CODEBLOCKS 快速上手教程

你是不是在想某一天 IAR 不再让 XX 的时候我该怎么办呢? KEIL 也不行!
(本人已不用 KEIL 很久了,现在用 SDCC)

你不是认为开源.LINUX 都是牛人玩的?是不是也想成为牛人!而不是用开源的东西仅仅是因为他不要钱?
是不是想哪一天也能为开源奉献点力量?

而这一切的开头都得你起码得会用这些软件吧!
一次和朋友聊天的时候朋友给我推荐了 CODEBLOCKS!
于是下载下来!天啦,英文! 不会,丢一边,这一放就是一年
不过当时记得里面有个 AVR 的!

一年后,也就是前几天,买了块 AVR 学习板的 PCB,是初版,上面 BUG 好多!还好本人焊接还可以,不成问题!

然后开始对开发软件选型!
一打听太多了
BASIC(不喜欢没理由! XX 掉)
IAR(用不起!虽然目前不是专门搞这个的,随便用用没问题,但如果这么想那么以后永远都只是随便搞搞)
Codevision(听说还可以!不过没用过,)
GCC FOR AVR(也就是 WINAVR GCC 可是大名鼎鼎呀,以后也想玩玩 LINUX,那就是他了)

于是安装了 AVR STUDIO 并安装了 WINAVR
发现写代码的时候没提示,不爽!
于是代码还是用我一直喜欢的 C_FREE 来写
但是再换回到 AVR STUDIO 里编译的时候要等一秒 AVR STUDIO 才会提示文件已被更新
看来这样子不行,太没效率了!

这时候想起了一年前见过的 CODEBLOCKS! 试试先
于是在电脑找了半天没找着,只要搜索了一下才发现,忽然发现这已是一年前的版本
于是在其发布网站下载了一个新版!

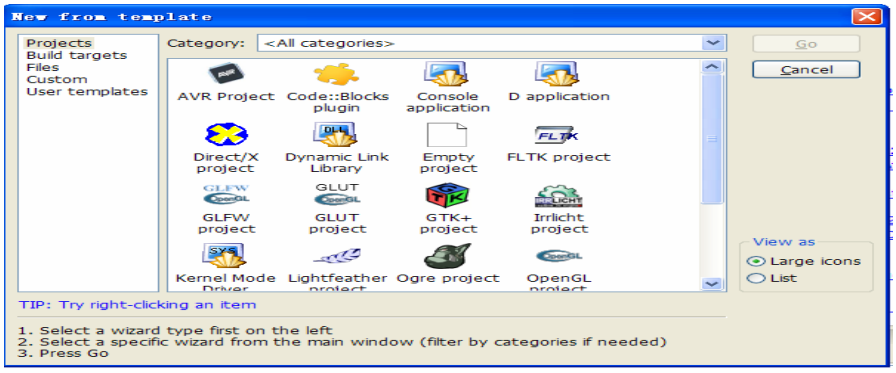
好了,我们先下载并安装好这次需要的!下载地址:
CodeBlocks: http://www.codeblocks.org/
WINAVR: http://winavr.sourceforge.net/

安装好! 

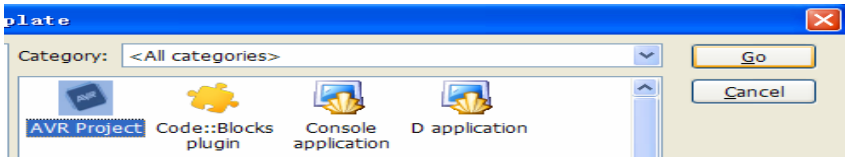

Create a new project

首先要新建一个工程!

哇!支持的还真多!

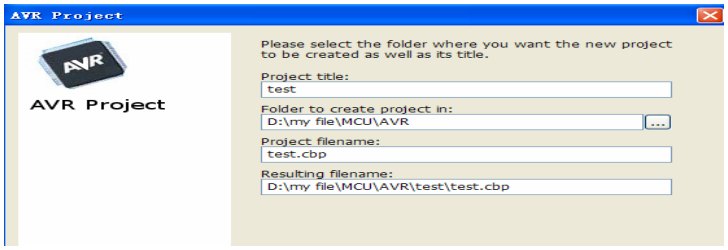


好!我们要的就是那个最上面的 AVR!



选择! 然后 Let’ s GO!

然后设置工程名称和存放路径!

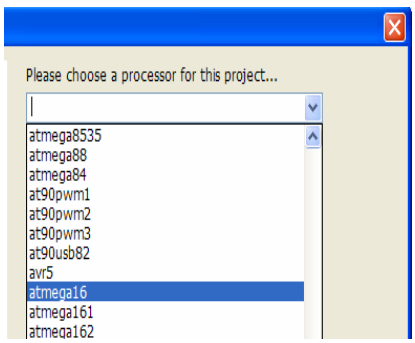


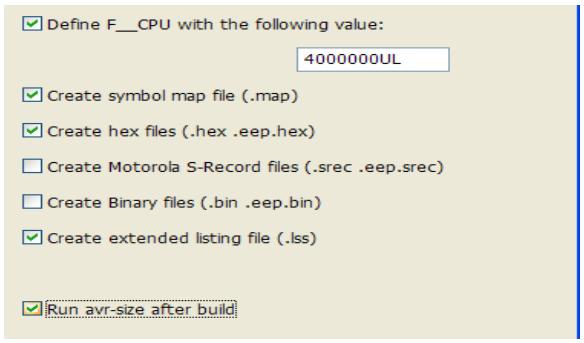
然后选择好 IC 型号和晶体频率(为 Delay 函数提供参数)

同时最好勾选上最下面的 RUN AVR_SIZA AFTER BUILD(编译成功显示程序的 SIZE)

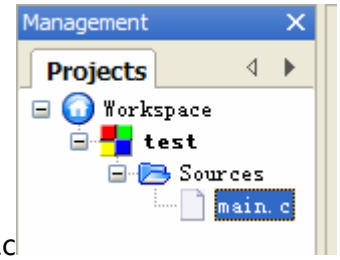
如果忘了选择待会儿也可以再手动添加

OK! Finish!



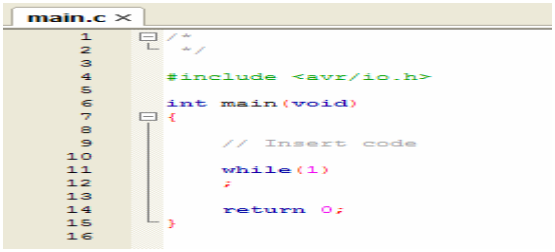


好了!现在可以写代码了!

看!默认她已经给我们创建了一个 main.c 

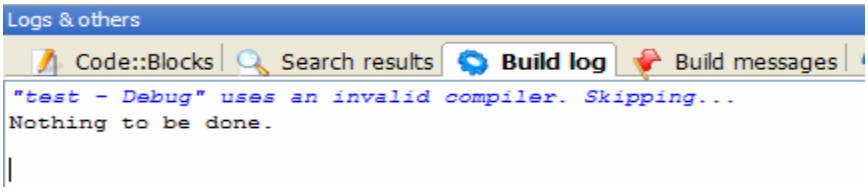
好!我们就用她作为主文件,双击以打开 main.c

OK! Very Good!

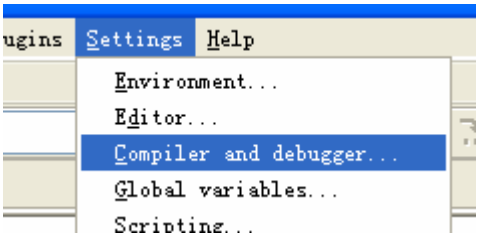


这已经是一个完整的程序了!直接编译下!

最左上角那个便是!点击下! 哇,不成功!



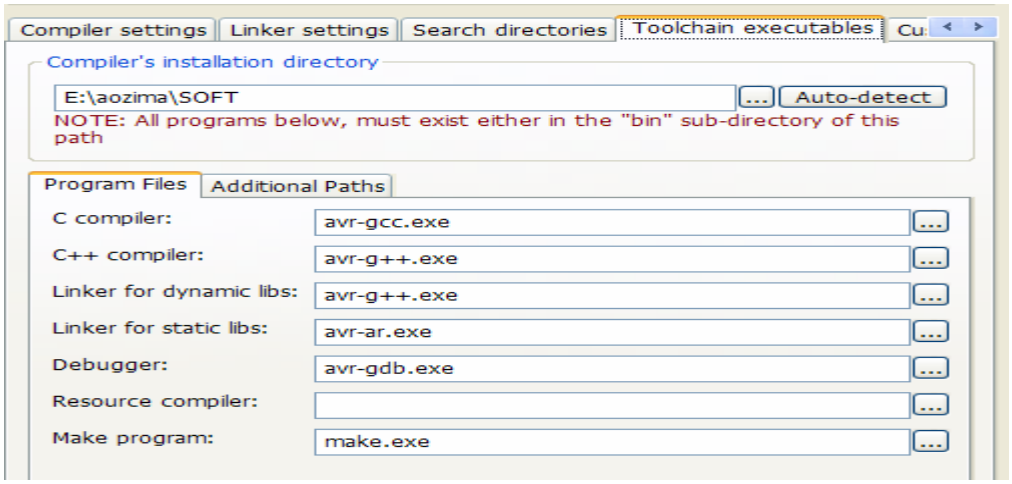
哦!原来没找到编译器! 那我们就设置下 WINAVR 的路径!(其实如果先安装 WINAVR 再安装 CODEBLOCK 的话她会自动搜索的)



哇,支持的好多!还有 ARM SDCC MSP430 是不是很 COOL?

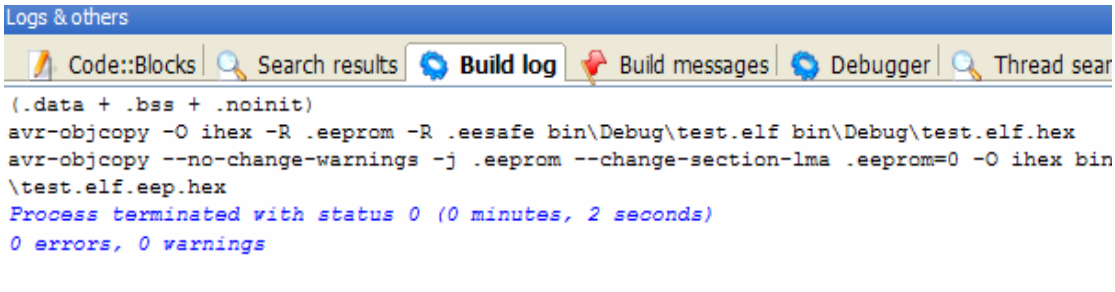
选择我们的 GNU AVR GCC(也就是 WINAVR)

然后确认 WINAVR 的目录



设置上面那个目录即可

好了,再编译下:



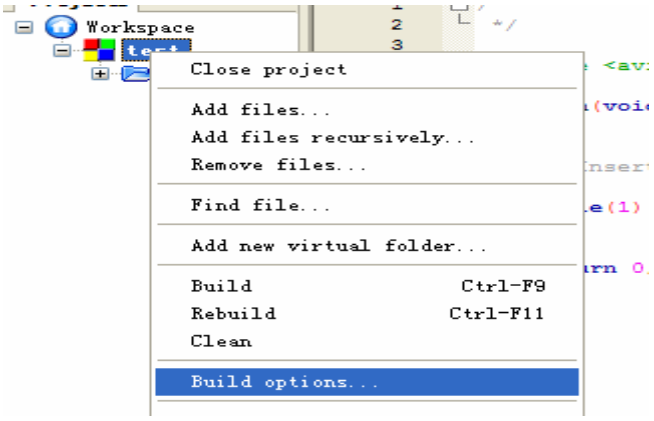
OK!~ 好了!

噫？怎么显示 HEX 在哪,有多大?嘿嘿,往上拉一拉即可,

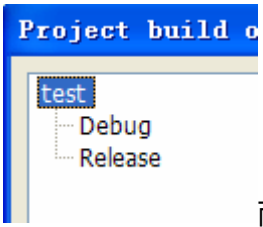
```
avr-size --mcu=atmega16 --format=avr bin\De
AVR Memory Usage
----------------
Device: atmega16
Program:     160 bytes (1.0% Full)
(.text + .data + .bootloader)
Data:          0 bytes (0.0% Full)
(.data + .bss + .noinit)
```

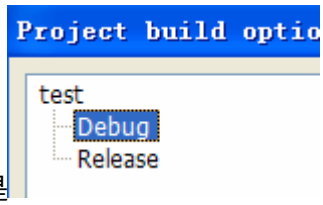
每次都得拉一拉呀?

嘿,有办法的!



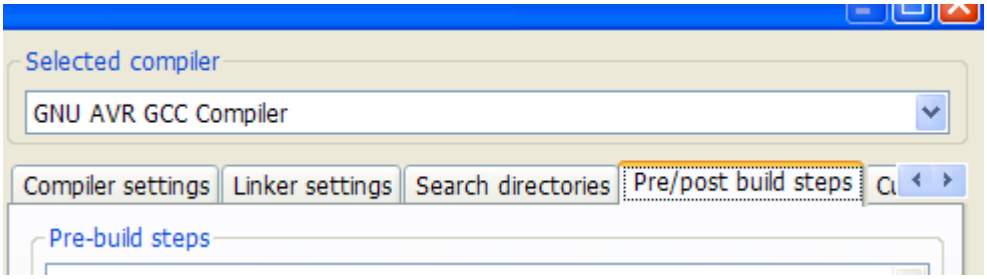
然后选择



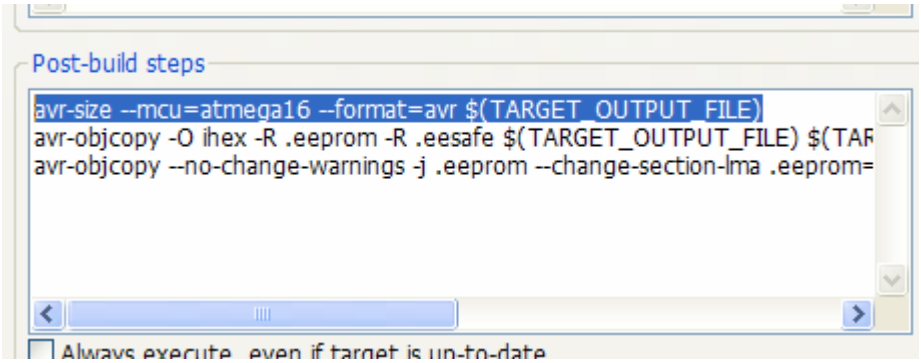
而不是


这将是全局的意思(这里的全局只是针对这个工程哦!)

然后切换到:



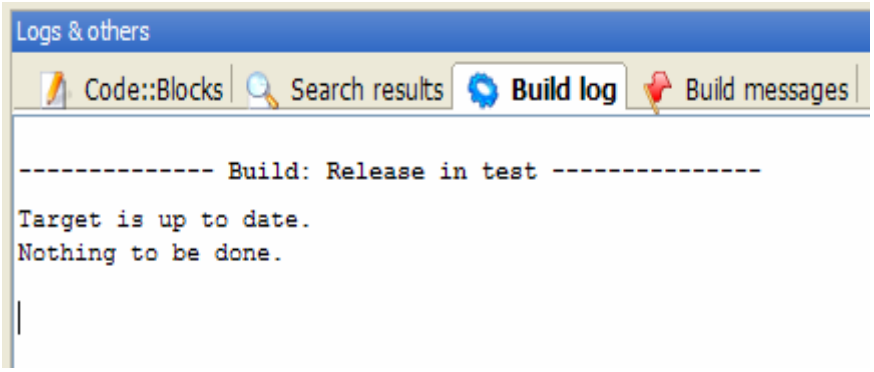
把这一行剪切到最下面即可



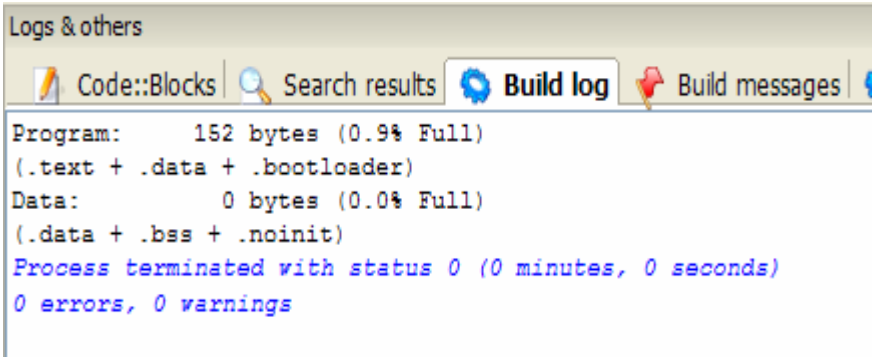
(小提示:下次新建工程的时候忘了选择 RUN AVR_SIZA AFTER BUILD 也可以自己添加这么

一行哦,嘿,一般人我不告诉他)

然后再编译下:



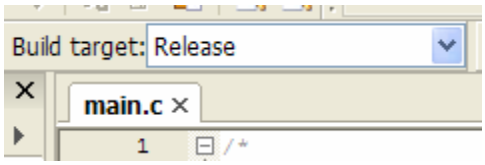
噫? 哦! 原来这个刚刚已经编译过了,我们并没修改他,那么只有选择  重新编译次了!



看,直接显示在最底下,是不是我们最想要的?

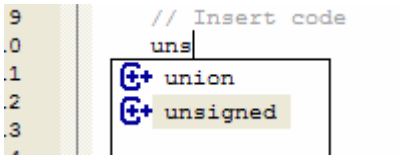
目前 HEX 文件在\bin\Debug 目录下

调试成功后发行的时候选择:



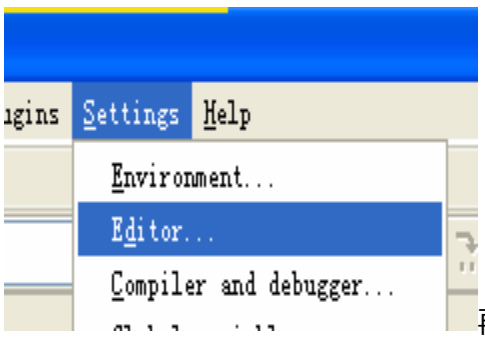
HEX 文件将产生在\bin\Release 目录下,并对程序进行最大优化(优化类型和等级也可以自定义)

你问我为什么最喜欢这个? 第一次确认选择这个的时候就是因为她有代码提示功能

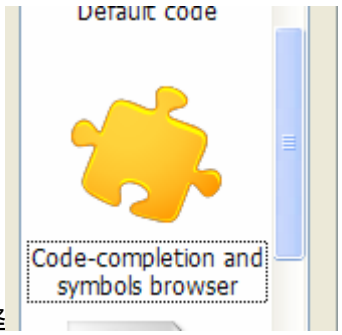


不过默认的提示要输入 4 个字符才会出现,而已比较慢

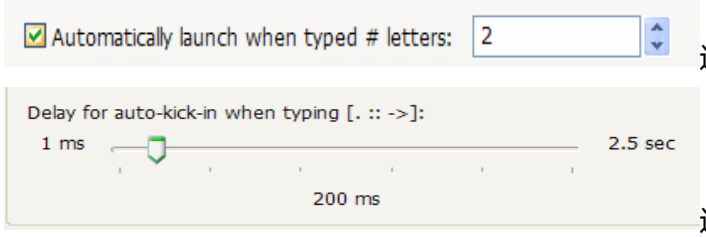
那么我们:



再选择



然后:



这里设置输入几个字符后提示

这里设置提示延时

同时也喜欢她的自由,开源

我可不是因为她是自由免费的才用的哦

总有一天,我会换到 LINUX 下面的,现在起就开始用能换到 LINUX 下面也能用的软件不是到

时候要上手快一些!

再个她支持的东西多,换对象不用换 IDE 是不是又要好一些?

快来用 CODEBLOCKS 吧!

以上愚见仅代表个人意见!

残剑饮血

master@aozima.cn

2008 年 8 月 31 日 18 时 54 分