

Keil MDK 开发环境

MDK介绍

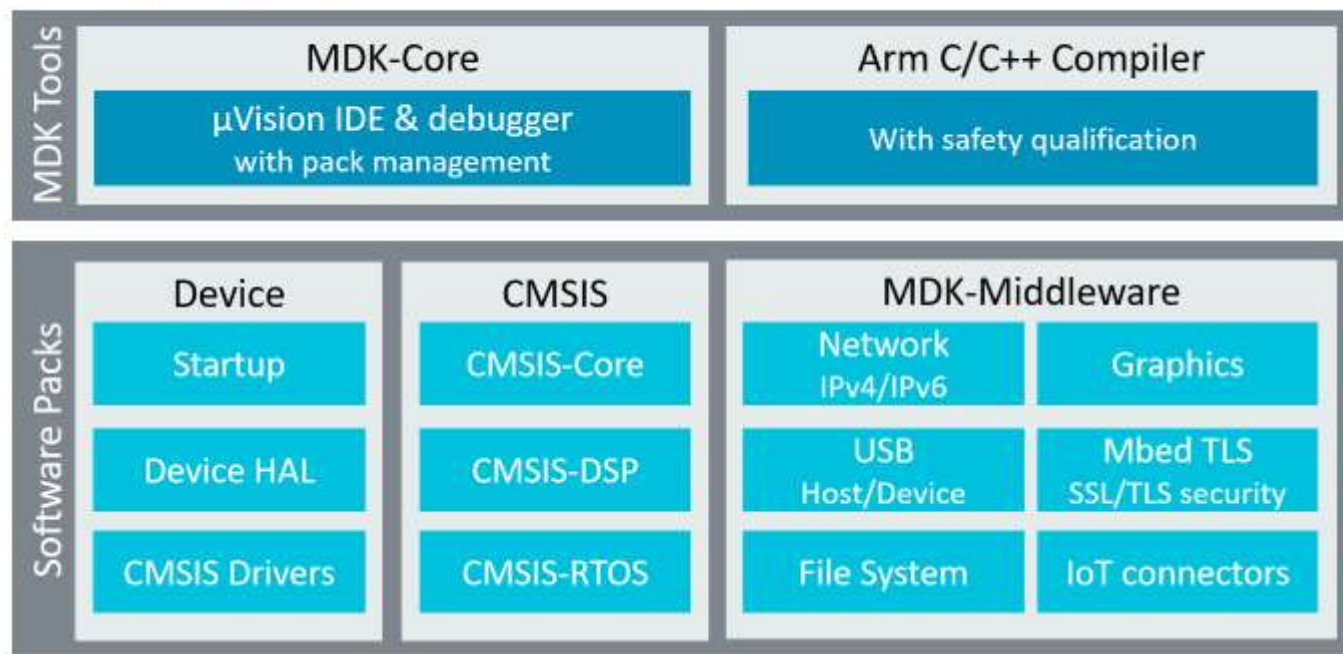
- Keil是业界最受欢迎的51单片机开发工具之一，它拥有流畅的用户界面与强大的仿真功能。
- RealView MDK —— RealView Microcontroller Development Kit （简称为 MDK）。
- RealView MDK出众的价格优势和功能优势，势将成为ARM软件开发工具的标准，预计一年之内，RealView MDK将占有国内ARM开发工具市场的90%以上。
- MDK-ARM软件为基于Cortex-M、Cortex-R4、ARM7、ARM9处理器设备提供了一个完整的开发环境。MDK-ARM专为微控制器应用而设计，不仅易学易用，而且功能强大，能够满足大多数苛刻的嵌入式应用。
- MDK-ARM有四个可用版本，分别是MDK-Lite（免费评估版）、MDK-CortexM、MDK-Standard、MDK-Professional。所有版本均提供一个完善的C/C++开发环境，其中MDK-Professional还包含大量的中间库。
- 与Keil MDK4及之前版本不同，Keil MDK5分成MDK Core和Software Packs两部分。MDK Core主要包含uVision5 IDE集成开发环境和ARM Compiler5。Software Packs则可以在不更换MDK Core的情况下，单独管理（下载、更新、移除）设备支持包和中间件更新包。

MDK特点

- 完美支持Cortex-M、Cortex-R4、ARM7和ARM9系列器件。
- 业行领先的ARM C/C++编译工具链
- 确定的Keil RTX，小封装实时操作系统（带源码）
- μ Vision5 IDE集成开发环境，调试器和仿真环境
- TCP/IP网络套件提供多种的协议和各种应用
- 提供带标准驱动类的USB 设备和USB 主机栈
- 为带图形用户接口的嵌入式系统提供了完善的GUI库支持
- 显示关于程序运行的完整代码覆盖率信息
- 执行分析工具和性能分析器可使程序得到最优化
- 大量的项目例程帮助你快速熟悉MDK-ARM强大的内置特征
- 符合CMSIS (Cortex微控制器软件接口标准)
- 完善的开发工具手册、设备数据手册和用户向导

MDK5组成

Product Components



1.MDK Core 又分成四个部分：uVision IDE with Editor（编辑器），ARM C/C++ Compiler（编译器），Pack Installer（包安装器），uVision Debugger with Trace（调试跟踪器）。uVision IDE 从 MDK4.7 版本开始就加入了代码提示功能和语法动态检测等实用功。

2.Software Packs（包安装器）又分为：Device（芯片支持），CMSIS（ARM Cortex 微控制器软件接口标准）和 Mdkidleware（中间库）三个小部分，通过包安装器，我们可以安装最新的组件，从而支持新的器件、提供新的设备驱动库以及最新例程等，加速产品开发进度。

MDK5组成

MDK5 较比5之前的MDK对比：

MDK5把芯片的器件支持包与MDK分离，所以在 MDK5 安装完成后，要让 MDK5 支持 STM32F407 的开发，还要安装 STM32F4 的器件支持包：Keil.STM32F4xx_DFP.1.0.8.pack（STM32F4 的器件包）。



若要支持 STM32F103的开发，也同样的需要F1的支持包：Keil.STM32F1xx_DFP.2.4.0.pack。

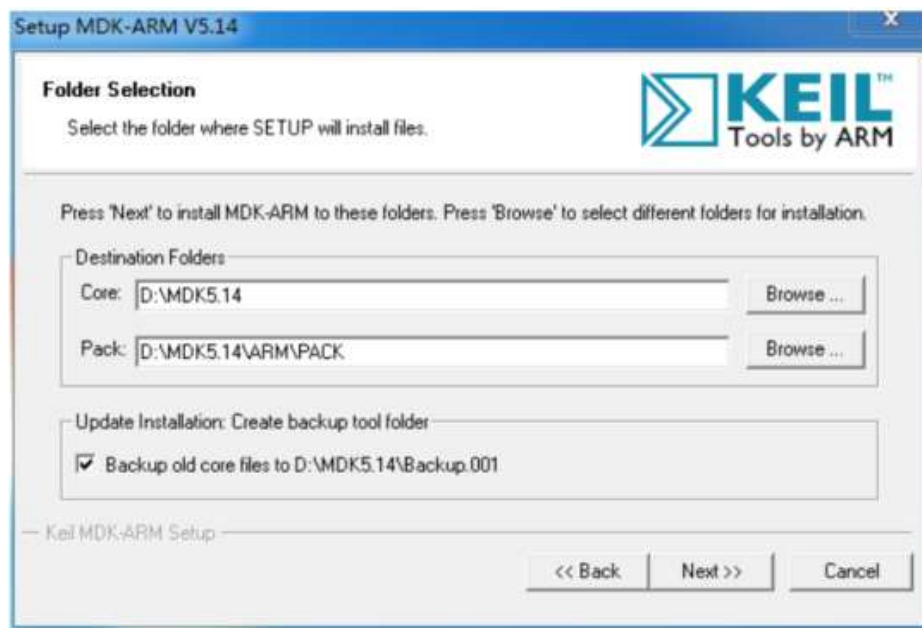


MDK5平台的搭建~安装

双击



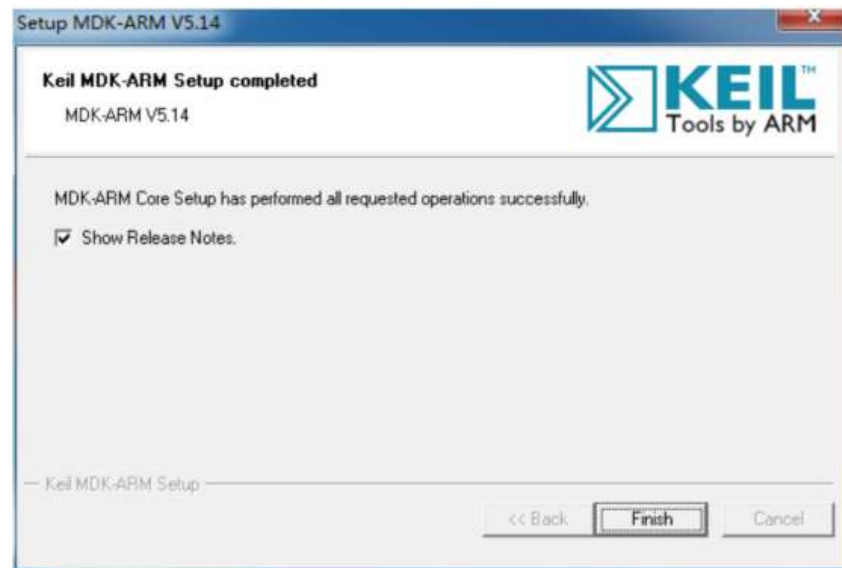
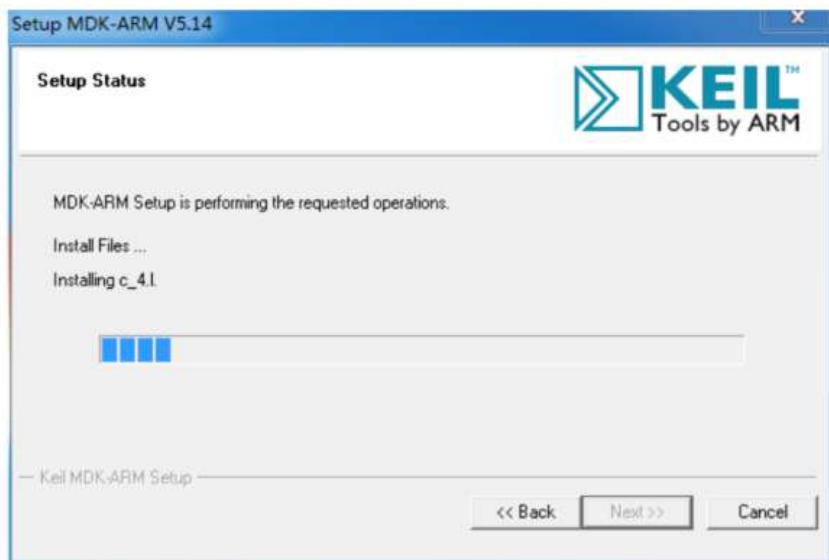
进入安装->next



这里以安装到D盘目录下为例，注意的是安装的路径一定不能有中文名字

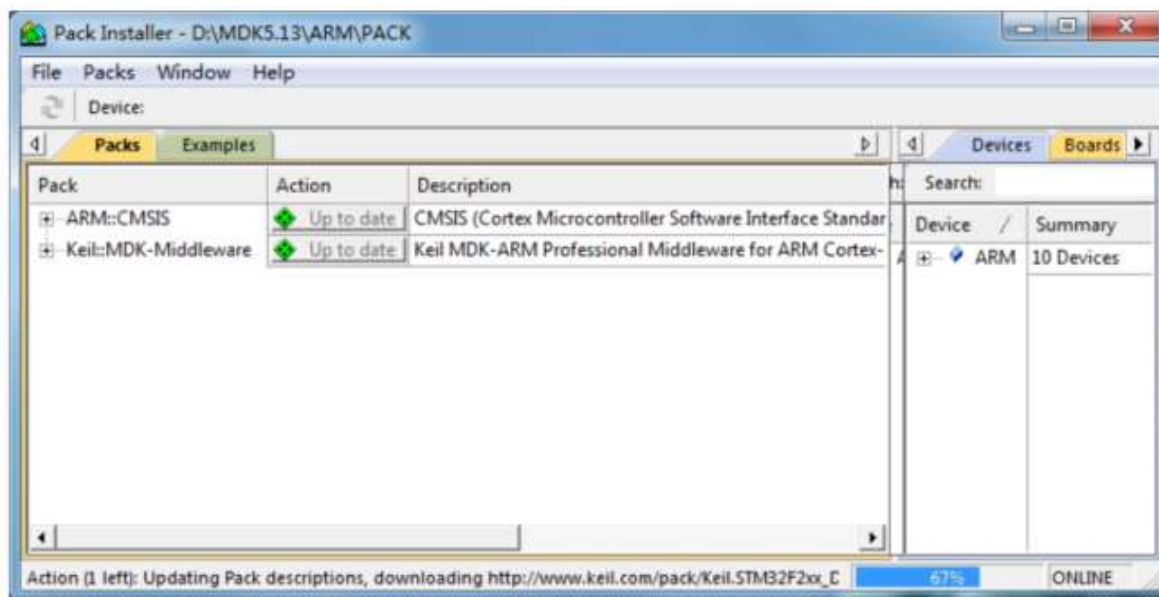
MDK5平台的搭建~安装

按要求的步骤，就可以开始安装的，安装界面如下：



MDK5平台的搭建~安装

最后点击**Finish**完成安装后，MDK可能会自动弹出**Pack Installer**的界面



这是更新芯片支持包的界面，可能会出现**File download failed**的错误，关闭即可。由于已经提供了**STM32F4**的支持包，关闭即可。

（支持包可在官网下载<http://www.keil.com/dd2/pack>）

MDK5平台的搭建~安装

安装完成后，我们要进行对MDK的破解：

点解File ----->License Management 调出注册管理界面，复制CID号



The image shows a 'License Management' dialog box with a blue title bar and a red close button. It contains several tabs: 'Single-User License', 'Floating License', 'Floating License Administrator', and 'FlexLM License'. The 'Single-User License' tab is selected. Inside this tab, there are two main sections. The left section is titled 'Customer Information' and contains three text boxes: 'Name:' with the value 'liu liu', 'Company:' with the value 'ALIENTEK', and 'Email:' with the value 'liujun6037@foxmail.com'. The right section is titled 'Computer ID' and contains a text box with the value 'CZ6WA-U1Z18' and a button labeled 'Get LIC via Internet...'. Below these sections is a table with three columns: 'Product', 'License ID Code...', and 'Support Period'. The first row of the table contains the text 'MDK-Lite' and 'Evaluation Version'. At the bottom of the dialog, there is a text box labeled 'New License ID Code (LIC):' followed by two buttons: 'Add LIC' and 'Uninstall...'. At the very bottom, there are two buttons: 'Close' and 'Help'.

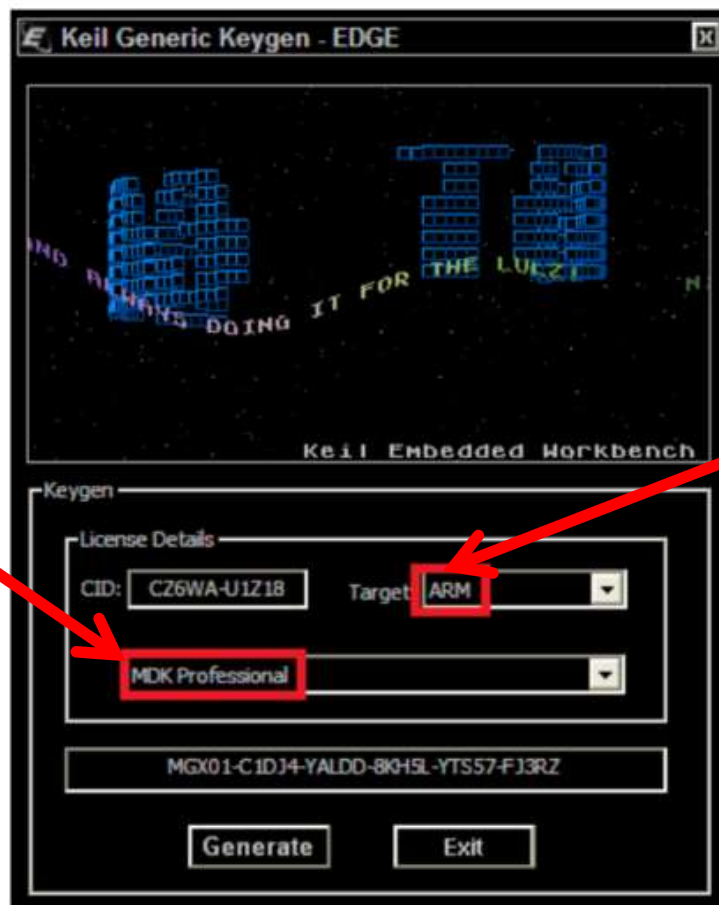
| Product | License ID Code... | Support Period |
|----------|--------------------|----------------|
| MDK-Lite | Evaluation Version | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

MDK5平台的搭建~安装

然后打开 keygen.exe 进行对MDK5 的破解，需要用到前一步的CID号，点击 Generate 获得破解码

注意

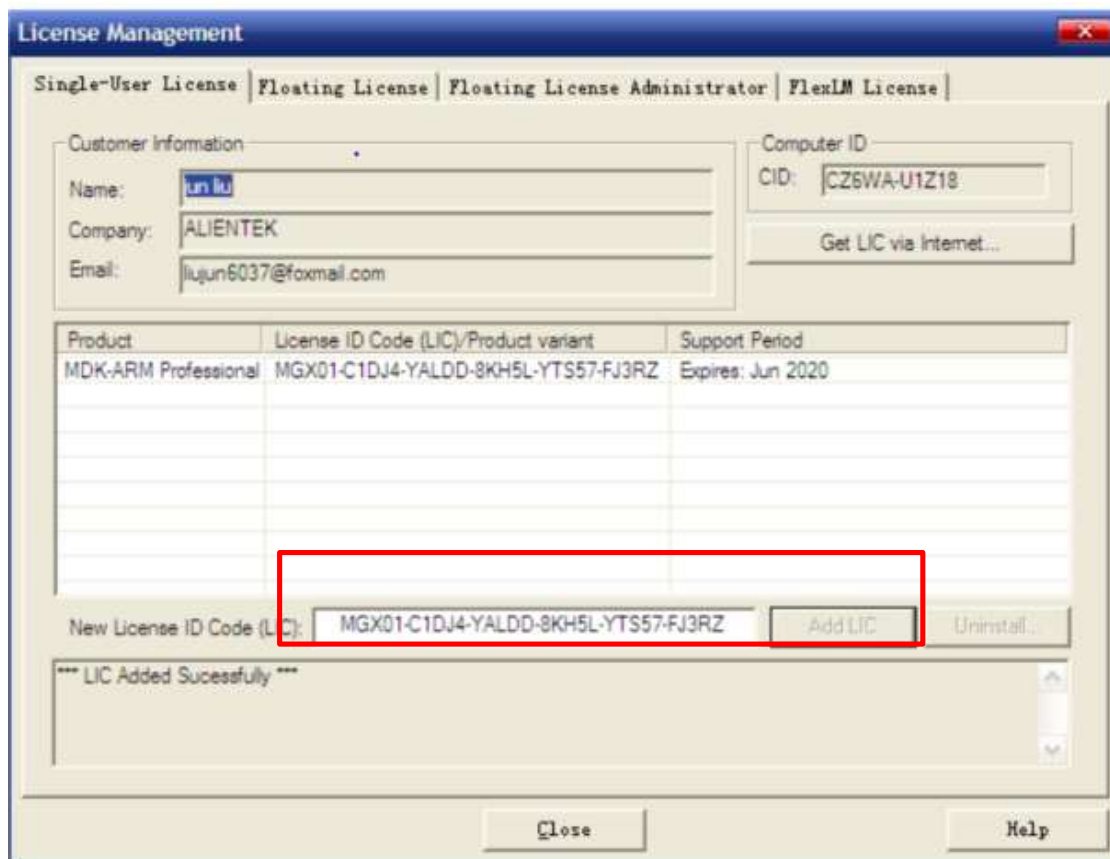
注意



MDK5平台的搭建~安装

把破解码粘贴到下图File ----->License Management中，进行破解

注意：步骤正常，若点击Add LIC没有反应，请用管理员权限打开MDK5在进行破解。



搭建工程环境框架1

1. 在建立工程之前，我们建议用户在电脑的某个目录下面建立一个文件夹，后面所建立的工程都可以放在这个文件夹下面，这里我们建立一个文件夹为 **Template**。作为工程的根目录文件夹。

2. 然后为了方便我们存放工程需要的一些其他文件，这里我们还新建下面 5 个子文件夹：

apps -- 用来存放应用层文件

doc -- 用来存放日志（例如修改的部分，增加的部分）

drivers -- 用来存放片外外设驱动程序

libraries -- 用来存放标准外设库中移植过来的文件

project -- 用来存放我们开发环境所生成的一些文件（例如可执行文件，新建的工程也保存在此文件夹下。）

至于这些文件夹名字，实际上是可以任取的，我们这样取名只是为了方便识别。

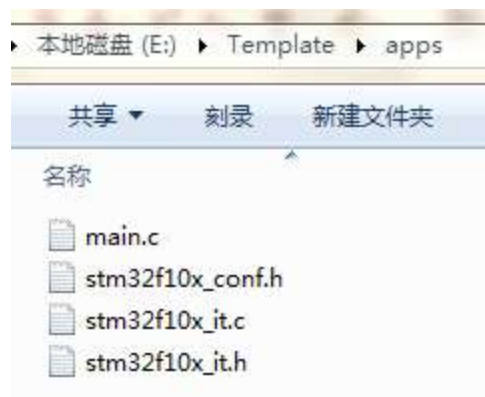


搭建工程环境框架2

接下来我们要复制工程模板需要的一些其他头文件和源文件到我们工程。
首先定位到目录：

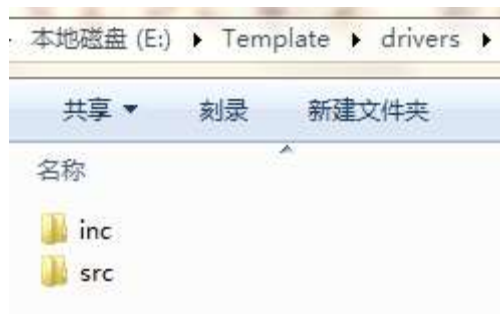
`\STM32F10x_StdPeriph_Lib_V3.5.0\Project\STM32F10x_StdPeriph_Examples\GPIO\IOToggle`

将里面的 4 个文件 `main.c`，`stm32f1xx_conf.h`，`stm32f1xx_it.c`，`stm32f1xx_it.h`，复制到 `apps` 目录下面。这是STM32F10x验证板的GPIO示例的应用层程序。



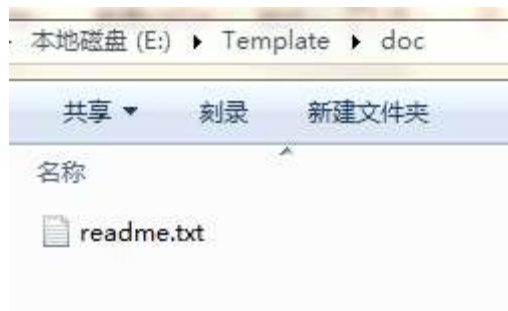
搭建工程环境框架3

在**drivers**目录下创建两个子目录**inc**和**src**，分别用于存放外设驱动程序的头文件和源码。



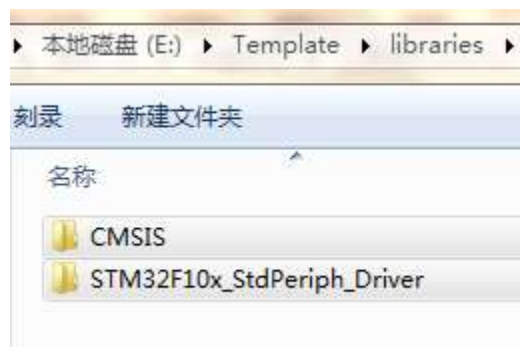
搭建工程环境框架4

在doc目录下创建一个文本文档，如：**readme.txt**，用于记录日志信息。



搭建工程环境框架5

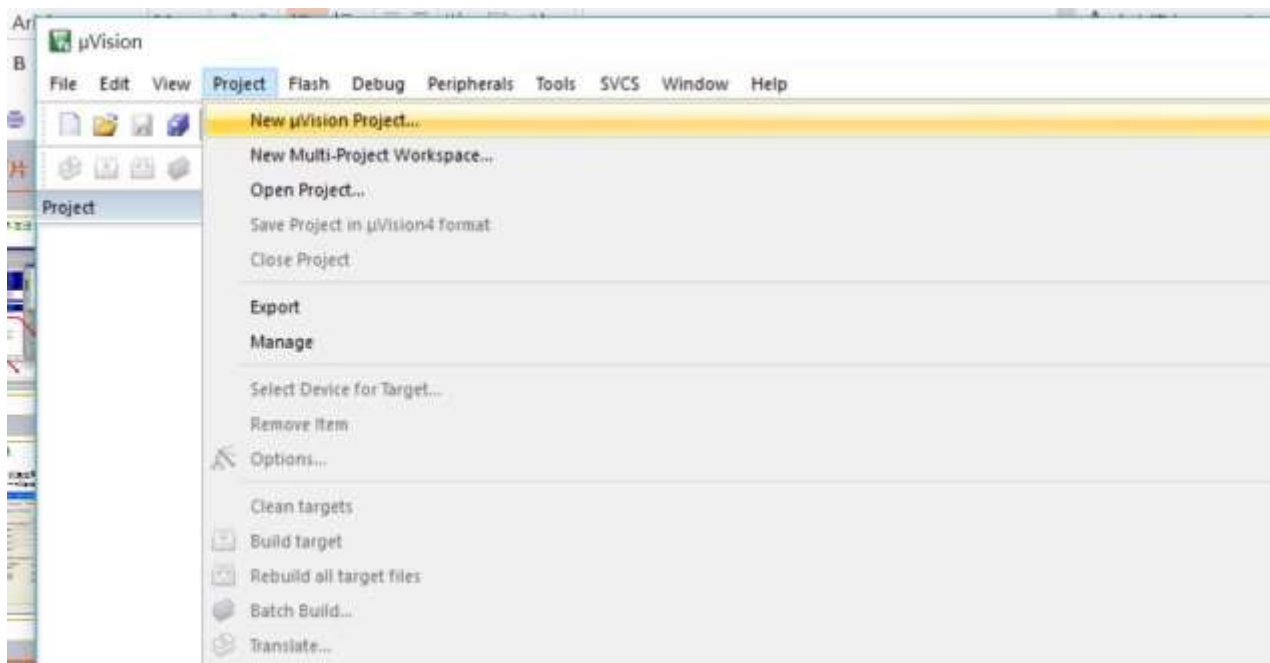
打开官方固件库包，定位到我们之前准备好的固件库包的目录：
\\STM32F10x_StdPeriph_Lib_V3.5.0\Libraries 下面，将目录下面的 CMSIS，
STM32F10x_StdPeriph_Driver两个文件夹 copy 到我们刚才建立的 libraries 文件夹下面。



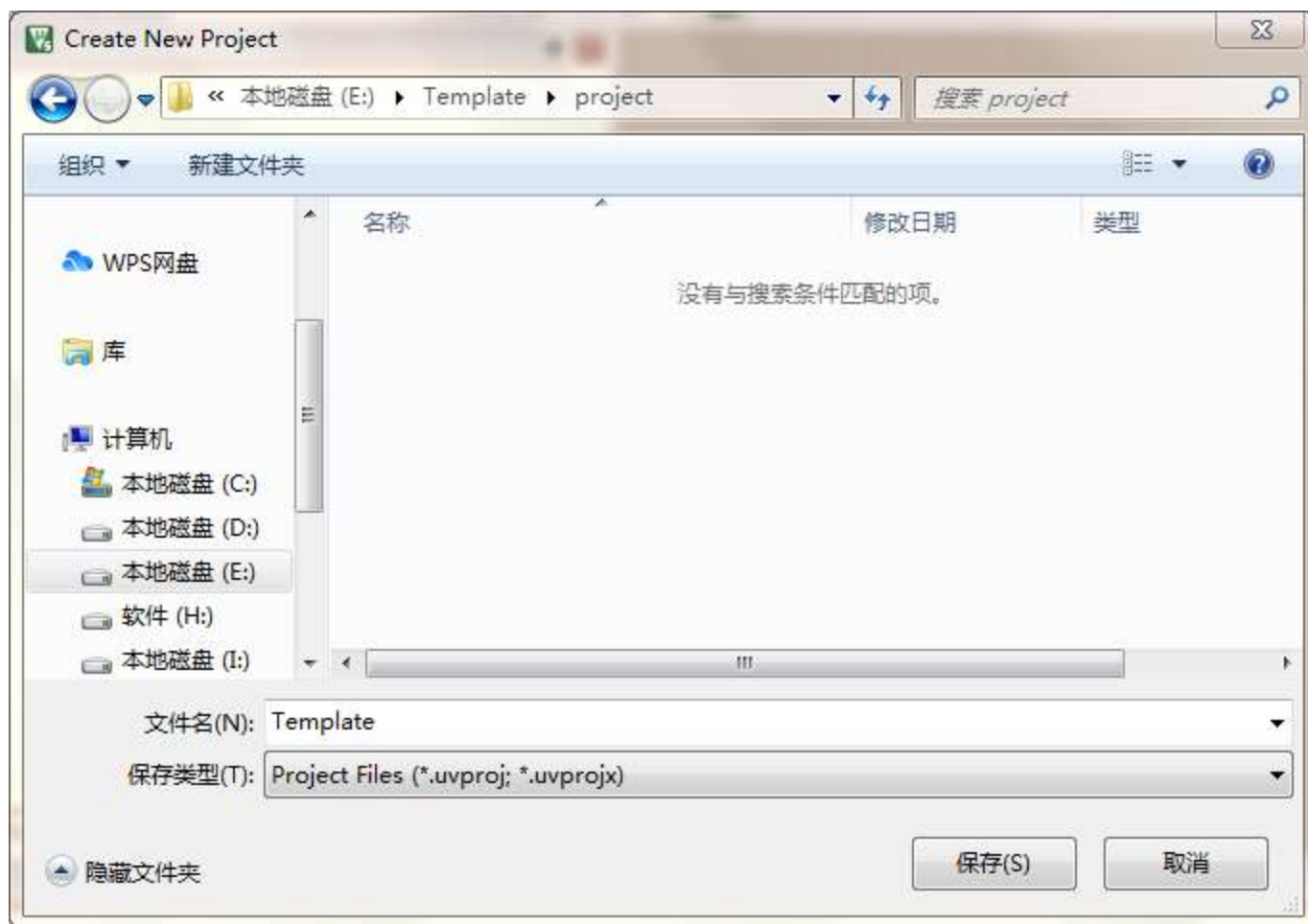
搭建工程环境框架6 - 创建新工程1

1. 打开 Keil，点击 Keil 的菜单：Project → New Uvision Project，然后将目录定位到刚才建立的文件夹Template之下的project子目录，工程的名字可以根据自己的要求设置，如：Template。project文件夹就用来存放工程的文件。

■ 创建新的工程 Project->New uVision Project

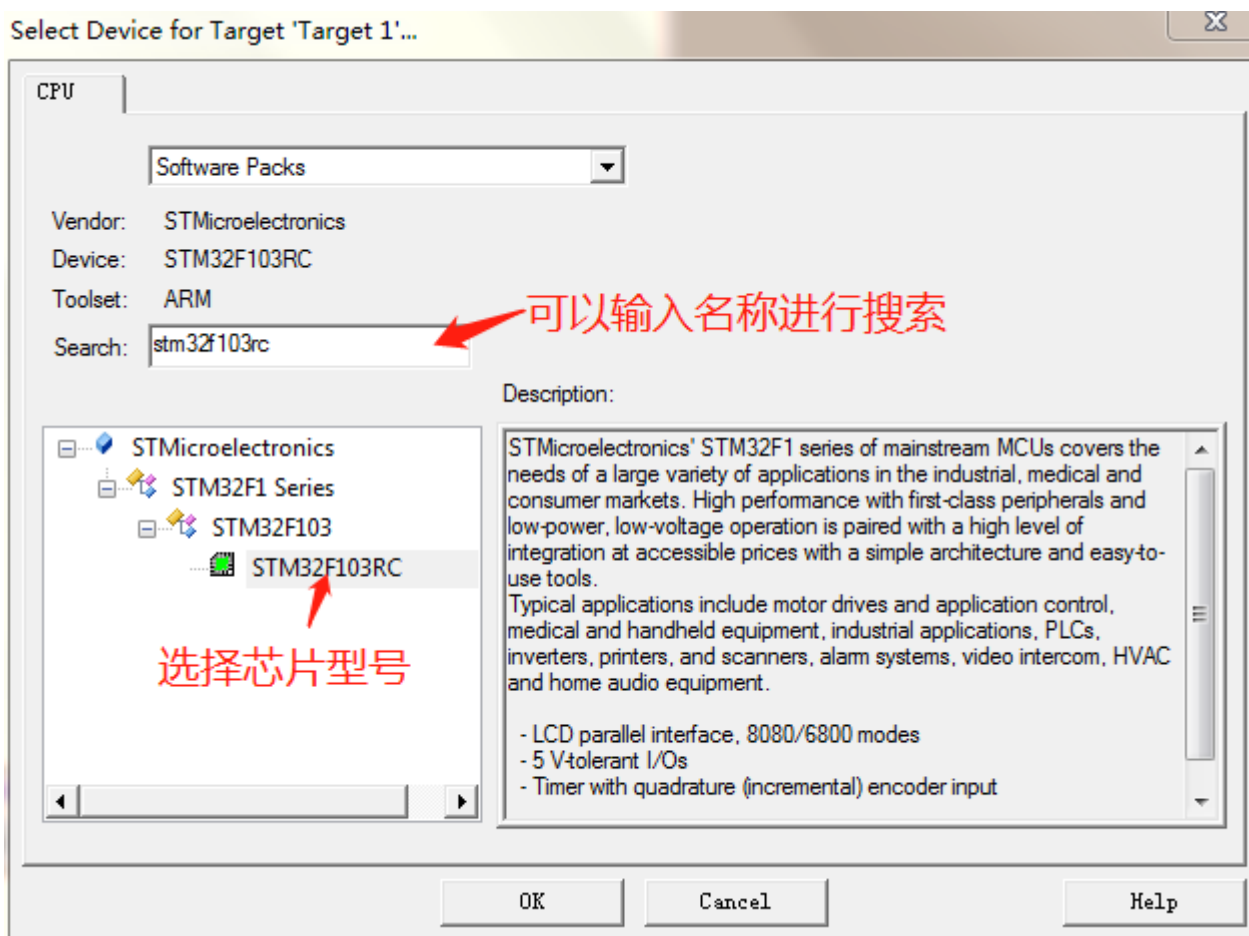


搭建工程环境框架6 - 创建新工程2

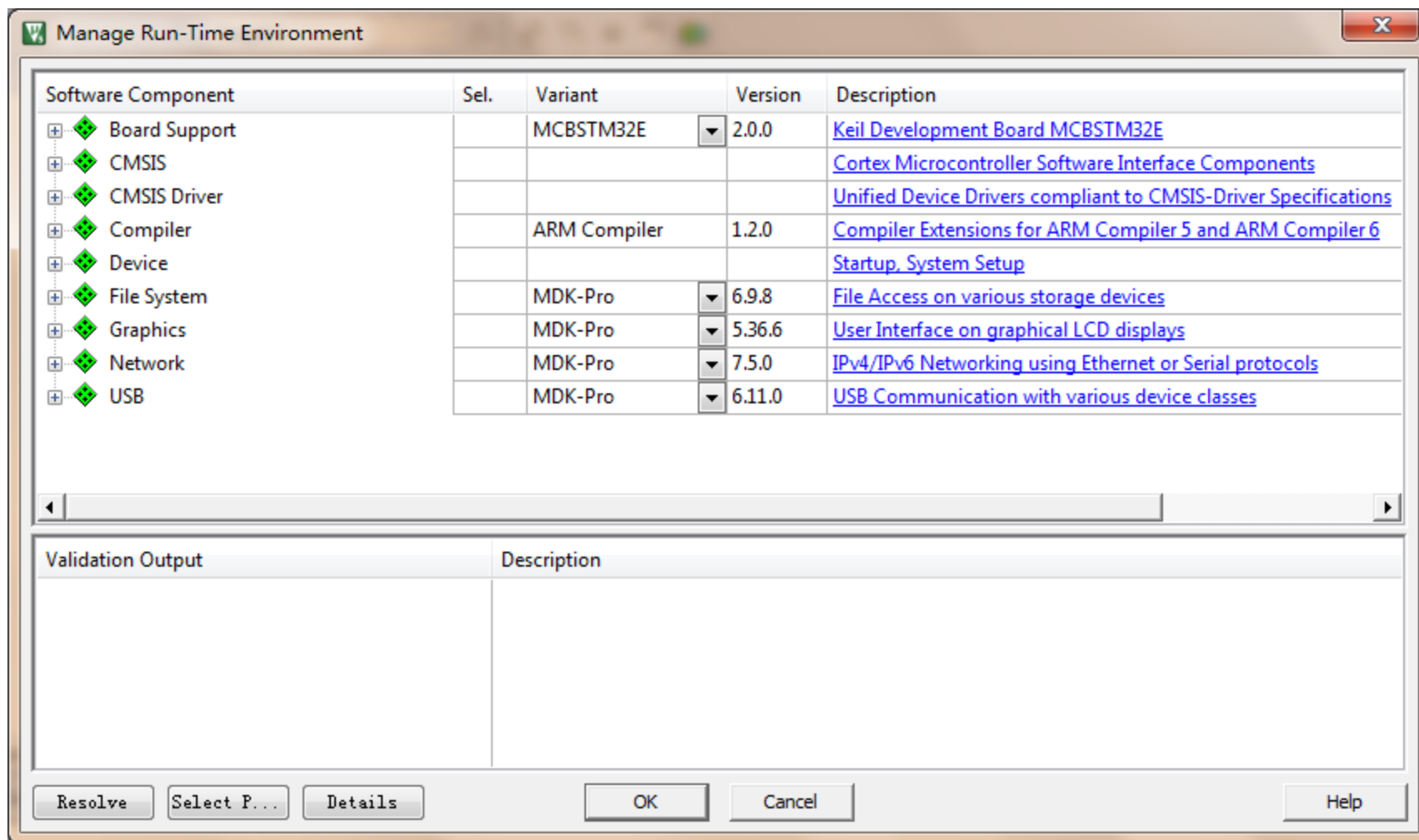


搭建工程环境框架6 - 创建新工程3

■ 选择处理器型号STM32F103RC



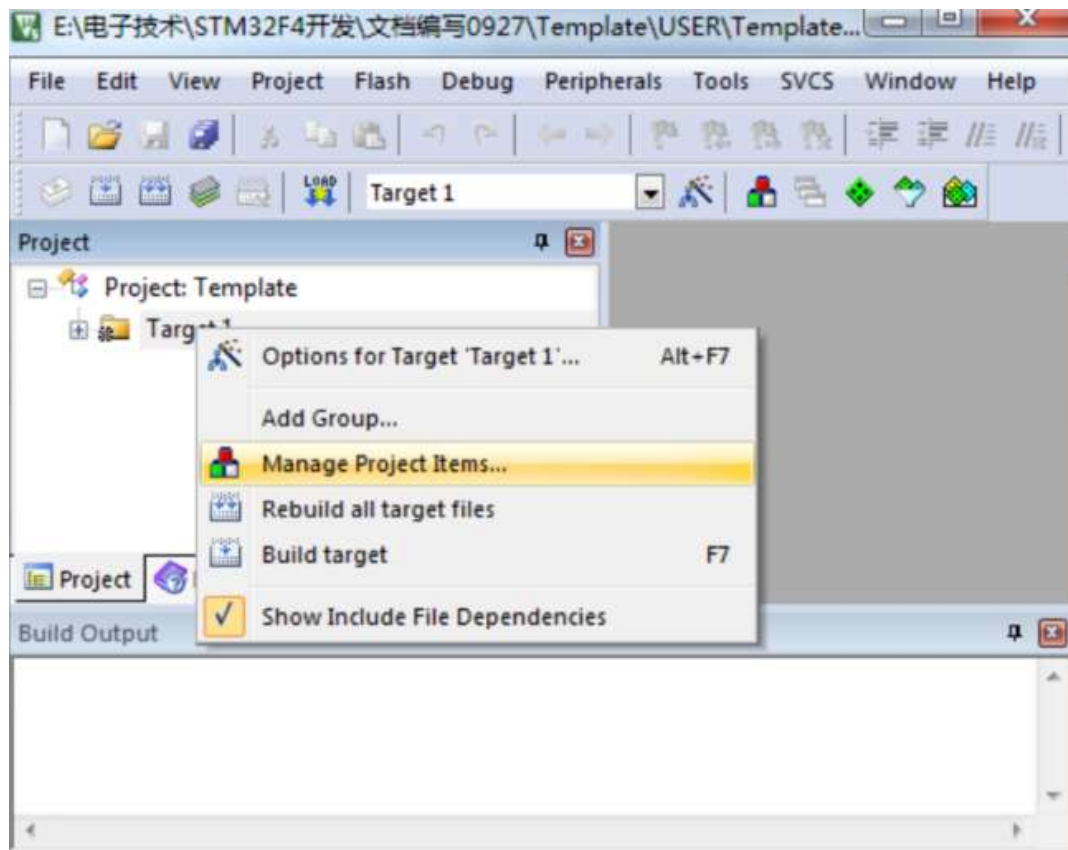
搭建工程环境框架6 - 创建新工程4



■ 点OK即可

搭建工程环境框架7-管理项目

完成以上步骤后，下面我们需要把之前的那些文件加入我们的工程中去。右键点击 **Target1**，选择 **Manage Project Items**，如下图：

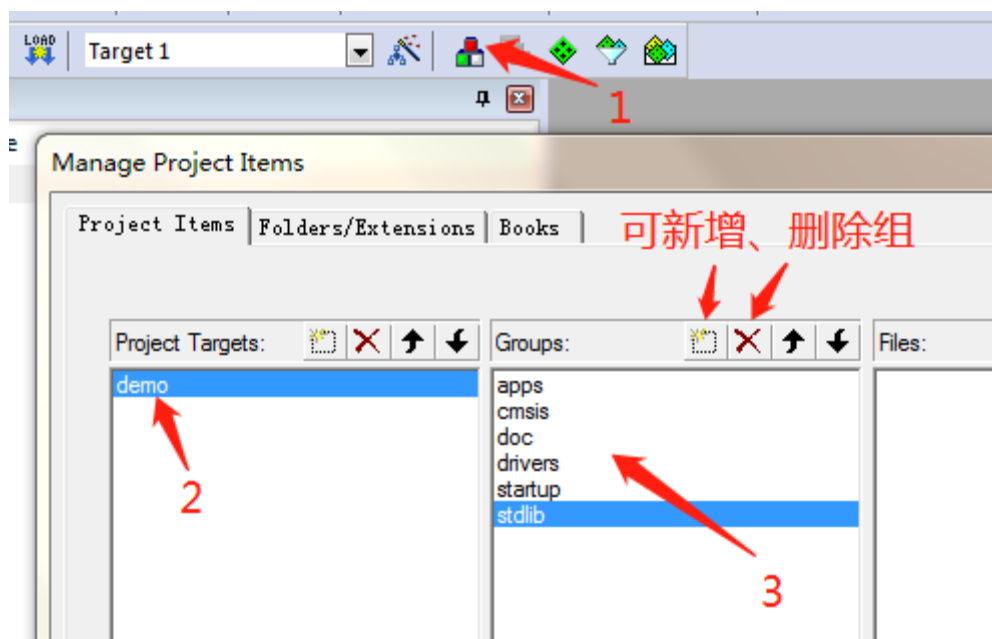


或点击品字形图标也可：



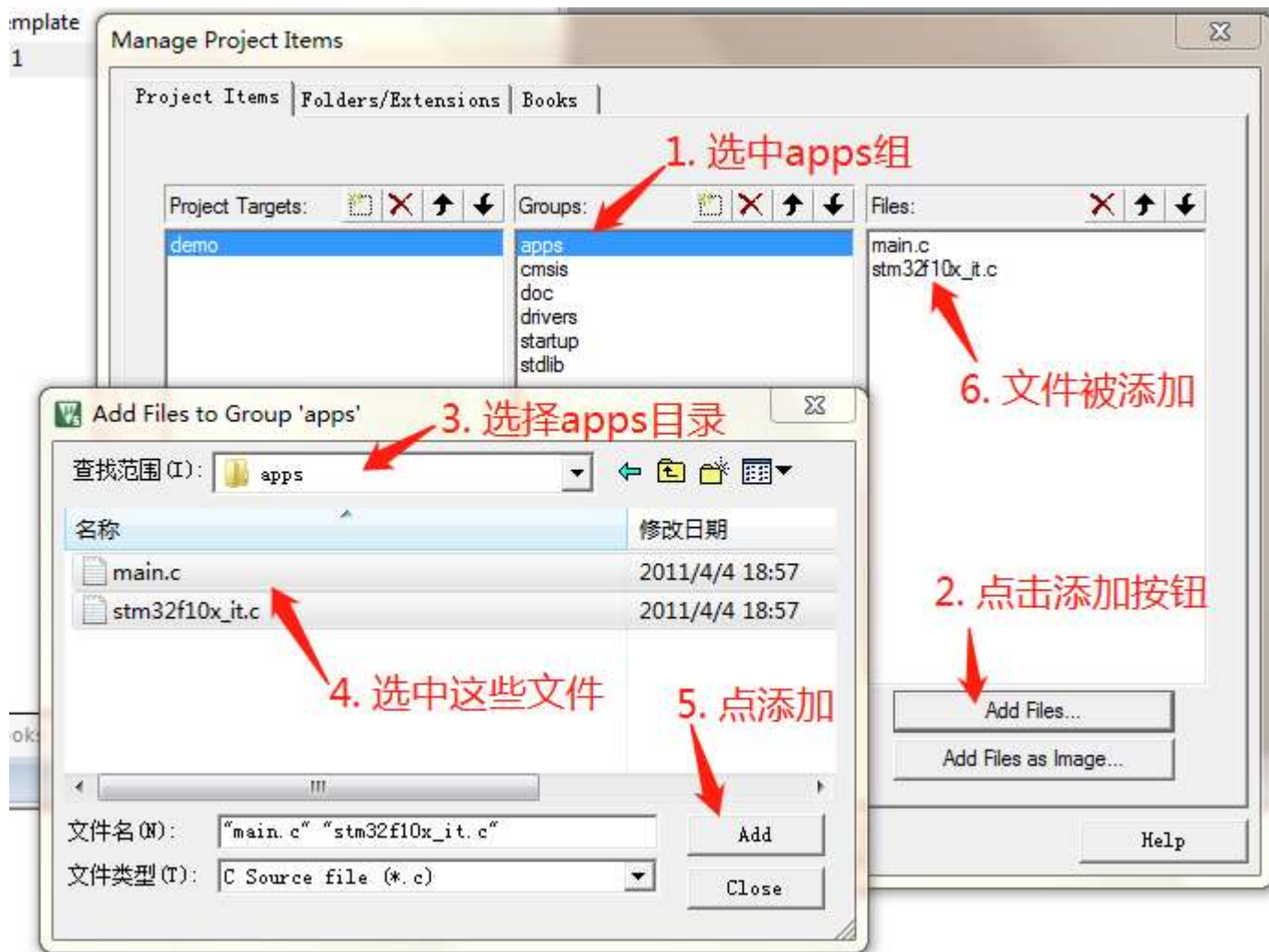
搭建工程环境框架7-目标下建组

Project Targets一栏，我们将Target1名字修改为demo，然后在Groups一栏删掉SourceGroup1，建立6个 Groups: apps, cmsis, doc, drivers, startup, stdlib。然后点击 OK，可以看到我们的Target名字以及 Groups 情况如下图：



搭建工程环境框架7-添加文件到各组

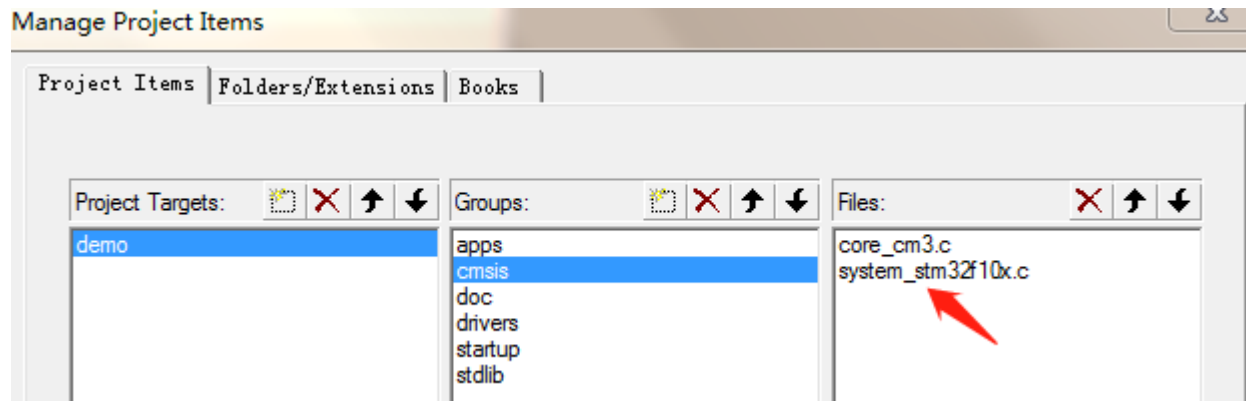
添加文件到
apps组。



搭建工程环境框架7-添加文件到各组

添加文件到cmsis组。

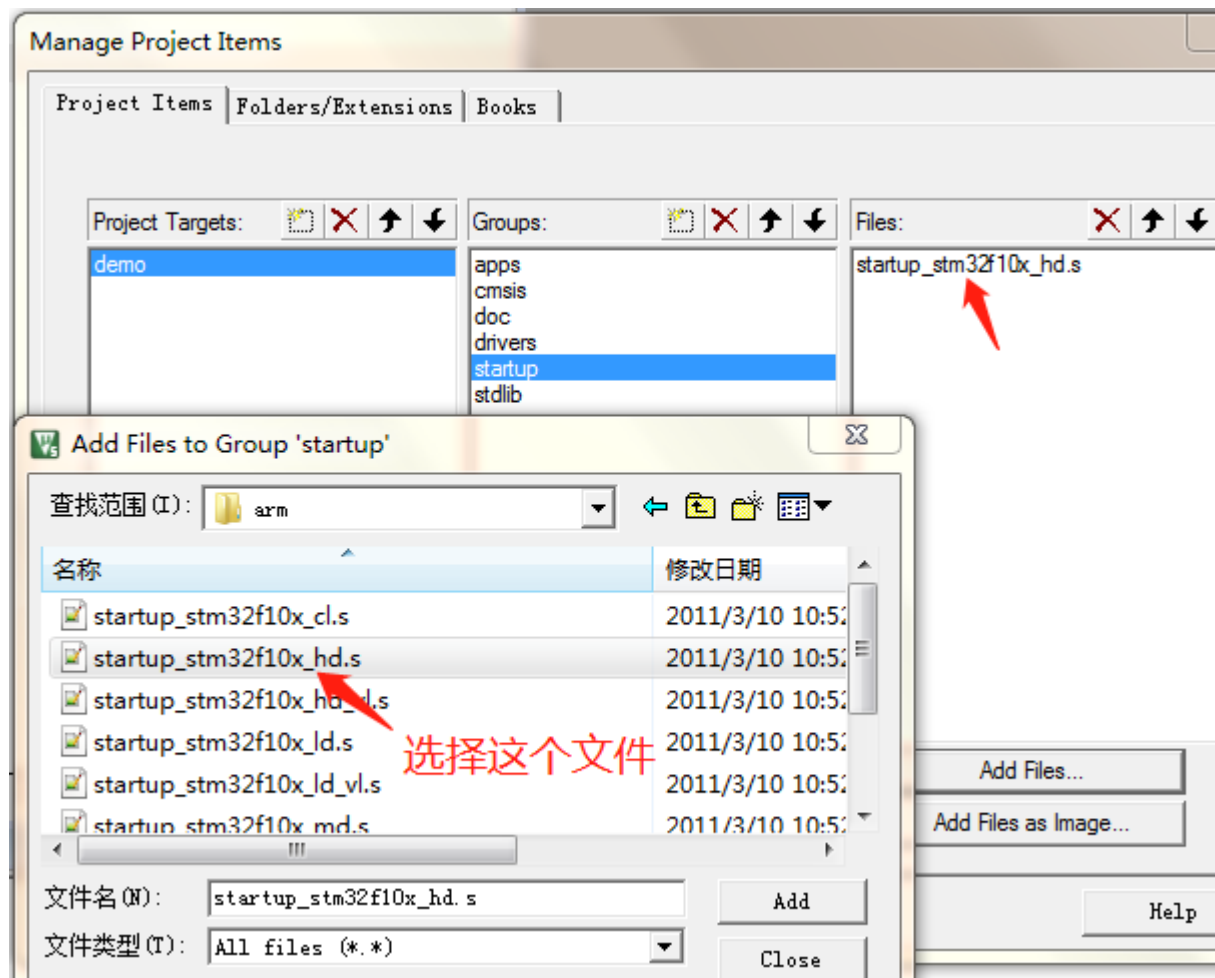
- core_cm3.c的路径：
 \libraries\CMSIS\CM3\CoreSupport
- system_stm32f10x.c的路径：
 \libraries\CMSIS\CM3\DeviceSupport\ST\STM32F10x



搭建工程环境框架7-添加文件到各组

添加文件到startup组，路径：

\\libraries\CMSIS\CM3\DeviceSupport\ST\STM32F10x\startup\arm



搭建工程环境框架7-添加文件到各组

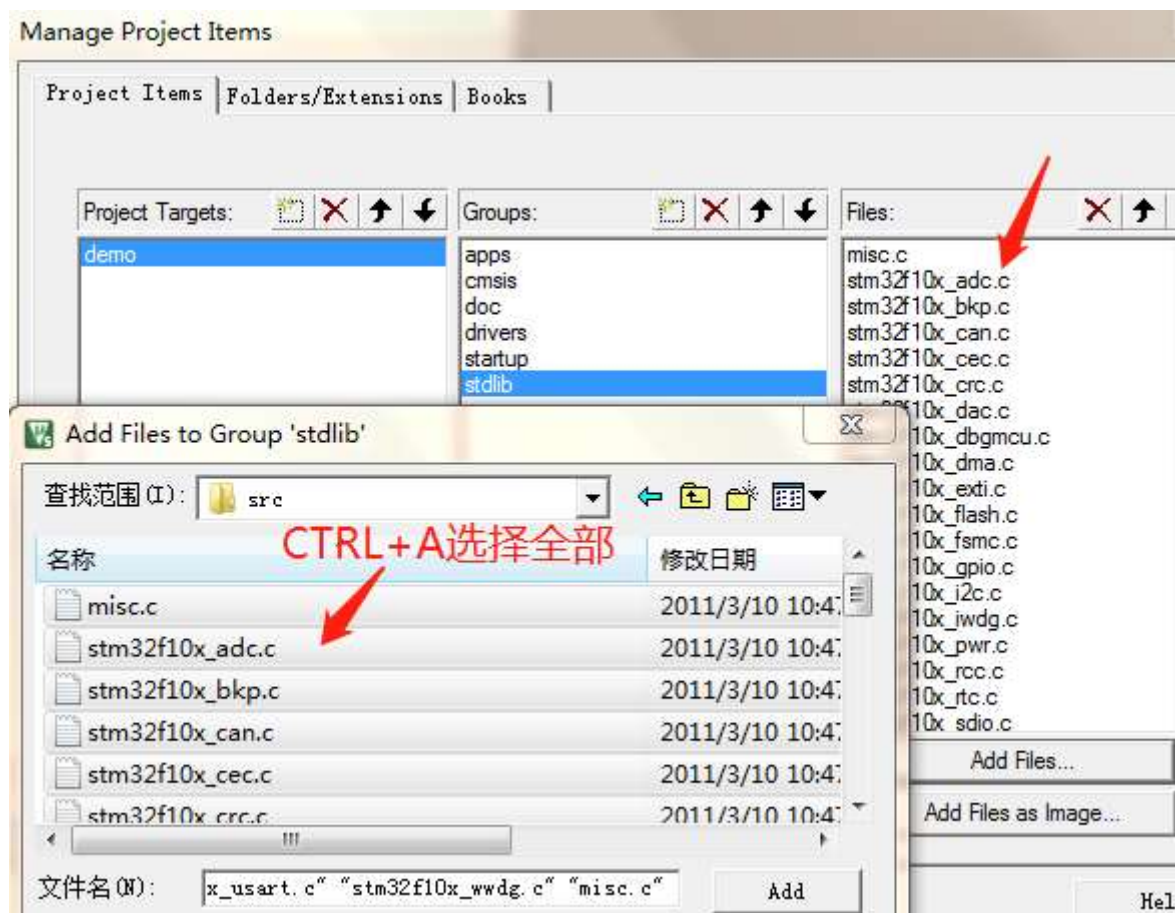
启动文件目录下有多个文件，该如何选择呢？
要根据容量、互联、超值等进行选择。

| 启动文件 | 区别 |
|--|--|
| startup_stm32f10x_ld.s | ld: low-density 小容量，FLASH 容量在 16-32K 之间 |
| startup_stm32f10x_md.s | md: medium-density 中容量，FLASH 容量在 64-128K 之间 |
| startup_stm32f10x_hd.s | hd: high-density 大容量，FLASH 容量在 256-512K 之间 |
| startup_stm32f10x_xl.s | xl: 超大容量，FLASH 容量在 512-1024K 之间 |
| 以上四种都属于基本型，包括 STM32F101xx、STM32F102xx、STM32F103xx 系列 | |
| startup_stm32f10x_cl.s | cl:connectivity line devices 互联型，特指 STM32F105xx 和 STM32F107xx 系列 |
| startup_stm32f10x_ld_vl.s | vl:value line devices 超值型系列，特指 STM32F100xx 系列 |
| startup_stm32f10x_md_vl.s | |
| startup_stm32f10x_hd_vl.s | |

搭建工程环境框架7-添加文件到各组

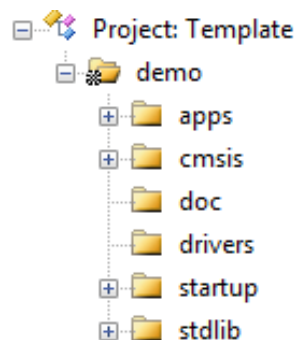
添加文件到**stdlib**组。路径：

`\libraries\STM32F10x_StdPeriph_Driver\src`



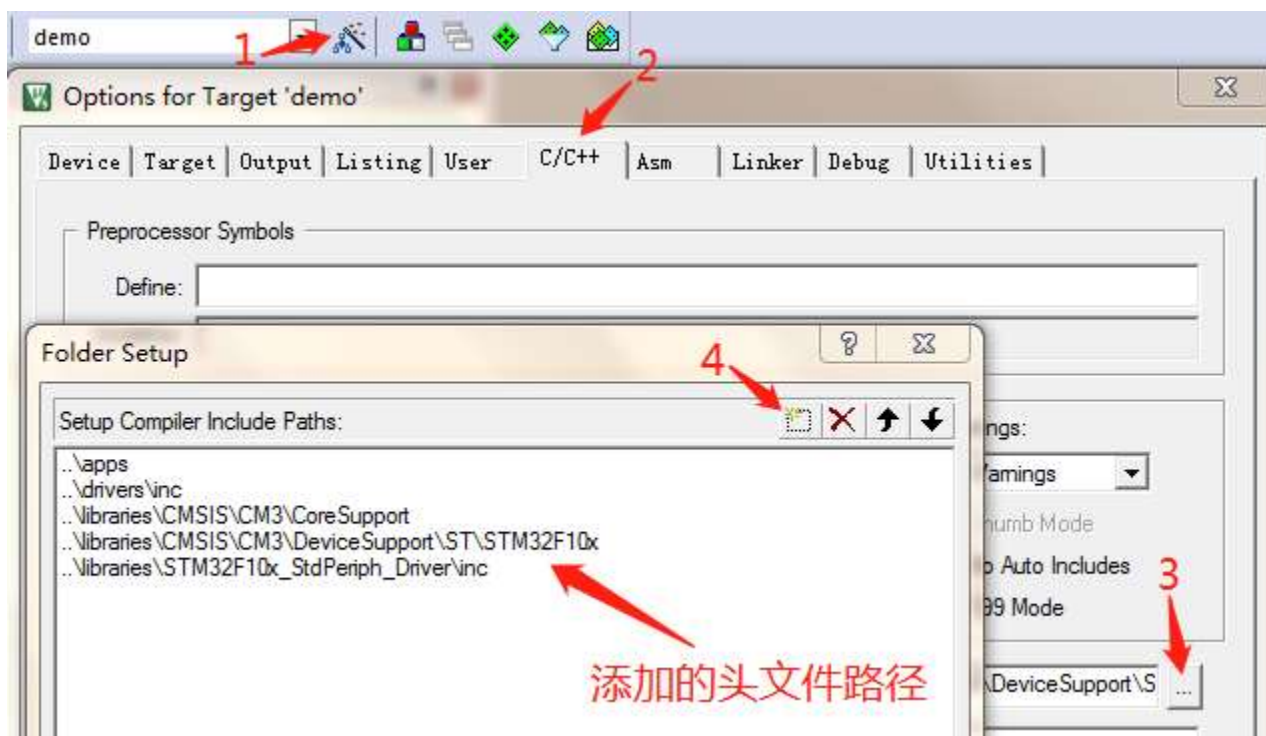
搭建工程环境框架7-添加文件到各组

所有文件添加完成，点击 **OK** 键。项目结构如下：



搭建工程环境框架8-添加头文件

头文件只需要设置路径即可，无需添加到各组。



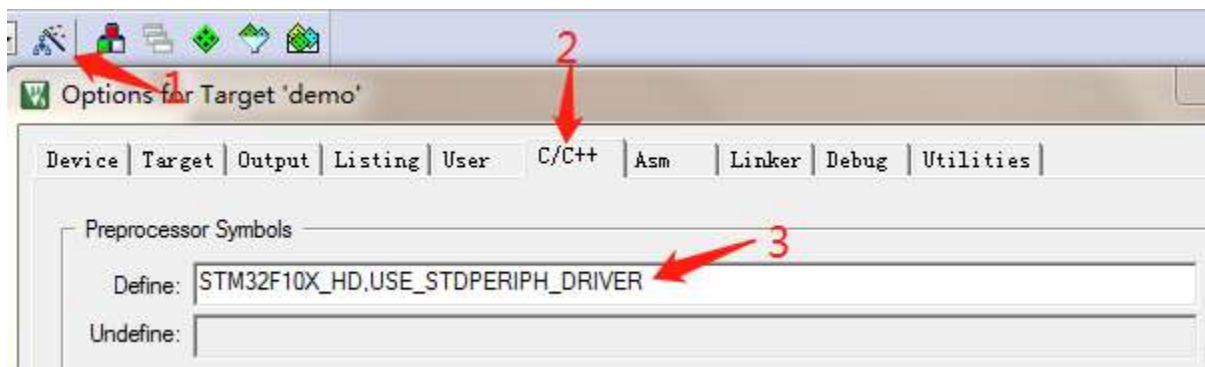
搭建工程环境框架9-全局宏定义

有两个全局宏需要定义：

STM32F10X_HD,USE_STDPERIPH_DRIVER。

STM32F10X_HD 表示大容量芯片，要根据不同型号进行不同配置；

USE_STDPERIPH_DRIVER 表示要使用标准库，不定义则表示不使用。



搭建工程环境框架10-配置输出

配置编译后生成可执行文件或库的路径、名称。**ISP**下载需要**hex**文件。



搭建工程环境框架11-配置生成bin文件

如果使用JLINK/STLINK烧录，则必须使用bin文件，而不能使用hex文件。

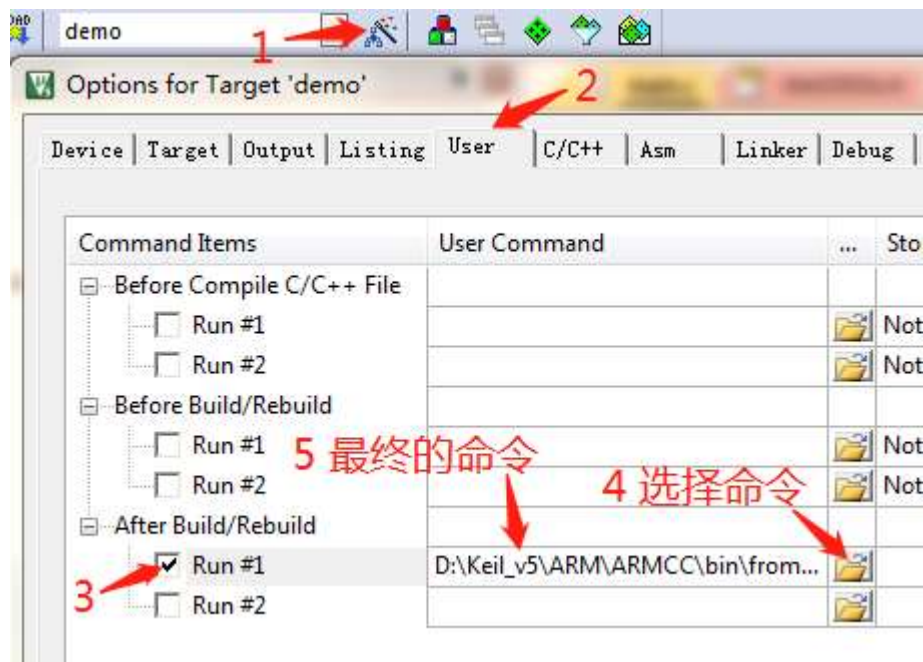
bin文件需要keil自带的fromelf.exe工具来生成。方法是输入命令：

```
D:\Keil_v5\ARM\ARMCC\bin\fromelf.exe --bin Objects\Template.axf -o  
Objects\Template.bin
```

fromelf.exe命令一般需要带路径，因为一般不会将其路径配置到PATH环境变量。注意这个路径不要照抄以上，必须按实际的路径。

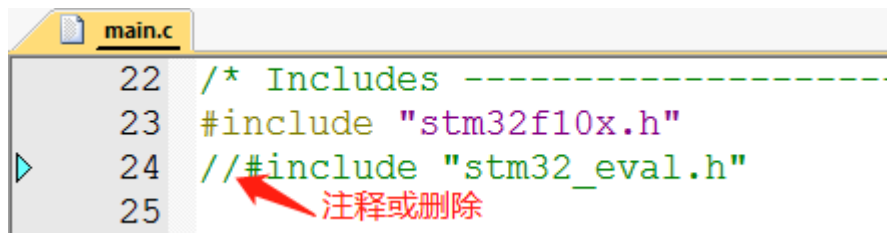
--bin 表示生成bin文件。-o 后跟输出文件路径和名称。

Objects\Template.axf 是输入文件的路径和名称。



修改main.c

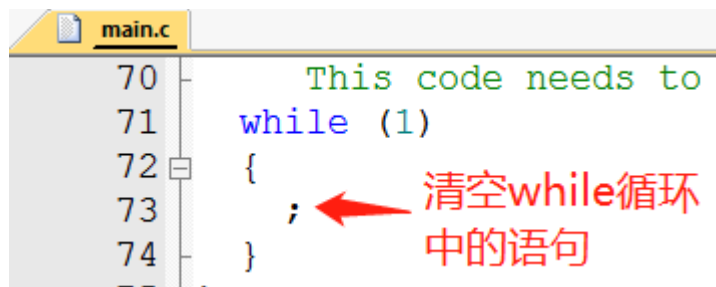
将main.c文件中的#include “stm32_eval.h”注释或删除掉。这是ST官方验证板的测试代码头文件，我们不需要。



```
22  /* Includes -----
23  #include "stm32f10x.h"
24  // #include "stm32_eval.h"
25
```

注释或删除

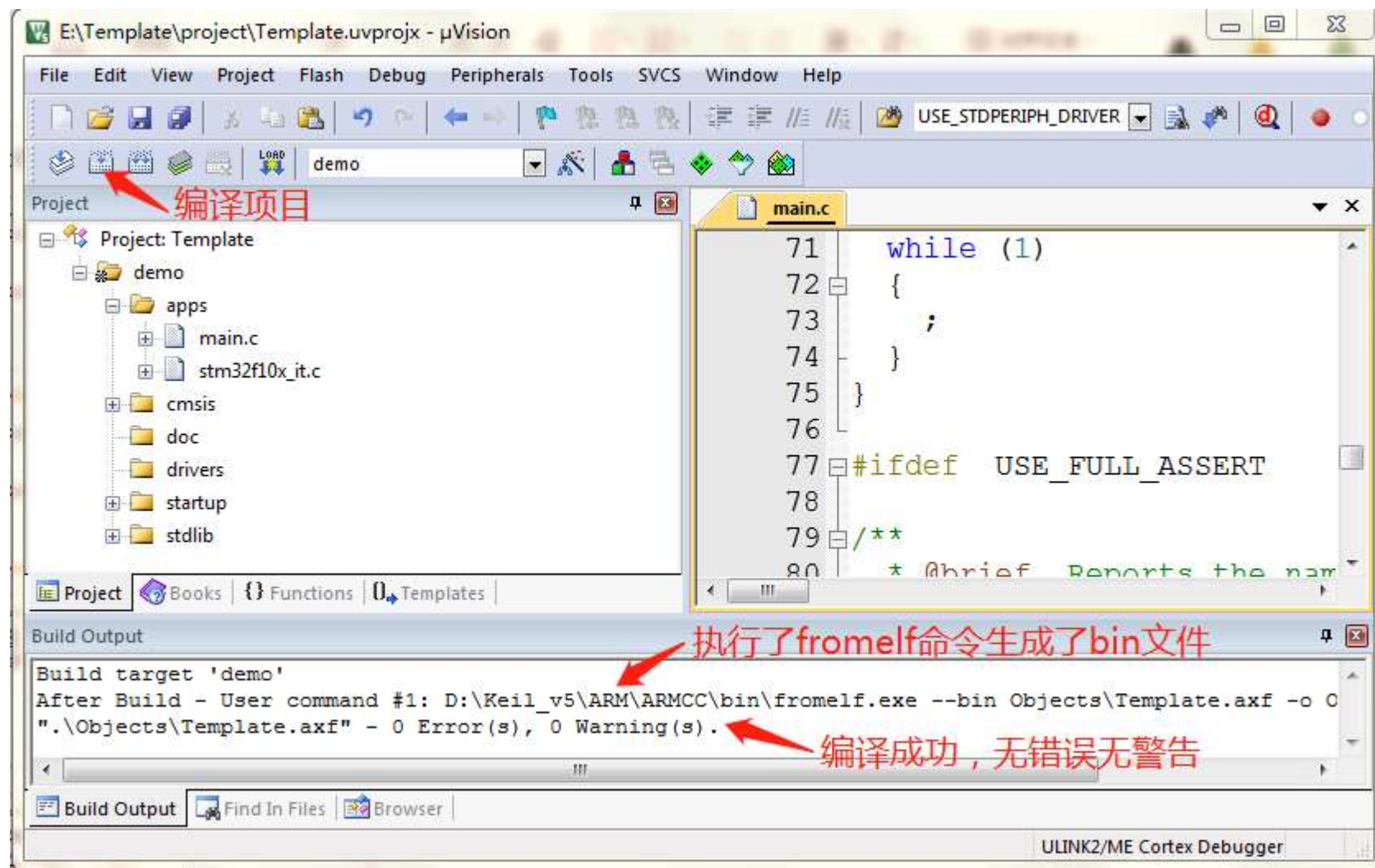
将main.c文件中的while循环内的语句清空。我们只需要验证配置是否成功，能否正常编译通过，因此不需要实现功能的代码。



```
70      This code needs to
71  while (1)
72  {
73      ;
74  }
```

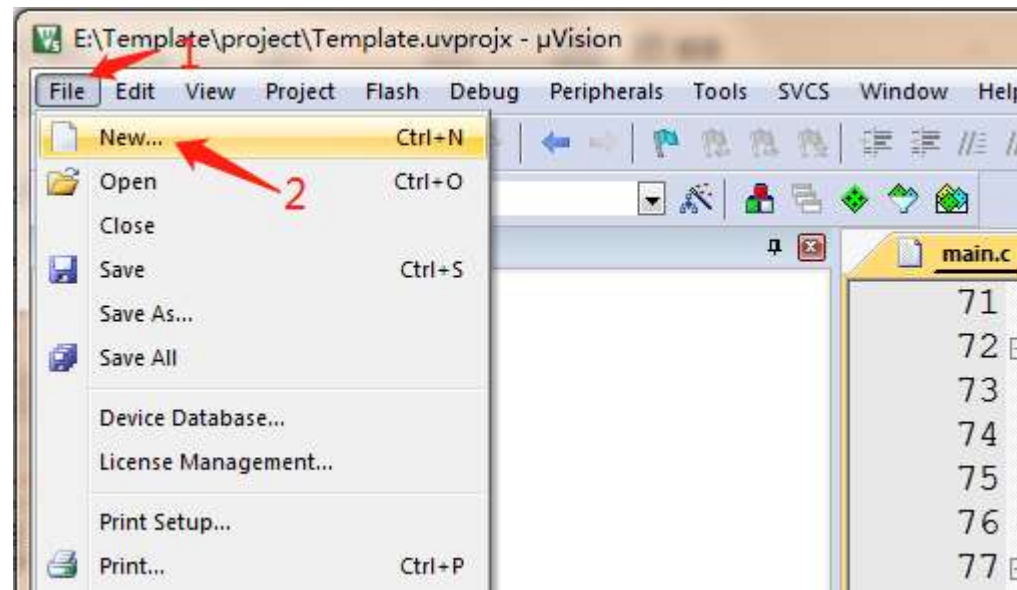
清空while循环中的语句

编译项目



创建源文件的方法

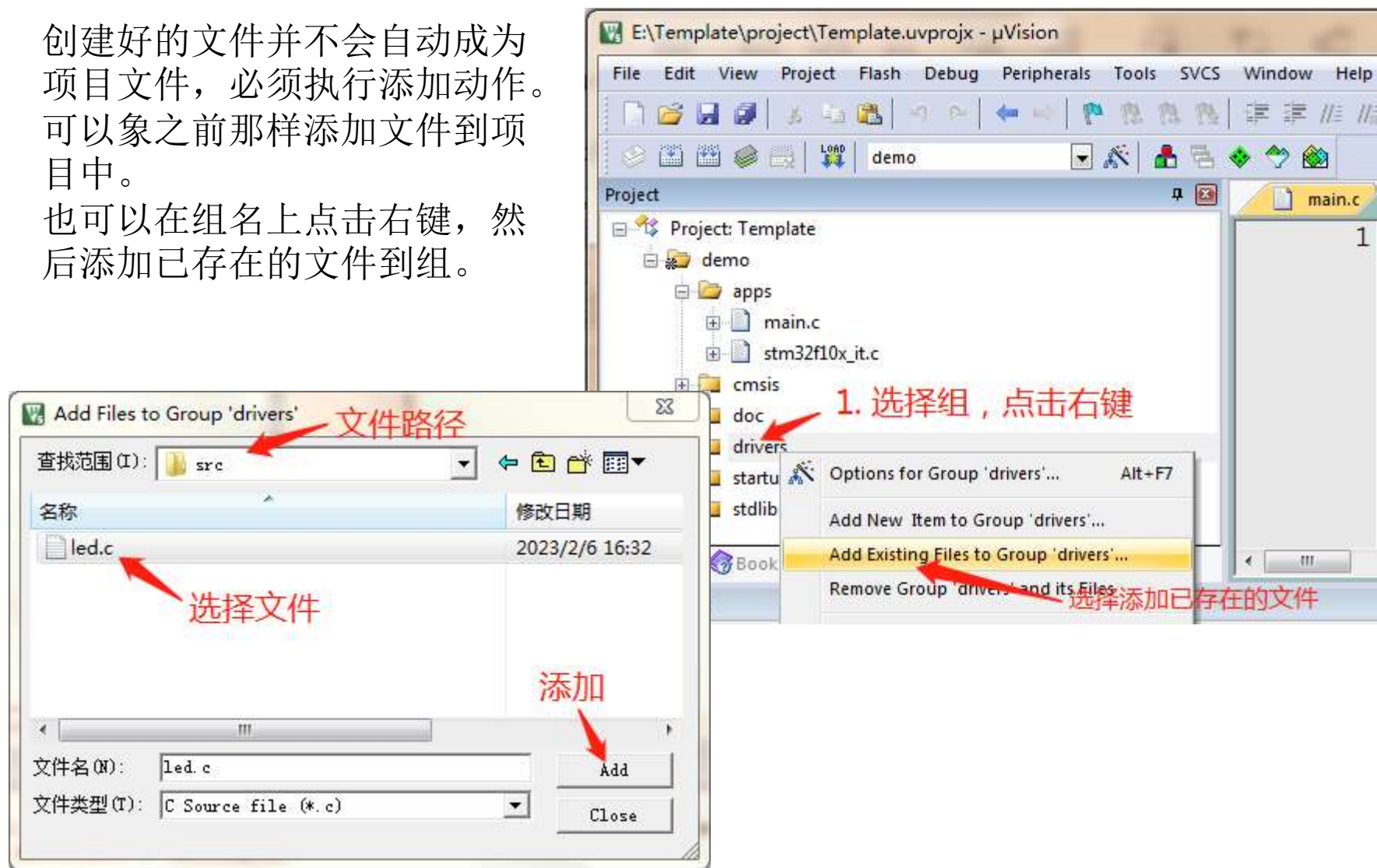
1. 创建源文件
File --> New...
2. 编写代码
3. 保存。选择保存路径及输入文件名字，如：
Template\drivers\
src\led.c



添加源文件到项目的方法

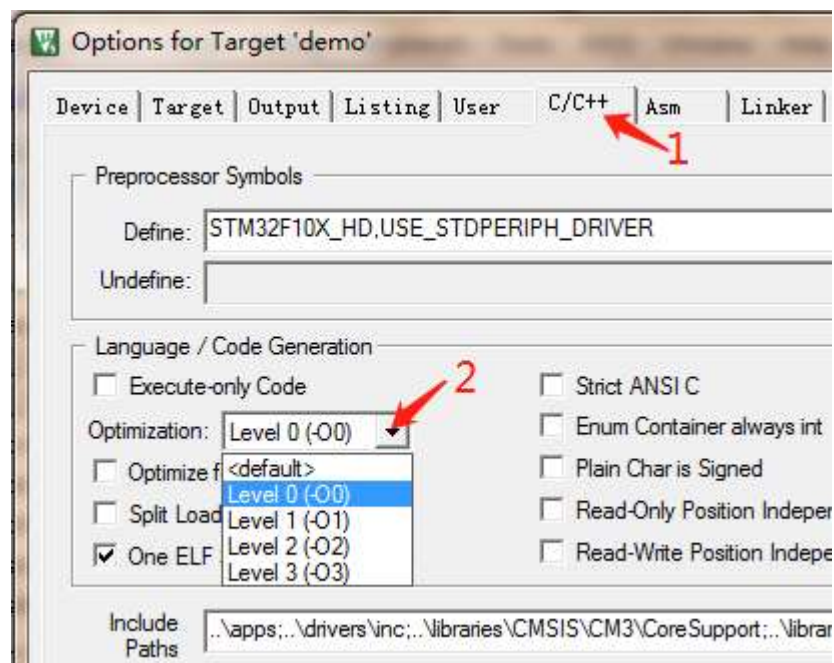
创建好的文件并不会自动成为项目文件，必须执行添加动作。可以象之前那样添加文件到项目中。

也可以在组名上点击右键，然后添加已存在的文件到组。



MDK编译优化级别的选择

点击Project ->
Options for demo
->C/C++



下载程序到STM32F1平台 - ISP方式

这里介绍两种下载程序到开发板的方式：**ISP** 和**J-Link**。

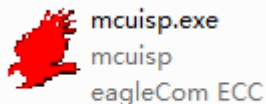
ISP（在系统编程）是通过**ST**在芯片生产线上写入内嵌的自举程序到芯片系统存储器中，启动时从系统存储器中运行这个自举程序，它通过**USART1**串口来将我们的程序写入闪存存储器中。

注意下载时要将开发板上的**boot**开关打到**ISP**模式，**ISP**模式下下载完成后可以立即执行程序。而如果要单独启动程序，则必须将**boot**开关打到**RUN**运行模式。

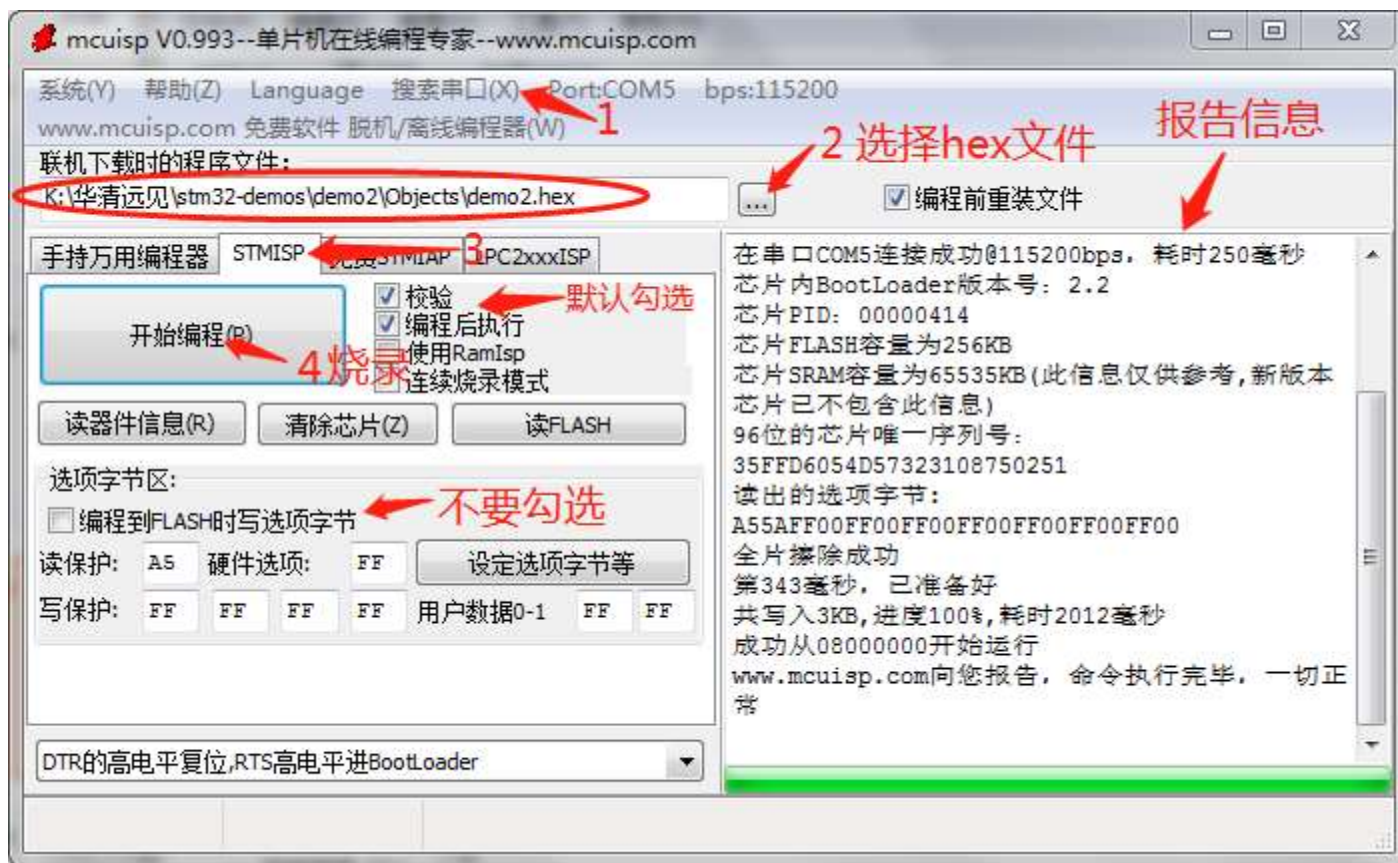


下载程序到STM32F1平台 - ISP方式

ISP下载需要借助第三方工具，如mcuisp.exe或FlyMcu.exe。



FlyMcu.exe
2017/9/1 9:22
2.52 MB



下载程序到STM32F1平台 - J-Link方式

另一种方式是使用J-Link下载。



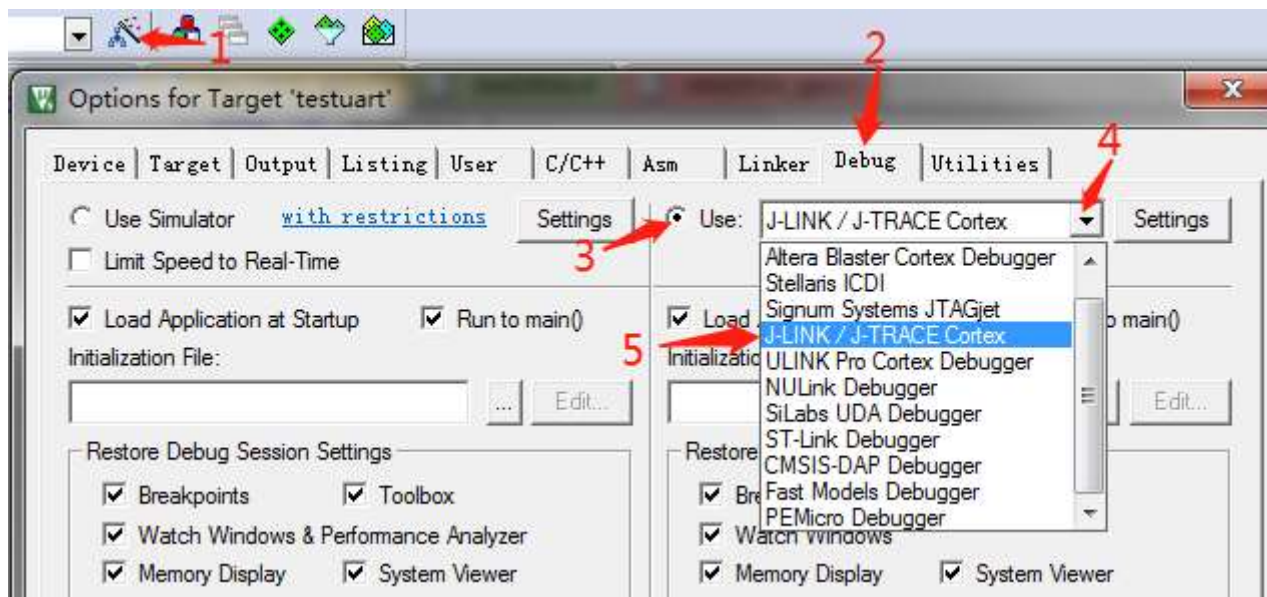
J-Link



J-Link与开发板连接

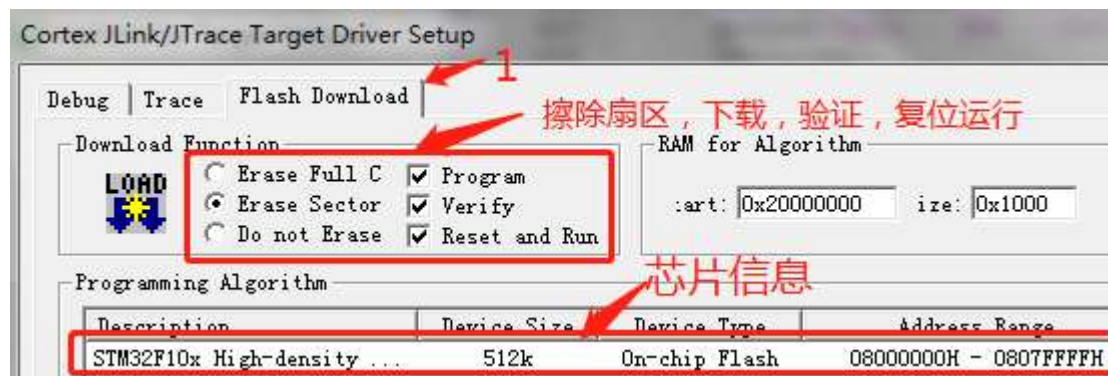
下载程序到STM32F1平台 - J-Link方式

J-Link可以直接通过keil下载。先选择J-Link。



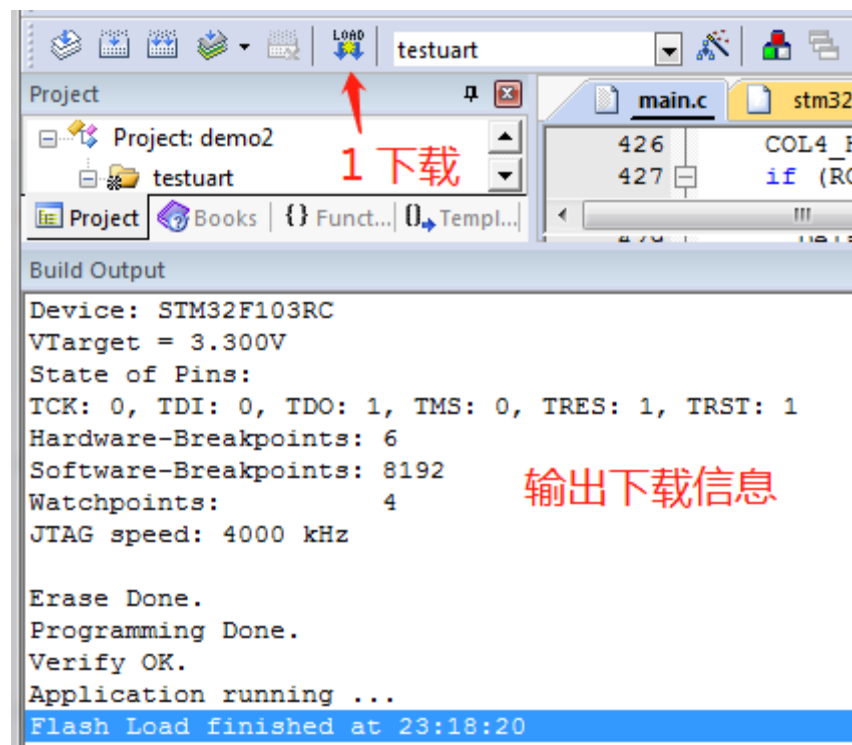
下载程序到STM32F1平台 - J-Link方式

再设置J-Link和芯片。如果无法显示芯片信息，则设置失败或未正确连接开发板。



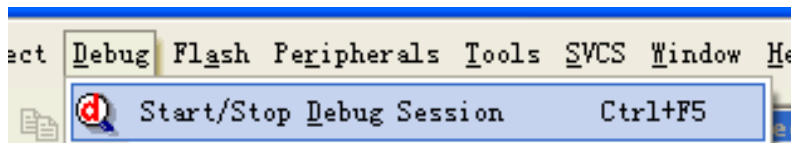
下载程序到STM32F1平台 - J-Link方式

keil下载程序到开发板。



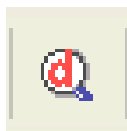
MDK平台软件下载调试

1. 选择

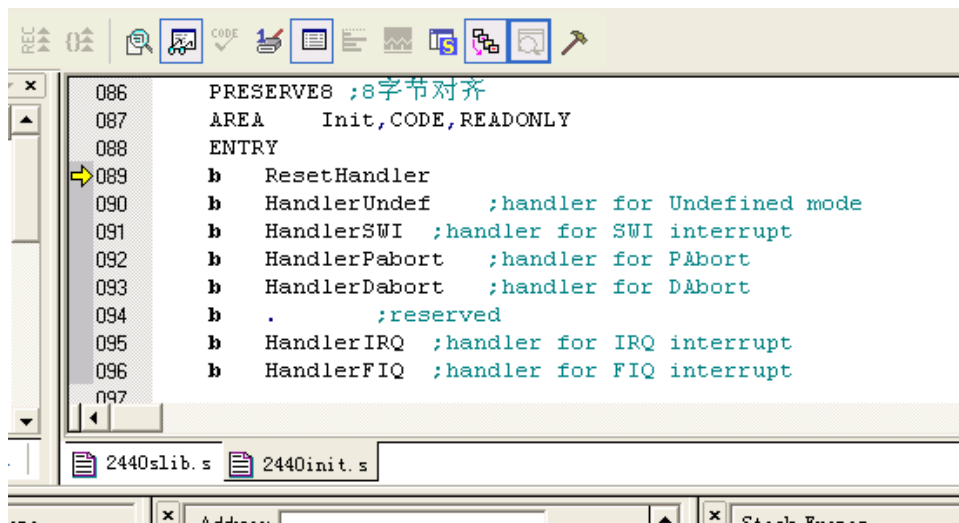


或快捷键Ctrl+F5

或点击



2. 将程序下载到开发板中调试



注意ISP方式不能调试。
这里是以J-Link方式进行调试。

MDK平台软件控制调试

单步调试

运行到断点

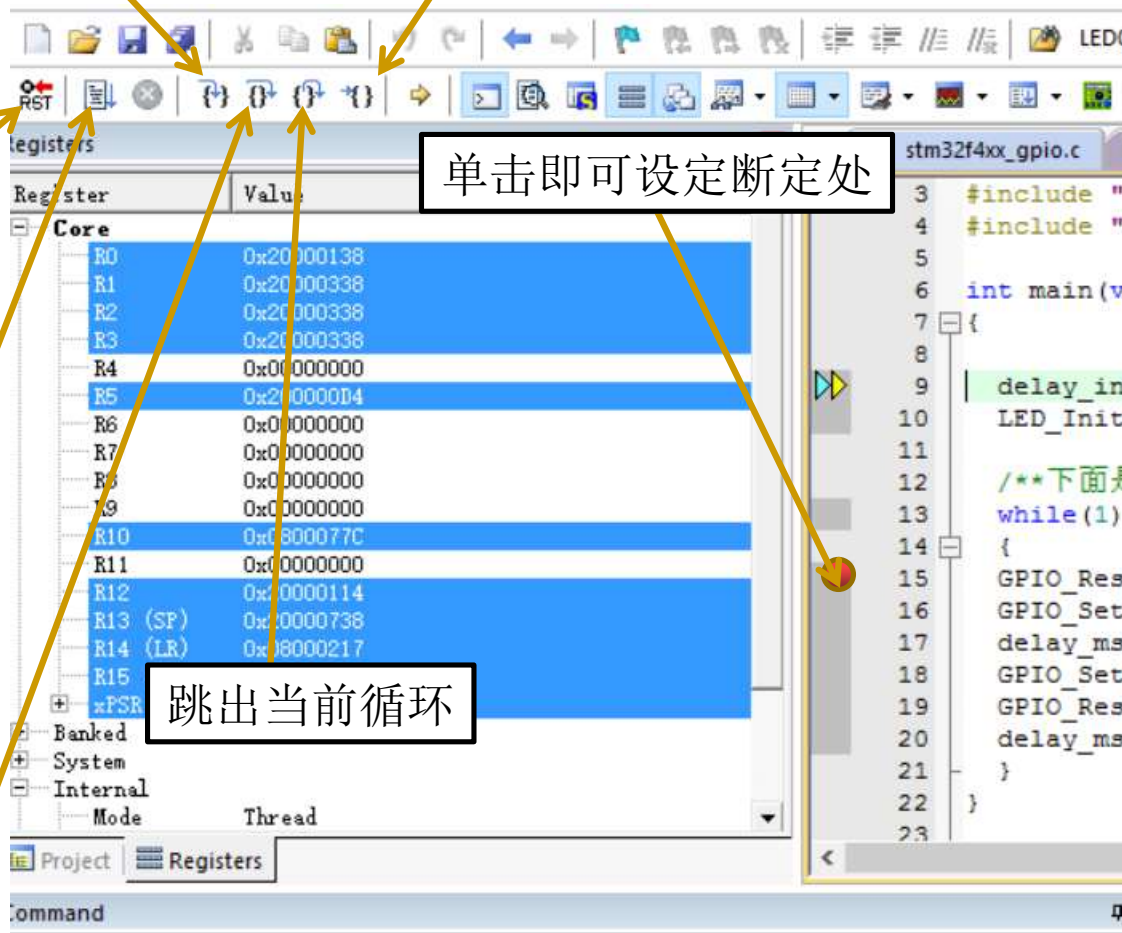
单击即可设定断点处

复位

全速运行

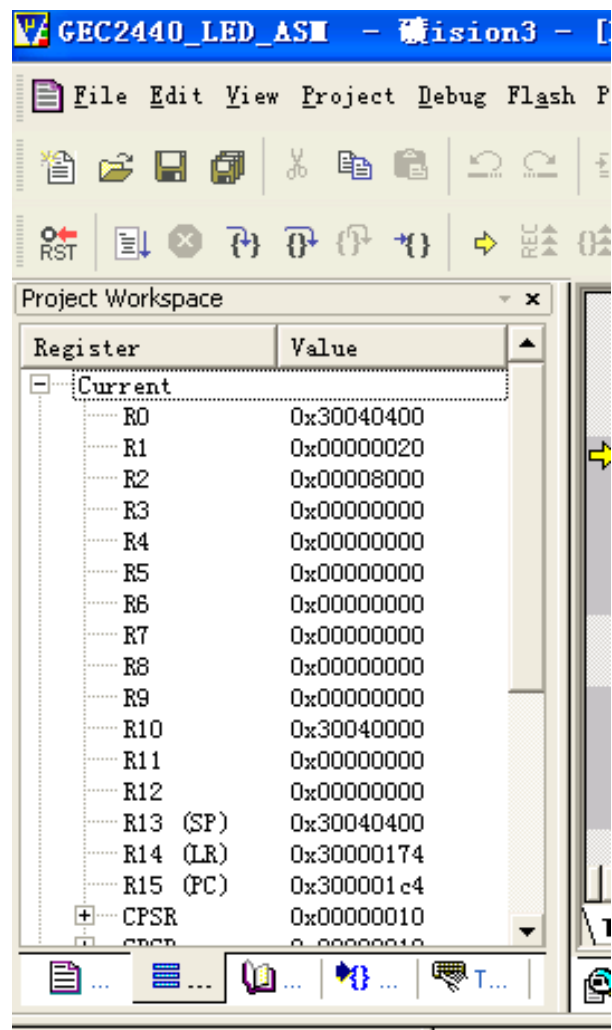
代码如有
循环嵌套，
直接跳过
进入下一
条指令

跳出当前循环



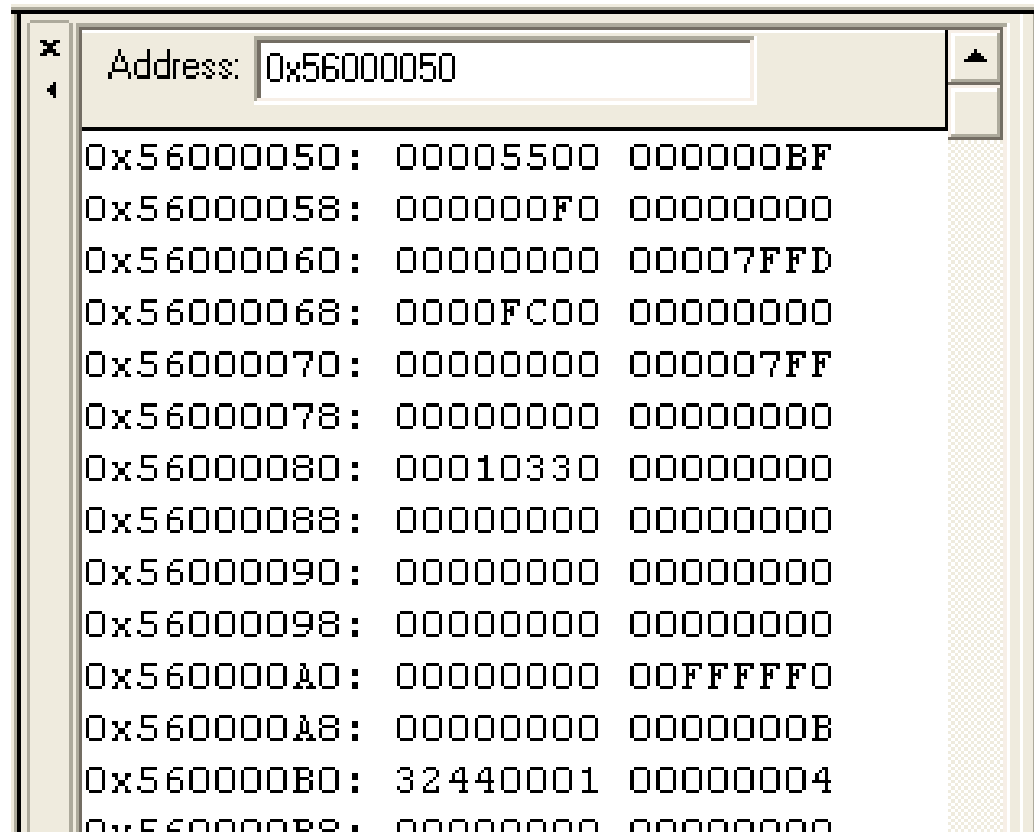
查看通用寄存器

查看通用寄存器中的内容



查看外部或内部存储器


查看存储器中的内容

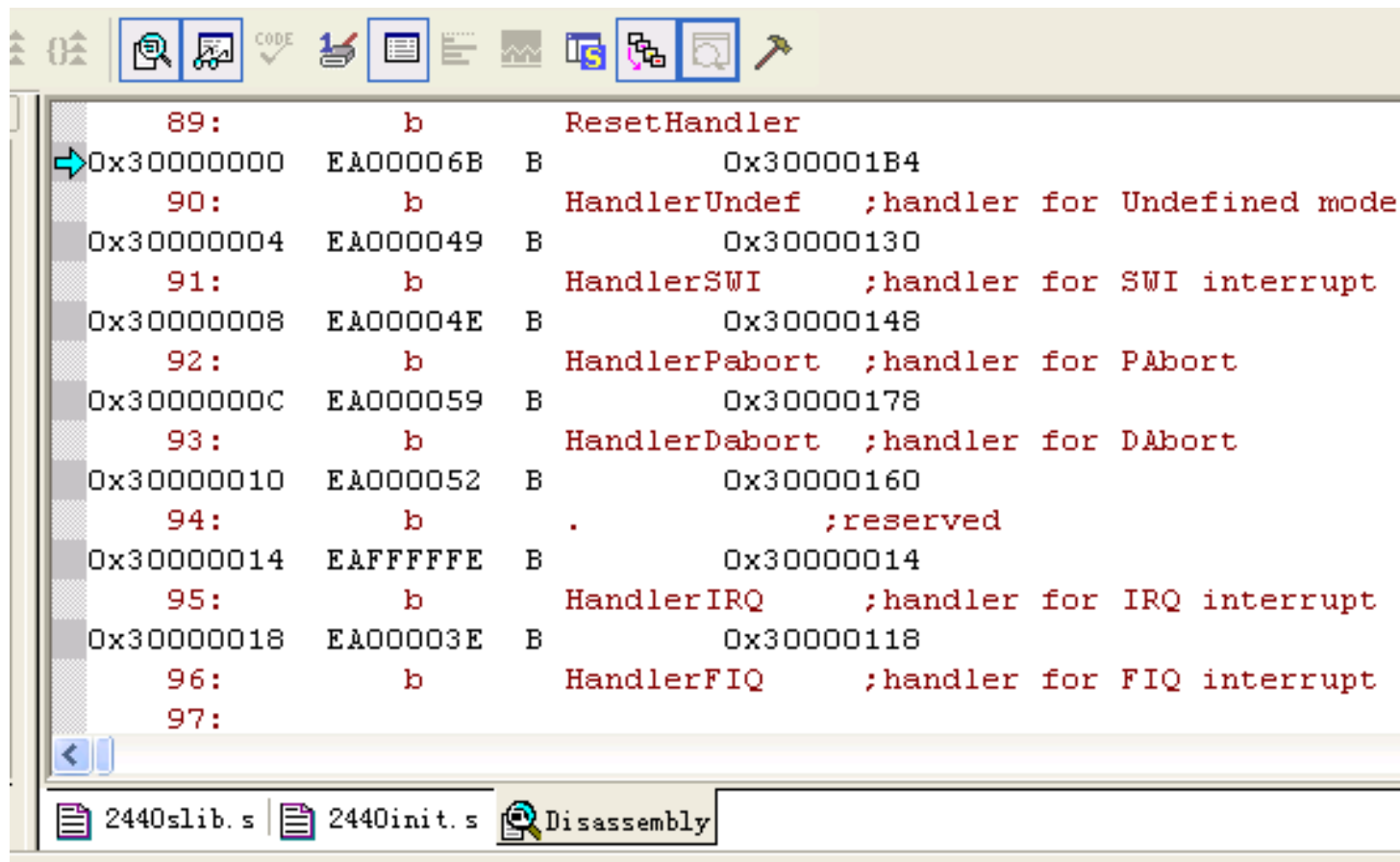


A screenshot of a memory viewer window. The window has a title bar with a close button (X) and a maximize button. Below the title bar is a search bar labeled "Address:" with the value "0x56000050" entered. The main area of the window displays a list of memory addresses and their corresponding hexadecimal values. The addresses range from 0x56000050 to 0x560000B8, with some addresses missing (e.g., 0x56000054, 0x5600005C, 0x56000064, 0x5600006C, 0x56000074, 0x5600007C, 0x56000084, 0x5600008C, 0x56000094, 0x5600009C, 0x560000A4, 0x560000AC, 0x560000B4). The values are displayed in two columns, with the first column showing the address and the second column showing the value. The values are in hexadecimal format, with some values being zero (00000000) and others being non-zero (e.g., 00005500, 000000BF, 000000F0, 00007FFD, 0000FC00, 000007FF, 00010330, 00000000, 00FFFFF0, 0000000B, 32440001, 00000004).

| Address | Value |
|------------|---------------------|
| 0x56000050 | 00005500 000000BF |
| 0x56000058 | 000000F0 00000000 |
| 0x56000060 | 00000000 00007FFD |
| 0x56000068 | 0000FC00 00000000 |
| 0x56000070 | 00000000 000007FF |
| 0x56000078 | 00000000 00000000 |
| 0x56000080 | 00010330 00000000 |
| 0x56000088 | 00000000 00000000 |
| 0x56000090 | 00000000 00000000 |
| 0x56000098 | 00000000 00000000 |
| 0x560000A0 | 00000000 00FFFFFFF0 |
| 0x560000A8 | 00000000 0000000B |
| 0x560000B0 | 32440001 00000004 |
| 0x560000B8 | 00000000 00000000 |

查看反汇编机器码

点击  查看反汇编机器码



The screenshot shows a disassembler window with a toolbar at the top containing icons for zooming, disassembly, and other functions. The main area displays assembly code for various handlers, with addresses ranging from 0x30000000 to 0x30000018. The code is organized into columns: address, instruction, and comment. The instructions are in hexadecimal, and the comments are in English. The window has a status bar at the bottom showing the current file (2440slib.s) and the current view (Disassembly).

| Address | Instruction | Comment |
|------------|-------------|--|
| 89: | b | ResetHandler |
| 0x30000000 | EA00006B B | 0x3000001B4 |
| 90: | b | HandlerUndef ;handler for Undefined mode |
| 0x30000004 | EA000049 B | 0x300000130 |
| 91: | b | HandlerSWI ;handler for SWI interrupt |
| 0x30000008 | EA00004E B | 0x300000148 |
| 92: | b | HandlerPabort ;handler for P&abort |
| 0x3000000C | EA000059 B | 0x300000178 |
| 93: | b | HandlerDabort ;handler for D&abort |
| 0x30000010 | EA000052 B | 0x300000160 |
| 94: | b | . ;reserved |
| 0x30000014 | EFFFFFFE B | 0x300000014 |
| 95: | b | HandlerIRQ ;handler for IRQ interrupt |
| 0x30000018 | EA00003E B | 0x300000118 |
| 96: | b | HandlerFIQ ;handler for FIQ interrupt |
| 97: | | |

查看自定义标识符

■ 点击

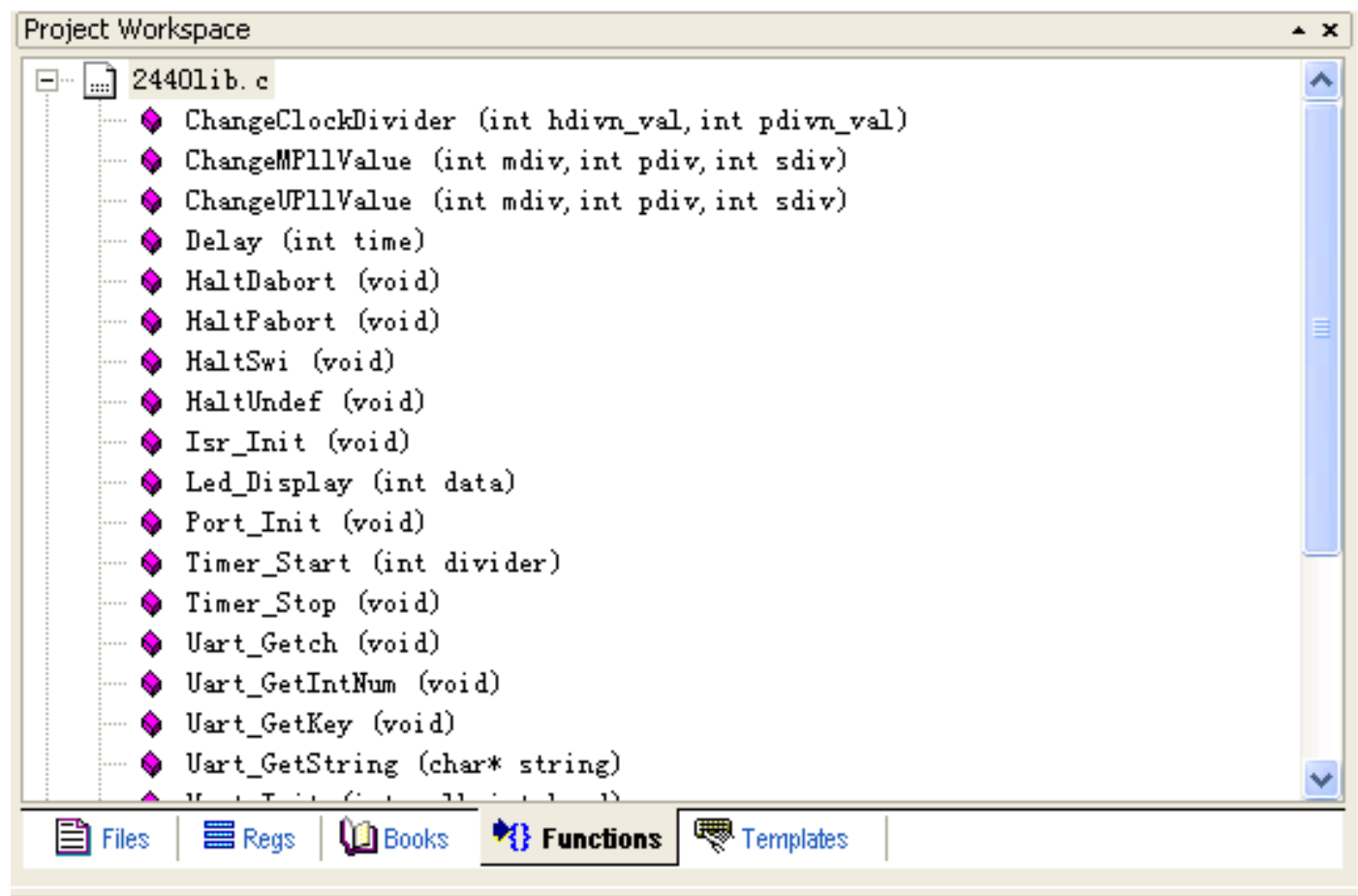


| Symbols | | |
|---|------------|-------------|
| Mask: * <input type="checkbox"/> Case Sensitive | | |
| Name | Address | Type |
| [-] GEC2440_LED_C | | Application |
| [-] Runtime Library | | |
| [-] 2440lib | | Module |
| delayLoopCount | 0x30040004 | int |
| whichUart | 0x30040000 | int |
| [-] ChangeClockDivider | 0x30000CB0 | Function |
| hdiwn | R7 | int |
| hdiwn_val | R0 | int |
| pdiwn | R6 | int |
| pdiwn_val | R1 | int |
| + ChangeMP11Value | 0x30000C98 | Function |
| + ChangeUP11Value | 0x30000E0C | Function |
| + Delay | 0x30000464 | Function |

查看工程中的函数

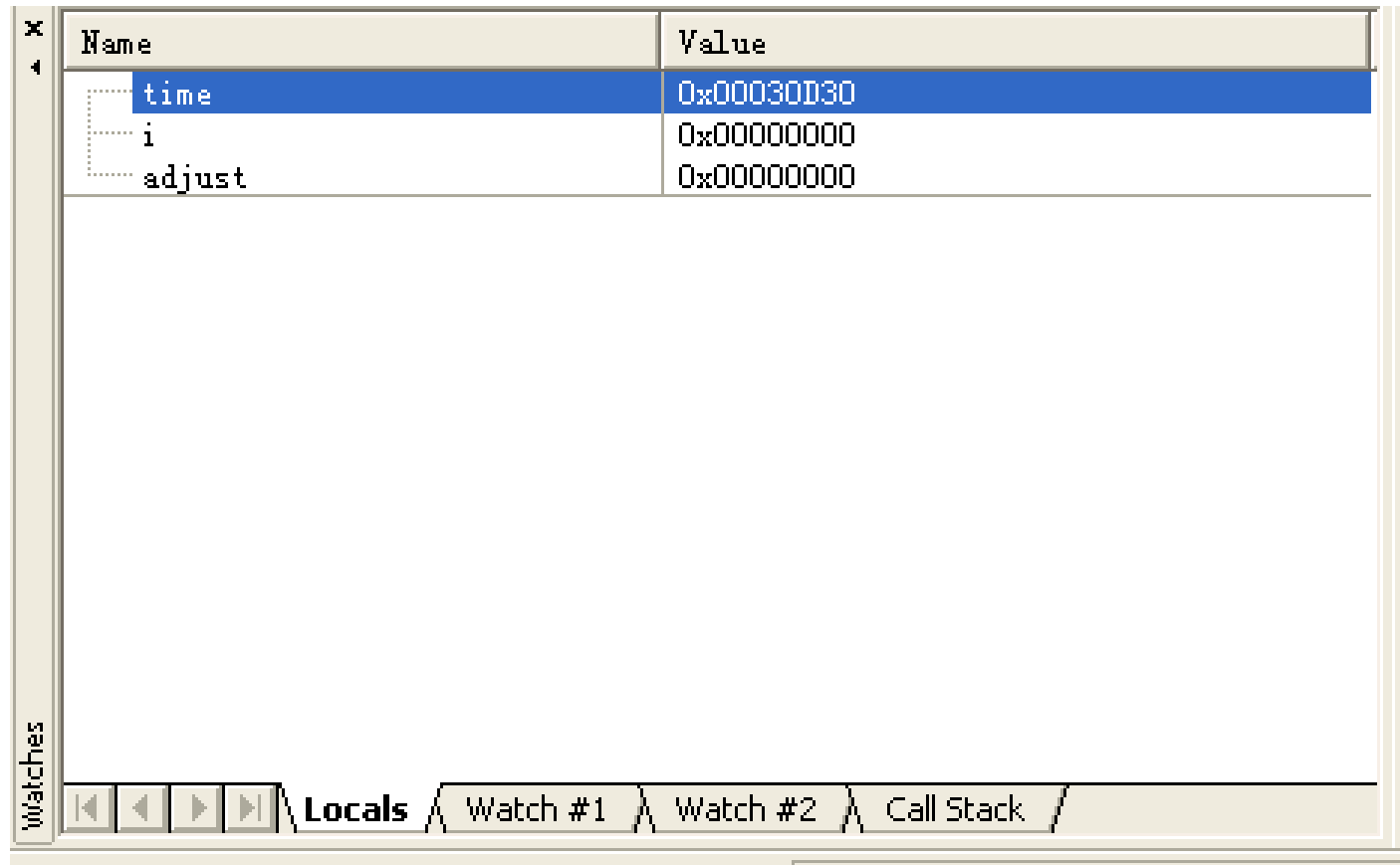
■ 点击

 Functions



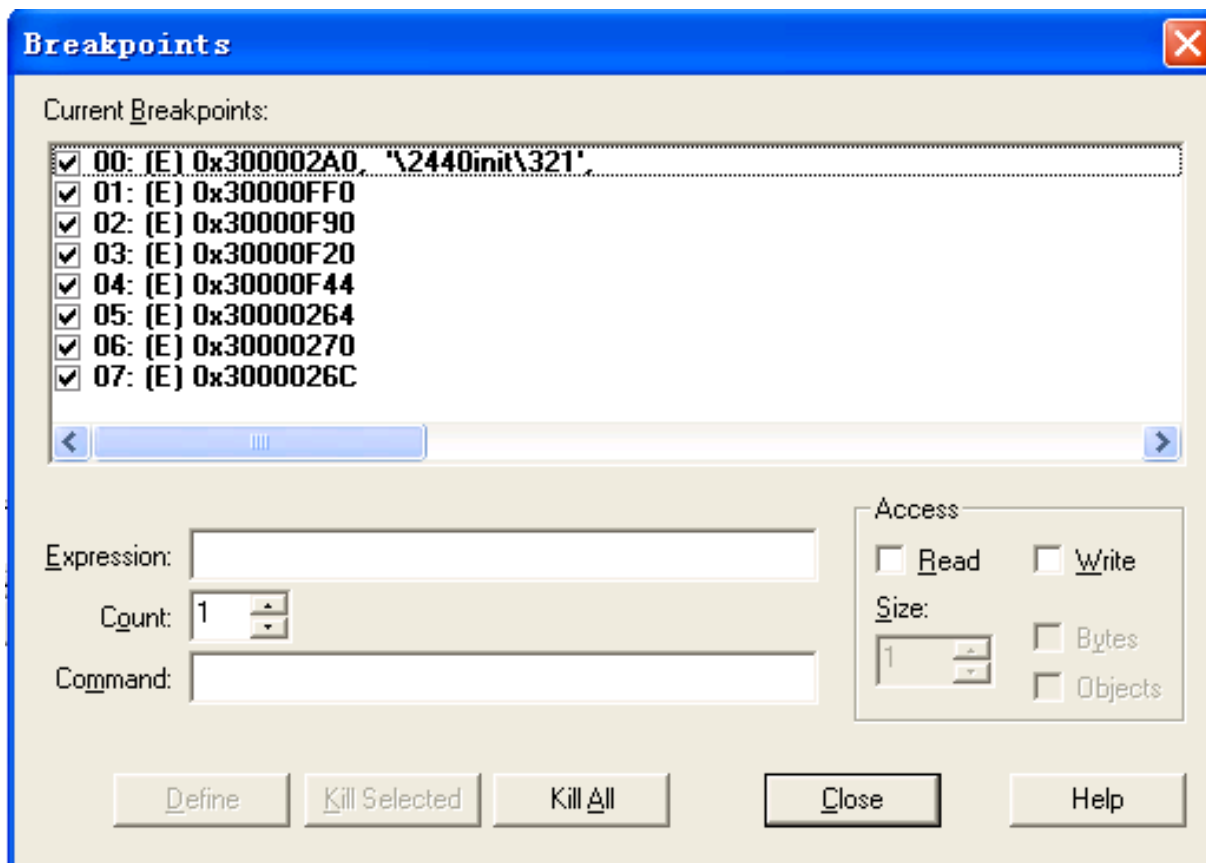
查看变量

- 点击 **Locals**



查看断点列表

选择Debug—>Breakpoints



Thank You !

www.gec-edu.org