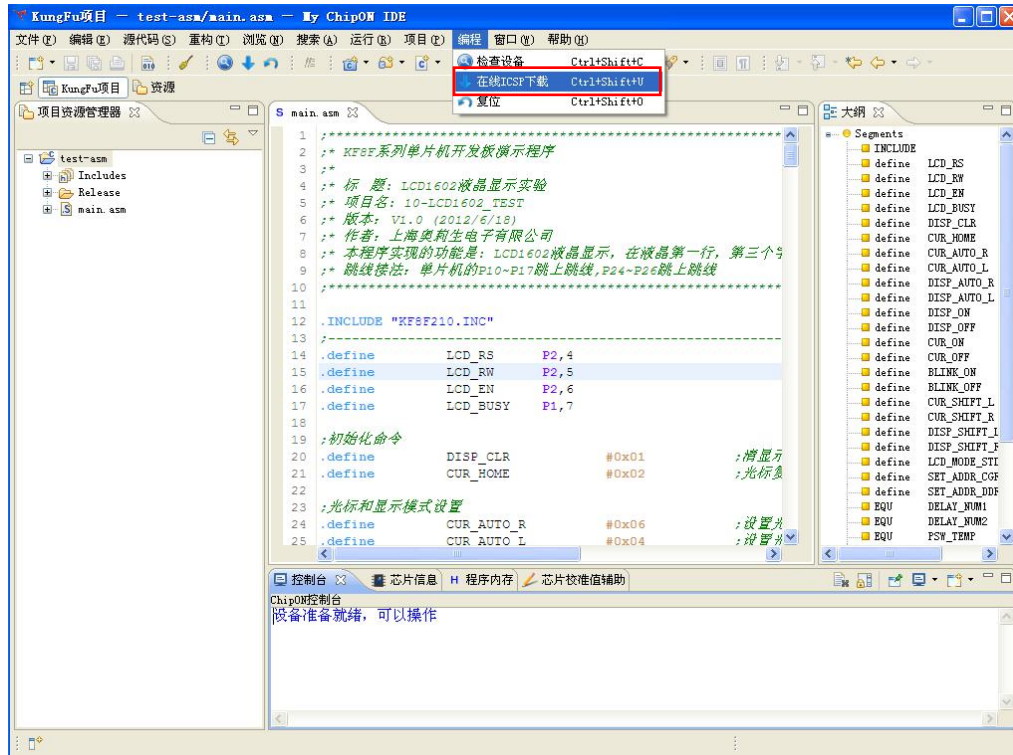


或者快捷键: Ctrl+Shift+c, 当检测到设备时菜单下下载或复位以及工具栏图标点亮。

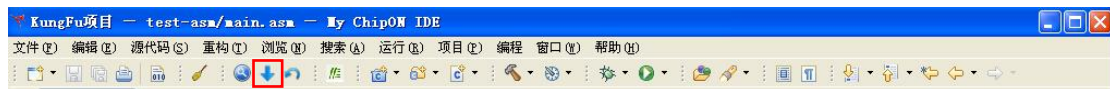
(2) 在线 ICSP 下载

检查设备成功后, 开始进行下载, 编程-->在线 ICSP 下载:





或者点击快捷工具栏图标:

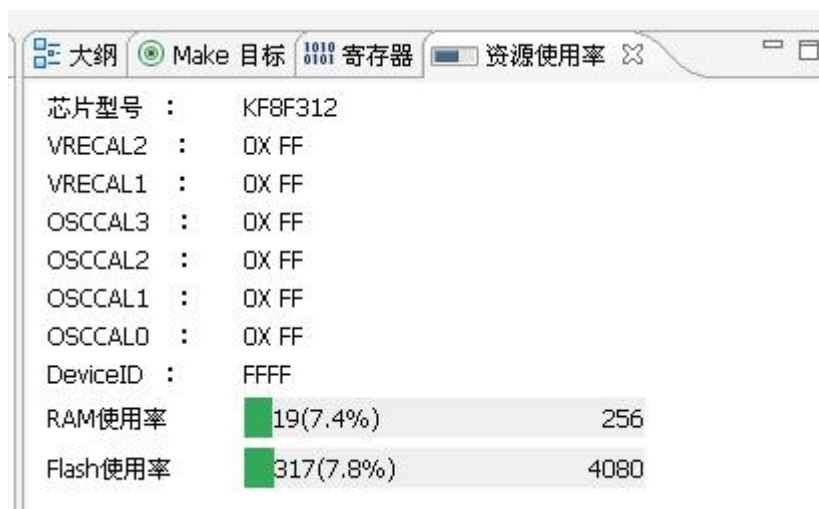


或快捷键: ctrl+shift+u

在“控制台”视图中显示下载信息：包括编程开始，编程成功或编程的失败信息。

## 12. 资源使用率

当在开发过程中，需要查看当前项目中对 Flash 的资源使用情况或 BlockEE 的资源使用情况时，可通过“资源使用率”视图进行查看，也可以在资源使用率视图中，设置配置位。



当使用率超过 95%时，使用率数值会以红色字体显示。

☐ 在代码中设置

程序数据加密	不加密
掉电监测	开
MODE引脚	作为复位
看门狗	开
BLOCK EE写保护	开

配置位已保存

应用

当配置位信息配置完成，点击应用按钮后，应用按钮旁边的字便由红色字体

变为蓝色字体，表示配置位信息已保存。

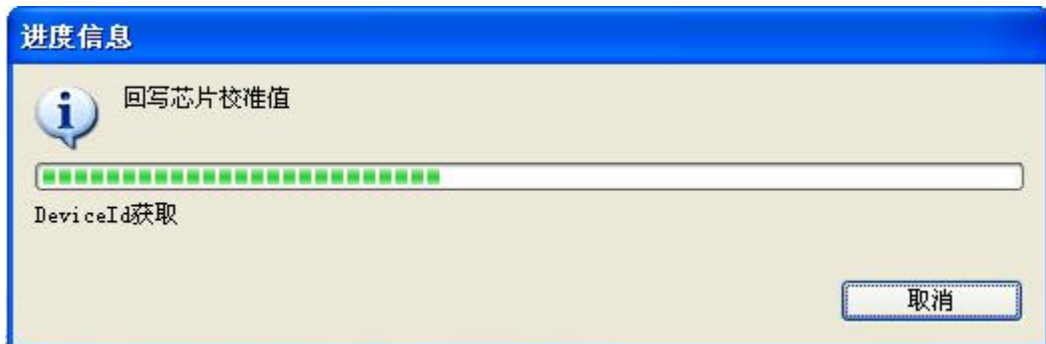
需要注意的是该页面同时负责了配置位的提供功能，当该视图不存在时新建项目会出现异常，可以通过窗口—》显示视图—》资源使用率或其他下选择 ChipON 类别下的资源使用率打开。也可以通过菜单的窗口—》复位透视图实现。透视图分调试和开发视图，这里应切换到开发视图下操作。

### 13. 回写校准值

当程序在下载的过程中发生中断时，如果提示“LD0 校准值和时钟校正值已丢失，请回写校准值或更换芯片”，请点击回写校准值按钮，将程序中保存的校准值写入芯片，如下图所示：



点击回写校准值按钮：



该功能已经添加到程序执行过程中，不需要额外操作，同时回写功能仅在读取到芯片有效信息后有效，当切换项目时，该信息会丢失，即原有校准值不在保持，也就是说不能支持回写功能。

## 14. 项目开发与调试

### 1、代码开发

在新建立的最新模板文件中进行代码的编写操作。可以建立不同的文件夹用来区分代码的用途。但针对电脑差异，该特性在不同的电脑上可能直接报编译器崩溃异常。此时需要将头文件和源文件放在项目名下的第一级路径下，删除定义的文件夹。

代码字体的设定可以查看 17 章节的改变编辑器字体实现。

不建议项目名称频繁改动，该操作涉及信息存贮一旦过程异常，会造成信息不一致，此时可能编译异常，或编译结果不能及时刷新，造成下载程序时识别不到存在 hex，可以在目录上执行刷新操作，或快捷键 F5。该异常建议新建立项目，然后直接复制原项目文件到新项目即可。

更多支持特性可以查看 17 章的技巧内容。

### 2、调试项目注意事项

#### (1) 调试配置

P0.0、P0.1 须保留给调试器使用，需要设置为数字输入口。MODE 脚需要使能，同时 P0.0、P0.1 不能作为程序功能使用，使用包括端口的操作，端口的 pwm 输出，端口的 AD 功能。

#### (2) 嵌入汇编

调试时可以嵌入汇编代码，但不能一个函数都为汇编代码。嵌入代码不能过程查看，会类似作为上一句 C 表达形式附加存在被过程运行。

如下：

```
void init_chip()
{
    __asm
    CALL    0XFFF
    MOV     OSCCAL0,R0
    CALL    0XFFE
    MOV     OSCCAL1,R0
    CALL    0XFFD
    MOV     VRECAL,R0
```

```
    __endasm;  
}  
需要修改为以下形式:  
void init_chip()  
{  
    volatile unsigned char i=0;  
    __asm  
        CALL    0XFFF  
        MOV     OSCCAL0,R0  
        CALL    0XFFE  
        MOV     OSCCAL1,R0  
        CALL    0XFFD  
        MOV     VRECAL,R0  
    __endasm;  
    i=1;  
}
```

原因: 嵌入汇编代码编译器产生的 coff 文件没有行号, 整个函数没有行号时单步进入后找不到当前行号。

(3) 嵌入汇编代码中不能有 CRET、IRET、RRET 之类的程序返回指令。单步跳过单步跳出会分析机器代码, 一个函数只能有一个返回指令。因为一个函数的退出遍历代码找寻到 CRET、RRET、IRET, 并在这里打断点使运行出函数, 提前的返回可能条件不成立, 退出函数意图操作会失败。

(4) 芯片调试时会额外追加监控代码和部分变量的开销, RAM 占用小于 16 字节, 程序地址占用小于 250 字。调试变量地址固定, 不能和程序显示定义的地址冲突。

(5) 本软件支持位结构体和联合体, 但针对结构体和联合体识别度不够友好, 如调试界面的变量值查看错位不能实现监控, 同时不建议结构体、联合体对象参与复杂运算, 否则可能产生错位的影响。

(6) 不建议设计项目使用指针以及浮点, 这两类的代码编译效率较低, 会占用较多的程序空间。

(7) 调试时不能使用 IDLE 指令, 即无法仿真睡眠情况。

(8) 调试时不能使用看门狗。

(9) 函数不能写为静态函数。

- (10) 项目名中不能存在特殊符号,如括号。项目空间不建议路径太深,同时不建议过长的名称。
- (11) 要调试的项目不支持定义静态函数。
- (12) 调试仅支持硬件在线仿真,当编程器连接缺失时,默认不执行调试及调试视图调整。

### 3、调试设置

选择窗口-->首选项菜单,在首选项中选择运行/调试-->调试设置,在右侧页面中选择调试前下载设置,在文本框中填入自动单步间隔时间值,如下图所示:



自动下载: 表示项目在进入调试模式前将会先下载项目到芯片中

手动下载: 表示项目直接进入调试模式

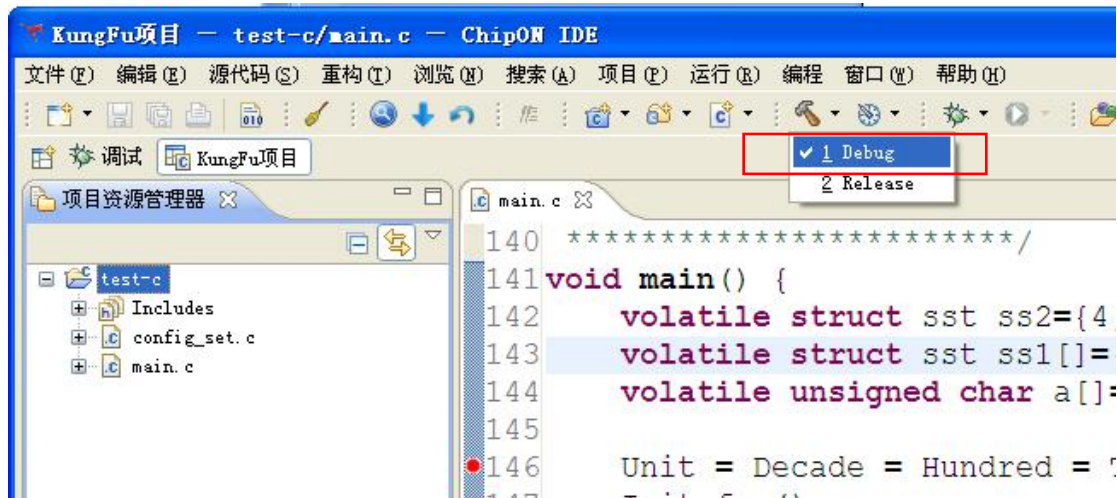
自动单步间隔时间表示项目完成一个单步后等待的时间



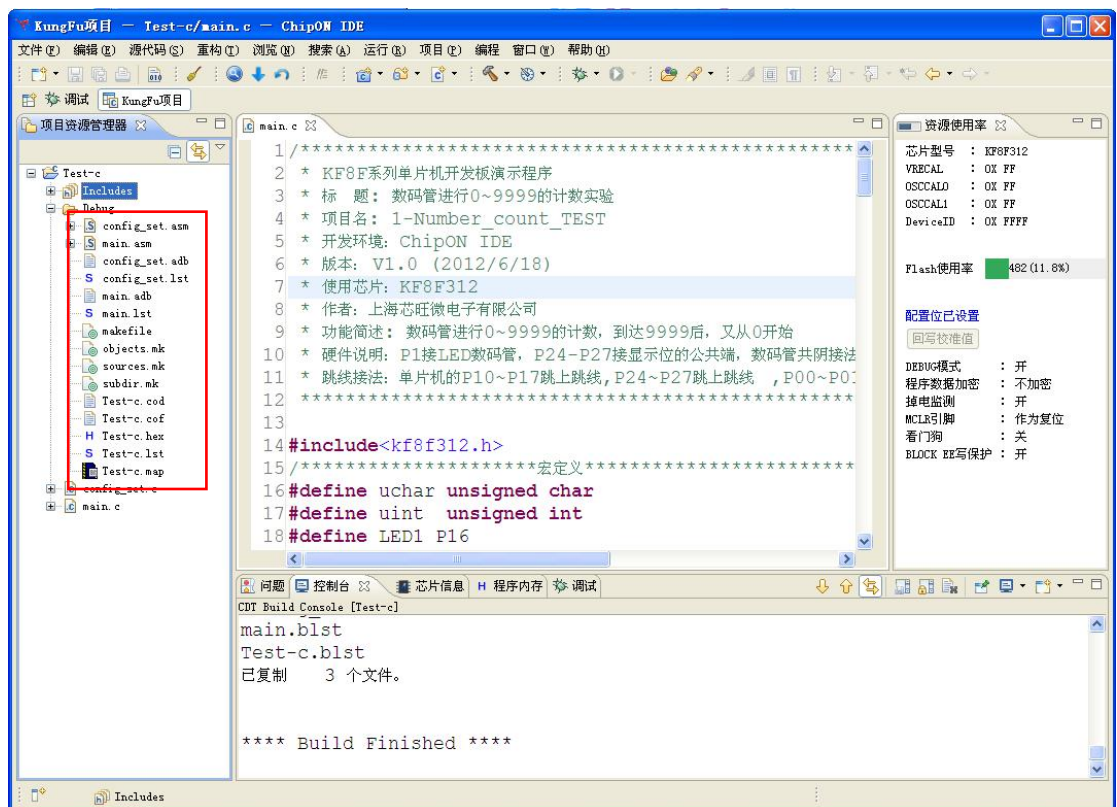
#### 4、编译项目

注意：首先应选中项目，并将编译模式选择为 Debug 模式。

通过快捷工具栏构建：



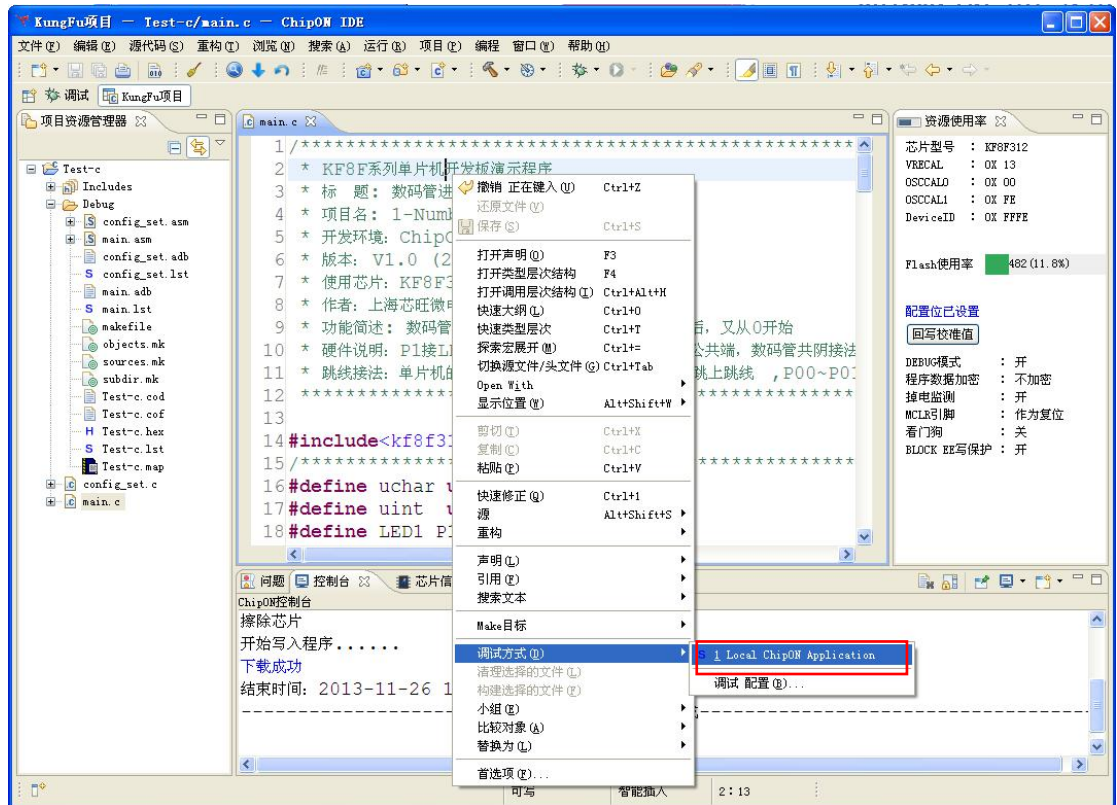
编译完成后，新增“Debug”目录和其目录下的文件



#### 5、下载 hex 文件到单片机

当选择手动下载模式时需要单独执行程序下载。选择自动模式时会在启动调试过程自动完成程序的下载。进入调试方法：在具有主函数入口的文件

上, 点击鼠标右键, 选择“调试方式(D)”, 选择本地调试。随后工具执行程序的下载(自动下载模式)和调试功能启动。



控制台视图显示过程的信息, 包括启动信息和异常信息。

## 6、设置断点

双击编辑器左侧栏即可添加断点或删除断点, 当前仅支持 1 个断点的设定, 同时断点信息作为项目信息存贮, 但可能存在不一致现象, 此时需要断点的重新设定。

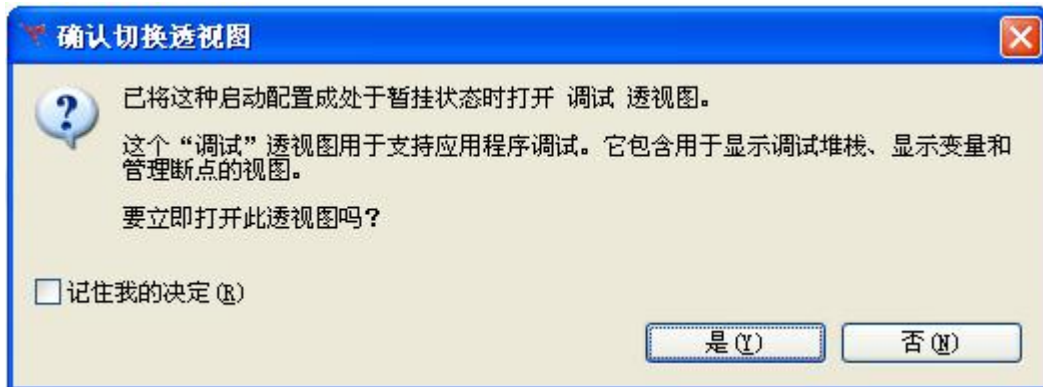
```

144     volatile unsigned char a[]={0,1,2,3,4,5,6};
145
146     Unit = Decade = Hundred = Thousand = 0;
147     Init_fun();
148
149     while (1) {
150
151         Unit++; //个位加1
    
```

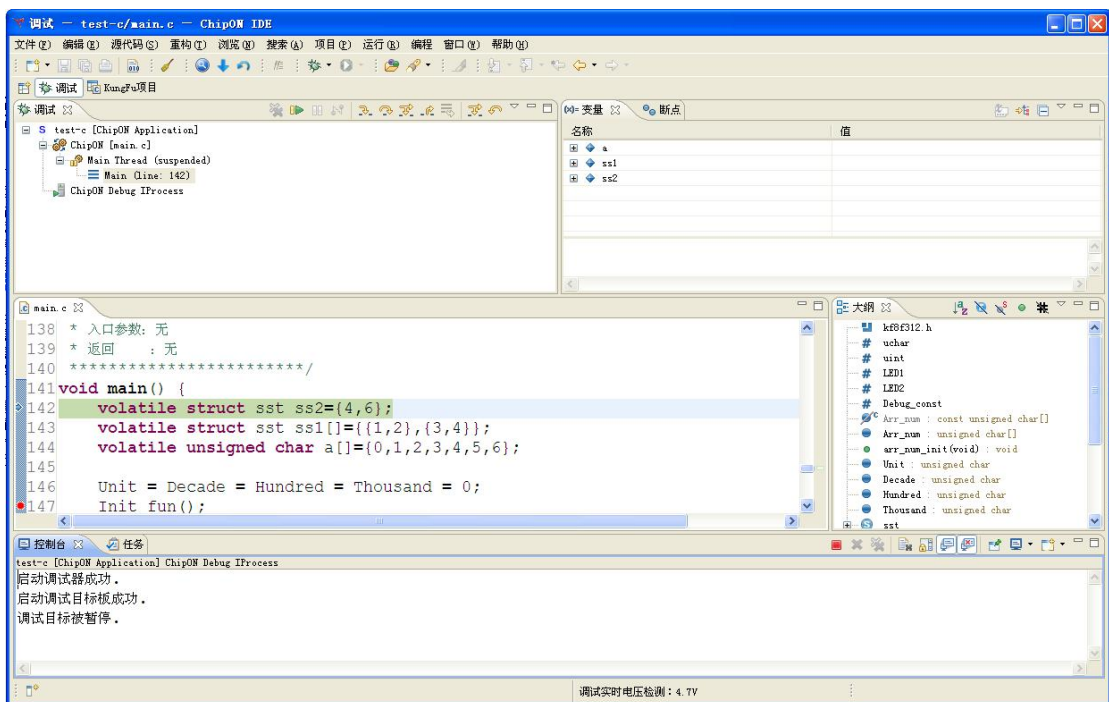
## 7、进入调试



开始运行程序后, 进入调试模式前弹出提示框, 选择是否进入调试透视图, 如图, 可以默认决定或点击是进入调试视图。

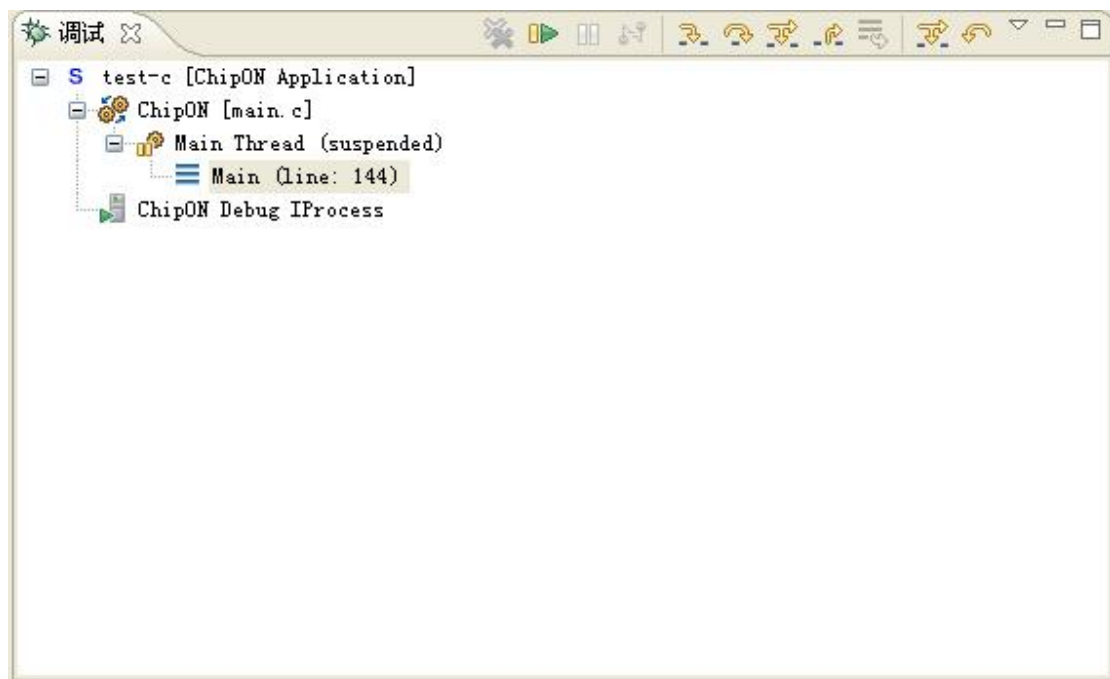



## 8、调试视图。





调试项目中所有的有关调试的视图:

1) 调试视图: 显示项目运行时的线程信息



: 去除所有终止线程

: 继续

: 暂挂

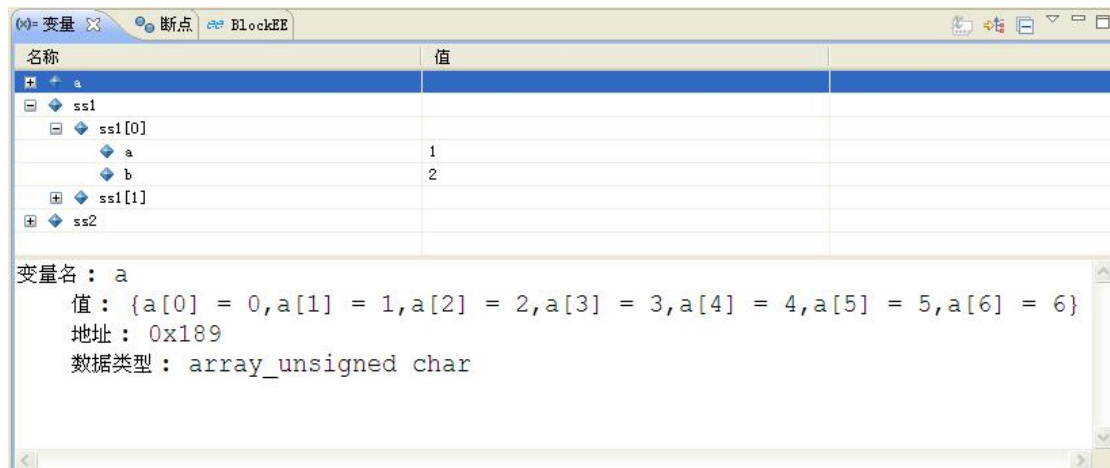
: 单步跳入, 单步跳过, 自动单步, 单步返回

: 复位

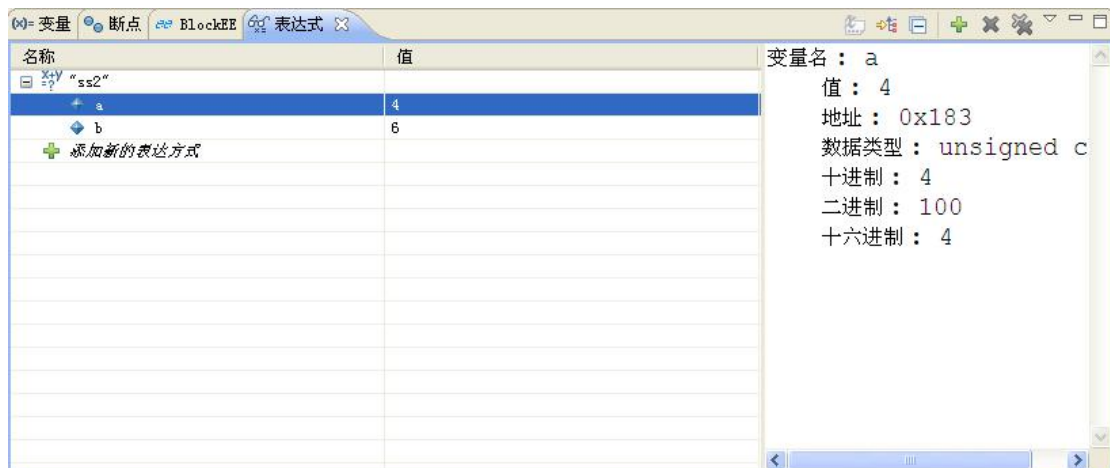
2) BlockEE 视图: 显示芯片中的 BlockEE 信息, 不可修改, 需要选择打开。

Address	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ASCII
F7	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	□□□□...
F8	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	□□□□...
F9	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	□□□□...
FA	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	□□□□...
FB	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	□□□□...
FC	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	□□□□...
FD	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	□□□□...
FE	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	□□□□...

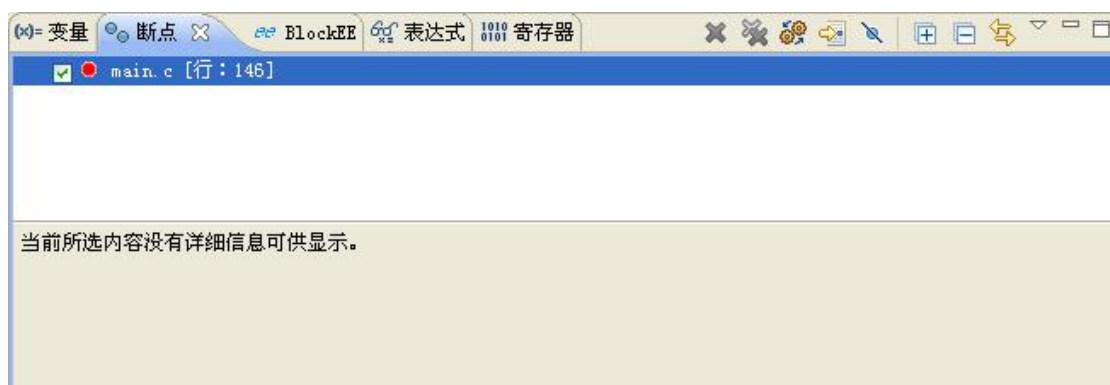
3) 变量视图: 显示项目中的局部变量, 值可修改。针对变量不能通过操作进行添加, 自动识别当前代码域中的变量并显示结果。该功能具有一定的局限性, 识别情况不够良好。



4) 表达式视图: 显示用户想看的任何变量或表达式的值, 重要的监控视图, 默认情况未打开, 需要手动打开, 或在代码中选择变量后通过右键菜单选择添加到表达式。



5) 断点视图: 管理项目中所有的断点。双击断点内容可快速定位所在代码位置。



6) 寄存器视图: 显示芯片中寄存器的信息, 除特殊地址外可以修改。

寄存器名	描述	地址	值
sfr TO	定时/计数器0 (T0)寄存器	01H	00
sfr PCL	程序计数器 (PC)低字节	02H	51
sfr PSW	状态字寄存器	03H	18
bit RPO	通用寄存器区选择位	5	0
TO	超时标志位	4	1
bit PD	上电复位标志位	3	1
bit Z	零标志位	2	0
bit DC	辅助进/借位标志位	1	0
bit CY	进位/借位标志位	0	0
sfr P0	P0口状态寄存器	05H	31
sfr P2	P2口状态寄存器	06H	EC
sfr P1	P1口状态寄存器	07H	06
sfr PCH	程序计数器 (PC)高字节	0AH	0E
sfr INTCTL	中断控制寄存器	0BH	00
sfr EIF1	中断标志寄存器	0CH	00
sfr EIF2	中断标志寄存器	0DH	00
sfr T1L	定时/计数器T1低字节寄存器	0EH	00
sfr T1H	定时/计数器T1高字节寄存器	0FH	00
sfr T1CTL	T1控制寄存器	10H	00
sfr T2	定时/计数器2 (T2)寄存器	11H	00
sfr T2CTL	T2控制寄存器	12H	00
sfr PWM1L	PWM1占空比设置寄存器	13H	FE
sfr PWM1H	PWM1寄存器	14H	00
sfr PWMCTL	PWM启动控制寄存器	15H	C0
sfr PP1	PWM1周期寄存器	16H	FF
sfr CMCTLO	CMCTLO控制寄存器	19H	00
sfr CMCTL1	CMCTL1控制寄存器	1AH	00
sfr VRECAL	内部参考电压校准寄存器	1CH	13
sfr ANSEH	模拟口设置寄存器	1DH	00
sfr ADCDATAH	ADC数据寄存器高字节	1EH	F9
sfr ADCCTLO	A/D控制寄存器0	1FH	00
sfr OPTR	选择寄存器	21H	FF
sfr TR0	P0方向控制寄存器	25H	FD
sfr TR2	P2方向控制寄存器	26H	FF
sfr TR1	P1方向控制寄存器	27H	FF

7) 内存视图: 显示芯片中所有的信息, 除特殊地址外可以修改。

内存	Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
	000	18	00	51	18	04	31	BC	06	FF	0F	0E	00	00	00	00	00	. Q . i . . . . .
	010	00	00	00	FE	00	C0	FF	18	18	00	00	00	13	00	F9	00	. . . . .
	020	18	FF	18	18	18	FD	FF	FF	FF	0F	18	18	00	00	10	30	. . . . .
	030	FE	00	FF	AF	00	F7	00	00	18	18	00	00	08	18	AE	00	. . . . .
	040	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	. . . . .
	050	FF	01	FF	00	00	89	DC	00	00	00	00	00	00	01	FF	00	. . . . .
	060	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	. . . . .
	070	18	18	18	18	18	18	18	40	02	00	00	18	18	18	80	DA	. . . . . @ . . . . .
	080	AE	42	68	B7	C1	0D	11	AA	23	5D	31	D7	DF	73	1A	F1	. Bh . . . . . #]1 . . . . .
	090	C6	6F	CF	FE	DF	F1	45	EF	3F	28	4D	DE	20	00	08	DF	. o . . . . . E ? (M . . . . .
	0A0	88	8F	F4	77	DF	8A	1C	AF	63	4F	13	AC	B3	25	18	FB	. . . . . w . . . . . c0 . . . . .
	0B0	80	A3	47	71	B3	47	27	9E	F4	5F	DF	B1	9F	02	25	5B	. . . . . Gq . G' . . . . .
	0C0	28	82	F3	EA	F9	4D	1F	46	66	30	F9	A4	BC	9E	40	EF	. . . . . M . F#0 . . . . .
	0D0	2A	26	35	F5	2F	B6	10	EA	84	34	3B	6D	58	F5	84	C6	*#5 . / . . . . . 4 : . . . . .
	0E0	69	26	3F	46	EA	CA	62	AD	82	CC	45	E3	B5	20	51	08	i#?F . b . . . . . E . . . . .
	0F0	80	49	EC	77	BF	49	96	A4	74	02	0E	FF	37	8D	3A	A4	. I . w . I . t . . . . .
	180	00	9D	2B	04	06	01	02	03	04	00	01	02	03	04	05	06	. . . . . + . . . . .
	190	9F	50	7C	FD	5D	3A	58	1F	45	0A	FE	E5	3F	06	5B	4F	. P   . ] : X . E . . . . .
	1A0	66	6D	7D	07	7F	6F	01	00	00	00	00	1E	DF	A4	E1	E2	f#} . o . . . . .
	1B0	36	18	0E	AB	2A	CF	9A	B7	38	57	79	A0	77	A2	1F	98	6 . . . . . * . . . . . 8Wy . . . . .
	1C0	14	33	EB	17	F6	76	A4	E8	35	D8	3E	73	B7	9C	96	2C	. 3 . . . . . v . . . . . 5 . > . . . . .
	1D0	A9	E3	63	7F	3E	C2	04	EB	CE	13	47	DF	79	B6	6A	77	. . . . . c . > . . . . . G . . . . .
	1E0	50	A0	A3	DB	55	98	4A	CF	85	C2	C0	F7	FC	C3	10	E4	P . . . . . U . J . . . . .
	1F0	13	9D	00	00	38	00	00	00	08	20	80	DA	DA	00	55	46	. . . . . 8 . . . . .

注：ChipON IDE 提供了默认的调试视图，但并非所有的视图在进入时打开，但可以通过操作完成视图的打开与关闭。常规的视图在菜单栏窗口下的显示视图下直接打开，部分需要选择其他，并根据类别执行选择。操作方法：菜单下窗口一》显示视图一》其他。如 ChipON 类别下的 芯片信息和资源使用率 视图。调试视图的芯片信息查看，包括表达式、寄存器、EEProm、通用寄存器等。



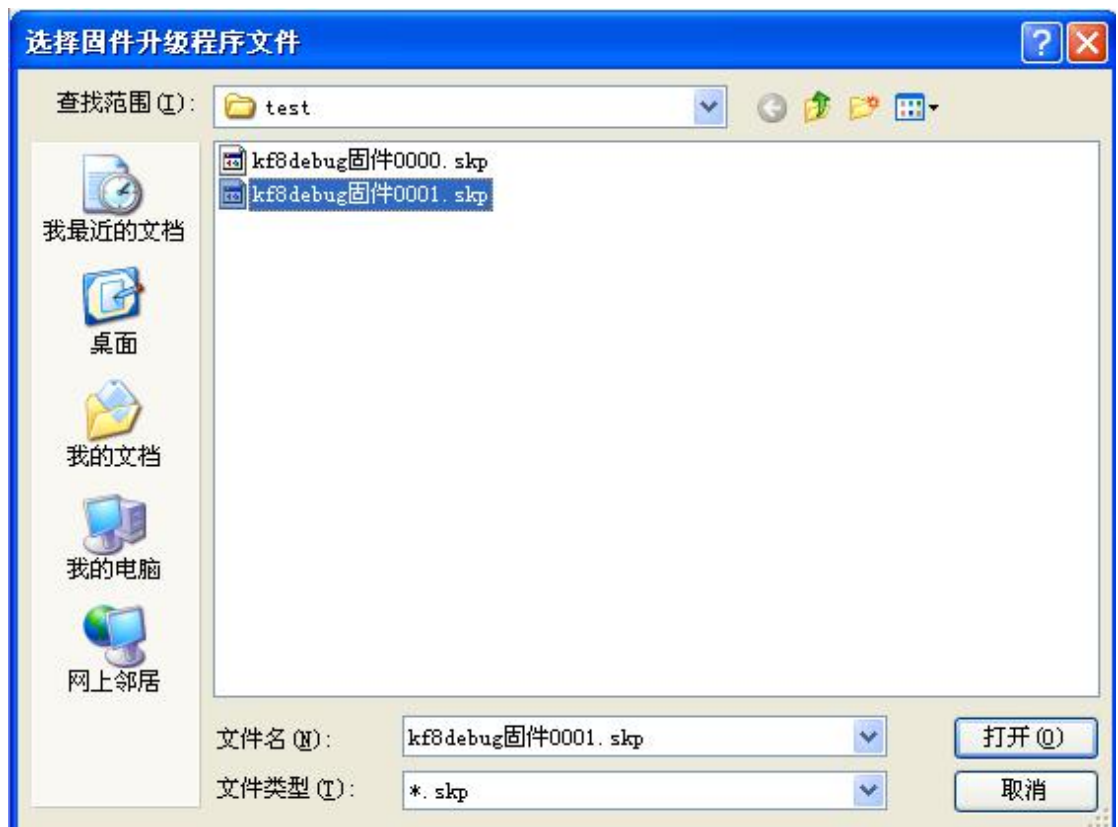
## 15. 固件升级

菜单: 帮助—>>固件升级

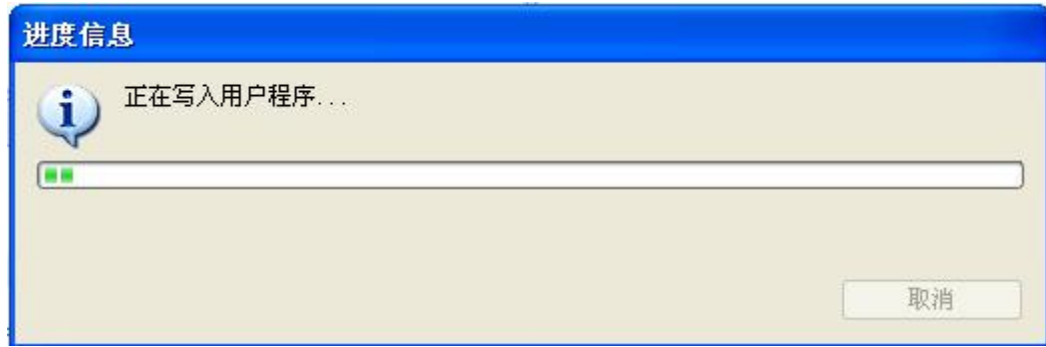


固件升级的方式有两种:

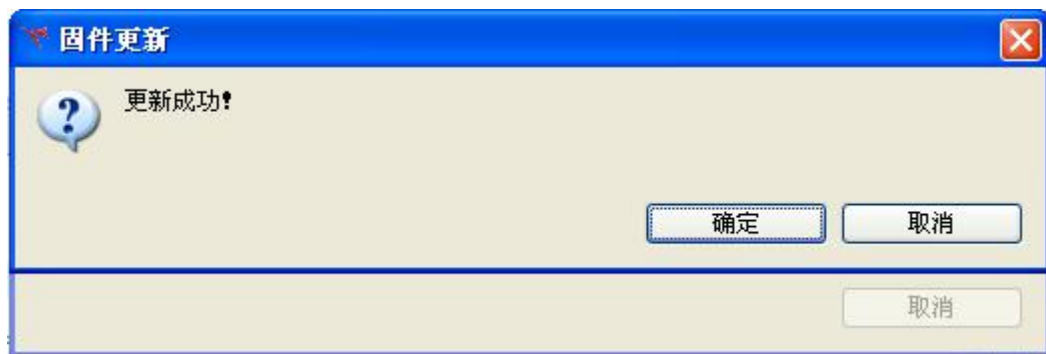
1. 自动升级  
将发布版带有的固件版本程序下载到编程器中。
2. 手动升级  
在弹出的对话框中选择“.skp”格式的文件, 如图:



不论自动和手动模式, 菜单操作完成后显示如下进度信息, 并等待升级完成。



完成后:



## 16. 硬件仿真（调试器）使用常见错误解决办法

### 1) Socket 建立连接失败!

建立 socket 连接超时，请关闭一些程序后再试，或者检测防火墙设置

### 2) 调试器未连接

未连接调试器或者未安装串口驱动程序

### 3) deviceICDPowerON() error!

调试器设定电压操作失败，确认调试正常或重试。

### 4) deviceReset() error!

请检查配置位调试功能是否打开，MODE 脚电路是否正常，ICSPCLK、ICSPDAT 脚电路是否正常。

## 5) gotomain() error!

一般不会出现此问题, 如果出现此问题有可能时编译器行号出错。无法再进行调试。需要在具有 main 函数的文件启动调试功能, 可以尝试改变 main 函数所在行号并重新编译后启动调试。

## 6) flush ram(0x00-0x7F) failed!

flush ram(0x80-0xFF) failed!

flush ram(0x180-0x1FF) failed!

通信异常, 获取芯片内通用寄存器值失败, 请检查连接后再试。

## 7) 程序已经跑飞, 请确认程序无误后再试!

出现这种错误原因较多:

硬件问题式程序跑飞

调试信息可能异常。

通信异常, 请检查连接后再试

## 8) 调试端口被占用, 请重新设置 P0.0、P0.1、和 MODE!

P0.0、P0.1 在调试时需要与调试器通信, 请不要设置成输出口或模拟口。MODE 脚需要做为复位使用, 请注意外部电路。

## 9) 调试目标已断开, 请检查连接后再试!

请检查连接后再试

## 10) 设置断点失败!

请检查连接后再试。如果多次尝试后依然不行, 应该是编译器信息出错, 或者此处不能设置断点。

## 11) 运行失败!

请检查连接后再试

## 12) 暂停失败!

请检查连接后再试

## 13) 复位失败!

请检查连接后再试

## 14) RAM 设置失败!

请检查连接后再试

## 15) 打印 BEE 失败!

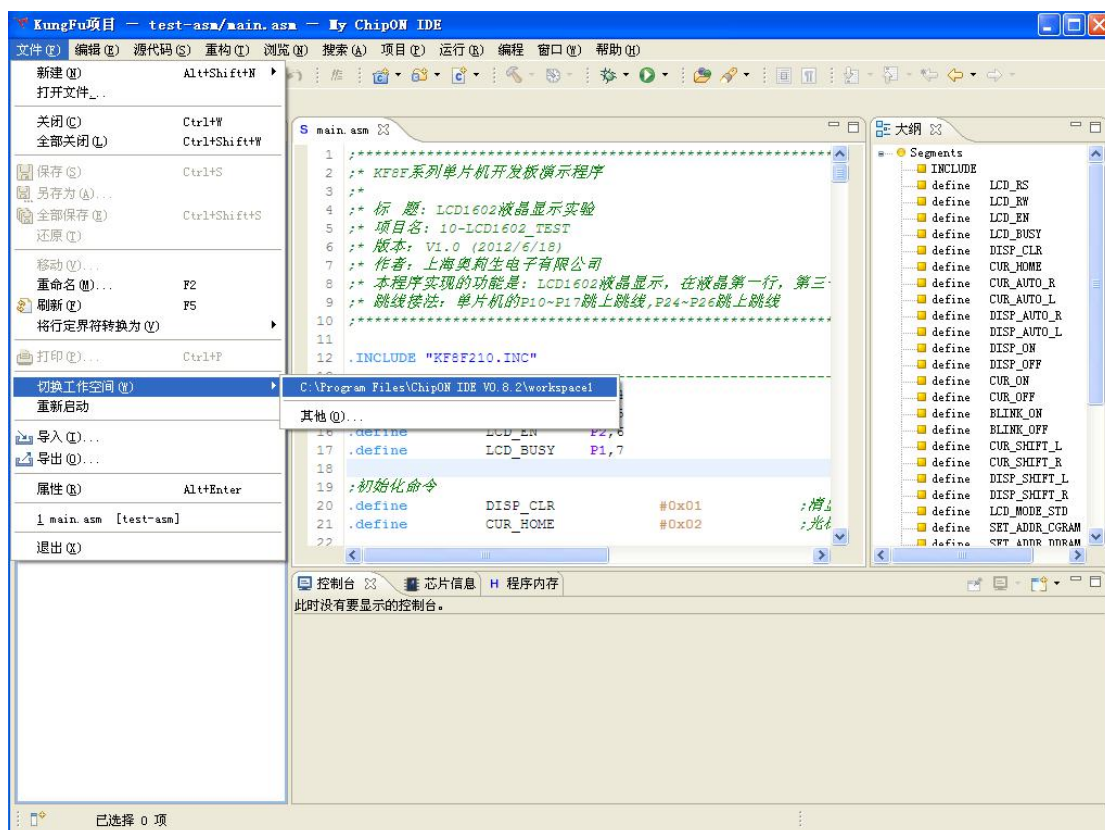
请检查连接后再试

- 
- 16) 串口连接已断开!请检测连接后再试.  
    串口接收数据异常  
    串口发送数据异常  
    串口接收数据通信异常!请检查连接后再试.  
    串口连接中断, 请检查 USB 连接是否正常。
- 17) 串口接收数据通信异常!请检查连接后再试!  
    串口通信抛出异常, 请检查 USB 连接是否正常。然后再试。
- 18) 请求内存失败!  
    内存分配失败, 请关闭部分软件后再试。
- 19) 读取 PC 通信异常!请检测连接后再试.  
    请检查连接后再试
- 20) 读取寄存器异常!  
    读取所有特殊功能寄存器失败!  
    请检查连接后再试
- 21) 单步运行失败!请检测连接后再试.  
    请检查连接后再试
- 22) 运行失败!请检测连接后再试.  
    检查连接后再试
- 23) 暂停调试目标失败!  
    检查连接后再试
- 24) 复位调试目标失败!  
    检查连接、配置字是否正常。
- 25) 调试设备忙!  
    Debugger busy!  
    检查连接后再试
- 26) 驱动识别类问题处理
- 1) 缺少驱动, 执行驱动的安装。
  - 2) 端口类, 更改端口号, 如将编程器端口号使用较小端口, 如 COM1。
  - 3) 驱动不兼容, 驱动程序较芯片落伍, 删除历史驱动, 安装最新驱动。
  - 4) 设备异常: 电脑设备管理器中找不到设备, 非驱动未安装情况。
- 注: 关于驱动使用的详细文档见独立下载包编程器驱动程序。

## 17. IDE 使用技巧

### 17.1 切换工作空间

用户可以建立多个工作空间，并可以在各个工作空间进行切换。选择 文件一>>切换工作空间一>>其他，如下，可以选择其他工作空间或者建立新的工作空间。切换的空间可以直接复制粘贴路径，或通过浏览按钮进行确认，需要注意：如果该路径是空间，进行引用；如果该路径不是空间，在此位置建立空间，但空间不能嵌套，即空间目录下建立新的工作空间。





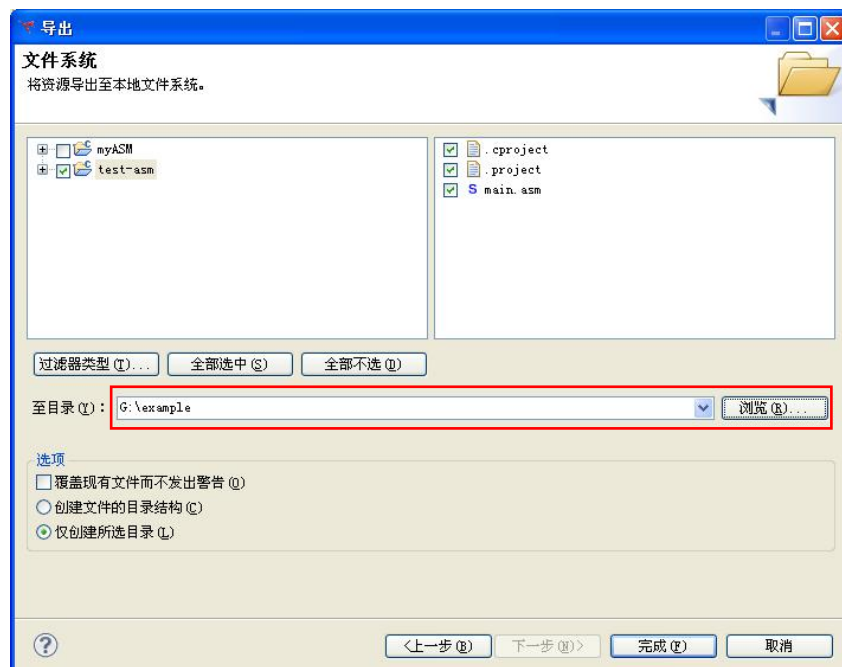
## 17.2 导出项目

通过导出功能, 用户可以将当前工作空间中的项目文件导出到本地指定路径下, 可以作为备份等。

操作: 选中工程文件, 右键打开菜单, 选择“导出”, 进入导出界面:



选择常规的文件系统, 下一步:



选择导出的文件目录后, 点击完成按钮, 成功将项目导出到 G:\example 文件路径下。

注: ChipON IDE 直接项目空间下的项目直接复制。支持复制粘贴的快捷键。

通过粘贴时可以追加日期信息作为项目的备份。不建议频繁直接更新项目名称。一旦存在更改名称过程异常,与项目信息可能存在不一致,此时会造成使用的不便,通过重新建立项目,直接复制项目代码文件到当前新建项目即可。

### 17.3 导入项目

通过导入功能,用户可以将现有的工程文件导入到当前工作空间的指定路径下。打开 文件-->导入,进入导入界面:



选择常规下面的现有项目到工作空间中,下一步:

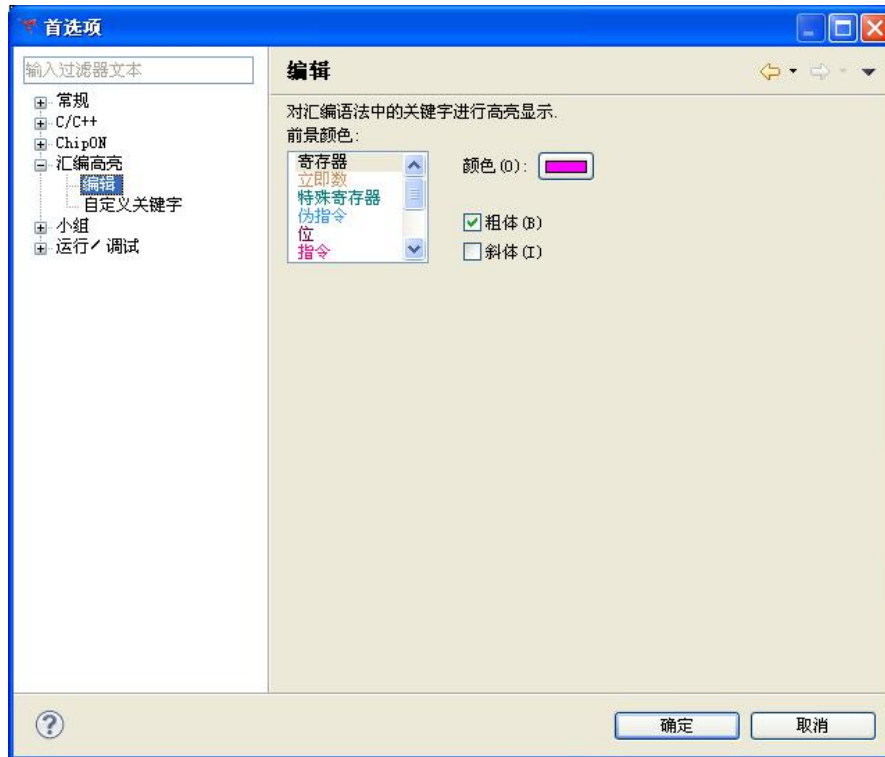


选择项目路径,点击完成按钮,成功将现有项目导入到当前工作空间中。

**注意:** 空间下不支持导入同名项目,不支持当前项目空间下项目的导入操作。建议该操作勾选将项目复制到工作空间,从而实现基于工作空间的项目管理。

## 17.4 关键字高亮显示设置

(1) 用户可以自定义汇编语法中的关键字在编辑器中显示的颜色。  
选择 窗口-->首选项，打开首选项设置：



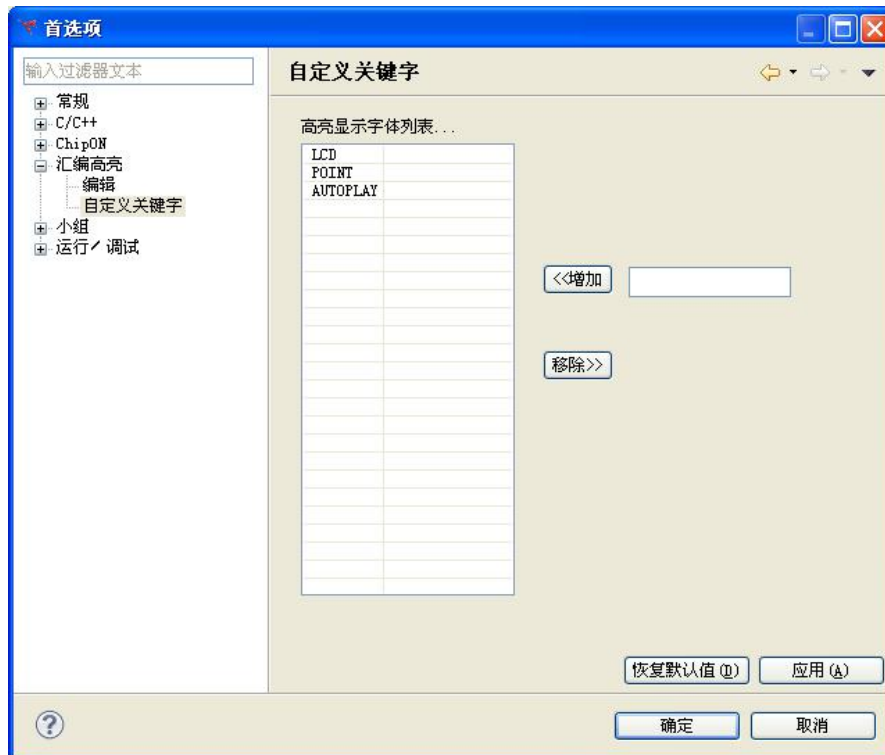
选择 汇编高亮-->编辑，在右侧页面中选择需要改变的关键字类型，选择颜色：



选择想要显示的颜色，设置字体是否粗体、是否斜体，设置完成后，点击确定按钮，完成设置。

(2) 用户可以自己添加需要高亮显示的关键字。

在首选项页面中, 选择 汇编高亮-->自定义, 在右侧页面中添加删除自己需要定义的关键字:



## 17.5 代码补全

在编辑代码的过程中,如遇到指令比较长,可通过代码补全功能进行简化输入,如下所示,若需输入一个标号名为 INIT\_ALL 的字段,先输入 I,IDE 会自动进行代码提示,如果代码提示框消失,可通过快捷键 ALT+/ 键进行显示:



选中 INIT\_ALL, 点击 Enter 键即可输入全部字段。

针对 C 项目的开发,结构体、联合体的. 会直接触发悬浮框,提示该对象所包含内容。如果需要通过上面示例的提示符号,需要手动通过按键执行弹窗指示。快捷键同样为“ALT”+“/”键。



## 17.6 重构

重构是指对现有代码进行重新构造，统一修改变量名或者提取复用代码到函数中。

### (1) 重命名

如果现有代码中的变量所代表的含义发生改变，需要对变量名进行修改，在变量使用过多时，可统一对一个变量名进行修改。

选择所需要更改的变量名字，点击右键，在右键菜单中选择 重构-->重命名：



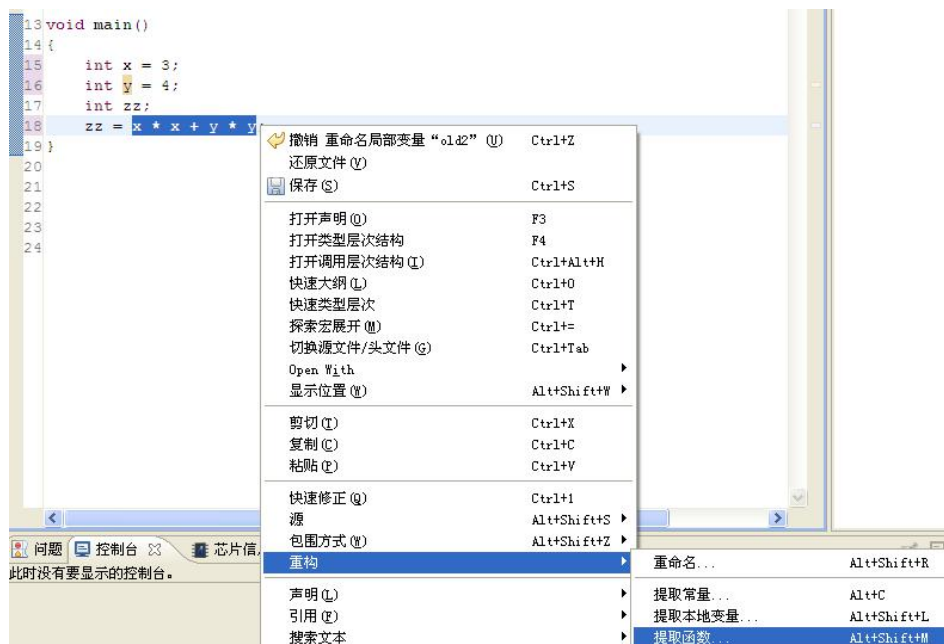
之后删除原有变量名 old1，直接输入新变量名 new，按 Enter 键完成修改：



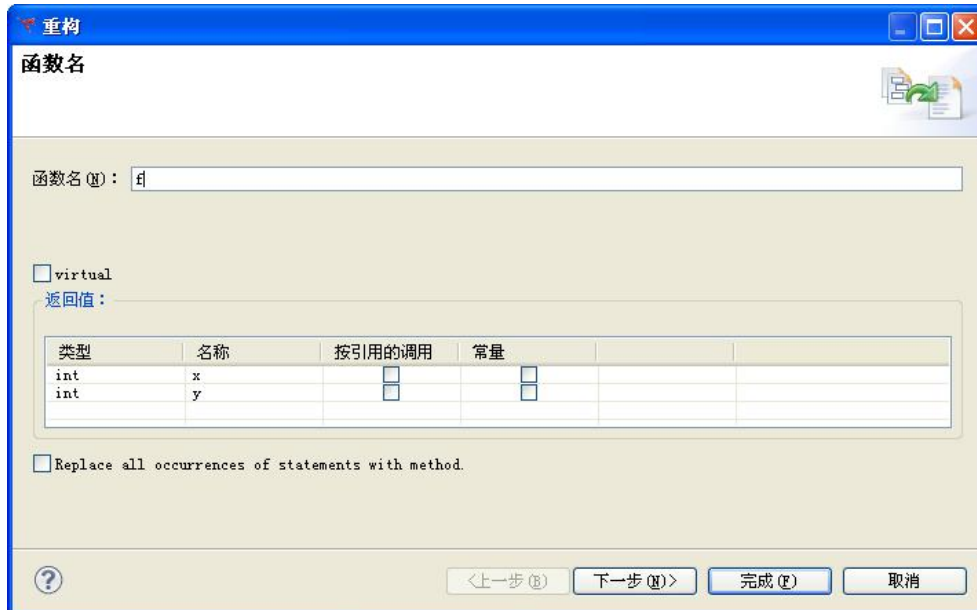
## (2) 提取函数

如果需要对现有函数中的一段代码进行提取，以方便代码复用，则可用提取函数功能。

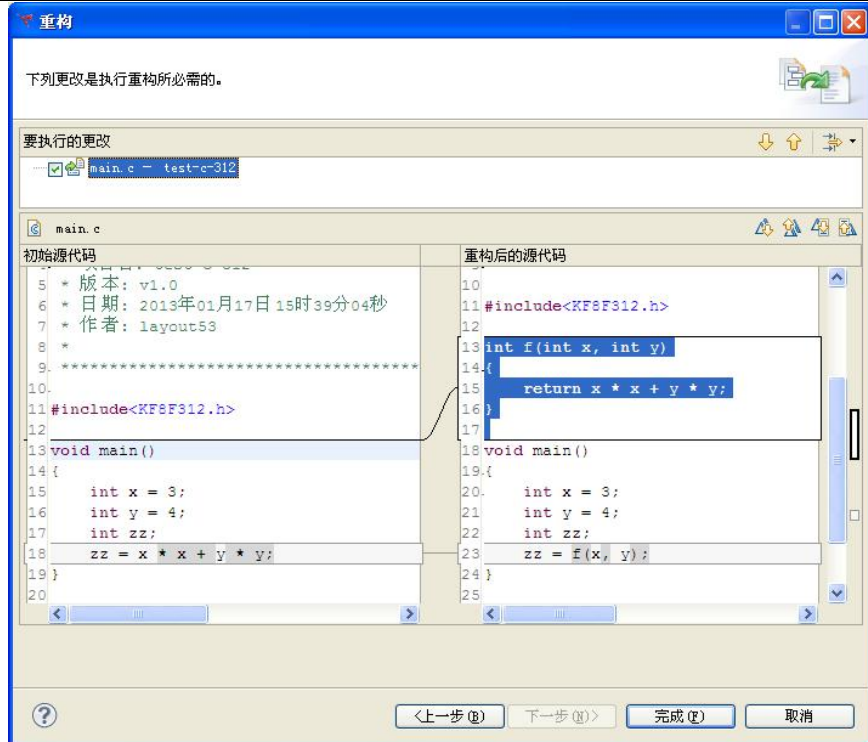
选中需要提取的代码块，点击右键，在右键菜单中选择 重构-->提取函数：



在打开的重构窗口中输入需要的函数名，根据开发需要对形参类型进行选择：



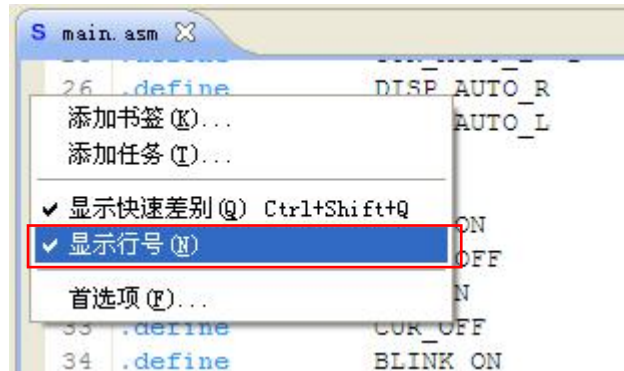
点击下一步，可对重构后的代码进行预览：



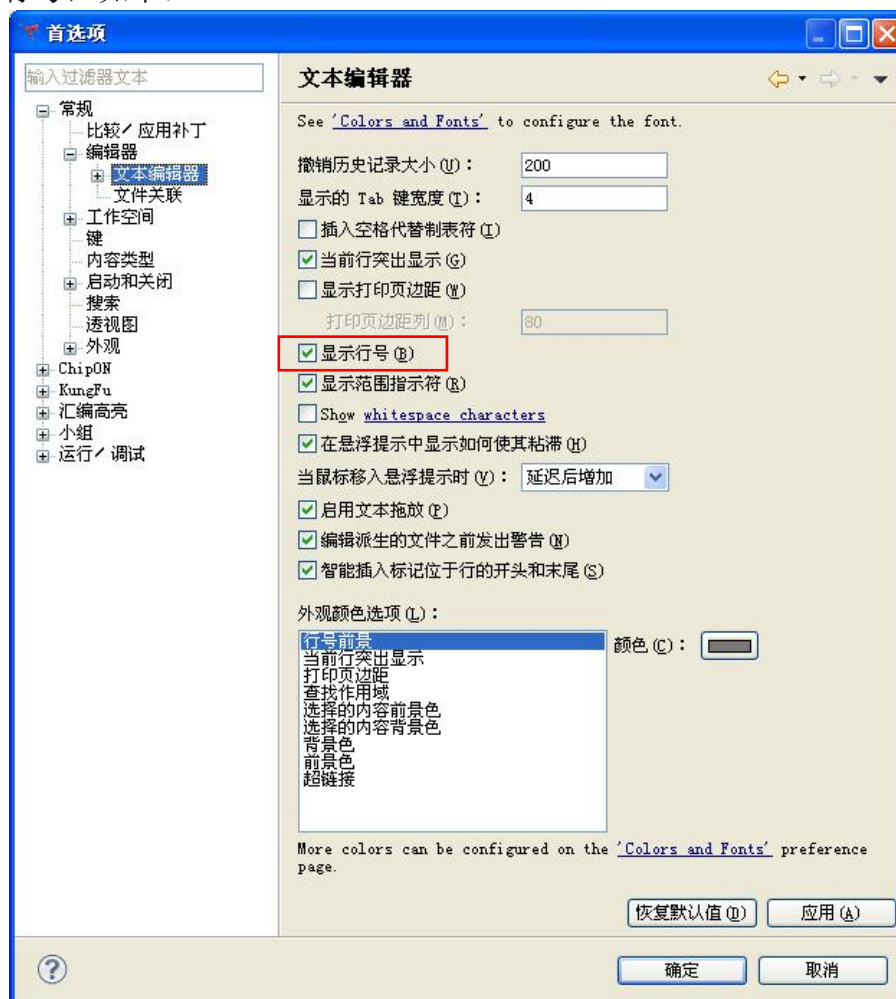
之后点击完成按钮，完成对函数的提取操作。

## 17.7 显示行号

右键点击编辑器左侧栏，选择显示行号即可，如下：



或者选择 窗口-->首选项-->常规-->编辑器-->文本编辑器，在右侧页面选择显示行号，如下：



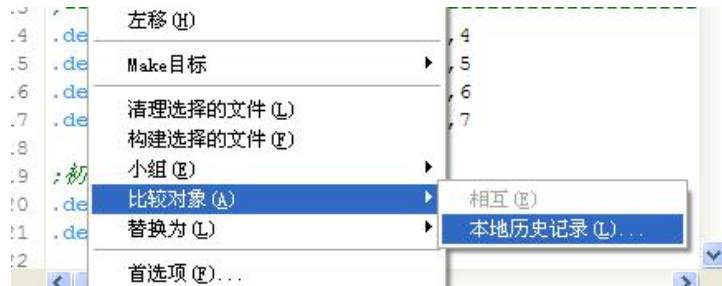
## 17.8 相关快捷键

快捷键	功能
Ctrl+/或 Ctrl+;	对编辑器光标所在行加注释或取消注释。
Ctrl+B	对工作空间中的所有工程进行构建。
Ctrl+Shift+C	下载 Hex 文件前, 检查下载器设备。
Ctrl+Shift+U	在线 ICSP 下载。
Ctrl+Shift+O	复位操作。
Ctrl+S	保存。
Ctrl+C	复制。
Ctrl+V	粘贴。
Ctrl+X	剪切。
Ctrl+Z	撤销上一步操作。
Ctrl+D	删除编辑器光标所在行。
Ctrl+W	关闭当前编辑文本。
Ctrl+F	查找替换。
Ctrl+H	搜索。
Ctrl+Alt+G	在工作空间对选中文本进行查找。
F5	刷新。
F2	重命名。
Ctrl+Shift+X	将选中字符串转换为大写。
Ctrl+Shift+Y	将选中字符串转换为小写。
Alt+/ 	代码补全功能

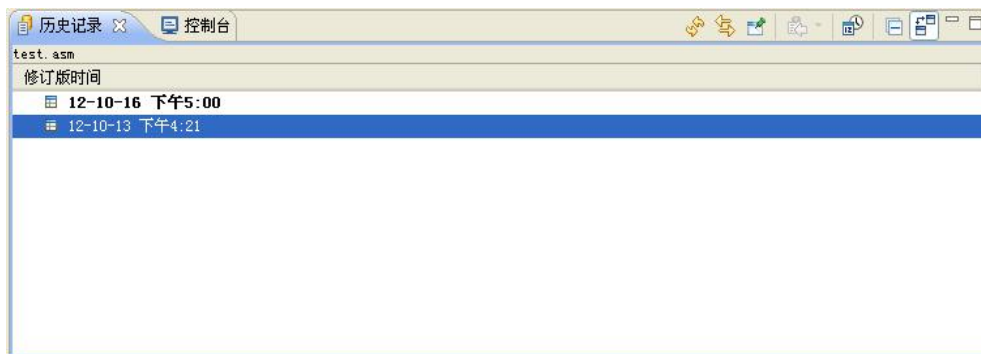
## 17.9 历史文件比较

在编辑文件的过程中，若需要查看历史记录中的某一个版本，可以通过比较对象完成。

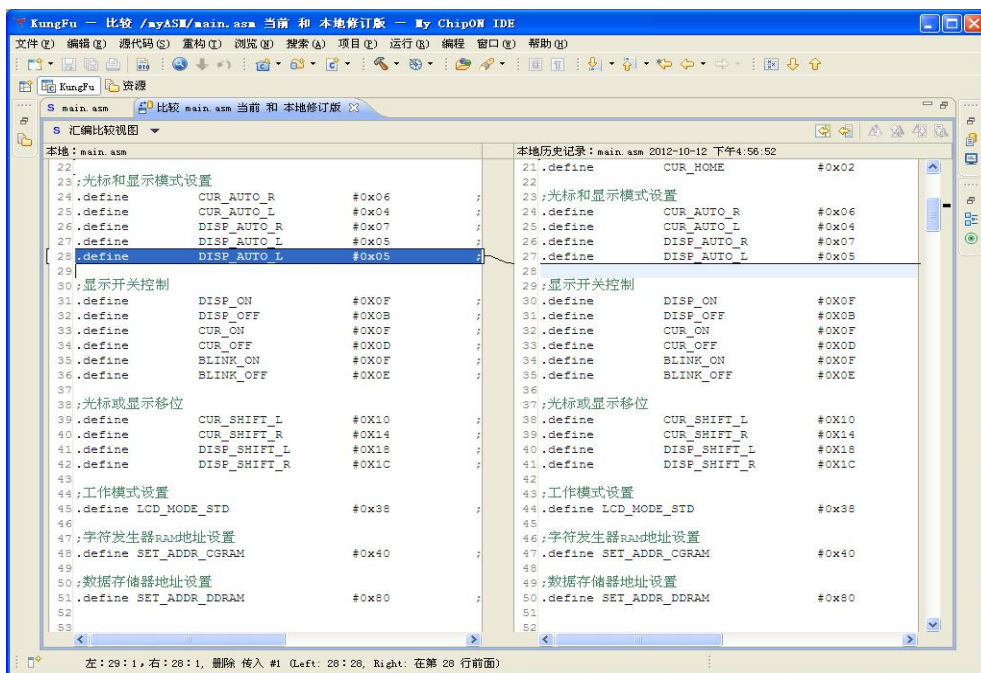
在右键菜单中选择 比较对象-->本地历史记录：



打开历史记录视图，在历史记录视图中，双击需要比较的版本，可对当前版本和历史版本进行比较：

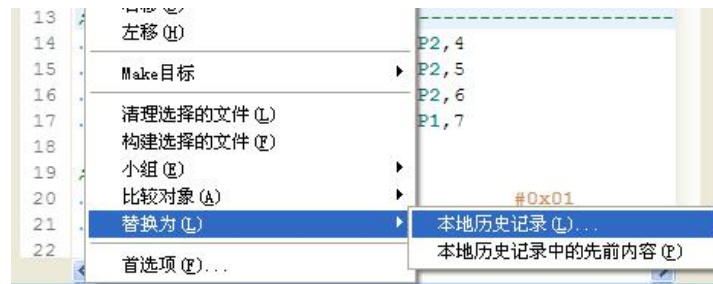


在比较视图中可清晰查看出两文本的差异：

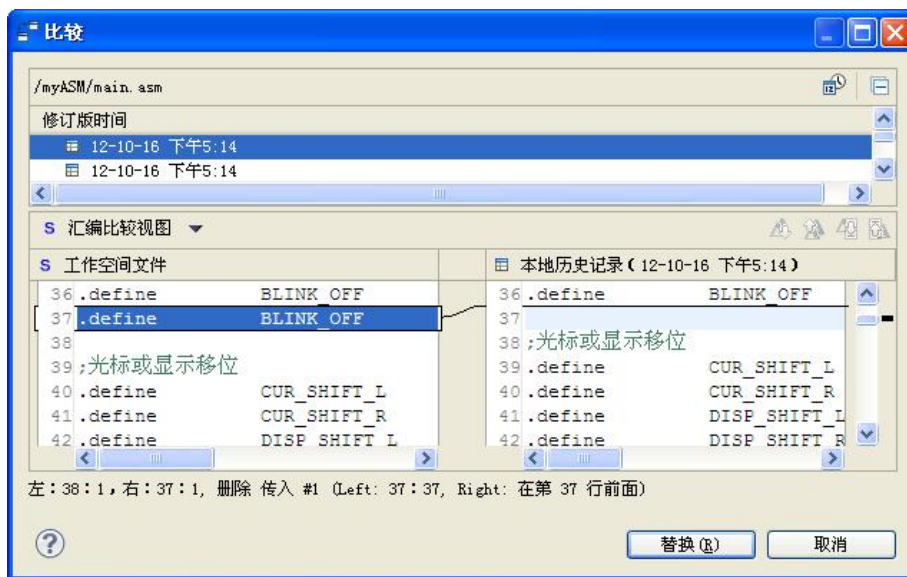




若需要回复到历史版本，右键菜单选择 替换为-->本地历史记录：



打开比较菜单项，双击选择需要恢复的历史版本，可在页面下方看到两个文件的差异：



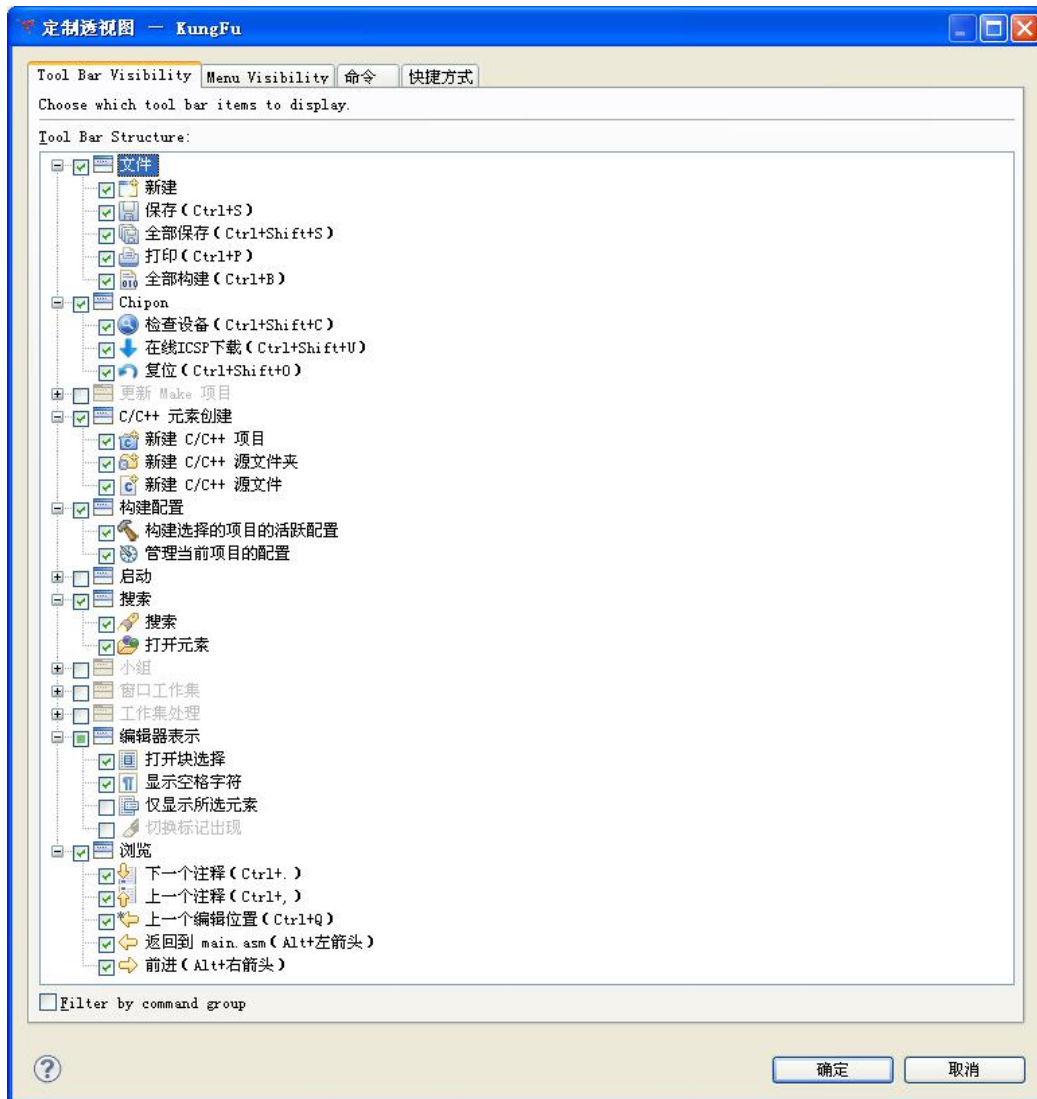
点击 替换 按钮完成替换任务。

## 17.10 定制透视图

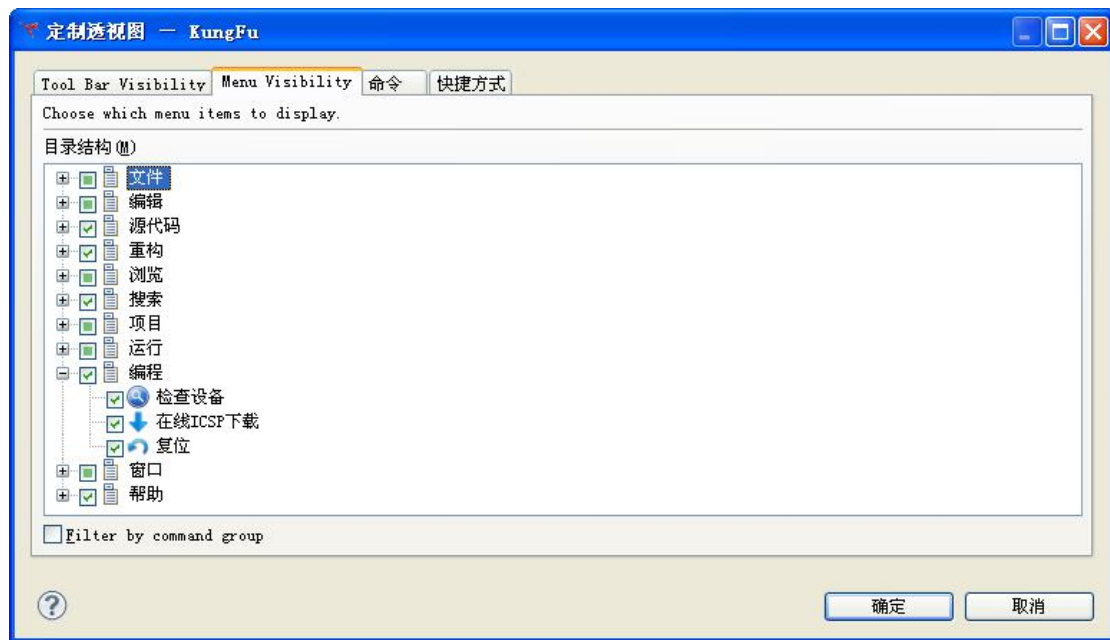
用户可根据需要自己添加删除在开发环境中显示出的工具，便于提高效率，定制自己个性的开发环境。

菜单项 窗口-->>定制透视图，打开定制透视图页面。

在“Tool Bar Visibility”页面中，用户可自定义工具栏中显示的图标。



在“Menu Visibility”页面中，用户可以自定义菜单项中显示的菜单。

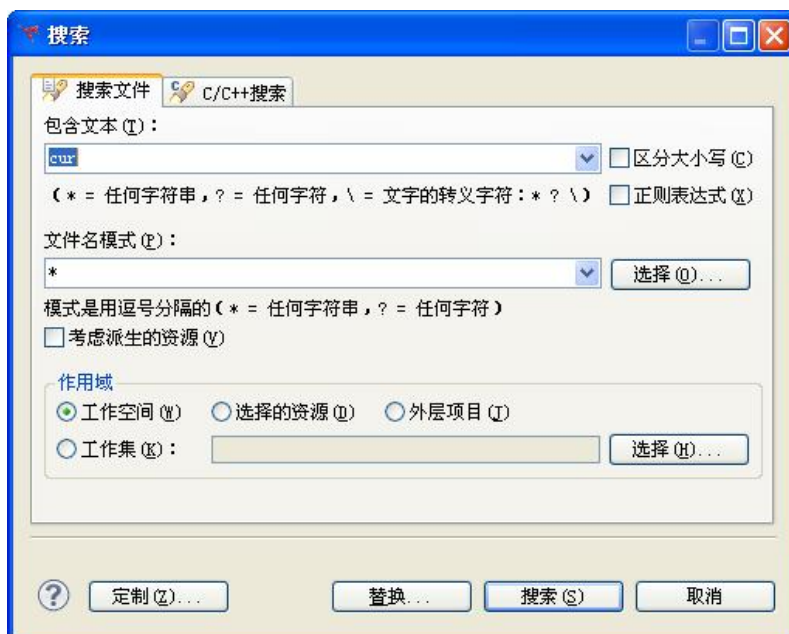


## 17.11 查找/替换

文件内关键字查找可通过菜单 编辑-->查找/替换, 或者快捷键 Ctrl+F , 在打开的页面中输入需要查找的关键字, 如下:



在整个工程中进行关键字查找, 可通过菜单 搜索-->搜索, 或者快捷键 Ctrl+H, 在打开的页面中输入需要查找的关键字, 如下:



## 17.12 查找特定位置的字符串

使用快捷键 CTRL+H 打开搜索页面，在包含文本中填入需要搜索的文本，在文件名模式中写入需要匹配的模式（默认为\*，即任何文件），如果需要在当前工作空间中的所有项目进行搜索，则在作用域中选择“工作空间”，如果希望在特定的文件或工程中进行搜索，则先选择对应的资源，之后在此页面的作用域中选择“选择的资源”，具体如下所示：



搜索完成后，若搜到所有的匹配位置，会在“搜索”视图进行显示出来，通过双击可以立刻定位到其位置，在编辑器视图的右侧栏中会显示一个白色图标显示其位置，如下所示：



### 17.13 返回到上一个操作位置

在编辑过程中有时会在源文件中进行大范围的跳转,若希望返回到上一个操作光标所在的位置,可通过工具栏中的  按钮实现该操作。

### 17.14 关闭工程

在“项目资源管理器”视图中,选中需要关闭的工程,右键单击打开右键菜单,选择关闭工程即可。

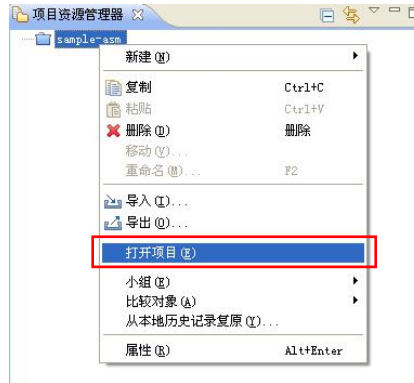
若需要关闭除此工程以外的其他工程,则选择关闭不相关工程即可。






## 17.15 打开工程

在“项目资源管理器”视图中，选中需要打开的工程，右键单击打开右键菜单，选择打开工程即可




## 17.16 最大化、最小化和关闭当前窗口

在每一个视图中，右上角都会有两个图标 ，分别代表最小化、最大化。

直接双击视图名称即可实现最大化和还原的操作。

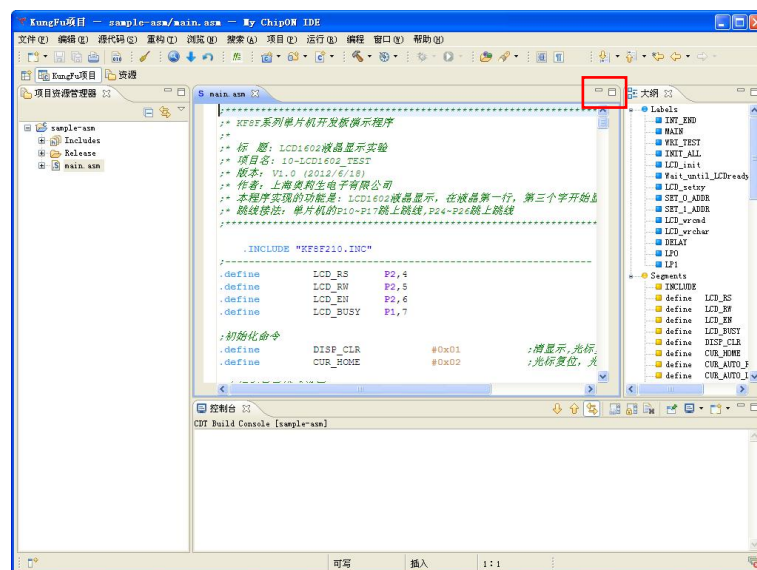
最小化后需要恢复可在 IDE 最右侧一列点击恢复即可。

若需关闭当前窗口，点击此窗口名称旁边的  即可。

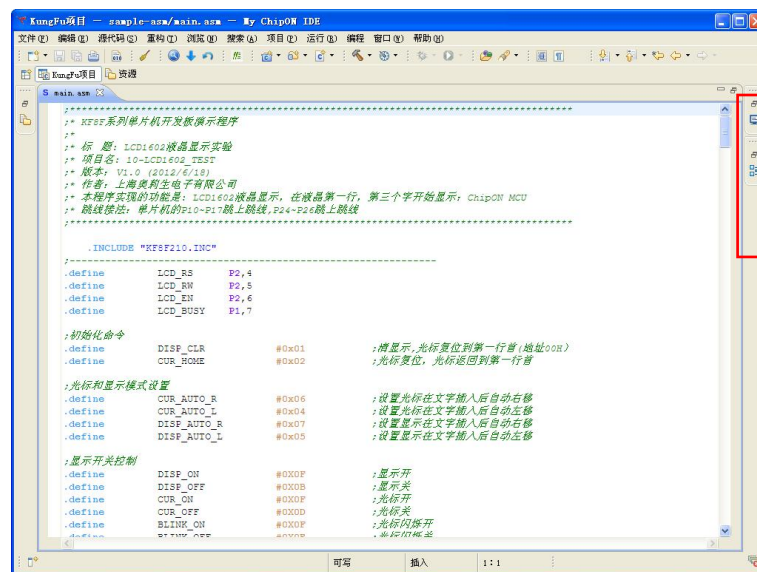
以上操作也可通过右键窗口名称在右键菜单中进行。

下面以最大化、最小化、恢复 main.asm 编辑器为例说明此操作：

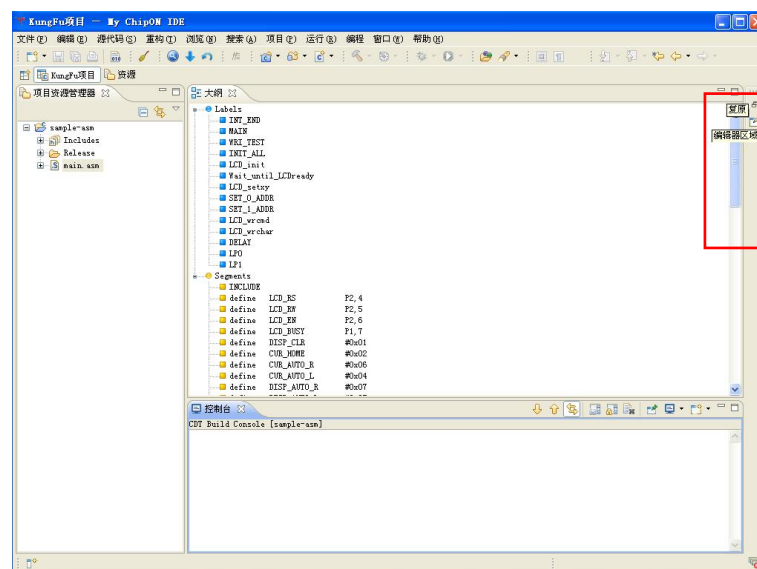
### ➤ 正常状态



➤ 最大化状态



➤ 最小化状态




## 17.17 快速的注释代码


### ➤ 单行注释

光标点中需要注释的行，快捷键 CTRL+； 或 CTRL+/或在工具栏点击 。

### ➤ 多行注释

选中需要注释的文本，快捷键 CTRL+； 或 CTRL+/或在工具栏点击 。

### ➤ 取消注释

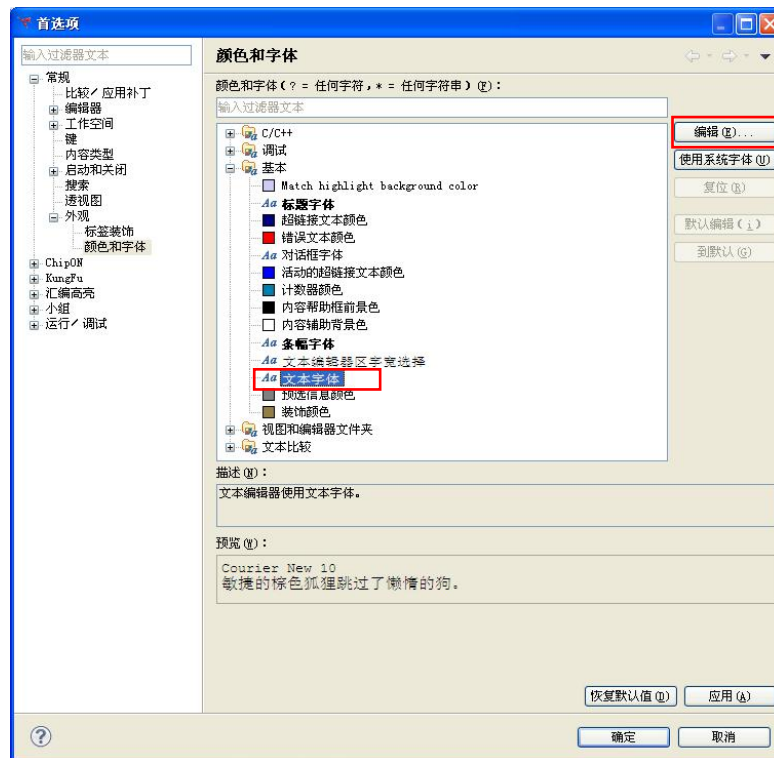
选中已经注释的文本，快捷键 CTRL+； 或 CTRL+/或在工具栏点击 。

## 17.18 改变视图窗口的位置

若需要更改视图窗口在 ChipON IDE 中的位置，可直接拖曳该窗口，放置在你希望放置的位置即可。

## 17.19 改变编辑器字体

选择窗口-->首选项菜单，在首选项中选择常规-->外观-->颜色和字体，在右侧页面中选择基本-->文本字体，选中后点击编辑，如下所示：



在字体页面中，设置需要的字体，应用即可。

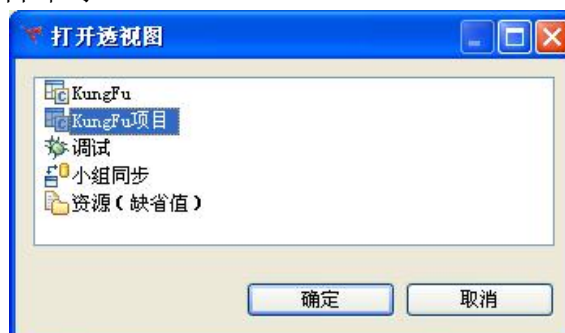


## 17.20 恢复初始视图布局

在 ChipON IDE 左上角右键点击 KungFu8 项目，选择复位即可，如下所示：




若找不到 KungFu8 项目按钮，选择窗口-->打开透视图-->其他，在下面页面中选择 KungFu8 项目即可。



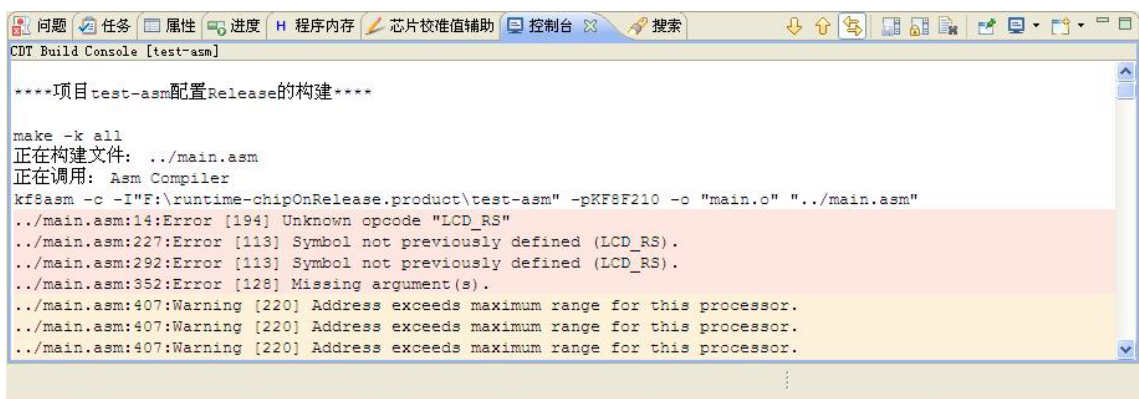
可以通过菜单栏的窗口下复位透视图菜单实现默认透视图的恢复。

## 17.21 清理项目

- 在工具栏点击  图标即可，（建立在选定的项目上）。
- 或选中项目后在右键菜单中，点击清理项目。

## 17.22 快速定位错误信息位置

构建完成后，在“控制台”视图中会显示构建信息，若出现错误或警告，可直接双击跳转到错误所在处，如下所示：



在“问题”视图中，也会显示错误或警告的具体信息，如下：



当不能快速定位时，还需要约定报错的内容，根据内容判断错误源。

## 17.23 恢复删除文件

在项目管理过程中,如果希望恢复以往删除的文件,可以通过“项目资源管理器”视图中提供的功能进行恢复。



选中需要恢复文件所在的文件夹,在右键菜单中选择“从本地历史记录复原 (Y)”,弹出“从本地历史记录复原”对话框,如下图:

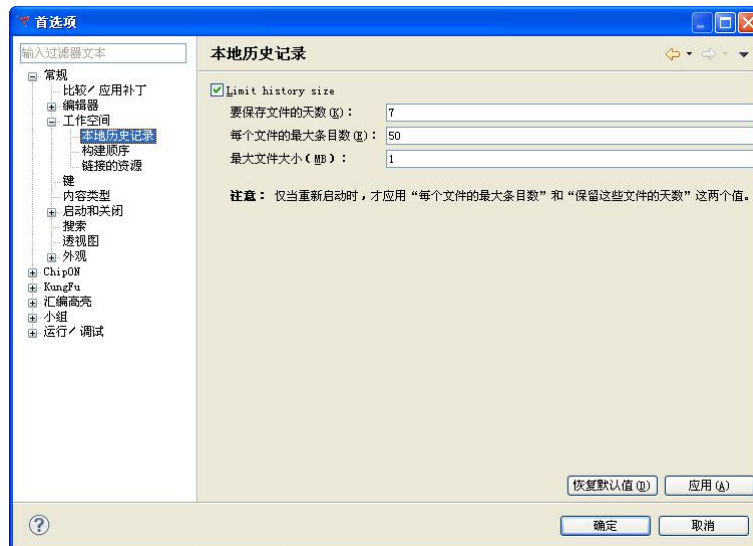


选中需要恢复的文件,点击“恢复”按钮即可将文件恢复。



用户在使用恢复功能时，恢复的文件有一定的时效以及个数限制，此设置可以在首选项菜单中进行设置。

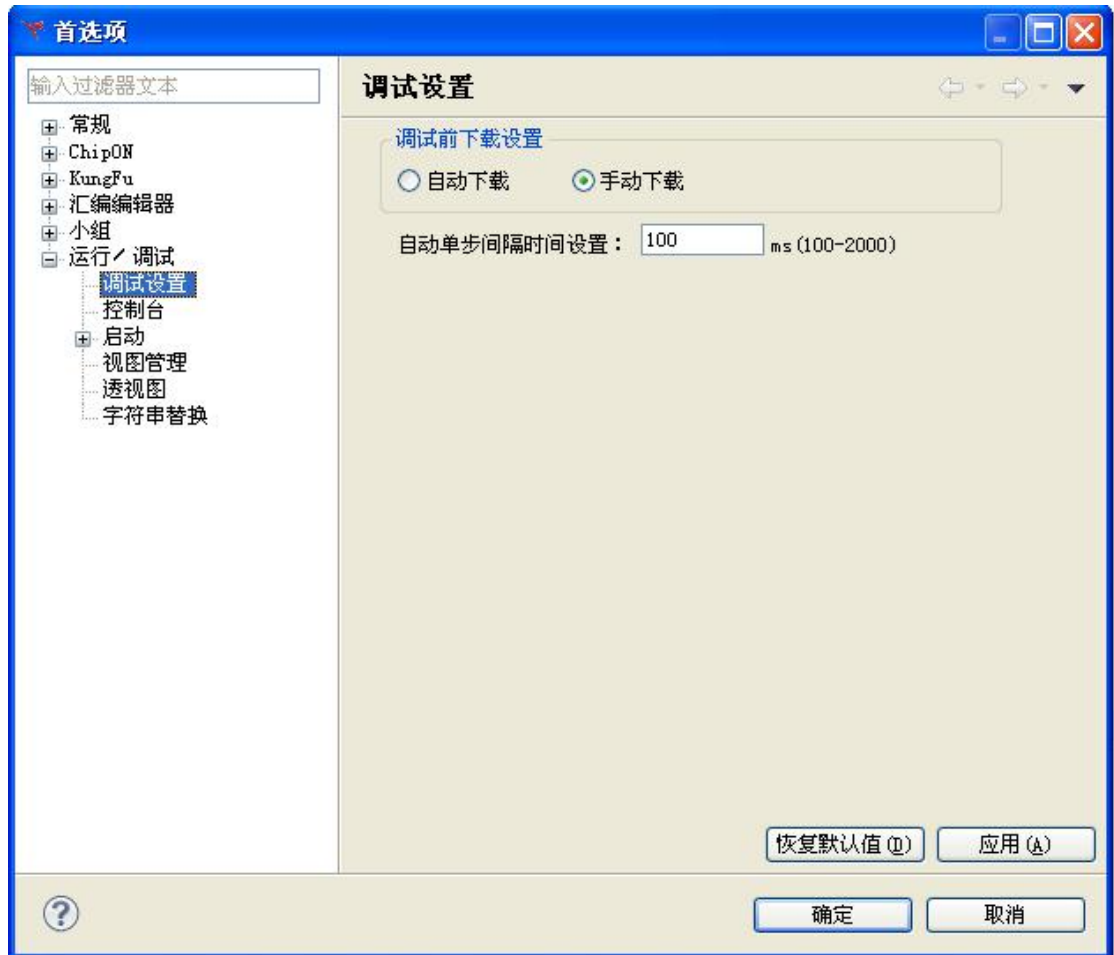
打开窗口-->首选项-->常规-->工作空间-->本地历史记录，显示如下：



用户可以根据自己的需求定制自己的存储方式。

## 17.24 调试设置

选择窗口-->首选项菜单，在首选项中选择运行/调试-->调试设置，在右侧页面中选择调试前下载设置，在文本框中填入自动单步间隔时间值，如下图所示：

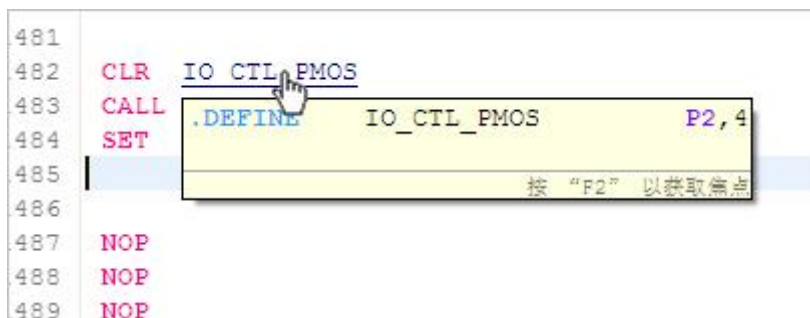


点击应用，确定；进入调试模式时，若是选择“自动下载”，则先向芯片中下载，再进入调试模式，若是选择“手动下载”，则直接进入调试模式，点击“连续单步”，其单步直接的间隔时间就是文本框中的值

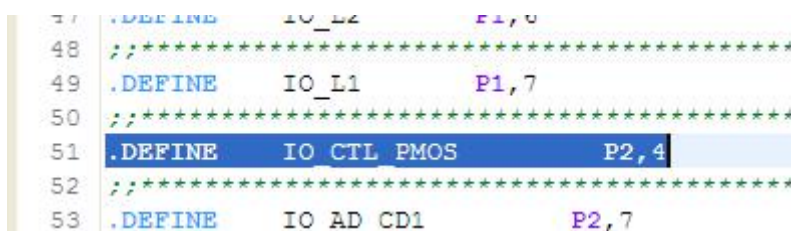
## 17.25 代码跳转

在汇编编辑器中，代码跳转方式有：

方法 1：按住 Ctrl 键，鼠标移动到要跳转的代码上方，如图：



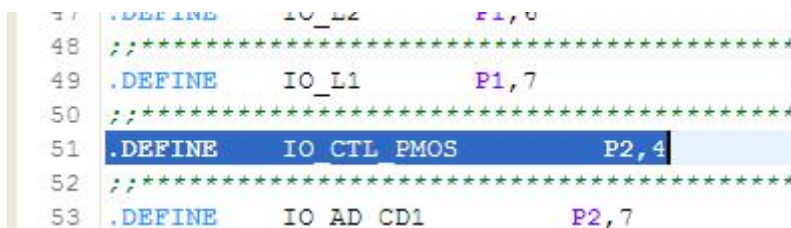
点击鼠标，在汇编编辑器中跳转到声明行，如图：




方法 2：选中要跳转的代码，点击鼠标右键，选择菜单中的“跳转到定义处 (O)”，如图：



点击鼠标，在汇编编辑器中跳转到声明行，如图：



## 17.26 管理配置

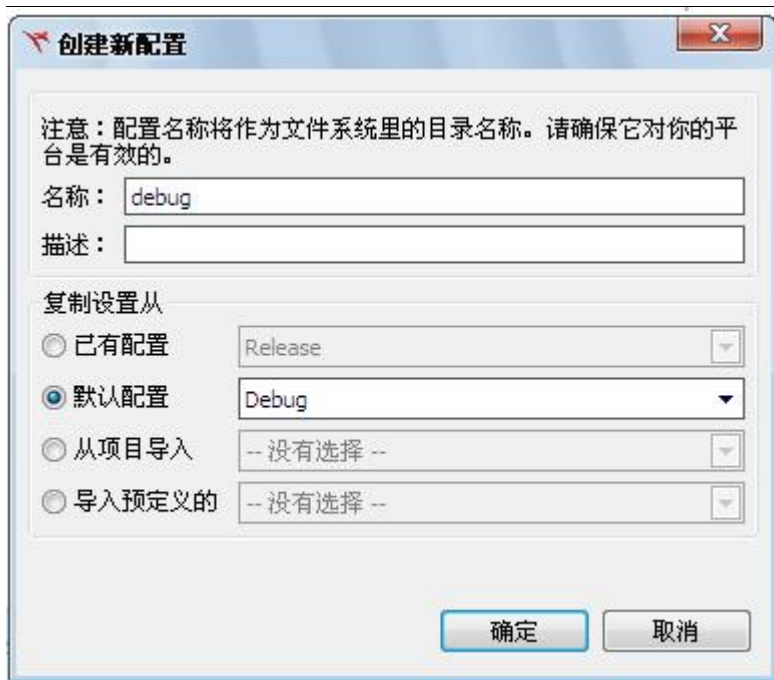
在新建项目时，如果忘记选择 debug 或者 release，可以按照以下方式找回：  
首先点击工具栏中的，就会出现如下图所示的界面（下图是忘记选择 debug）：



然后点击 **新建...**，就会出现如下图所示的界面：



输入名称，选择默认配置，如下图所示：



点击确定，管理配置中就新增了 debug 选项：



点击确定，就可以进行 debug 了。



注意:新建配置名称不做限定，建议且只实现图中的 2 个配置，确保名称与意义一致。

## 17.27 编译结果查看

项目编译不仅仅生成了 hex 的机器文件，实际存在 2 个非常有用的文件，在编译结果输出目录下输出了 以项目名称命令的 1st 文件 和 map 文件。其中 1st 包含了最多的信息，包括每个程序地址下的机器码，汇编指令，源码对应关系。这里也是查看编译结果是否正确的观察出发点。Map 文件的作用主要查看项目中定义的变量或编译过程使用的中间变量所分配的地址情况。根据变量所在地址，在调试时可以通过打开内存视图查看变量的实际结果，从而不局限于变量或表达式窗口。

## 17.28 使代码编译效率高

### a)、位取反:

```
Flagbit0 = ~Flagbit0 ;//效率低  
Flagbit0 = ! Flagbit0 ;//效率高
```

```
if(Flagbit0 ==0) //效率最高  
{   Flagbit0 = 1;}  
else  
{   Flagbit0 = 0;}
```

### b)、位判断:

```
if(DCIN!=1)    //效率低  
if(DCIN==0)    //效率高  
if(DCIN==1)    // 效率低  
if(DCIN)       // 效率高
```

### c)、定义

在RAM空间够的情况下,用结构体定义的位变量没有定义成字节型效率高.

采用指针表达效率较低,尽量少用。

可以定义简单联合结构体。如

```
AD_samp1=ADCDATAH<<256|ADCDATA  
没  
AD_samp1.hb= ADCDATAH; AD_samp1.lb= ADCDATA; 效率高。
```

### d)、表达式

```
if(vot_value<(vot_temp-1))      //效率低  
vot_temp=vot_temp-1;  
if(vot_value<vot_temp)          //效率高
```

```
-----  
a[i*j+2]=a[i*j];                // 效率低  
m=i*j;  
a[m+2]=a[m];                    // 效率高
```

### e)、逻辑运算

函数带返回值

```
dc_value=adc(2)/2;              //效率低  
dc_value=adc(2);  
dc_value=dc_value/2;            //效率高
```

### e)、数组参数

```
fun(a[], x, y)                  // 效率低  
fun(x, y, z)                    // 效率高, 其中 z 控制条件分支直接使用全局或局部数据直接操作。
```

### f) 嵌汇编原则

可使用的 R 寄存器限定使用 R0, R1, 如使用其他应确定此时关闭中断, 因为 R2-R5 作为中断现场的保护存贮使用。针对寄存器加 BANSEL 使编译器自动处理分区位置, 针对函数调用需要考虑支持 MOVP 指令系列芯片的切页, 可以使用 PAGESEL 修饰使编译器自动处理。前汇编变量只



能使用特殊寄存器或全局变量，C 变量到汇编转换原因，嵌汇编的操作变量需要加 “\_” 修改。

示例：

```
unsigned int a;
void fun1()
{
    __asm
        MOV R0, #0x33
        BANKSEL _a
        MOV _a, R0
    __endasm;
    Flag=1;
}
void fun2()
{
    Fun3();
    __asm
        PAGESEL _fun1
        CALL _fun1
        PAGESEL $
    __endasm;
}
```