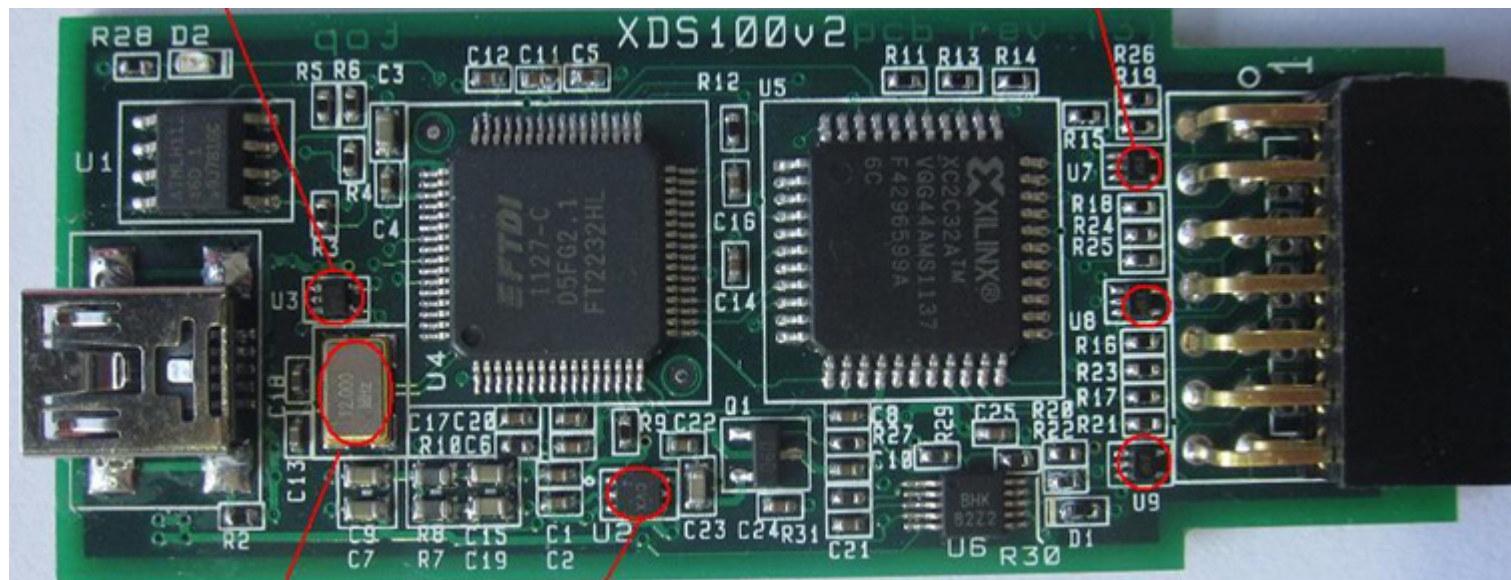


CCS5.5 的详细操作说明

说明：本描述是对 CCS5.5 一步一步地操作如何建立工程，配置仿真器、连接仿真器、烧录 RAM 与调试、烧录 flash，如何打开一个已有的 CCS5.5 工程。

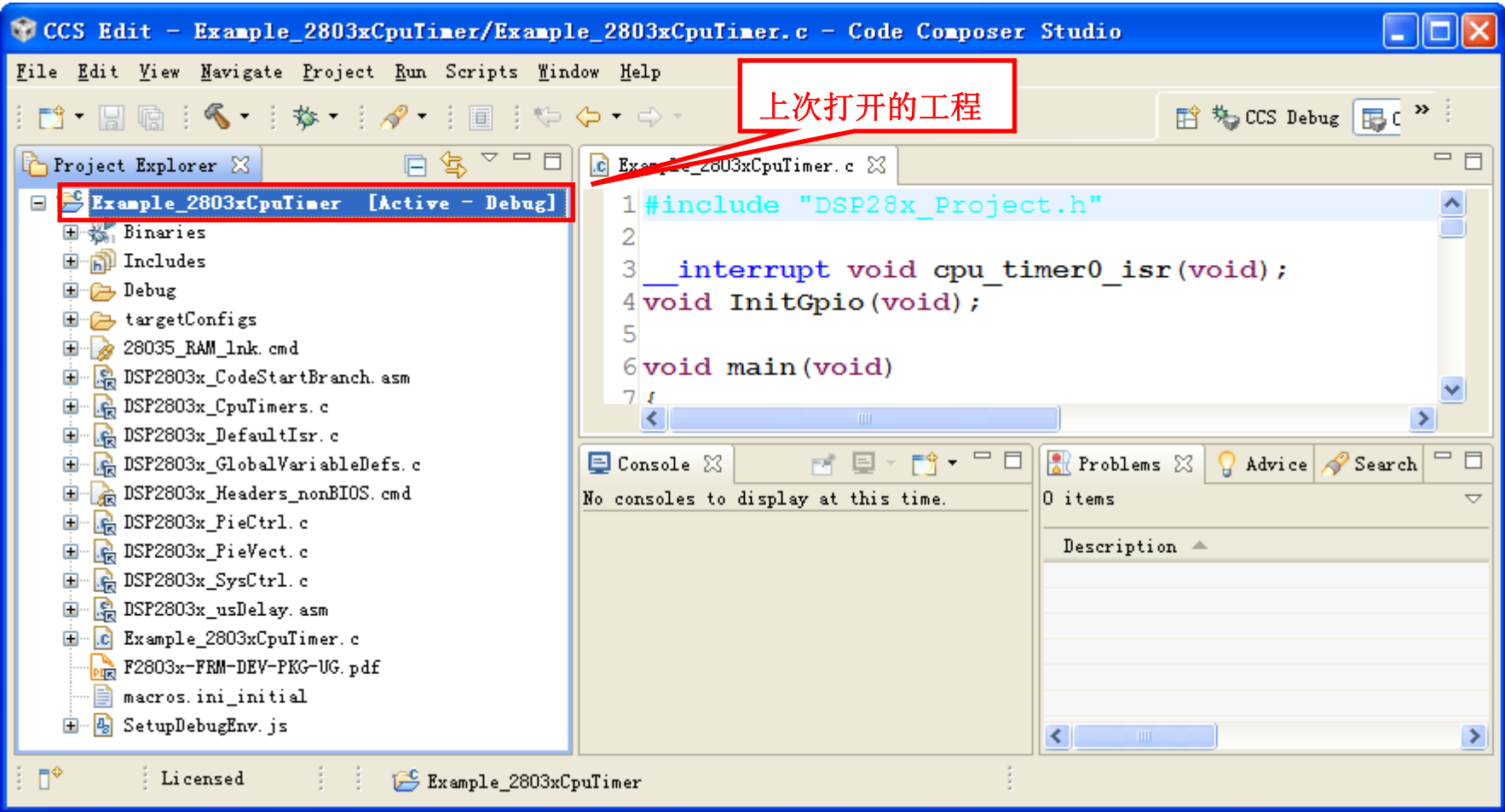
- 1、CCS 版本：CCS5.5.0.00077_win32。（CCS5.5 需要破解的，安装前其实不用卸载 CCS3.3 的，我的电脑就有 CCS3.3 和 CCS5.5 两个版本）
- 2、电脑操作系统：Windows XP（番茄花园 Ghost XP SP3 装机版）
- 3、开发板：以 TMS320F28035 为芯片的开发板。
- 4、仿真器：原装 TI 的 XDS100v2，如下图。



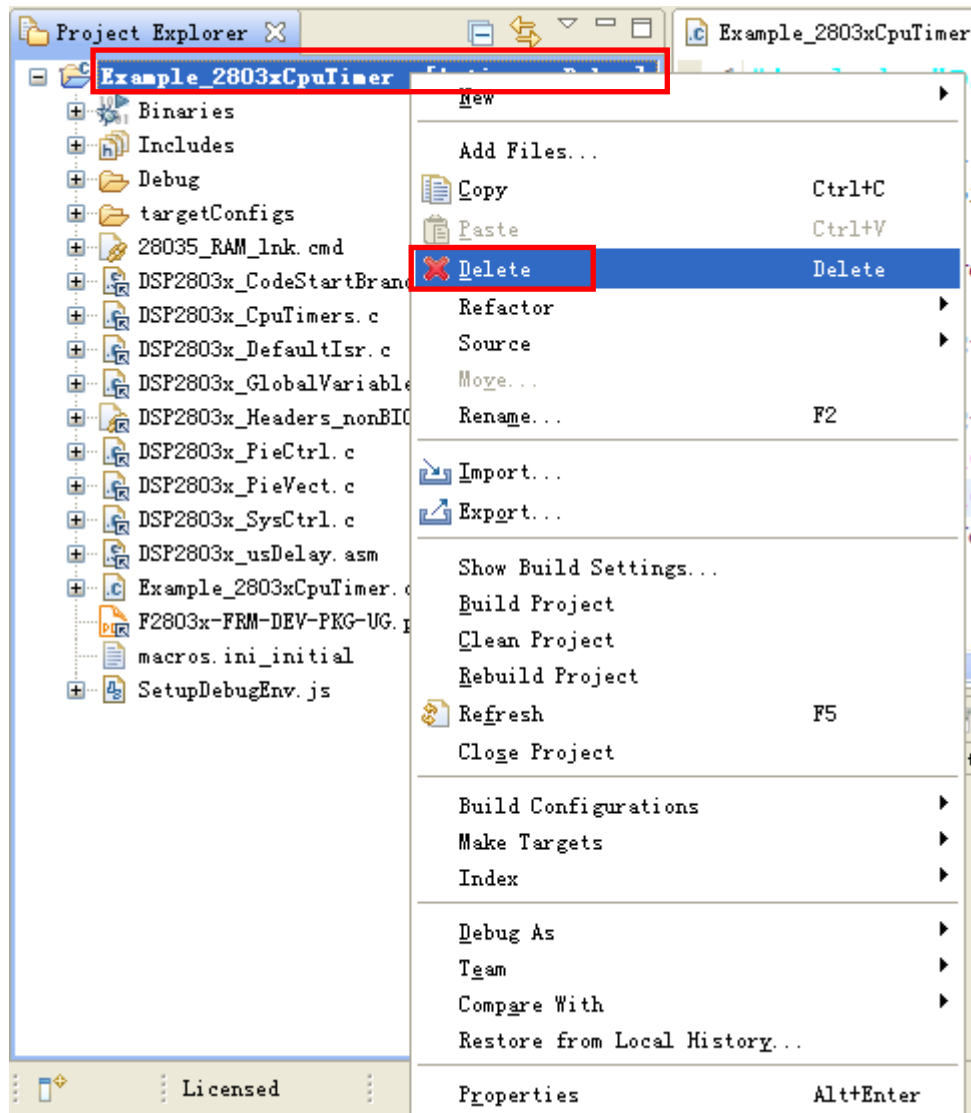
第一步、关闭上次打开的工程



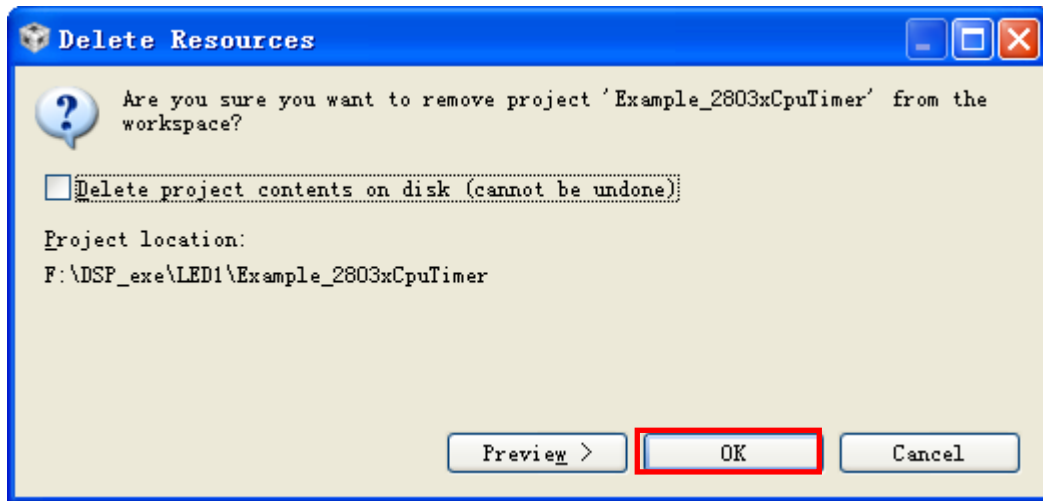
1、双击 ，出现如下界面，其中 **Example_2803xCpuTimer [Active - Debug]** 为上次打开的工程。

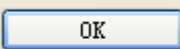
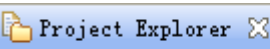


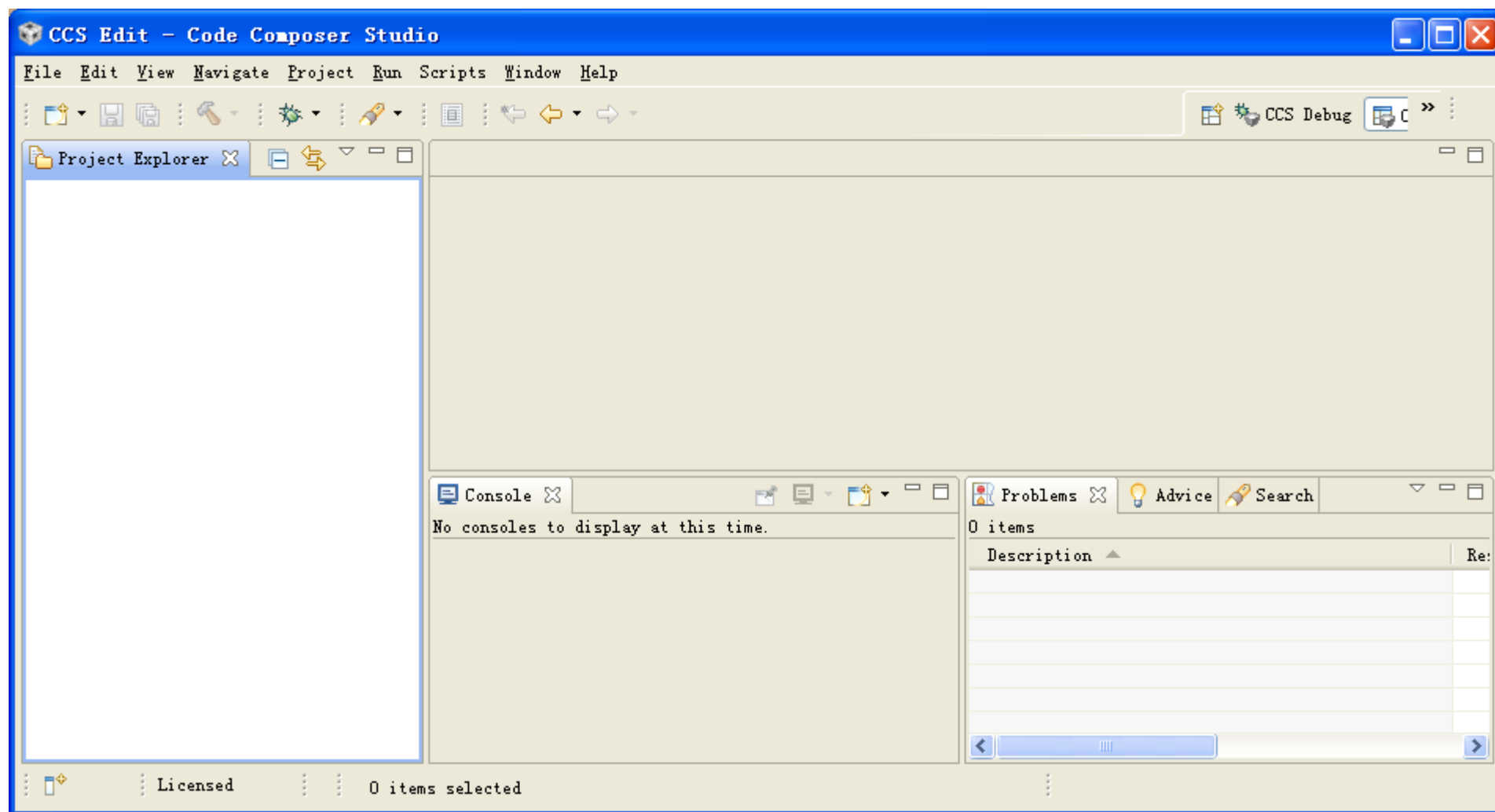
2、鼠标指向  **Example_2803xCpuTimer [Active - Debug]**, 点击鼠标右键, 出现如下对话框:



3、点击鼠标左键  Delete，出现如下对话框，如下图：

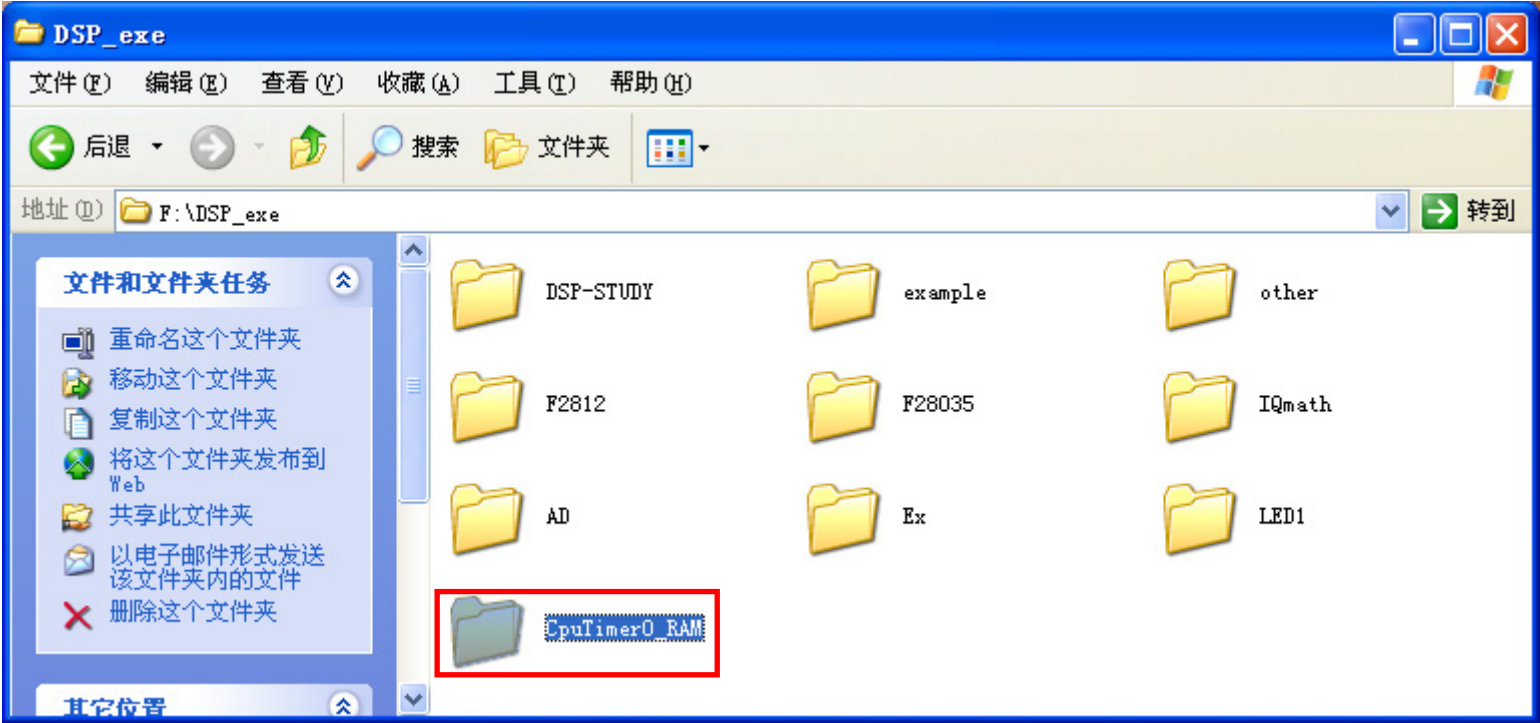


4、点击 ，即可删除  窗口下的工程，如下图：

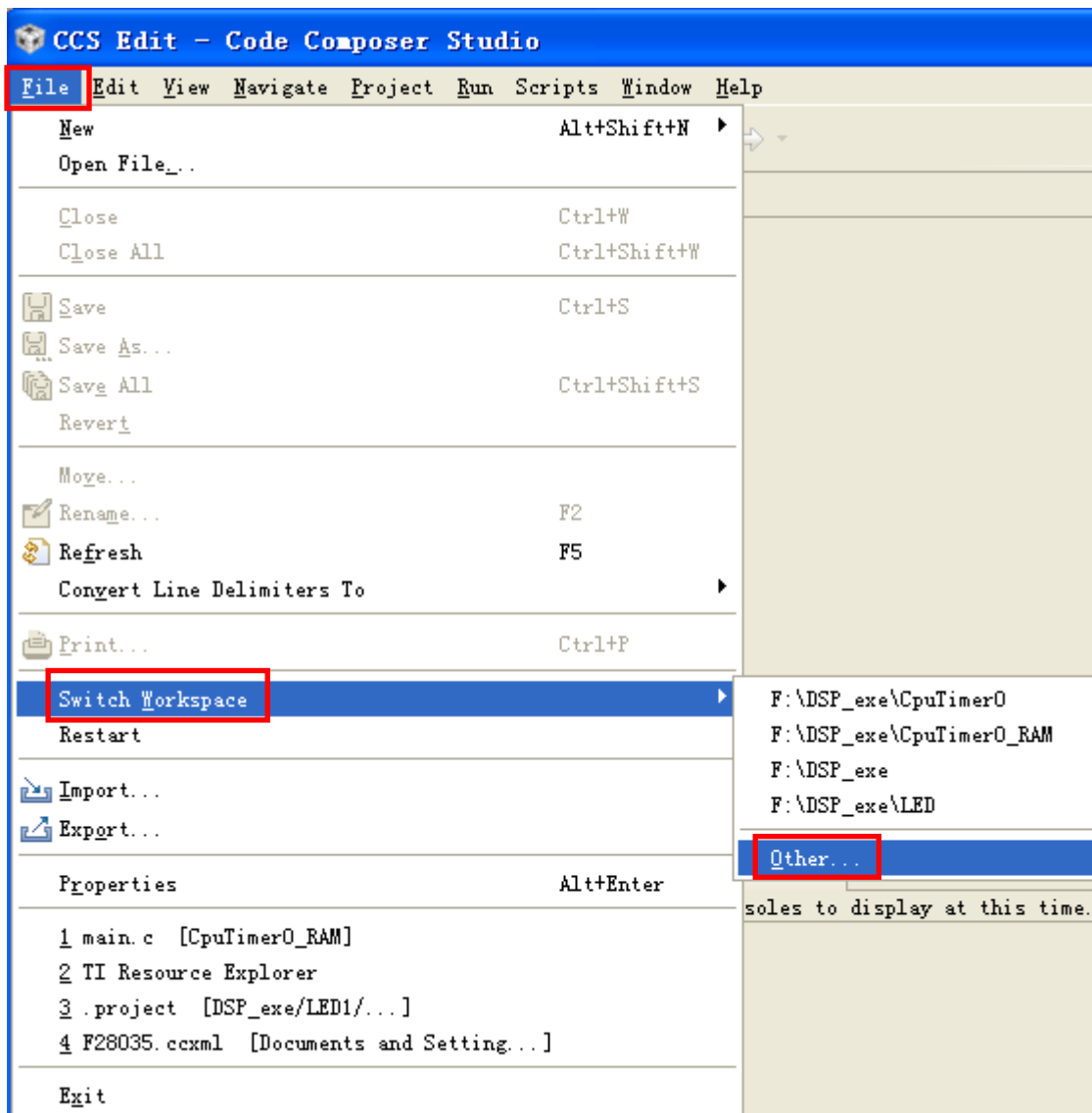


第二步：设置工程文件的存放路径

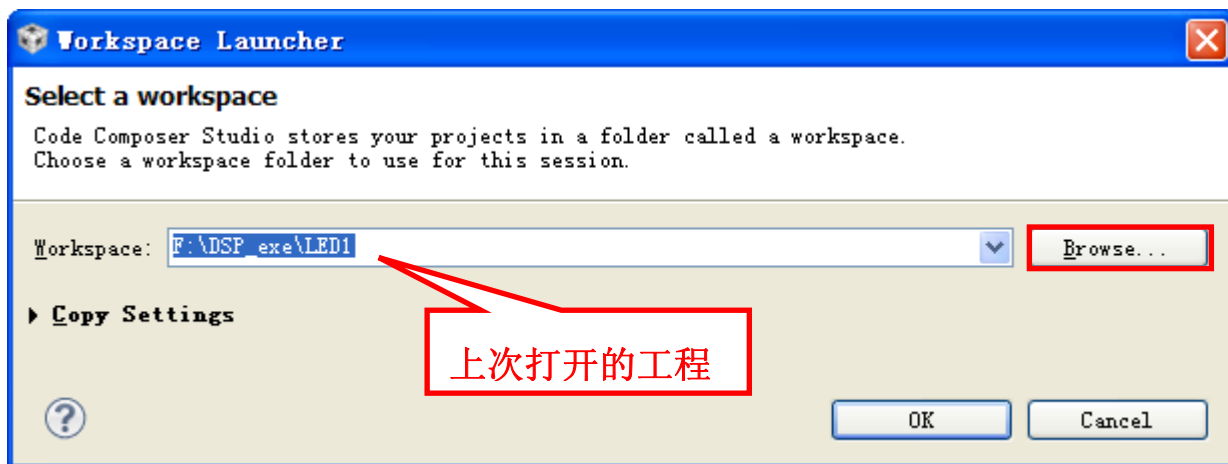
1、首先新建一个文件夹，如下图：



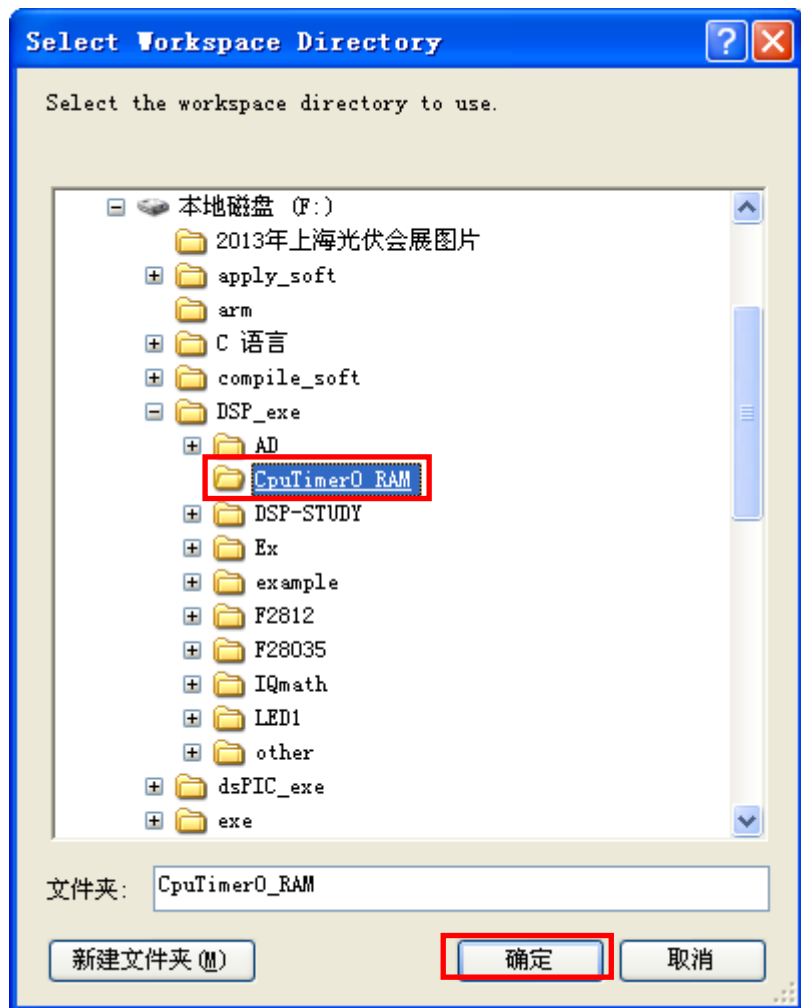
2、选择工程的储存位置，File→Switch Workspace→Other，如下图：

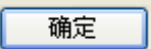



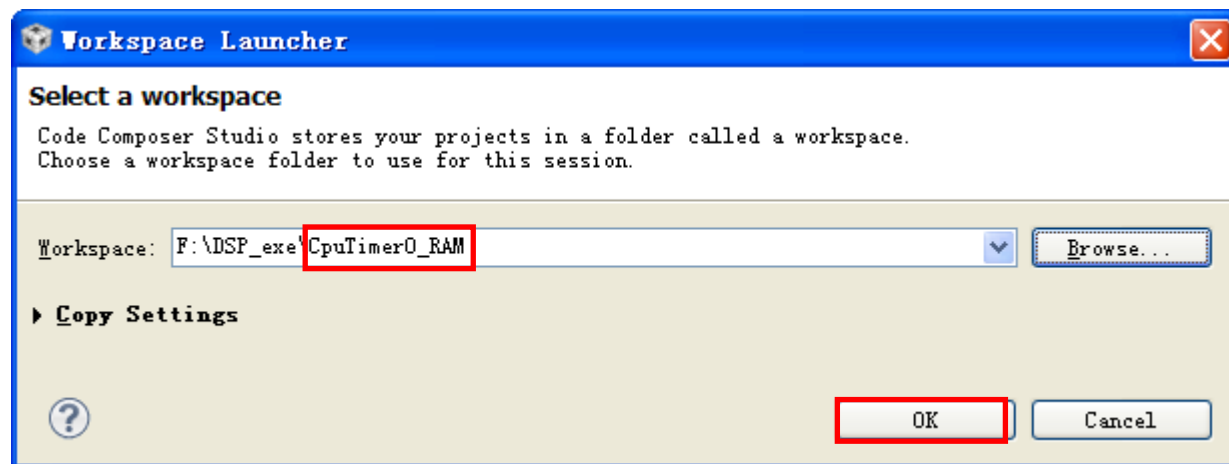
3、点击 **Other...** 出现如下对话框：




4、通过点击 Browse... 选择刚才新建的文件夹，如下图：



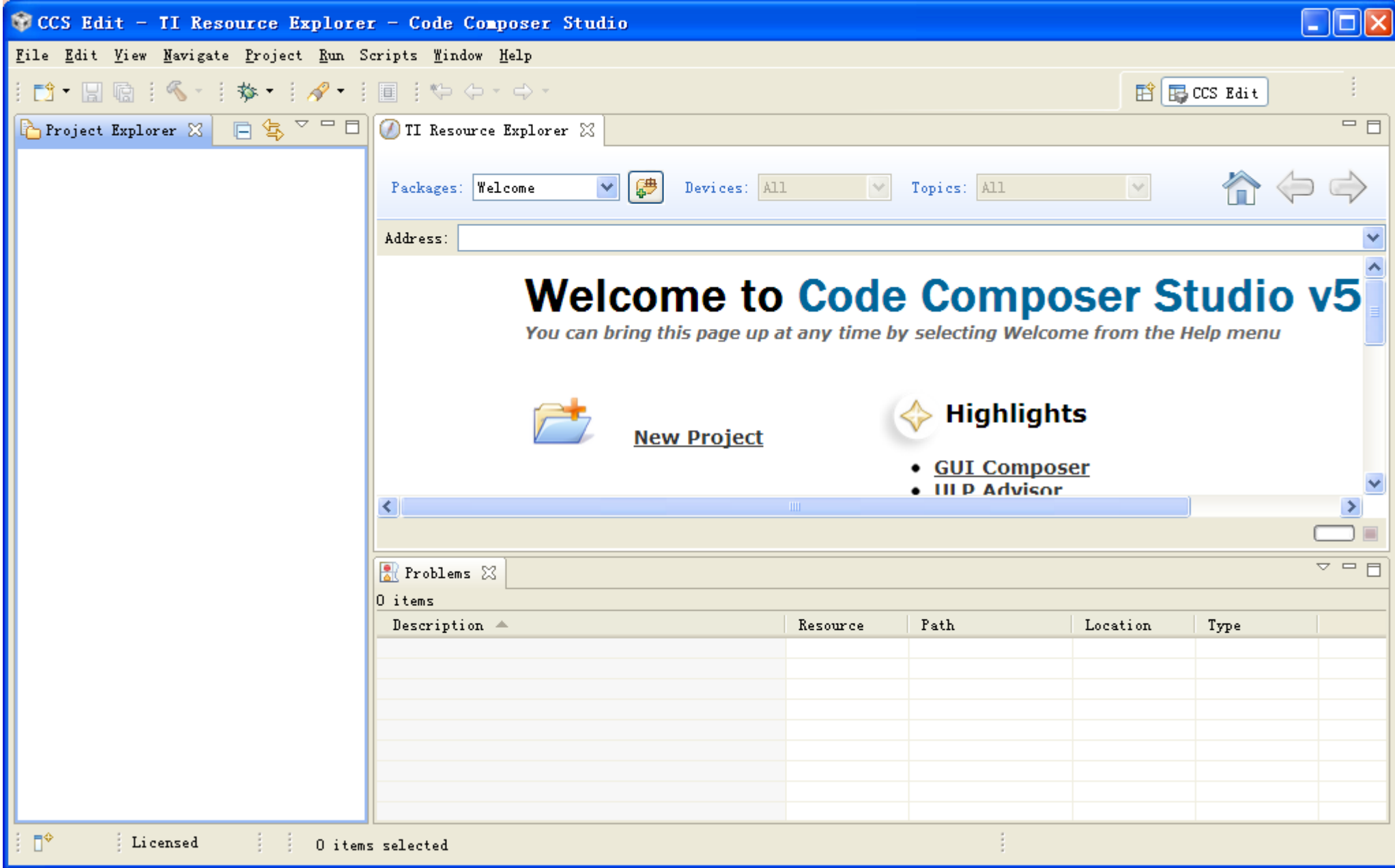
5、点击  后，工程文件的存放路径就在  **CpuTimer0_RAM** 位置，如下界面：



6、点击  后，将会重新打开 CCS5.5 界面，如下图：

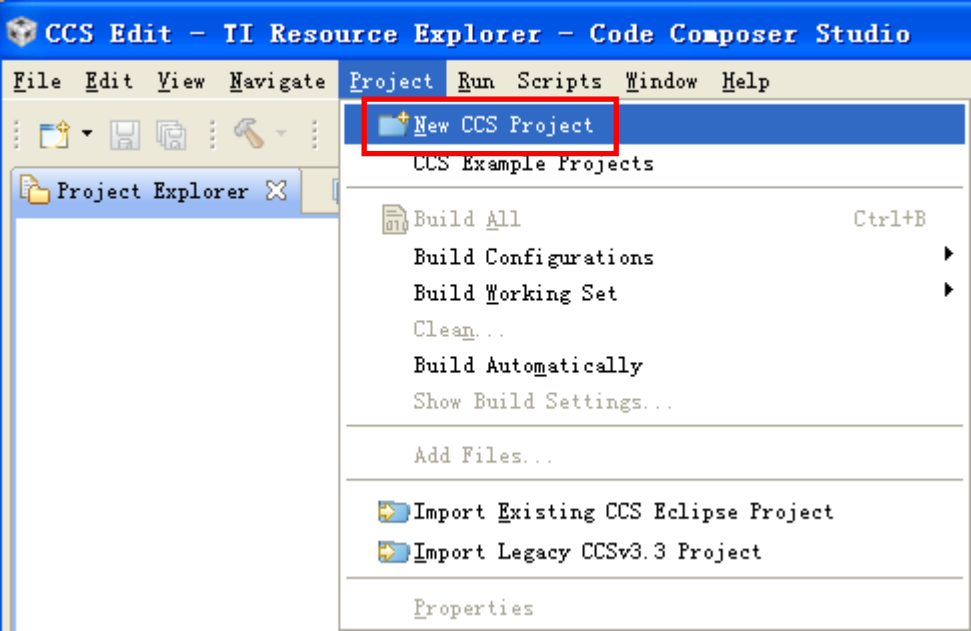


Project Explorer

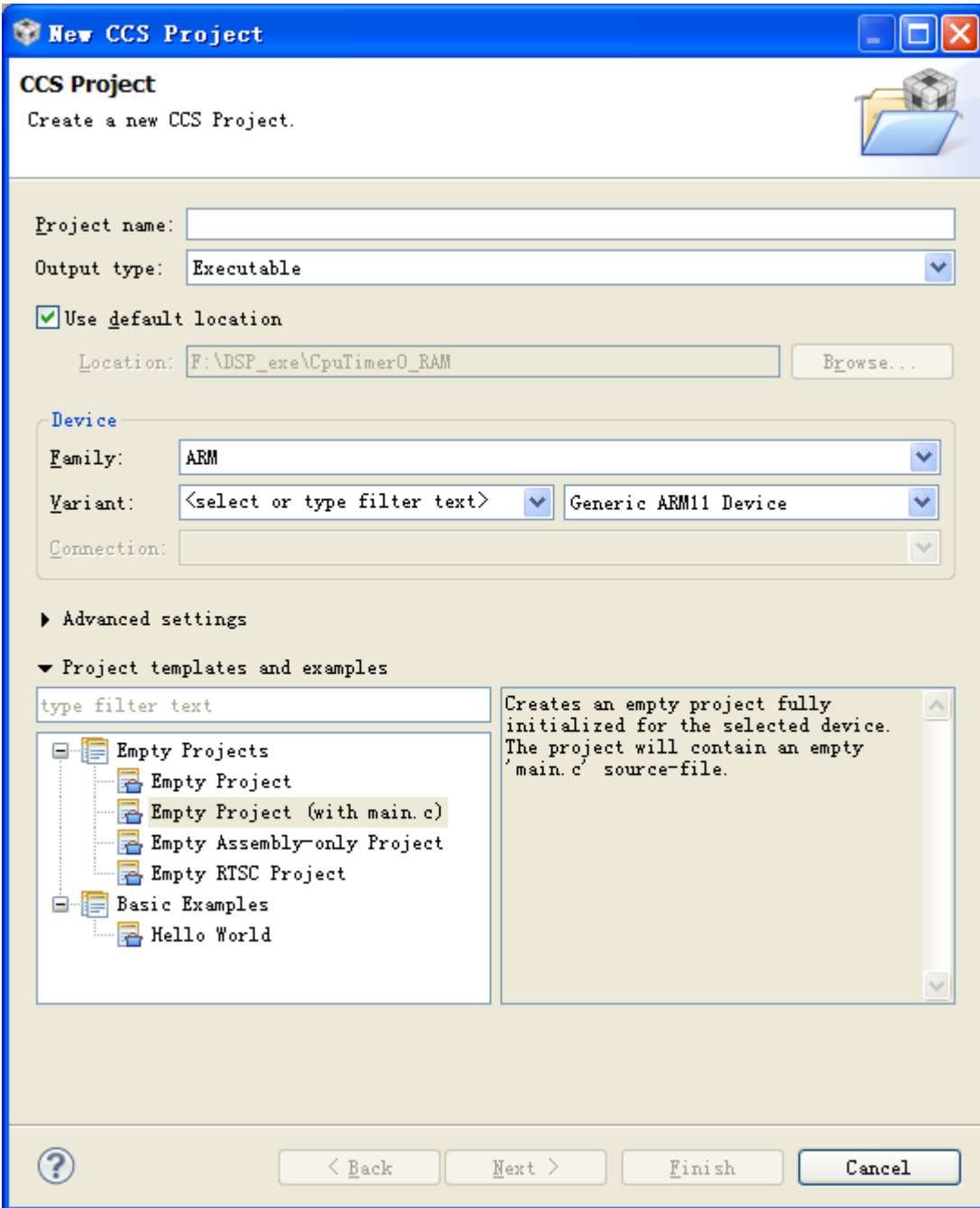


第三步：新建一个工程

1、新建一个项目工程：Project→New CCS Project，如下图：



2、点击  New CCS Project，出现如下对话框：



The image shows the 'New CCS Project' dialog box. It has a title bar with standard Windows window controls. The main area is divided into sections for project configuration. The 'Project name' field is empty. The 'Output type' is set to 'Executable'. The 'Use default location' checkbox is checked, and the 'Location' is set to 'F:\DSP_exe\CpuTimer0_RAM'. The 'Device' section has 'Family' set to 'ARM', 'Variant' set to 'Generic ARM11 Device', and 'Connection' is empty. There are sections for 'Advanced settings' and 'Project templates and examples'. The 'Project templates and examples' section shows a tree view with 'Empty Projects' and 'Basic Examples' expanded. The 'Empty Projects' list includes 'Empty Project', 'Empty Project (with main.c)', 'Empty Assembly-only Project', and 'Empty RTSC Project'. The 'Basic Examples' list includes 'Hello World'. A description box on the right explains that the selected template creates an empty project with a 'main.c' file. At the bottom, there are buttons for '< Back', 'Next >', 'Finish', and 'Cancel', along with a help icon.

New CCS Project

CCS Project

Create a new CCS Project.

Project name:

Output type:

☒ Use default location

Location:

Device

Family:

Variant:

Connection:

► Advanced settings

▼ Project templates and examples

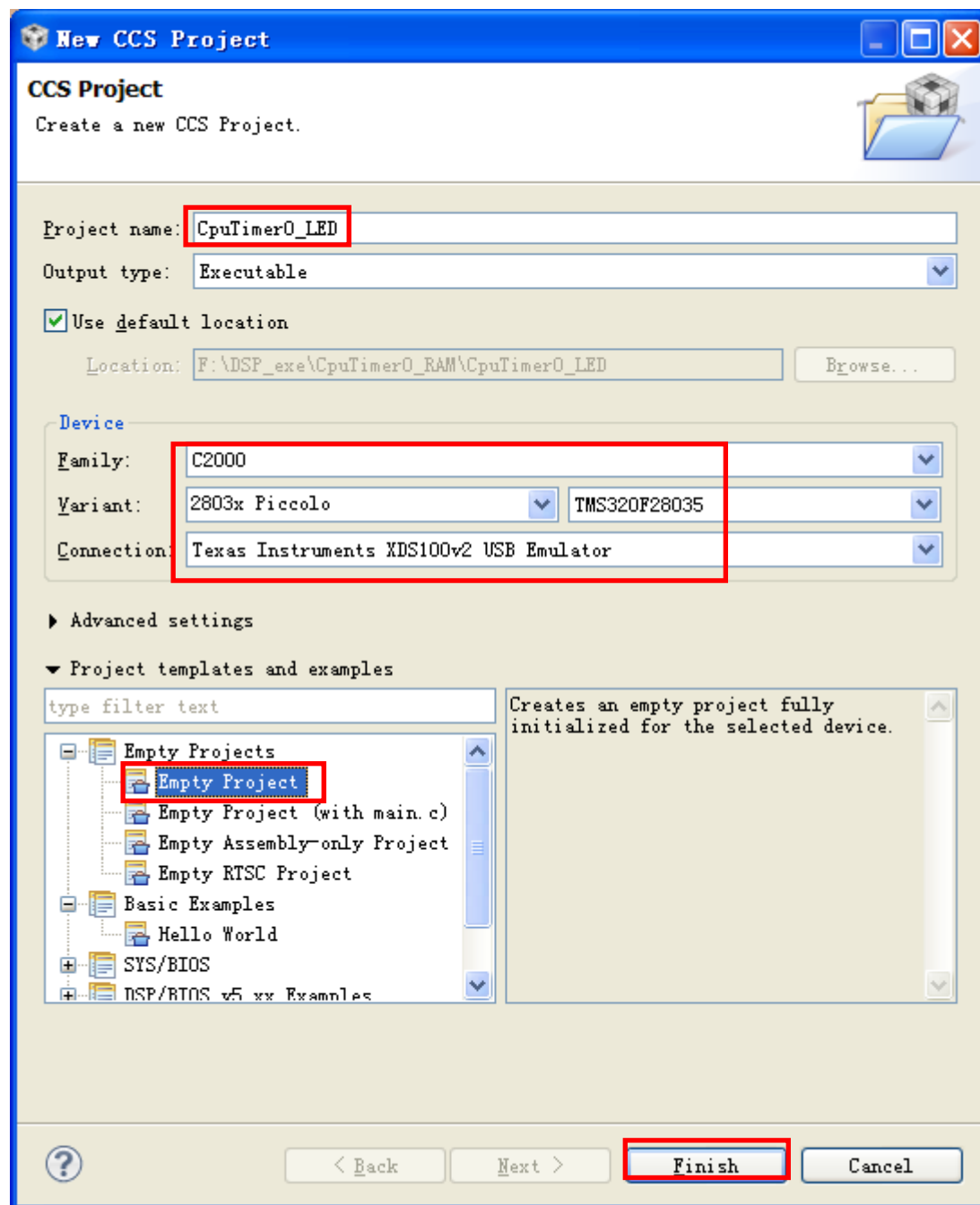
type filter text

- Empty Projects
 - Empty Project
 - Empty Project (with main.c)
 - Empty Assembly-only Project
 - Empty RTSC Project
- Basic Examples
 - Hello World

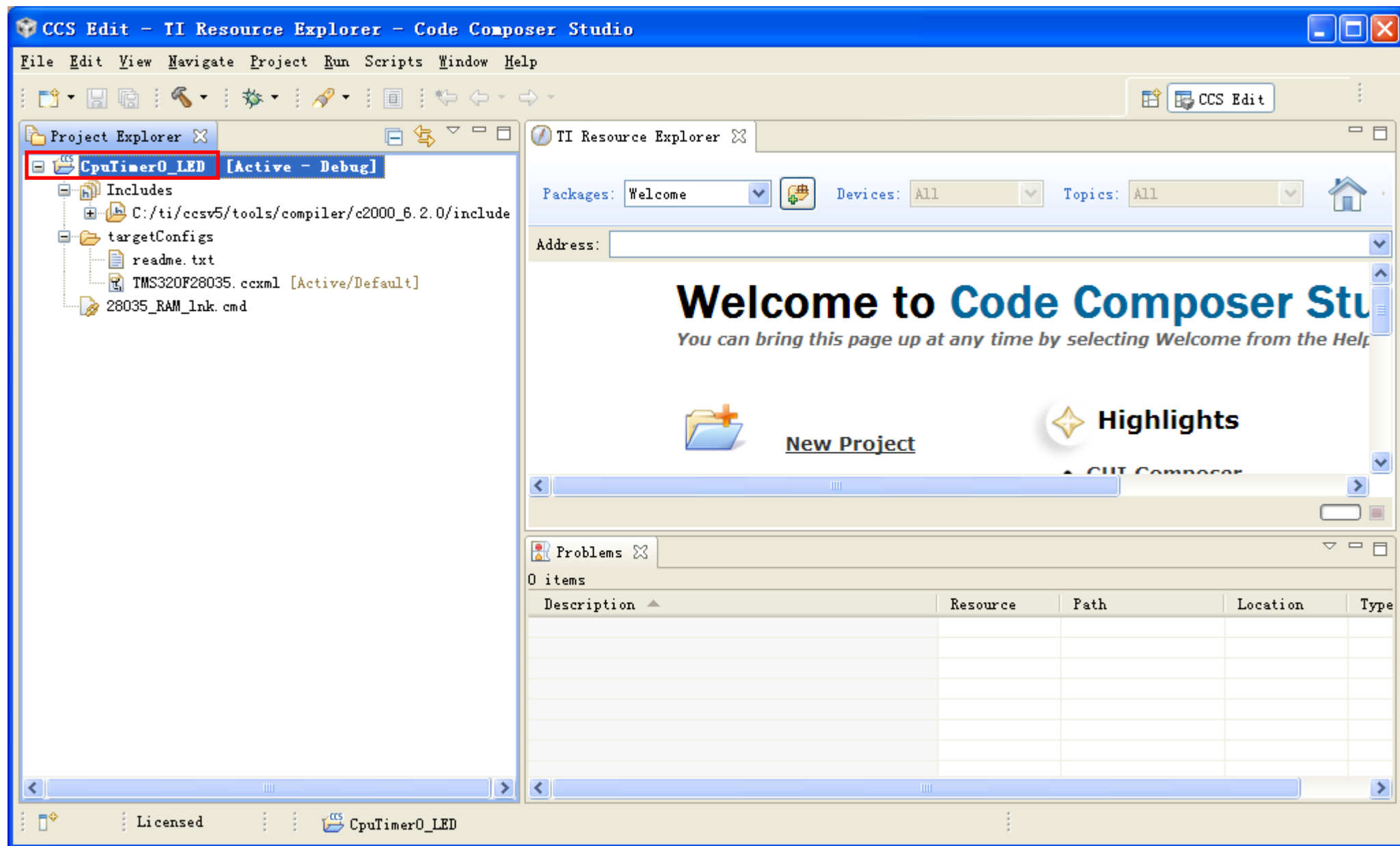
Creates an empty project fully initialized for the selected device. The project will contain an empty 'main.c' source-file.

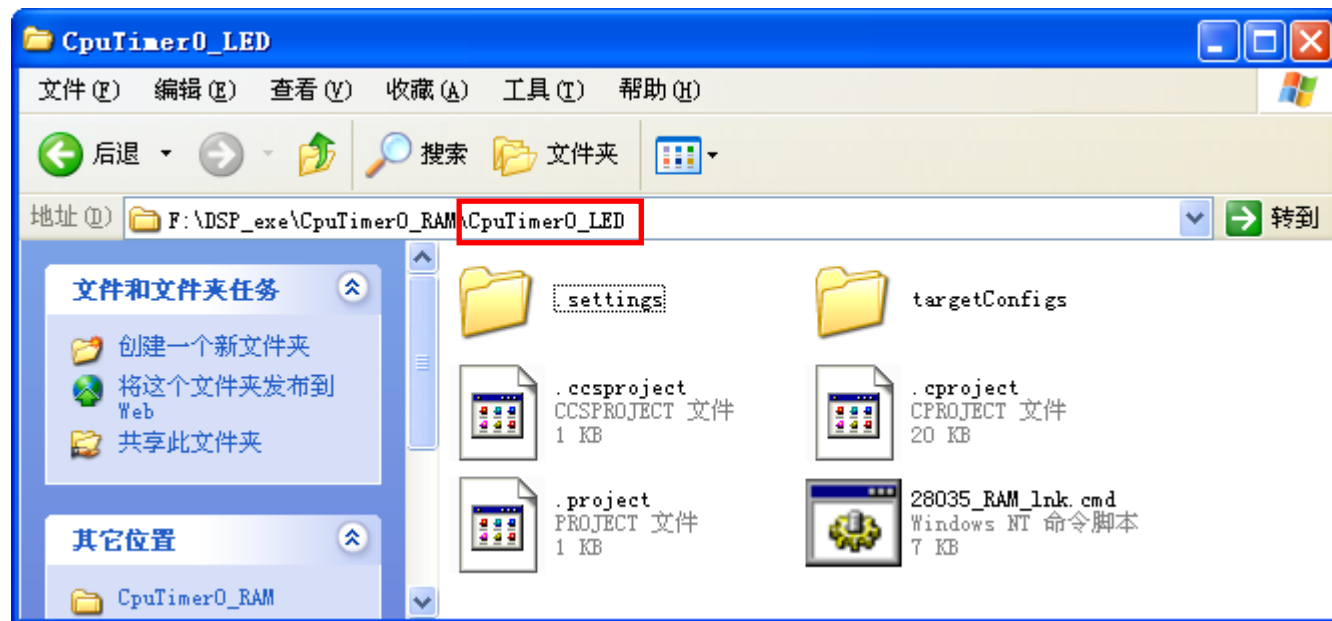
? < Back Next > Finish Cancel

3、将上面的参数修改后如下图：




- 4、点击 **Finish** 后，在 **Project Explorer** 窗口下出现了添加的 **CpuTimer0_LED** 项目，如下第一个图，并且在 CpuTimer0_LED 文件夹也自动添加了一些文件，如下第二个图。

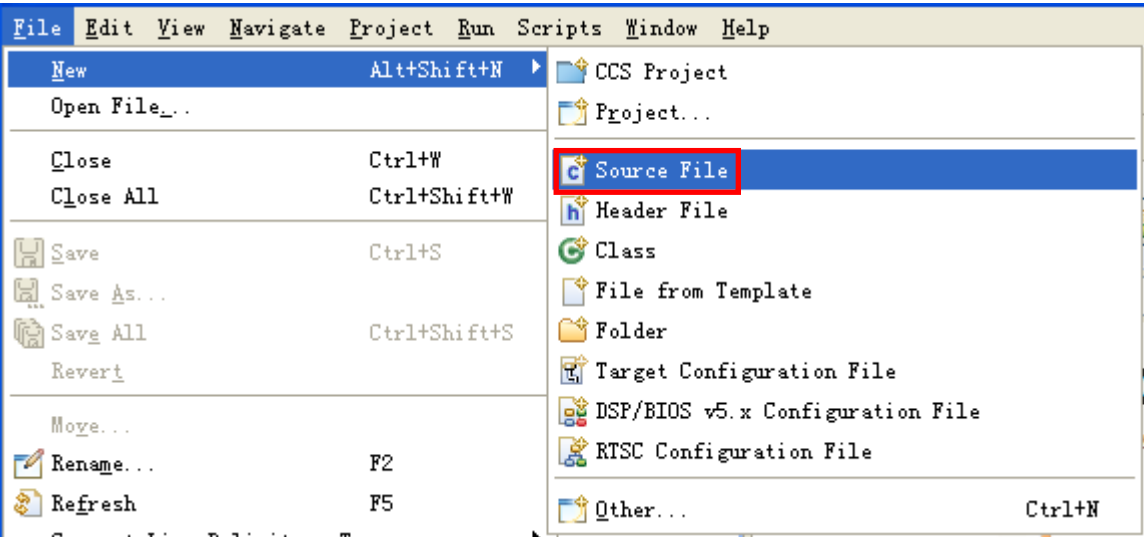




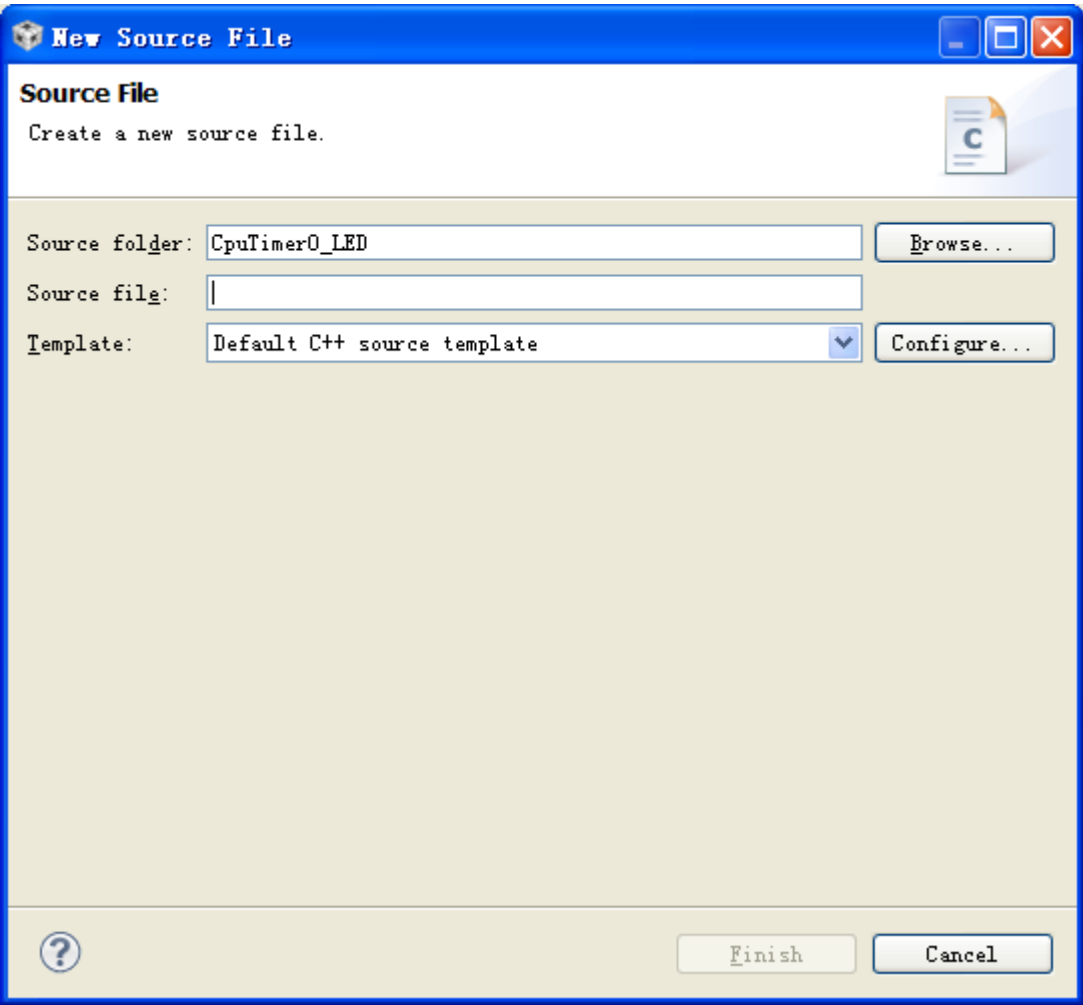
第四步：新建一个主函数源文件

1、File→New→Source File，如下面界面：

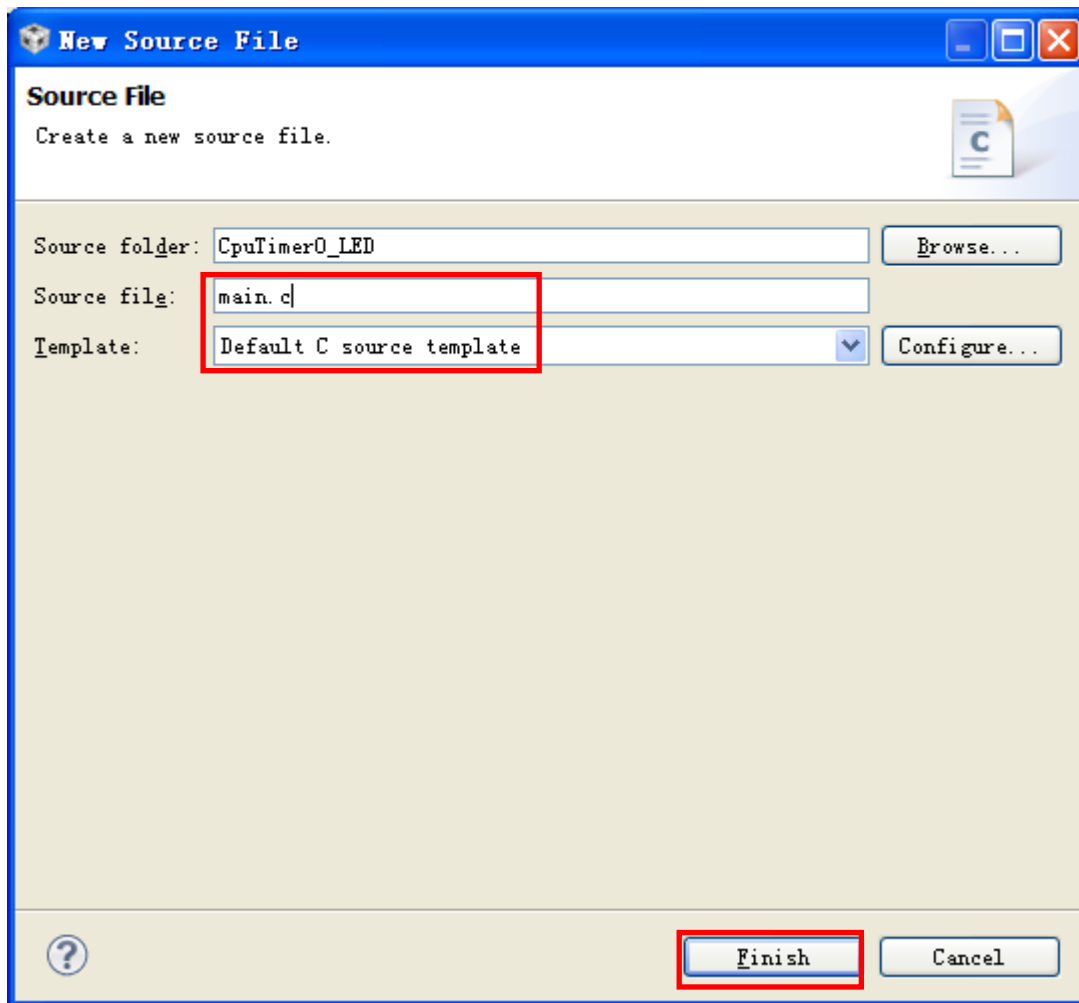
（提示：事实上，也可以从其它地方拷贝一个 main.c 文件到 “CpuTimer0_LED” 文件夹目录下，拷贝完成后，在  **CpuTimer0_LED** [Active - Debug] 下框会自动出现 main.c 文件。见下面的第 4 步）



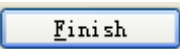
2、点击  Source File，出现如下对话框：

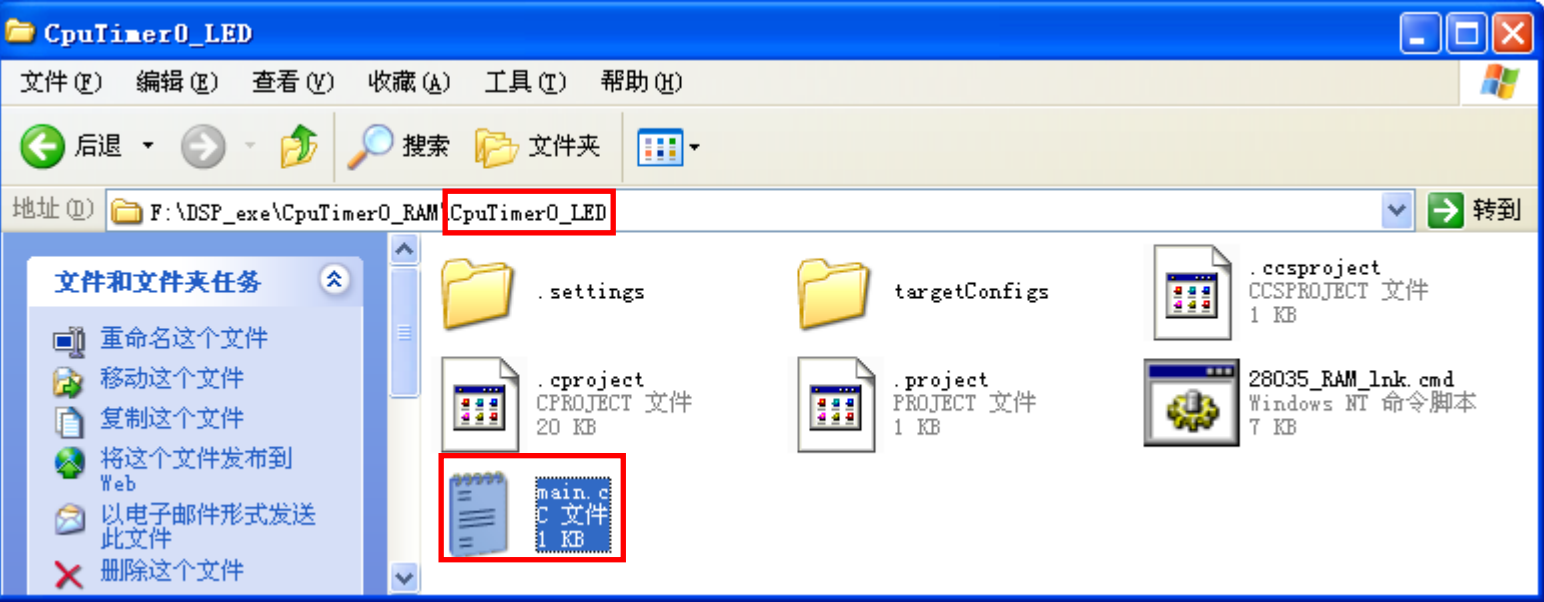


3、在 Source file: 填写源文件名“main.c”，在 Template: 选项选择 Default C source template，如下图：

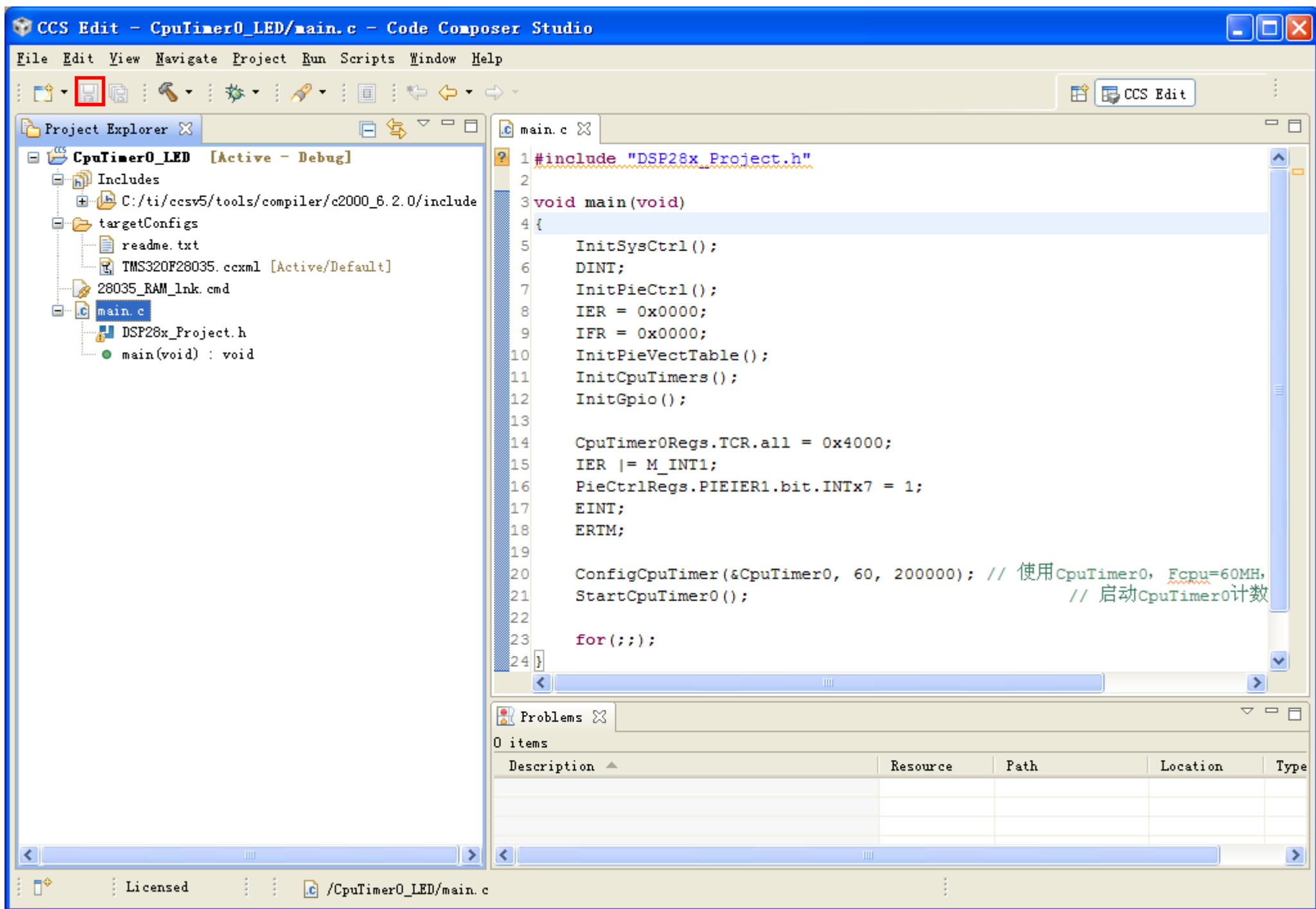


Finish



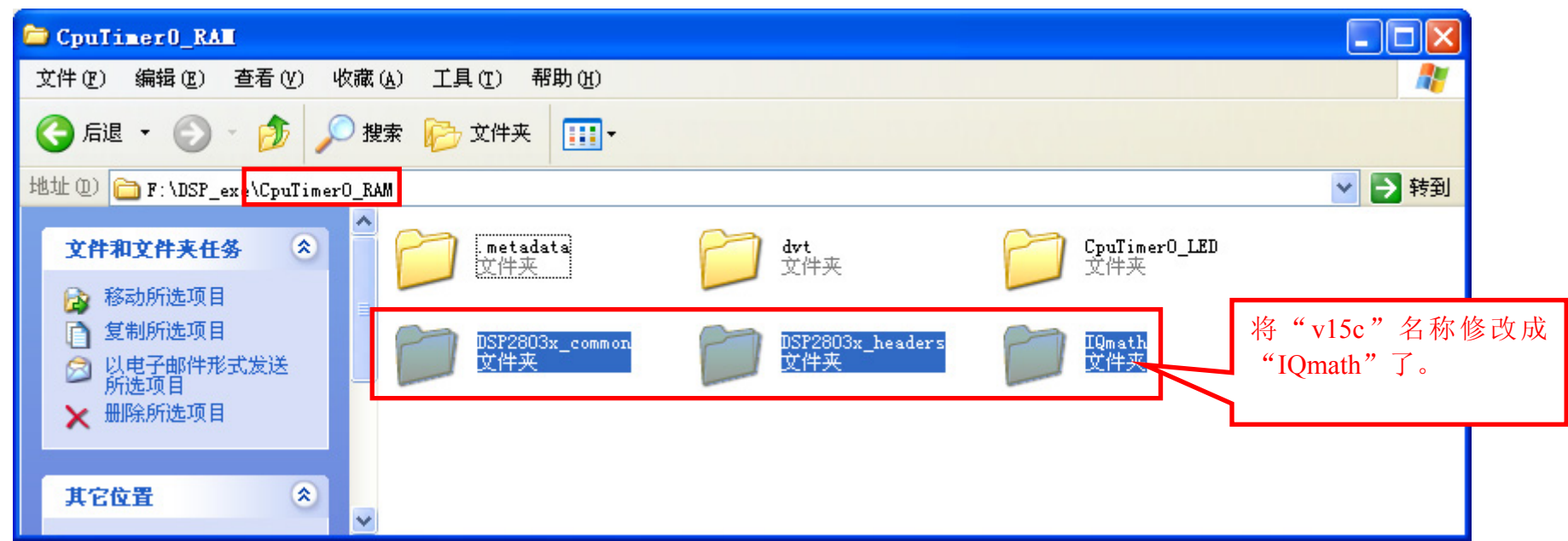


5、在 main.c 源文件编写或复制完程序后，点击  保存，如下图：

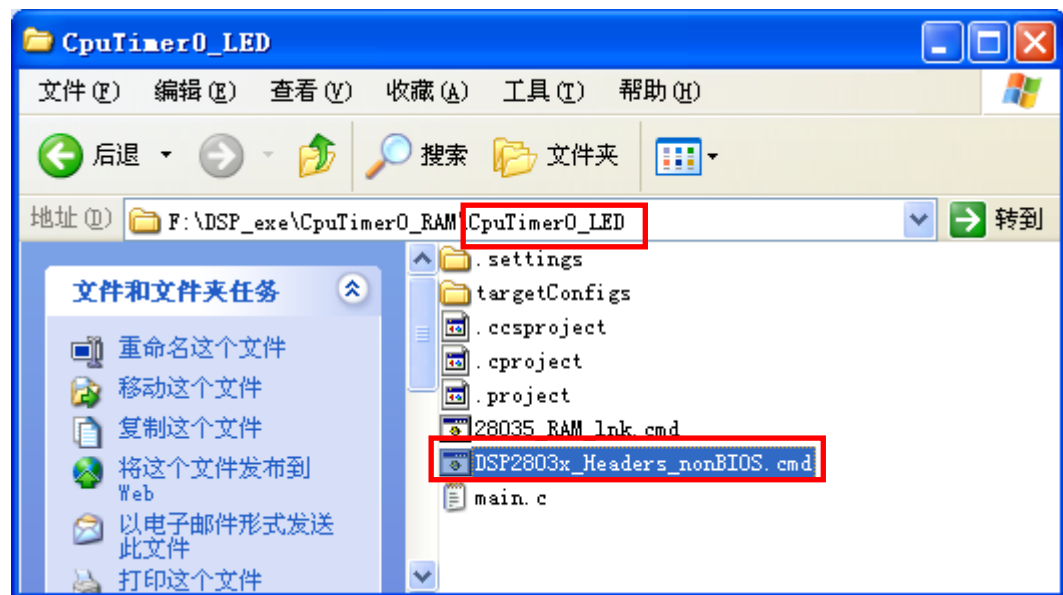


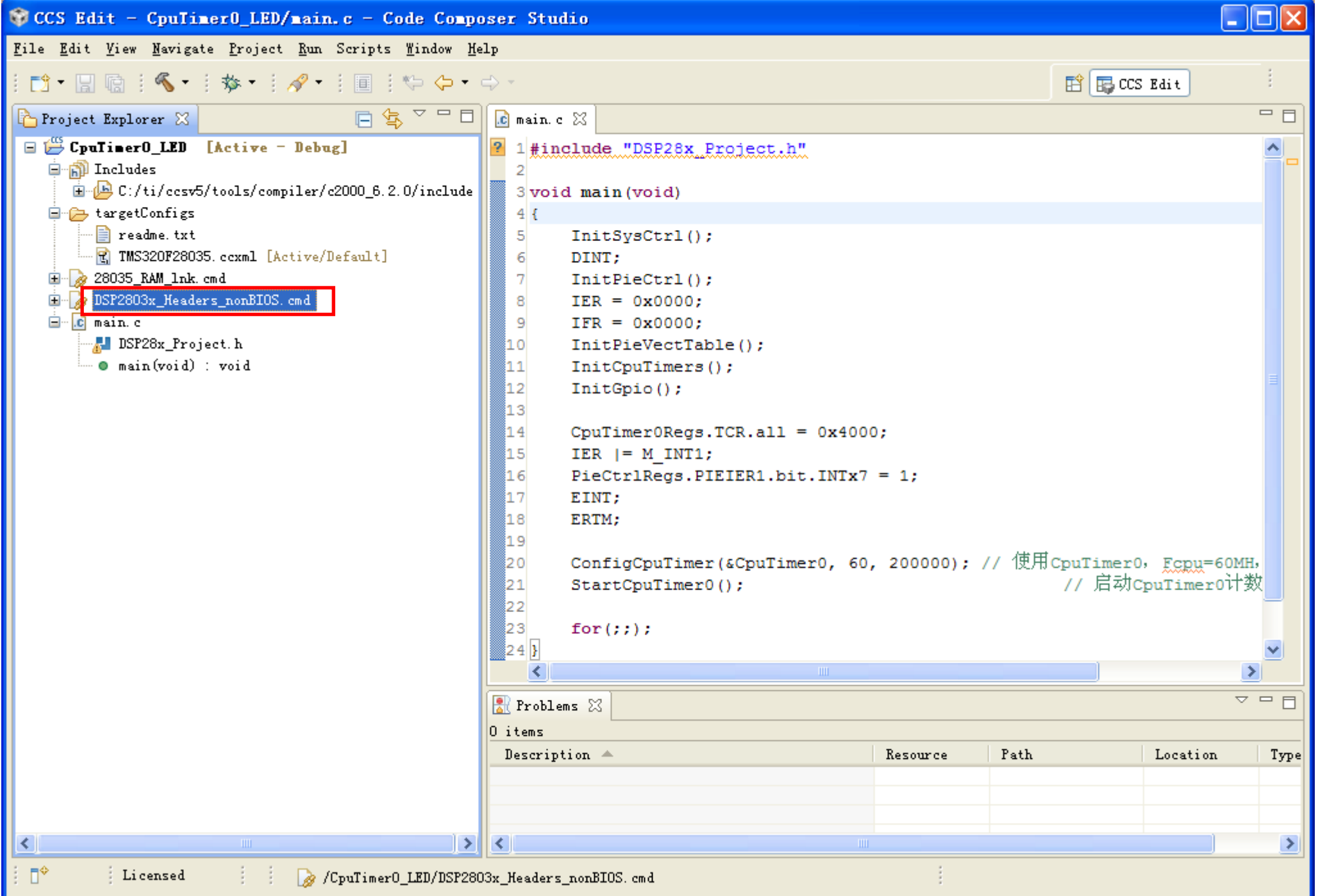
第五步：拷贝 TI DSP2803x 的标准文件

- 1、从 TI 官方网上下载 DSP2803x 所用的标准文件 DSP2803x_common、DSP2803x_headers、IQmath，并拷贝到“CpuTimer0_RAM”目录下，如下图：
- （提示：建议安装 TI 的“controlSUITE3.2.5setup.exe”软件，该软件包含 TI DSP 各式各样的文件，资料非常丰富。默认安装完成后，在
- ①、C:\ti\controlSUITE\device_support\2803x\v126 目录下拷贝“DSP2803x_common”、“ DSP2803x_headers”两个文件夹到“CpuTimer0_RAM”目录下。
 - ②、C:\ti\controlSUITE\libs\math\IQmath 目录下拷贝“v15c”文件夹（即 IQmath 文件夹）到“CpuTimer0_RAM”目录下。）

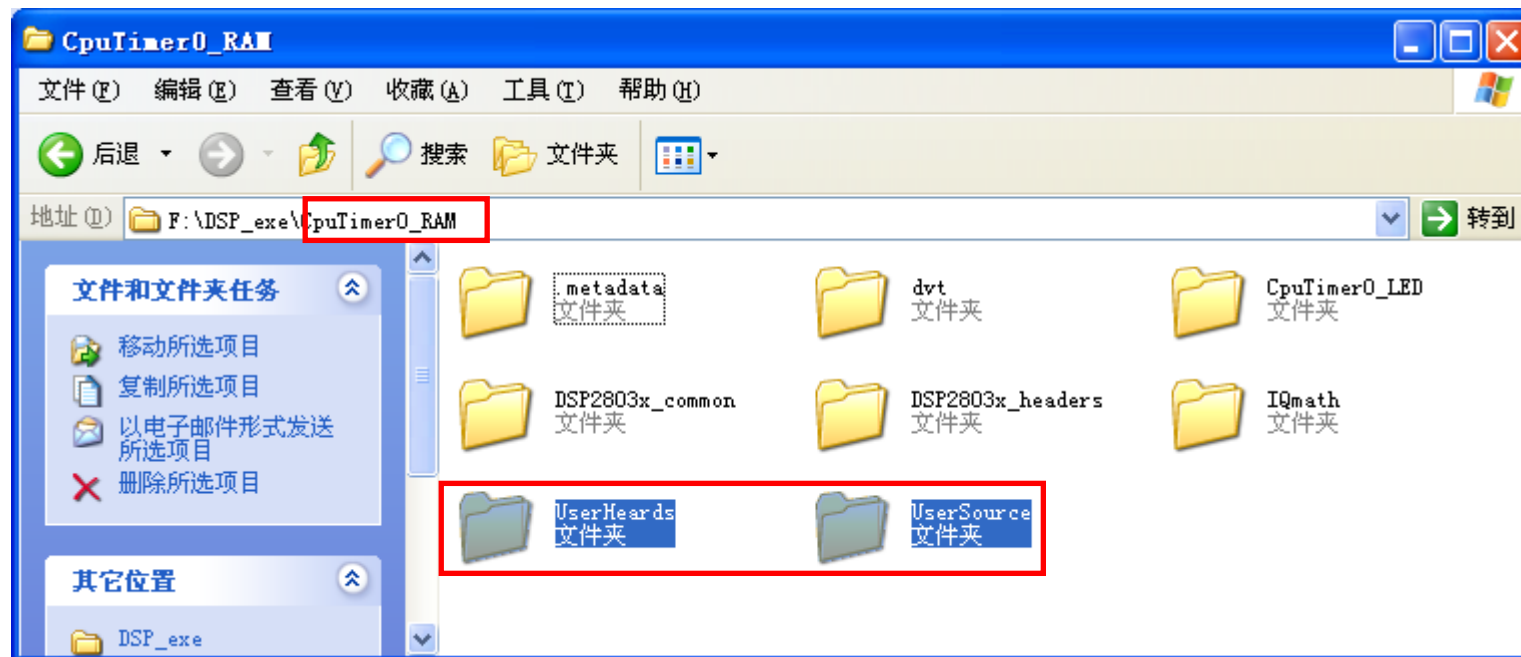


2、从“DSP2803x_headers”文件夹目录下拷贝“DSP2803x-Headers_nonBIOS.cmd”文件到“CpuTimer0_LED”文件夹目录下，如下第一张图，同时在 CCS 工程的“CpuTimer0_LED”下自动出现了添加的“DSP2803x-Headers_nonBIOS.cmd”文件，见下面第二张图。



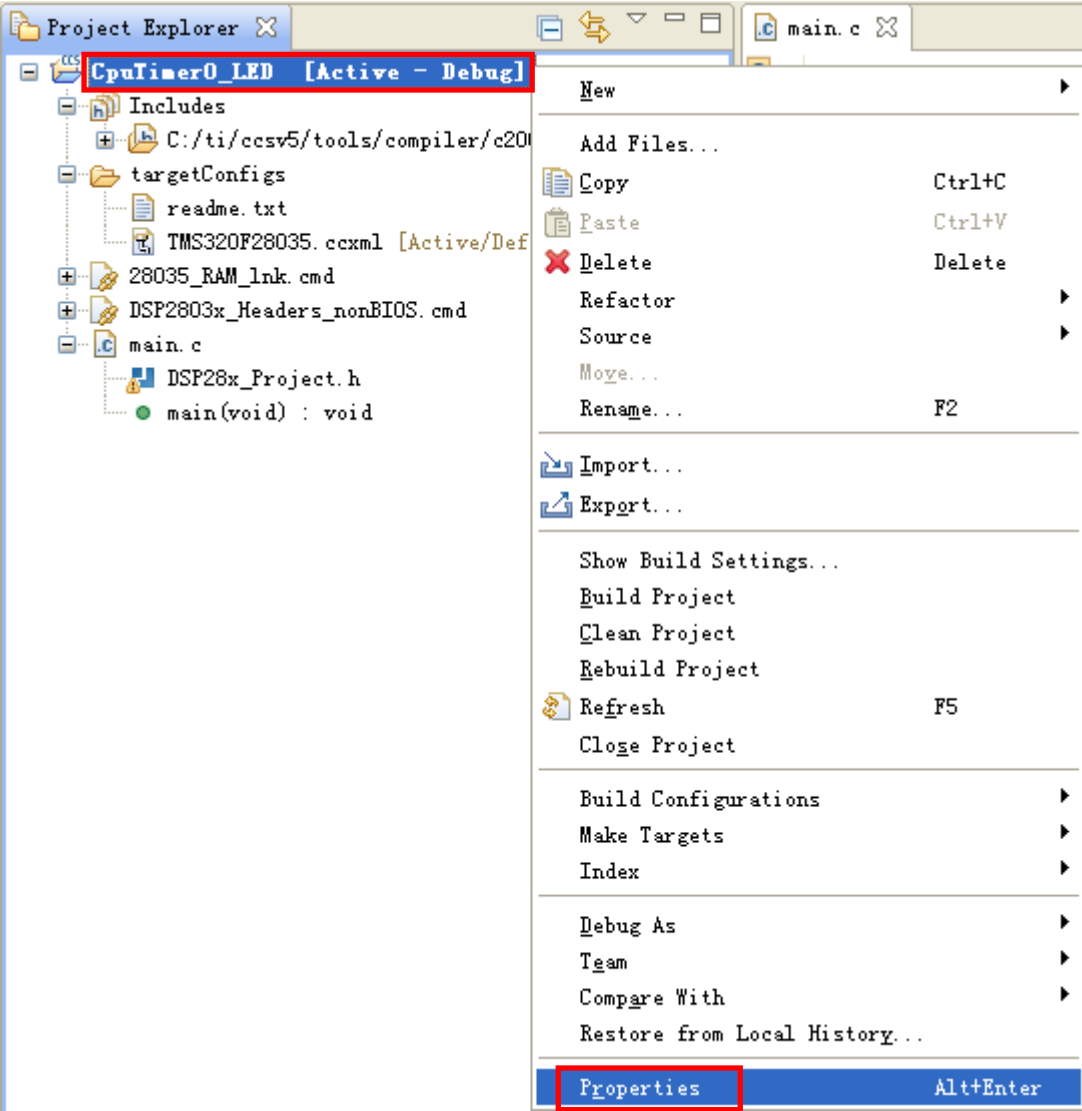


3、若有自己编写的头文件和源文件，则同样拷贝到“CpuTimer0_RAM”目录下，如下图：

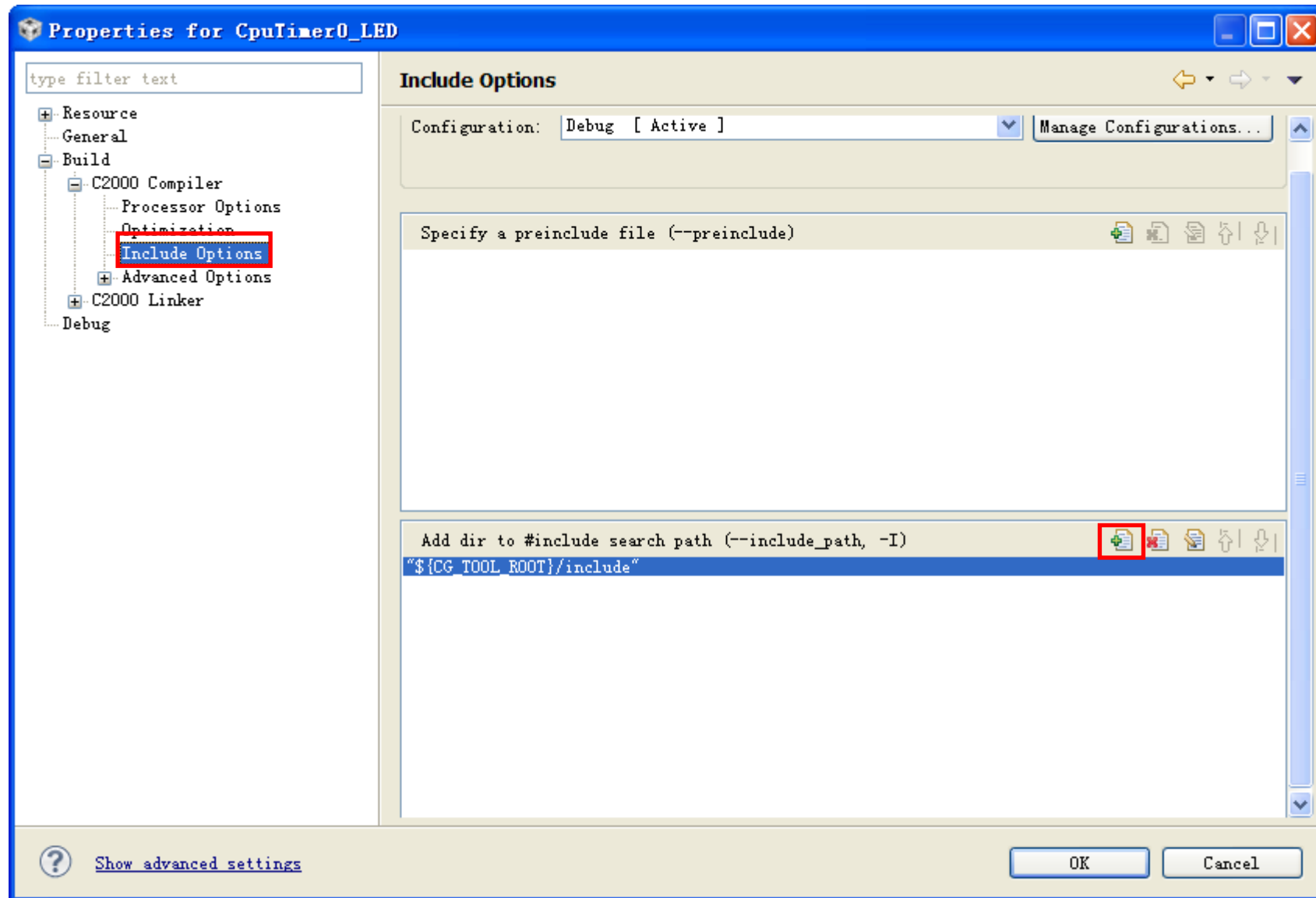


第六步：设置头文件（include）的搜索路径

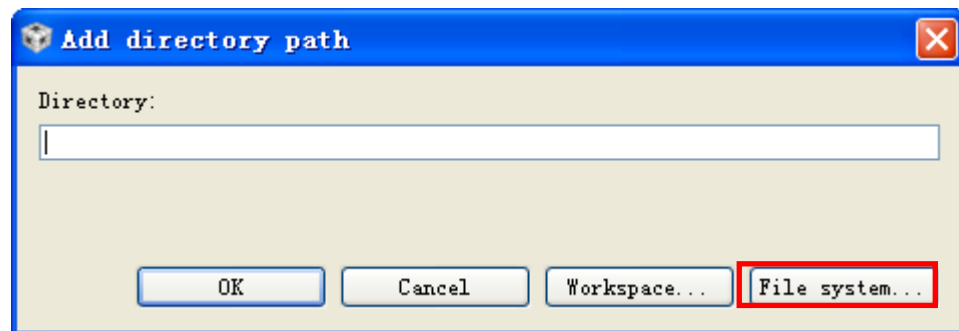
1、鼠标指向工程 **CpuTimer0_LED [Active - Debug]**，点击鼠标右键选中“Properties”如下图：



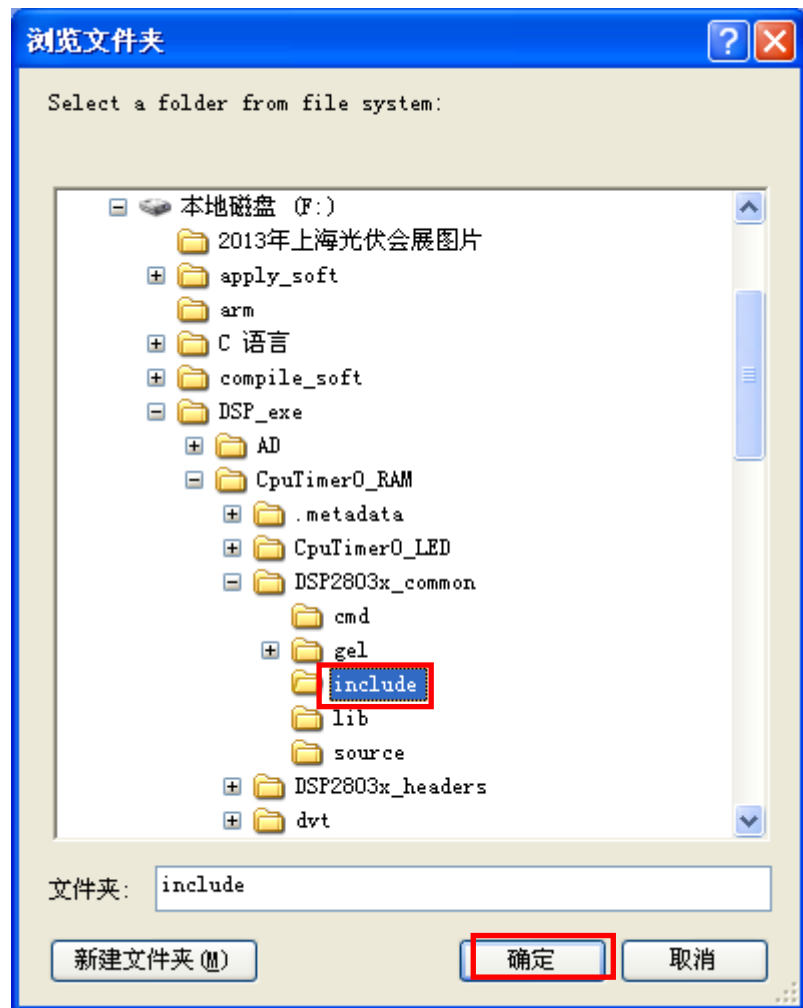
2、鼠标左键点击 Properties，出现如下对话框，并选择 Build→C2000 Compiler→Include Options，如下图：



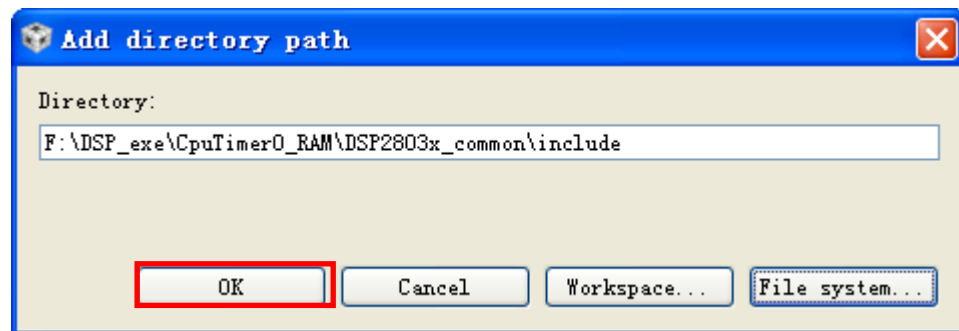
3、点击  出现如下对话框：




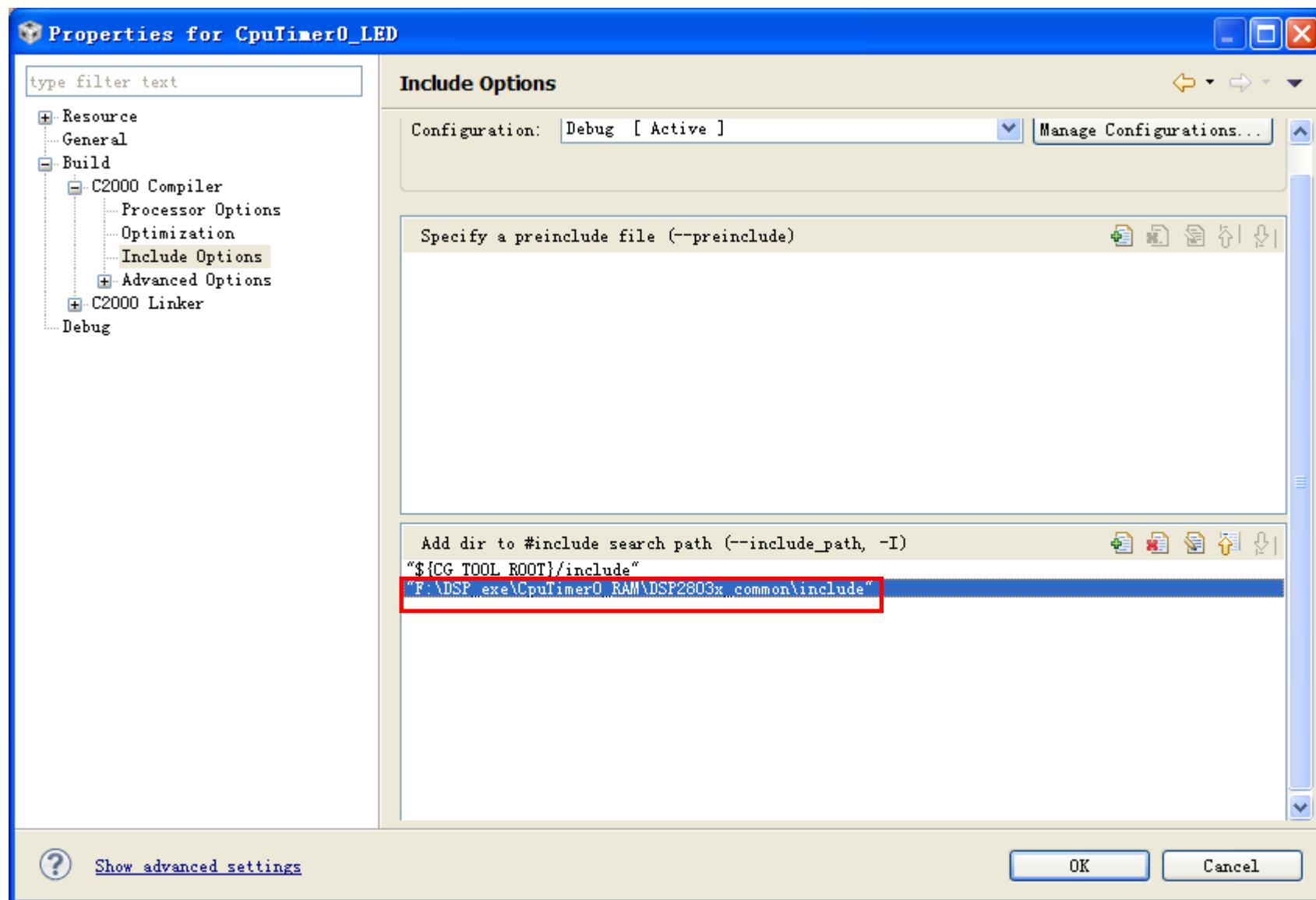
4、点击 **File system...** 出现如下对话框，并选择头文件路径位置 `CpuTimer0_RAM\DSP2803x_common\include`，如下图：



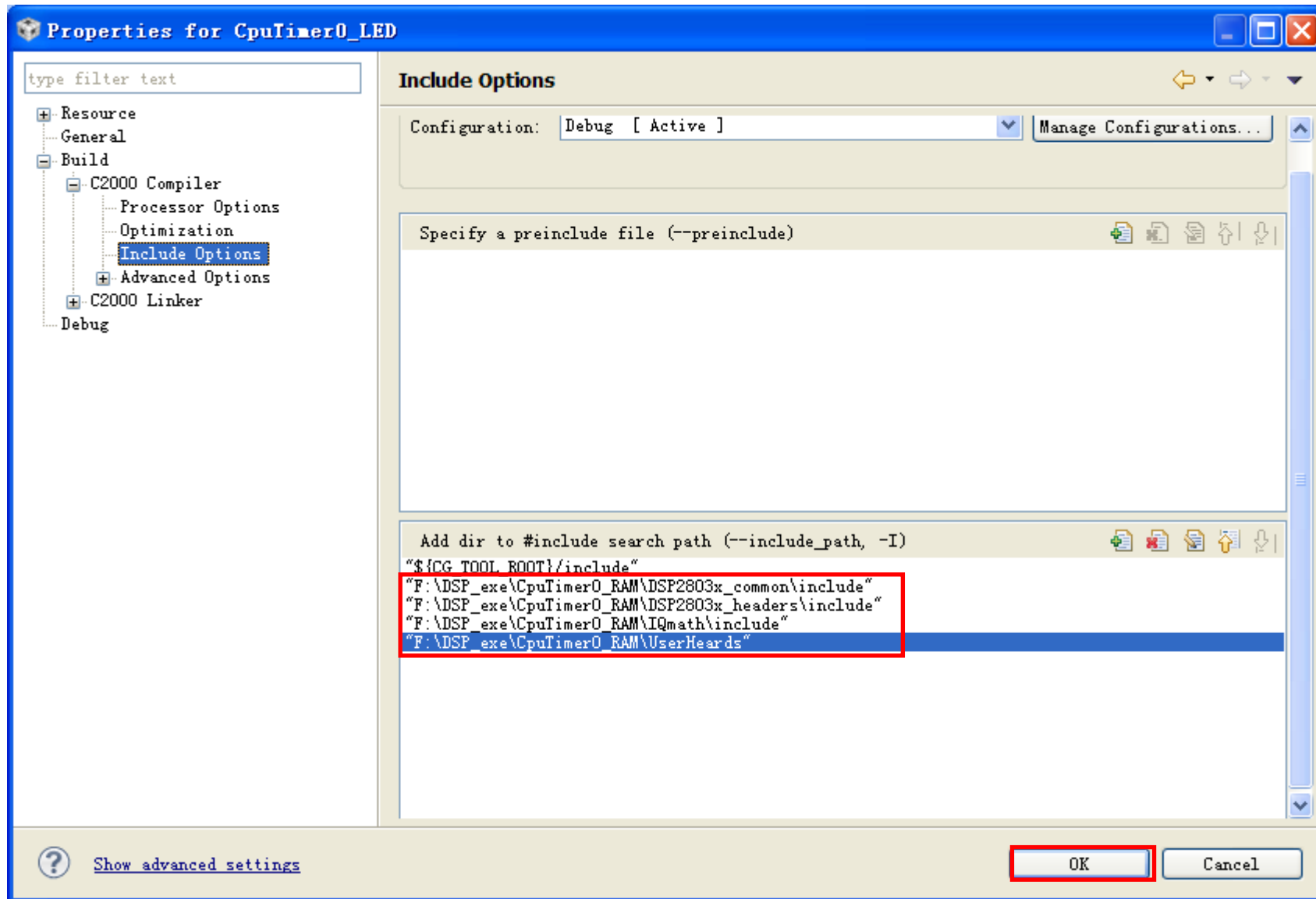
5、点击  后返回如对话框：



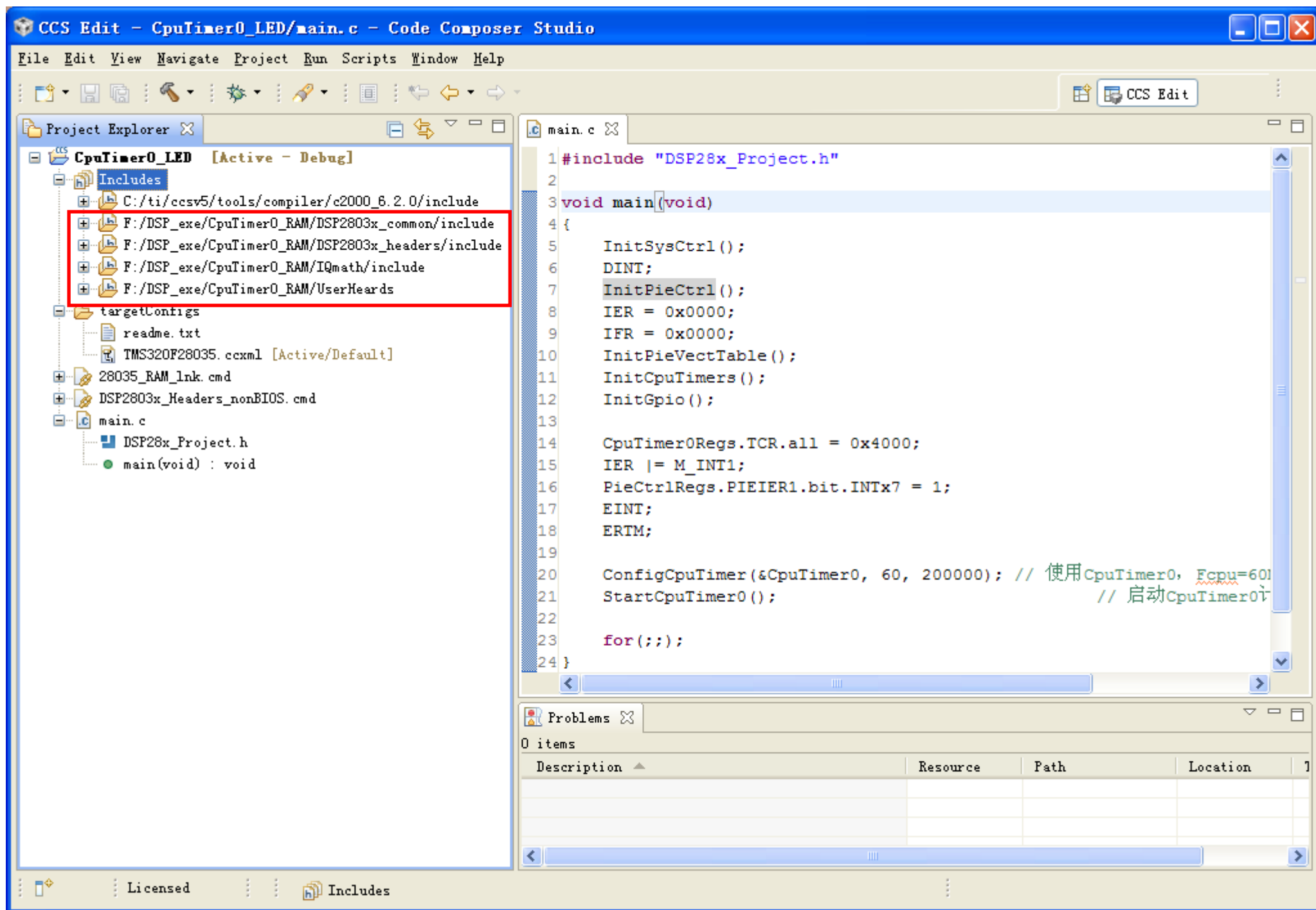
6、点击  后返回如下界面，即添加了头文件的搜索路径为 F:\DSP_exe\CpuTimer0_RAM\DSP2803x_common\include 目录下，如下图：



7、同理：按第 3~6 步骤，继续添加其它标准的或者自己编写的头文件路径，如下图：

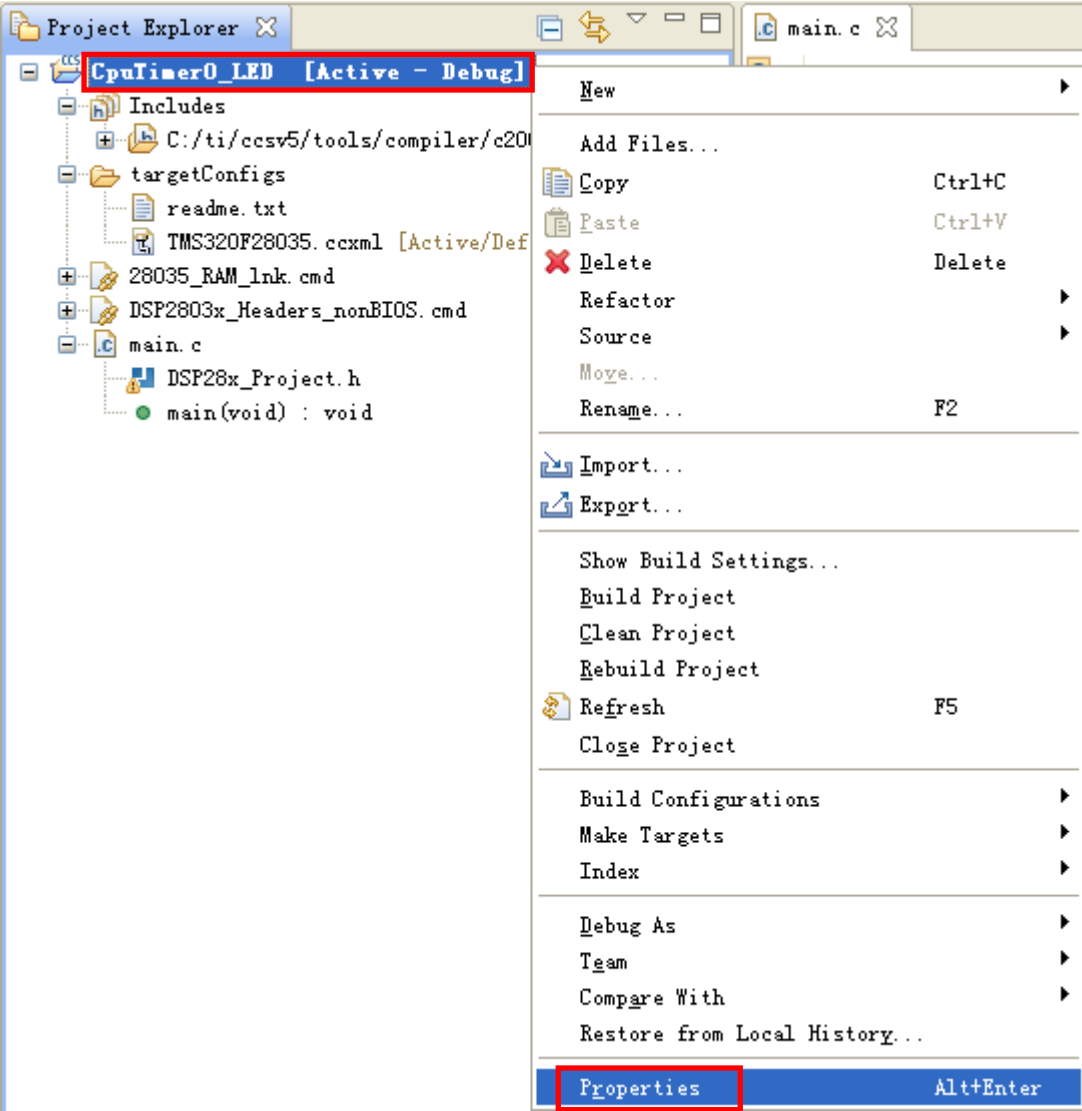


8、添加所需的头文件路径完成后，点击 ，返回如下界面，即  **CpuTimer0_LED** 下的 Includes 出现了刚才设置头文件的搜索路，如下图：

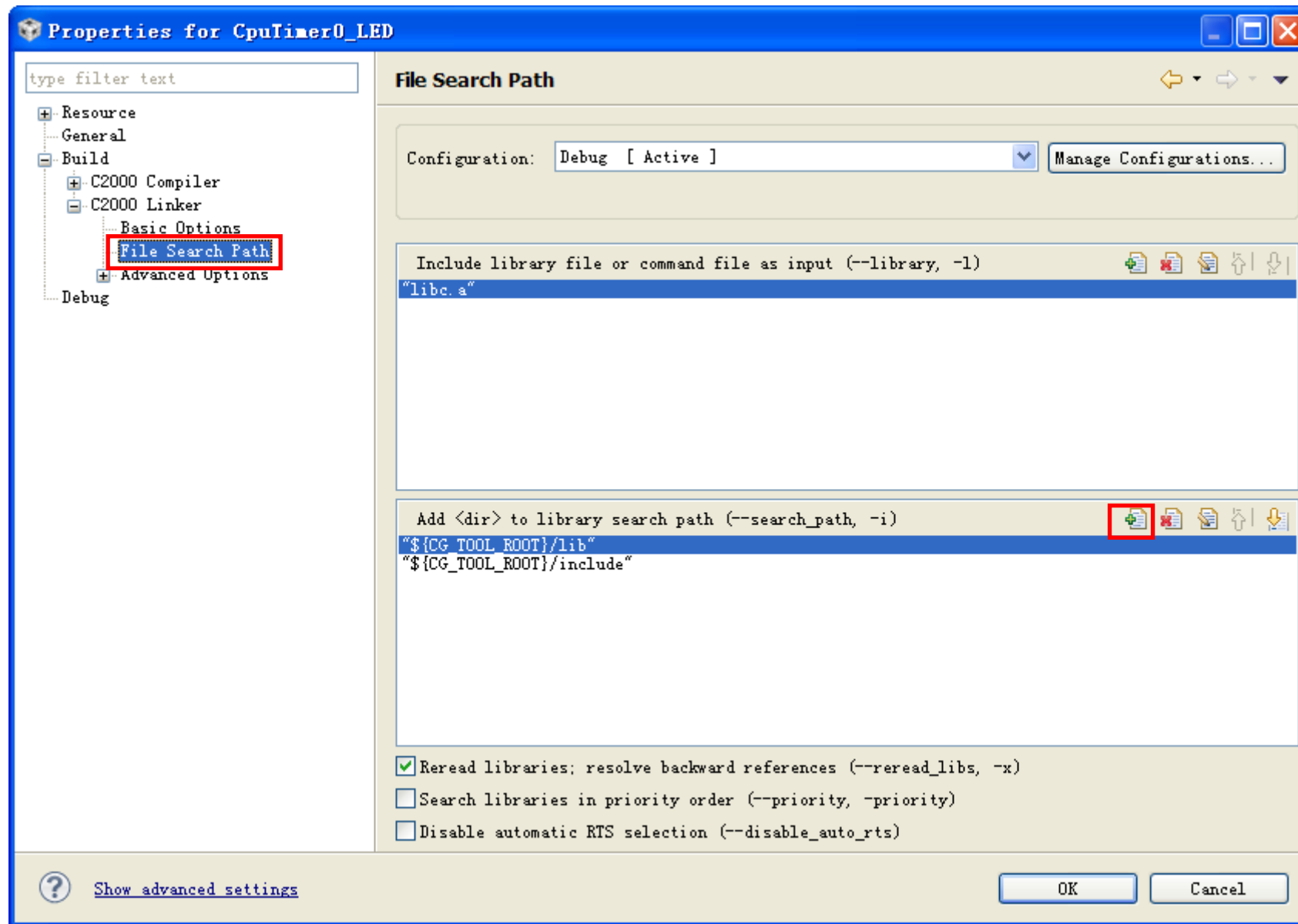


第七步：设置库文件（lib）的搜索路径

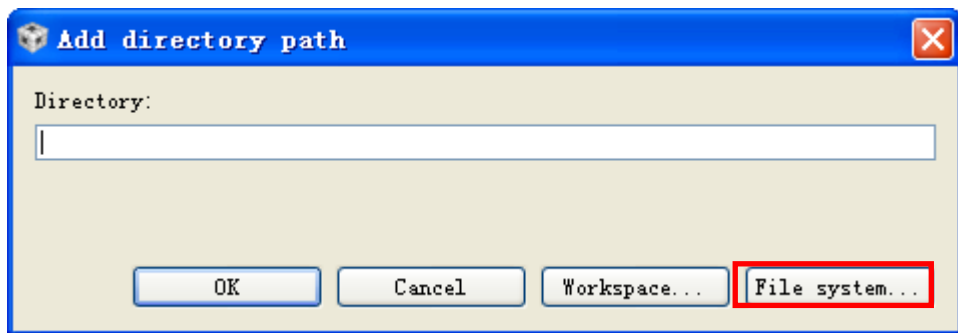
1、鼠标指向工程 **CpuTimer0_LED [Active - Debug]**，点击鼠标右键选中“Properties”如下图：



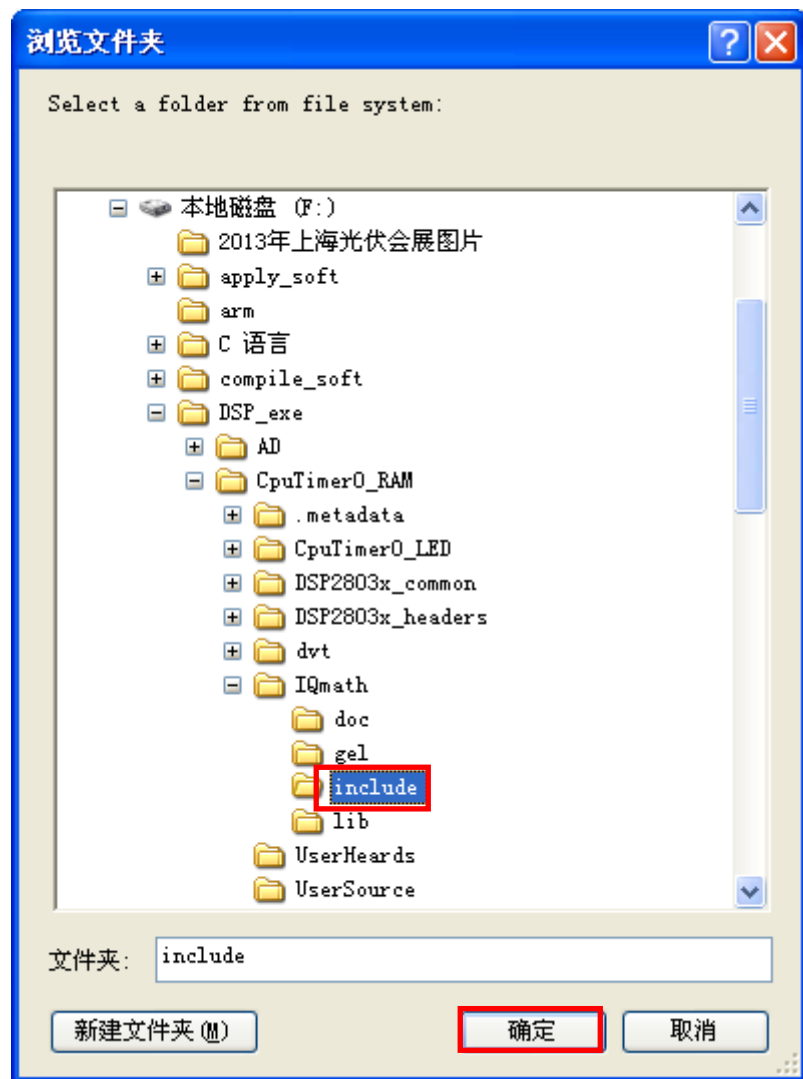
2、鼠标左键点击 Properties，出现如下对话框，并选择 Build→C2000 Linker→File Search Path，如下图：

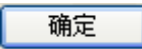


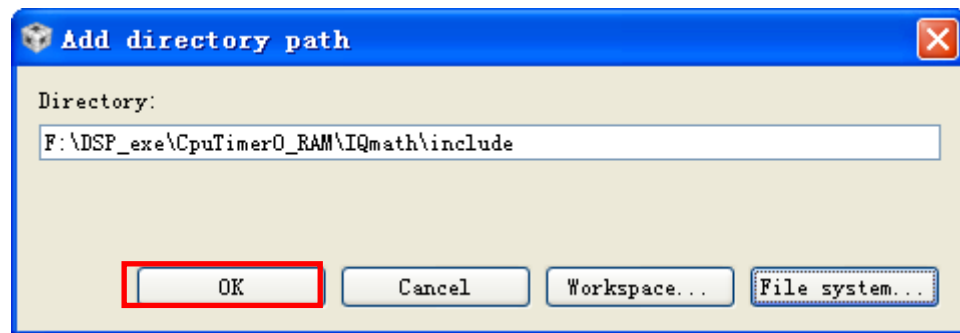
3、点击  出现如下对话框：



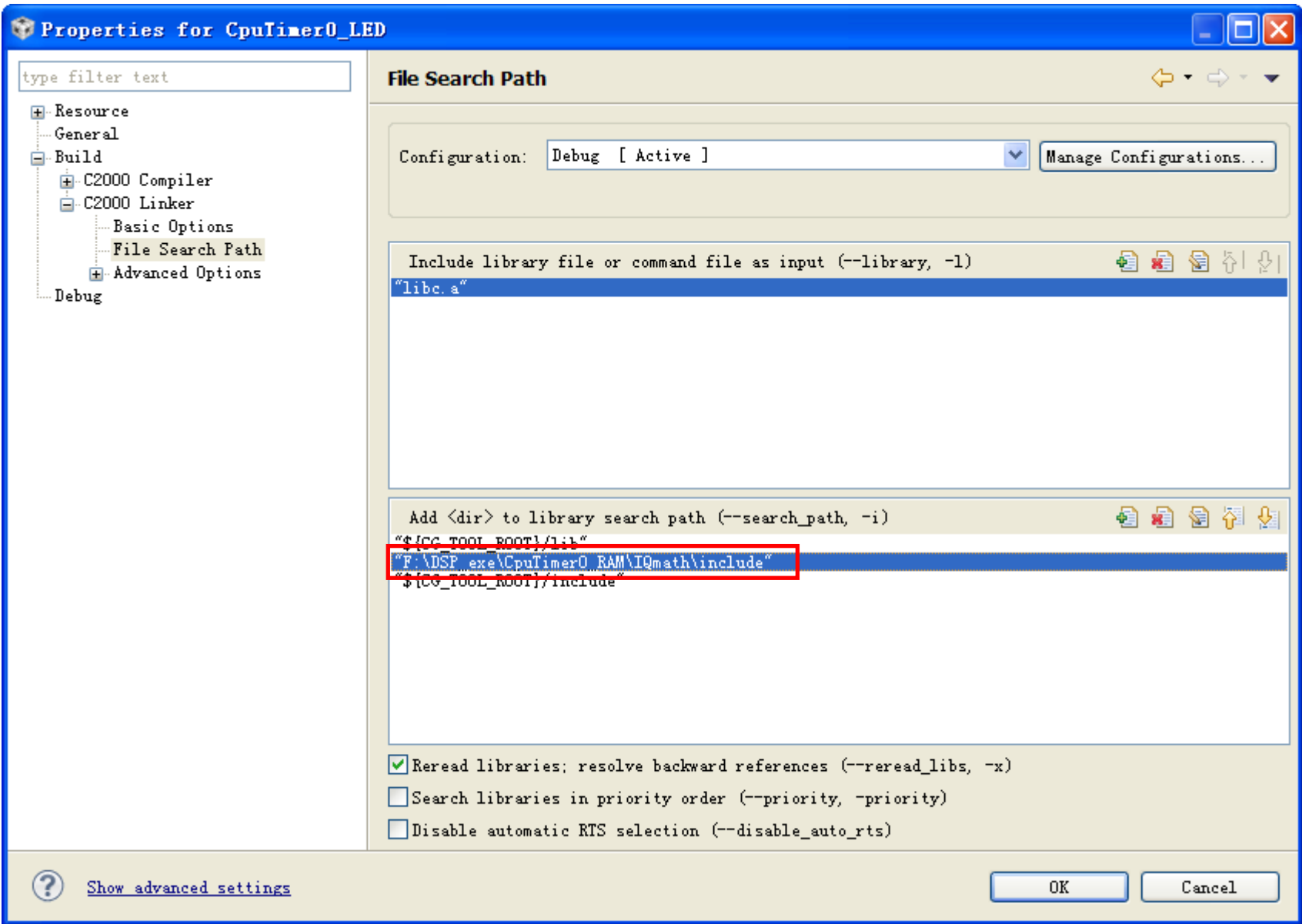
4、点击 **File system...** 出现如下对话框，并选择头文件路径位置 **CpuTimer0_RAM\IQmath\include**，如下图：



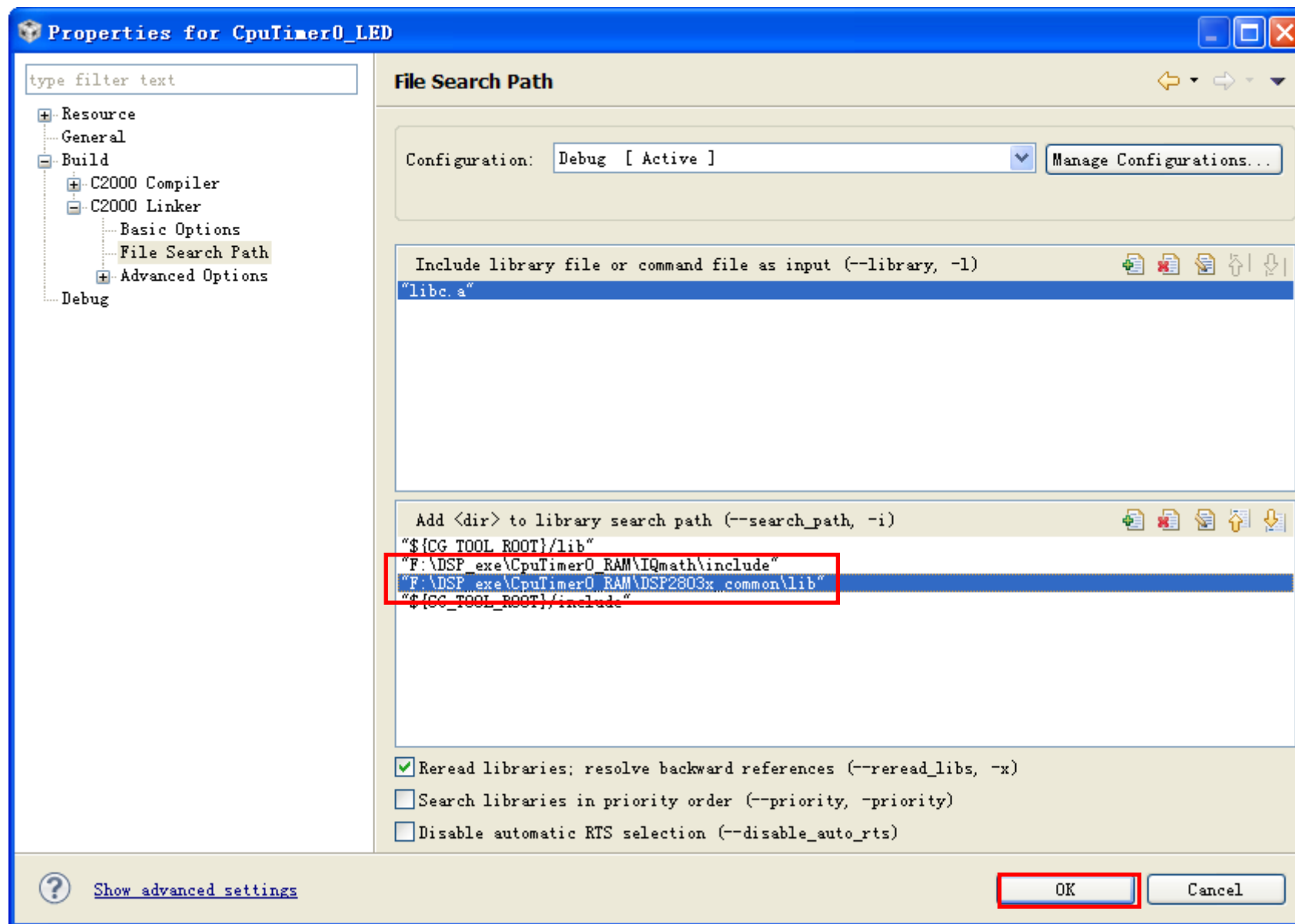
5、点击  后返回如对话框：



6、点击  后返回如下界面，即添加了库文件的搜索路径为 F:\DSP_exe\CpuTimer0_RAM\IQmath\include 目录下，如下图：

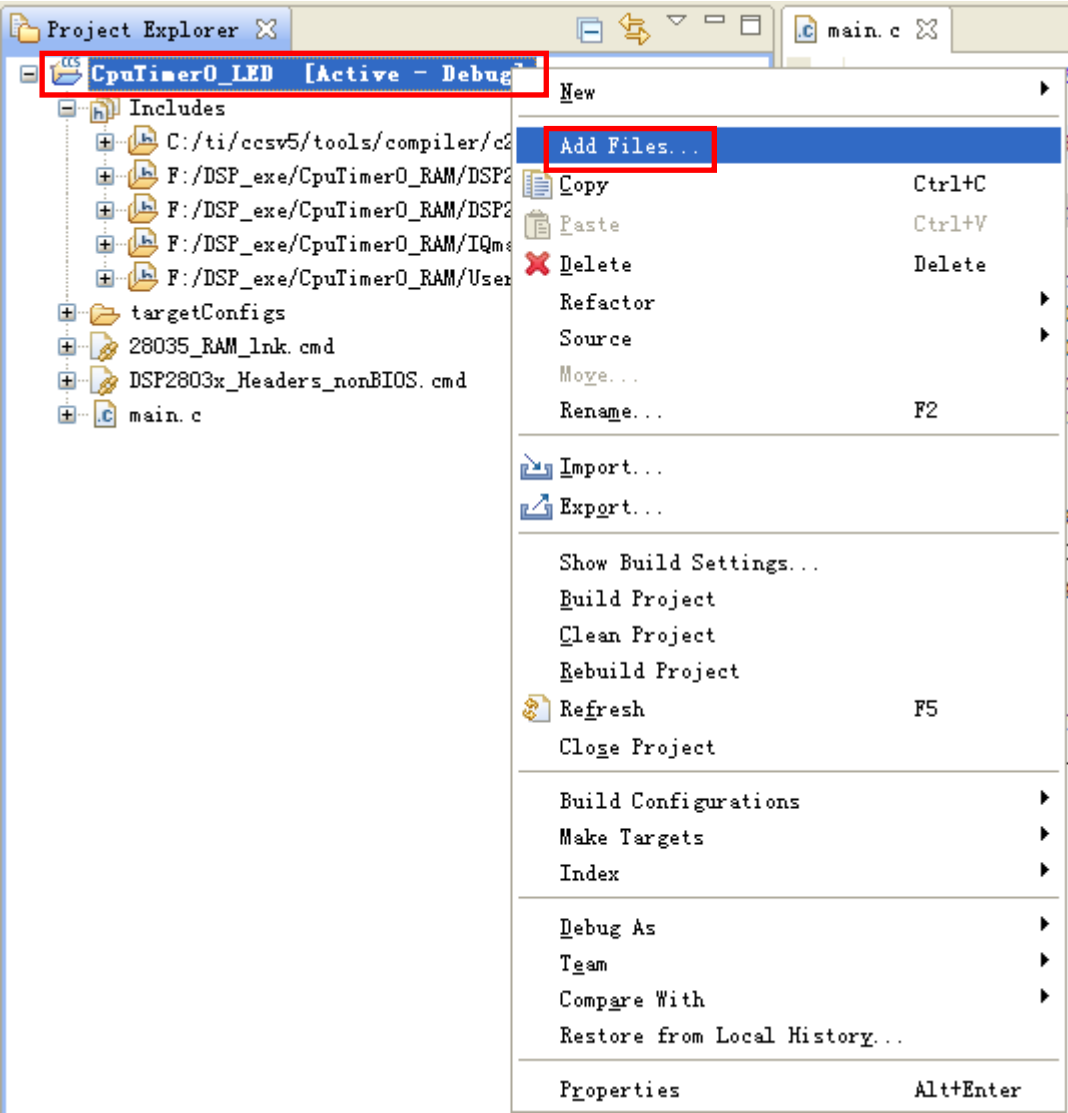


OK

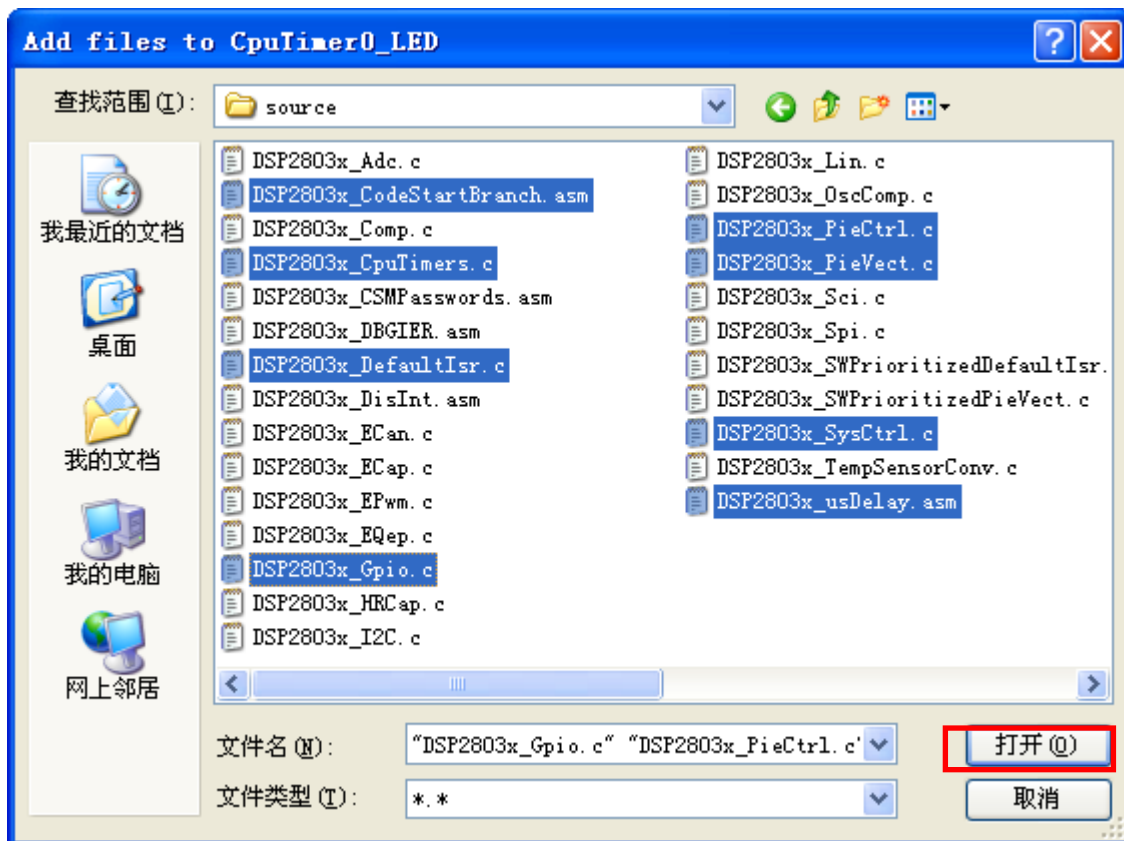


第八步：添加其它需要的源文件

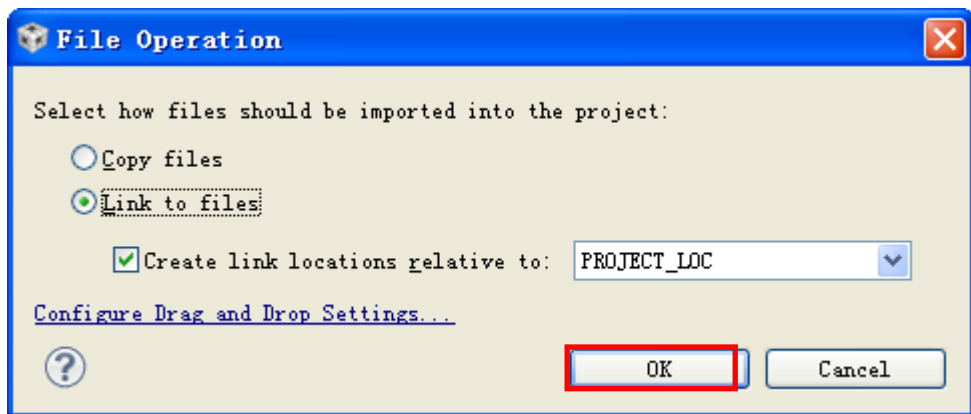
1、鼠标指向工程文件夹 `CpuTimer0_LED [Active - Debug]`，单击鼠标右键，选择 `Add Files...`，如下图：



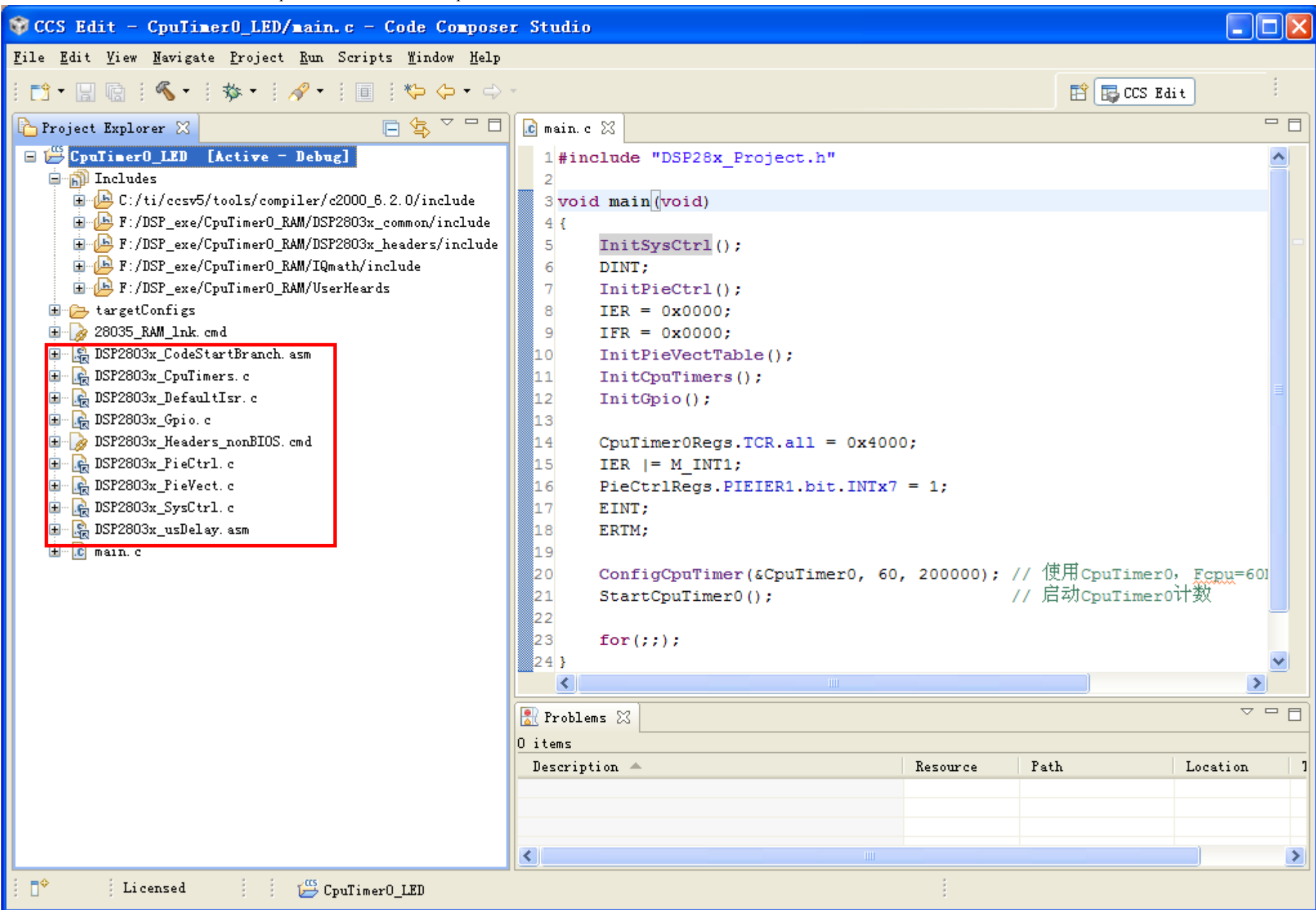
2、鼠标左键点击 **Add Files...**，出现如下类似的对话框，并选择 F:\DSP_exe\CpuTimer0_RAM\DSP2803x_common\source 目录下需要的源文件，如下图所示：



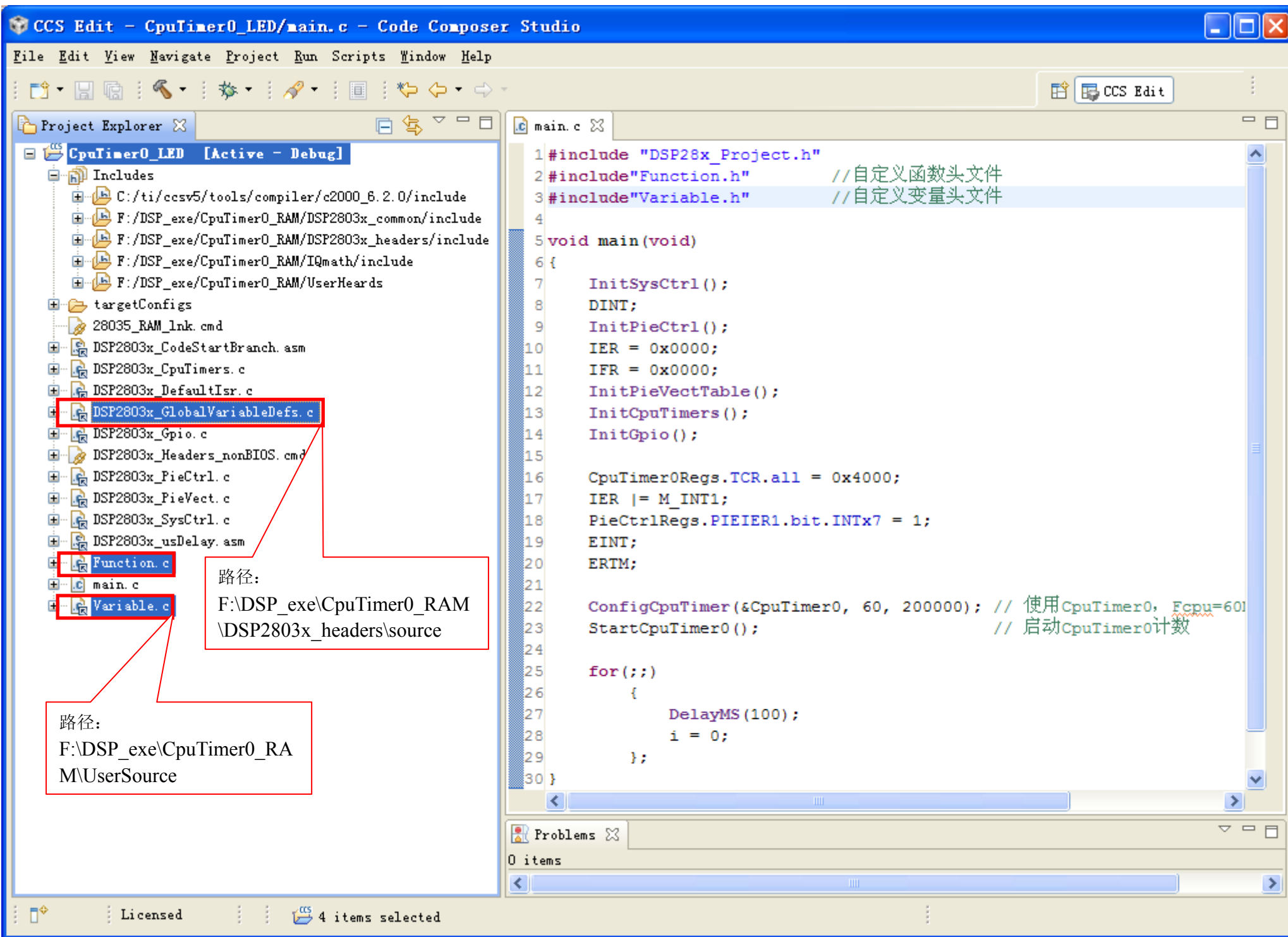
3、然后点击 **打开(O)**，出现如下对话框，并选择 **Link to files**，如下图：



- 4、然后点击 ，返回如下界面，即在 **CpuTimer0_LED [Active - Debug]** 下出现了添加的源文件，如下图：
(说明：本示例程是使用 CpuTimer0，定时进入 CpuTimer0 中断函数，周期性的控制 GPIO13 输出高、低电平使 LED 闪烁。)

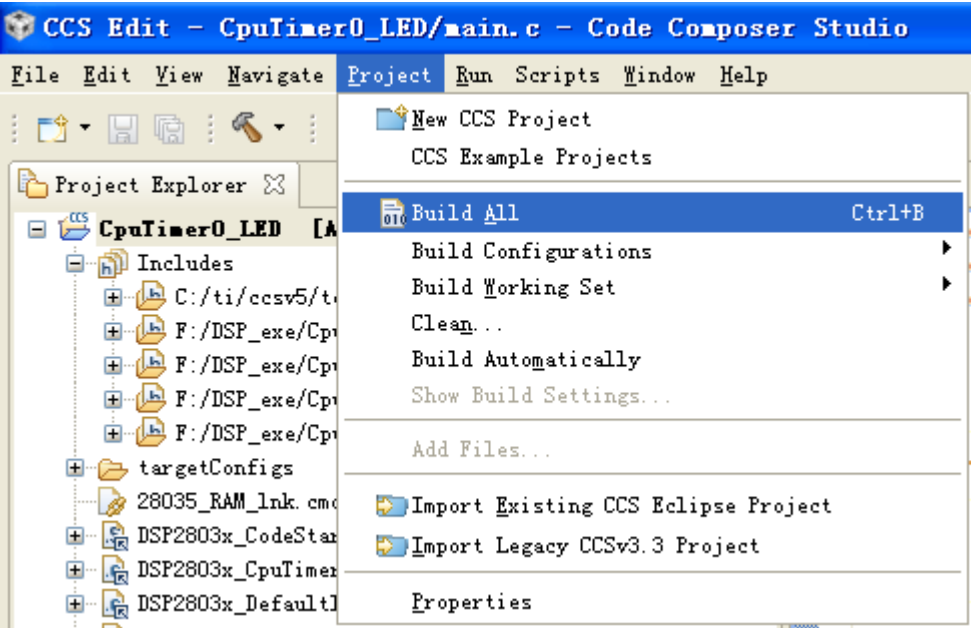


5、同理，按第1~4步继续添加源文件，添加完成后如下图：

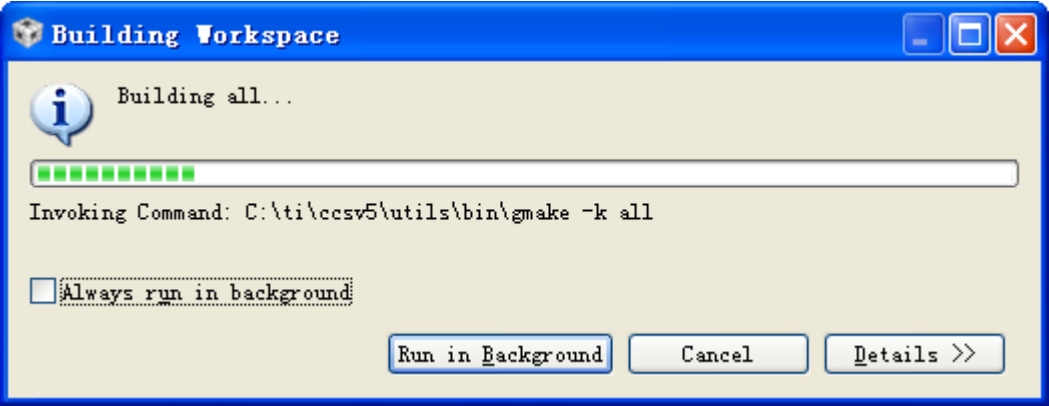


第九步：编译：当所有的文件都添加完成后，现在就是对源文件进行编译了。

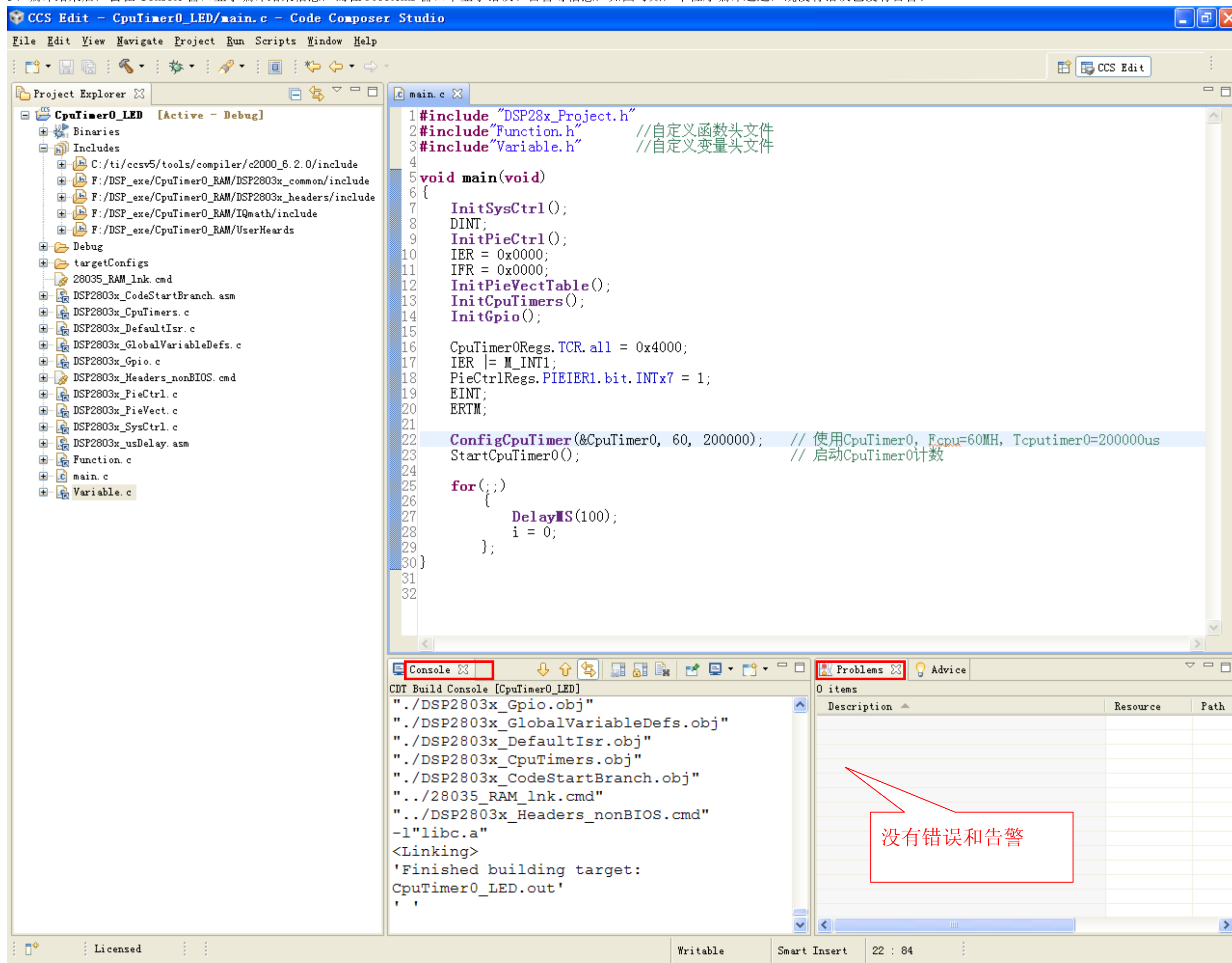
1、Project→Build All，如下图：



2、点击 **Build All**，编译所有源文件，出现如下对话框。

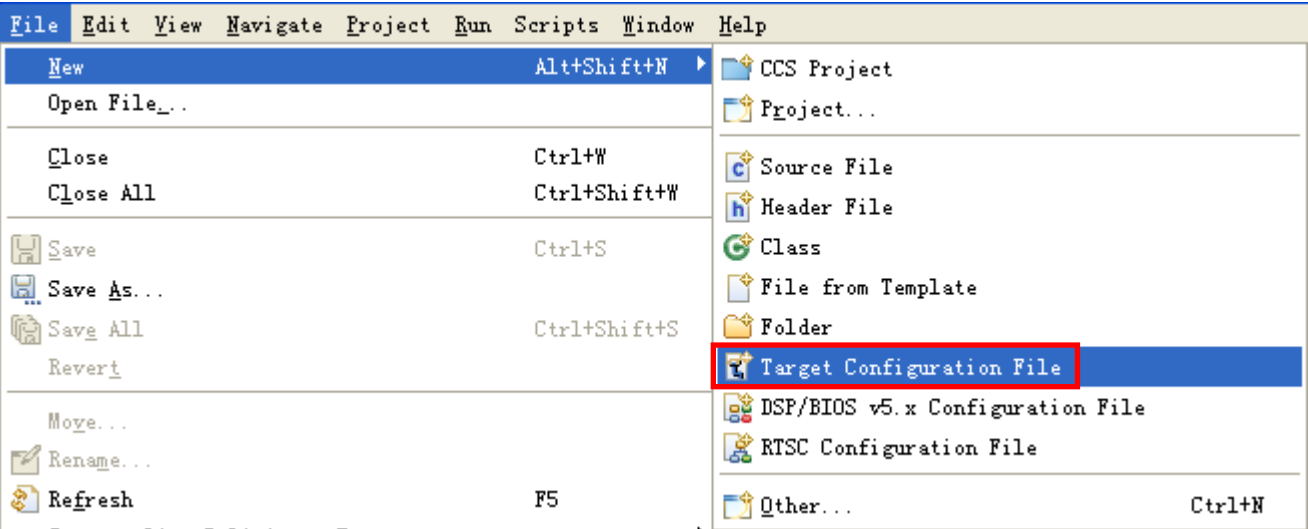



3、编译结束后，会在 Console 窗口显示编译结果信息，而在 Problems 窗口中显示错误、告警等信息，如图可知，本程序编译通过，既没有错误也没有告警。

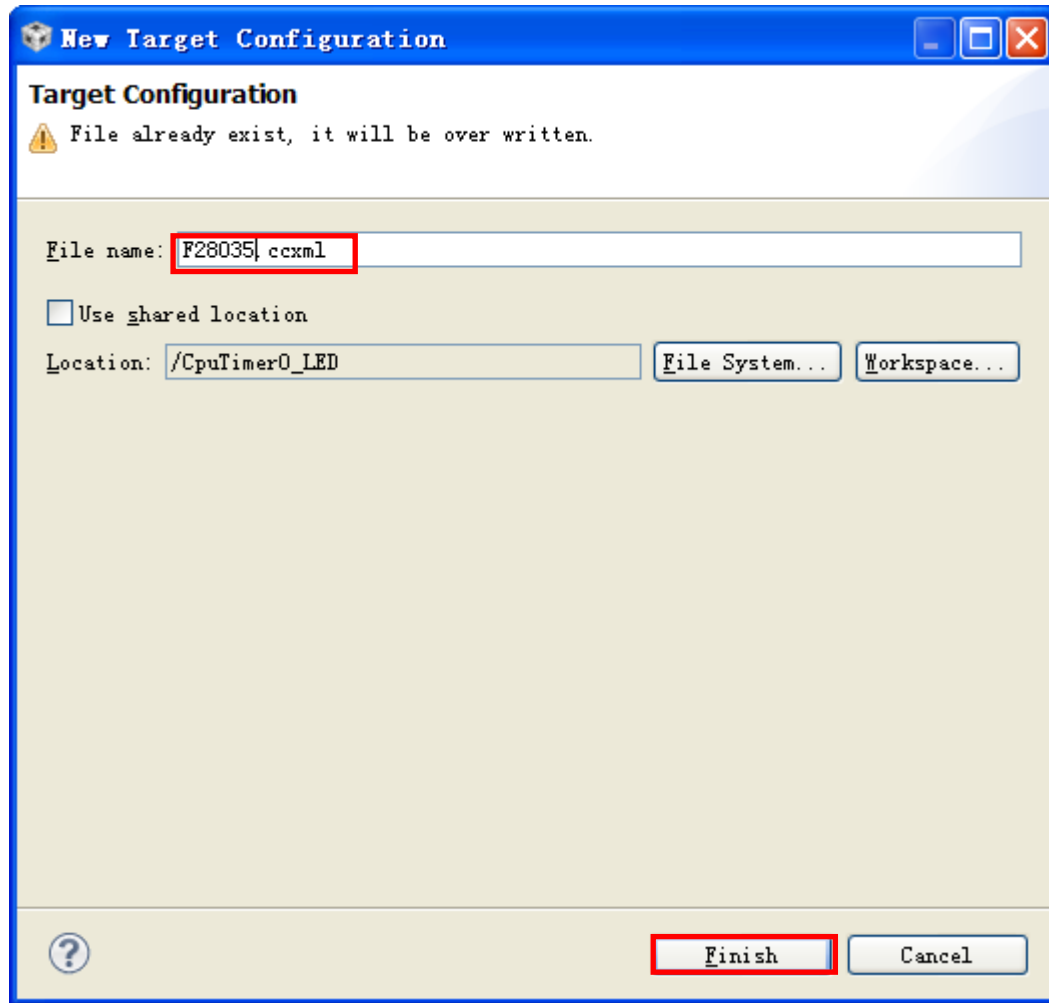


第十步：配置仿真器

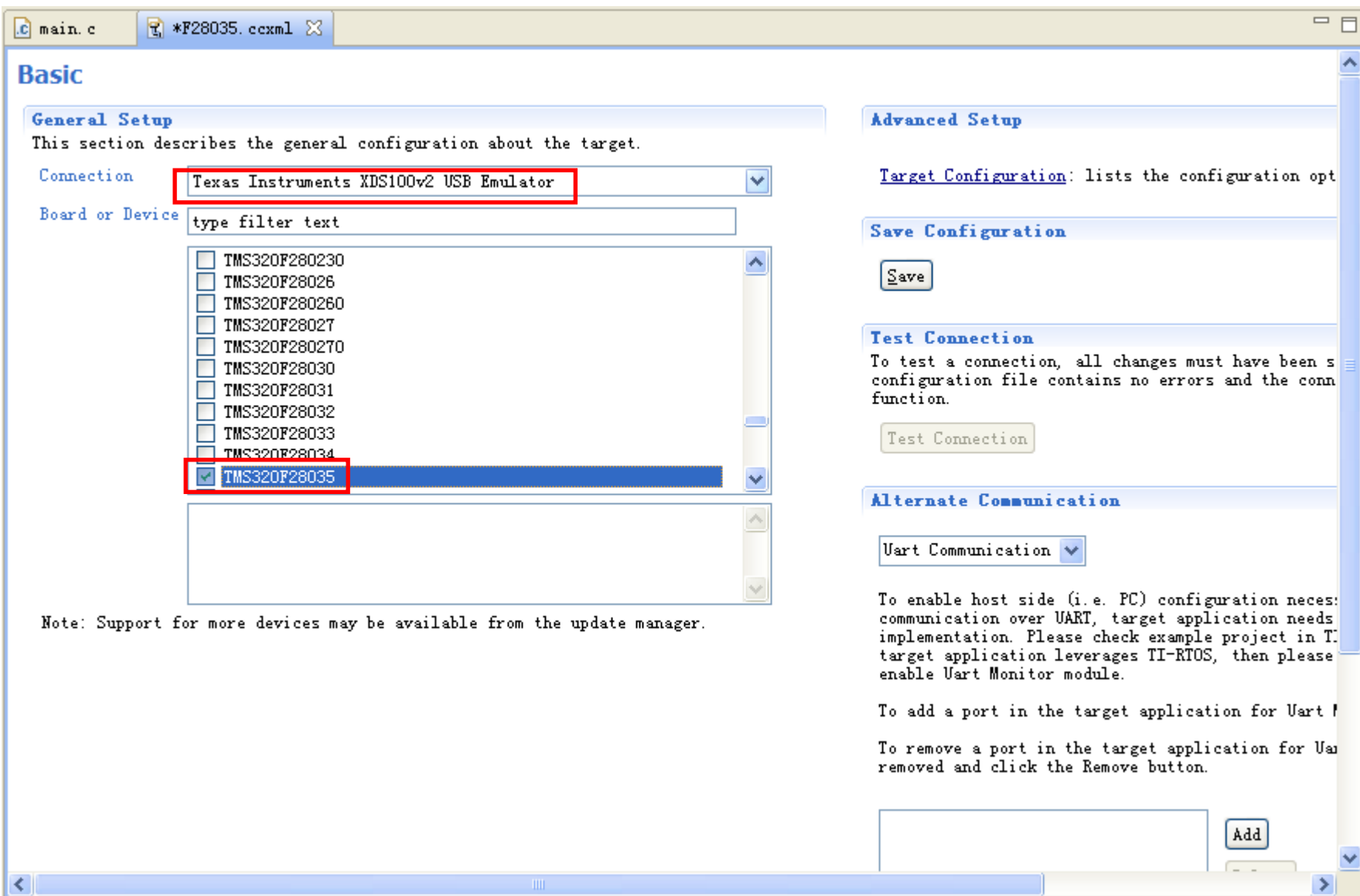
1、File→New->TargetConfigurationFile，如下图：






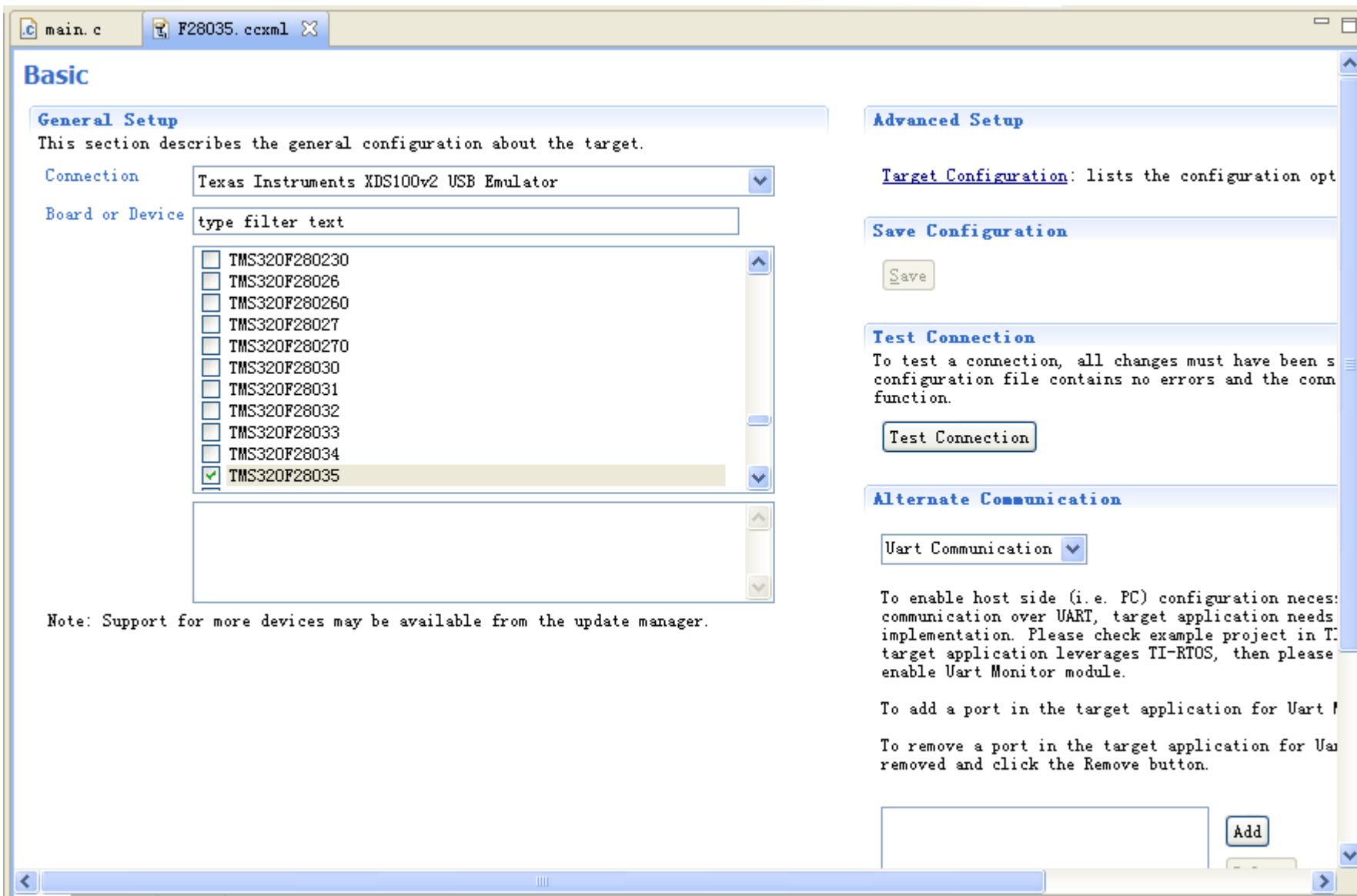
2、鼠标左键点击  Target Configuration File , 出现如下对话框, 建议把 File name 选项修改为自己仿真芯片的型号, 如下图:



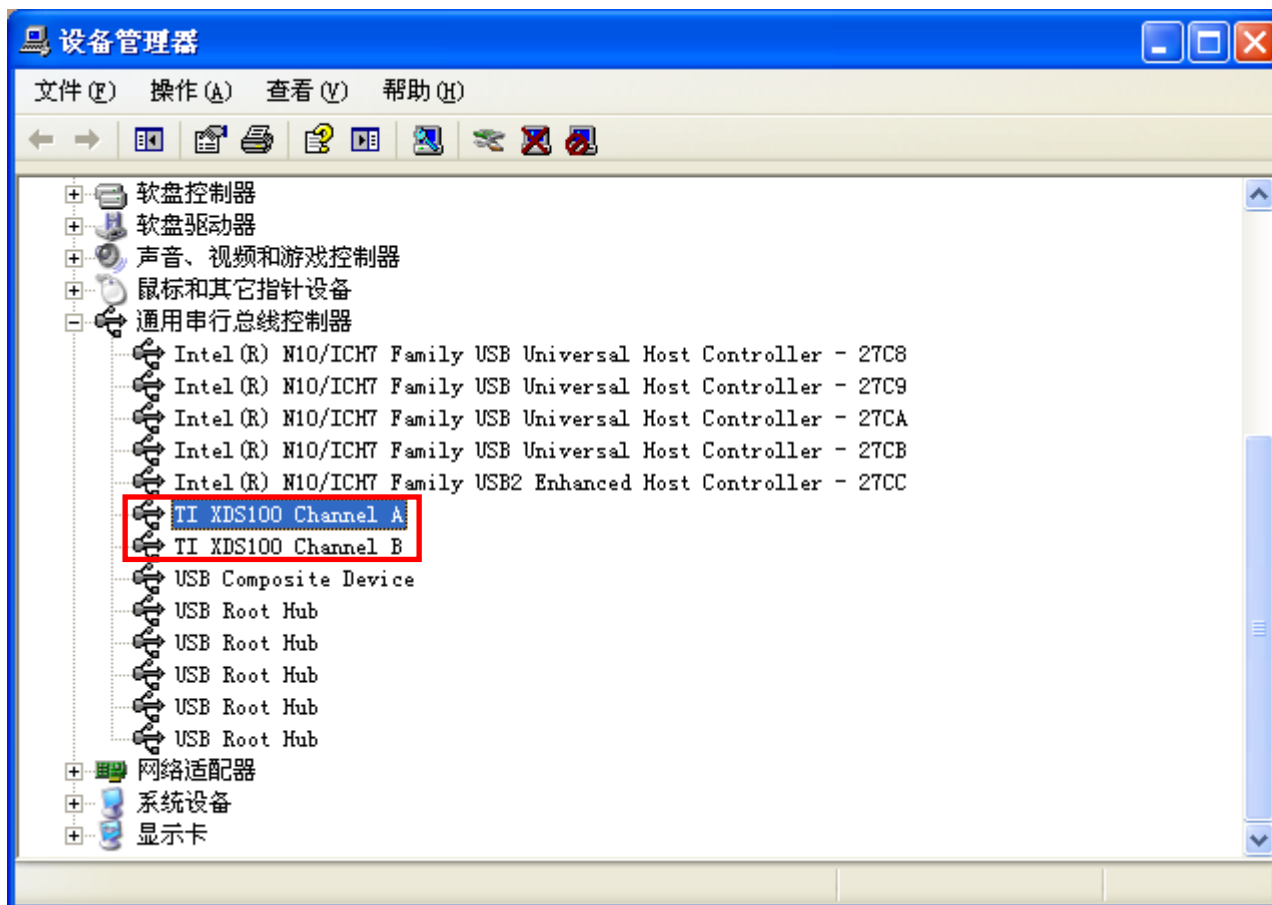
3、点击 **Finish** 出现如下对话框，修改选项后如下图：



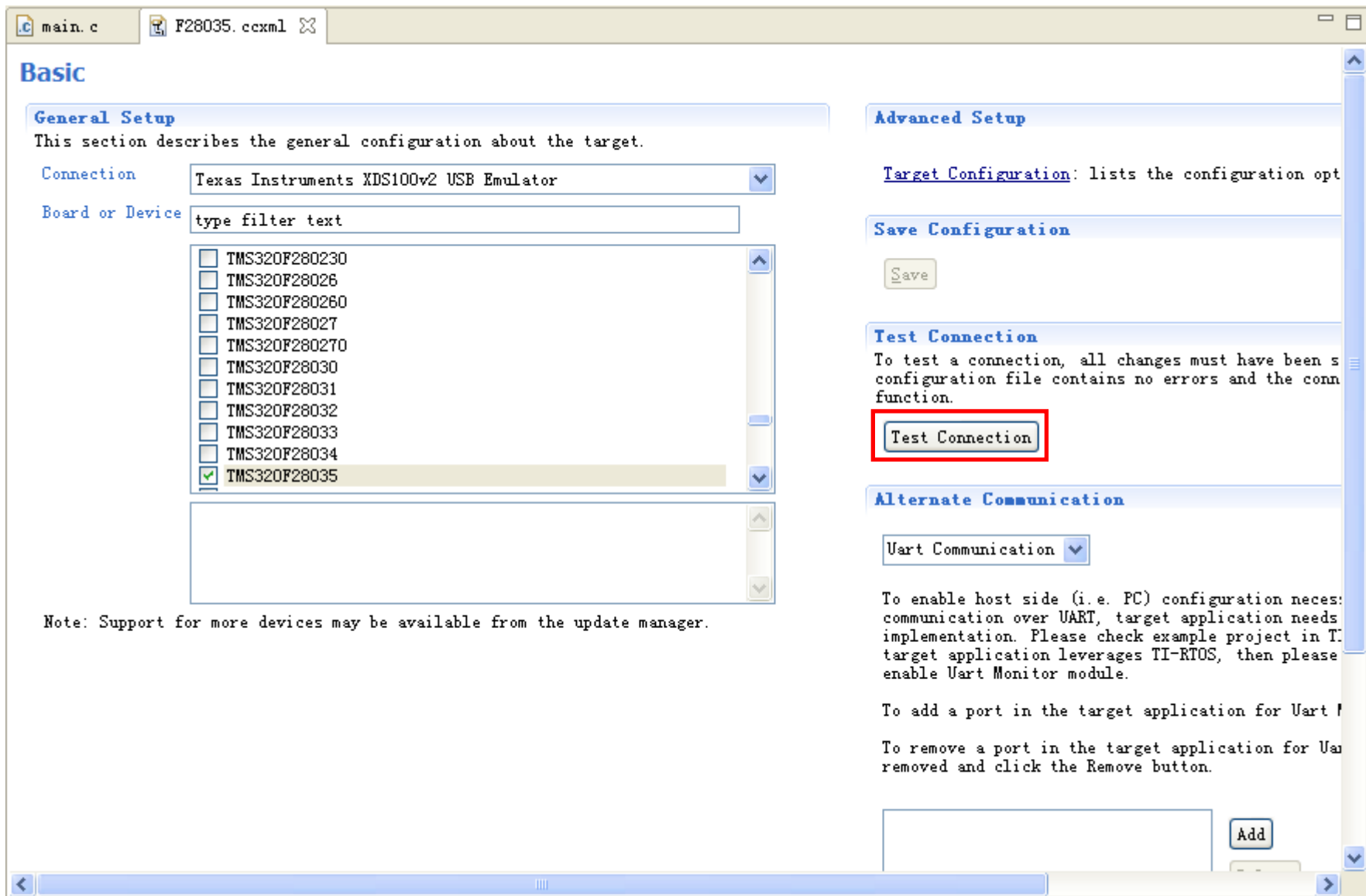
4、然后点击右边的  按钮，点击后会由高亮  变为灰色  如下图：



5、将仿真器 xds100v2 连接电脑 USB，连接成功后，会在设备管理器有 TI XDS100 指示，如下图：



6、将仿真器 xds100v2 与 TMS320F28035 开发板连接好，并通电，然后点击右边的 **Test Connection** 按钮，如第一张图。点击 **Test Connection** 后，会出现连接目标板信息，在信息最后当有 “The JTAG DR Integrity scan-test has succeeded” 指示时，请仿真器成功连接开发板，如第二张图。最后关闭 “Test Connection” 和 “F28035.ccxml” 对话框。



Test Connection

This test will use blocks of 512 32-bit words.
This test will be applied just once.

Do a test using 0xFFFFFFFF.

Scan tests: 1, skipped: 0, failed: 0

Do a test using 0x00000000.

Scan tests: 2, skipped: 0, failed: 0

Do a test using 0xFE03E0E2.

Scan tests: 3, skipped: 0, failed: 0

Do a test using 0x01FC1F1D.

Scan tests: 4, skipped: 0, failed: 0

Do a test using 0x5533CCAA.

Scan tests: 5, skipped: 0, failed: 0

Do a test using 0xAACC3355.

Scan tests: 6, skipped: 0, failed: 0

All of the values were scanned correctly.

The JTAG DR Integrity scan-test has succeeded.

[End]

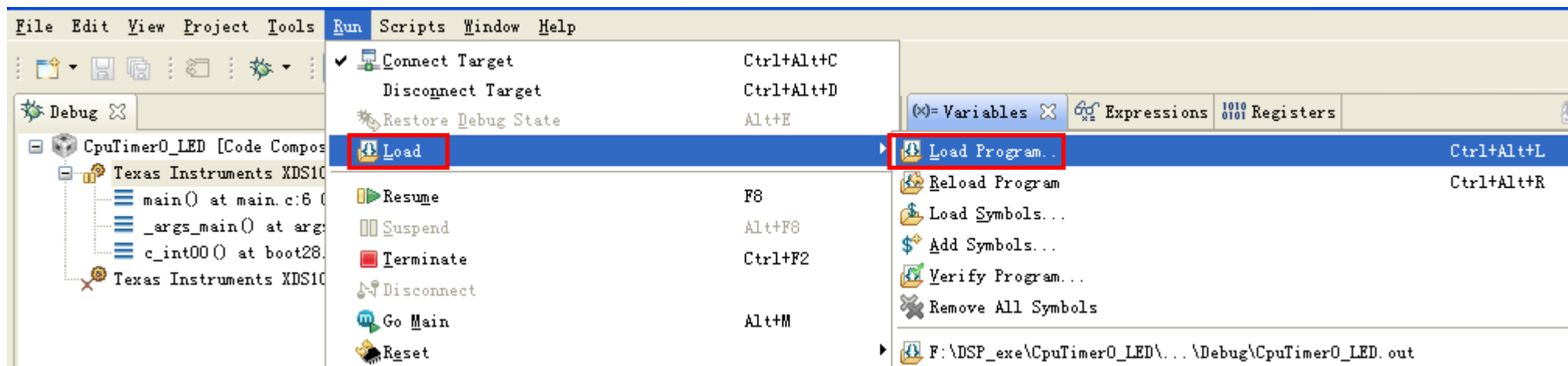
第十一步：煤录 RAM 并调试

1、Run→Debug，如下图：

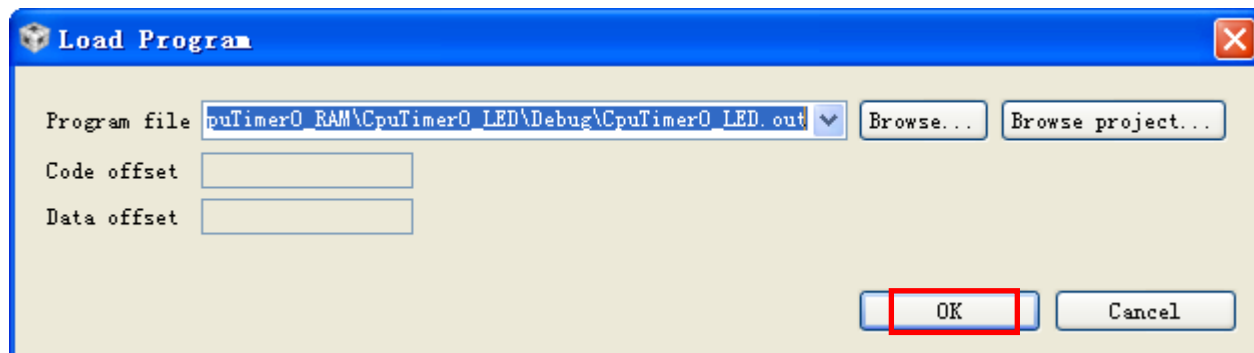


 Debug

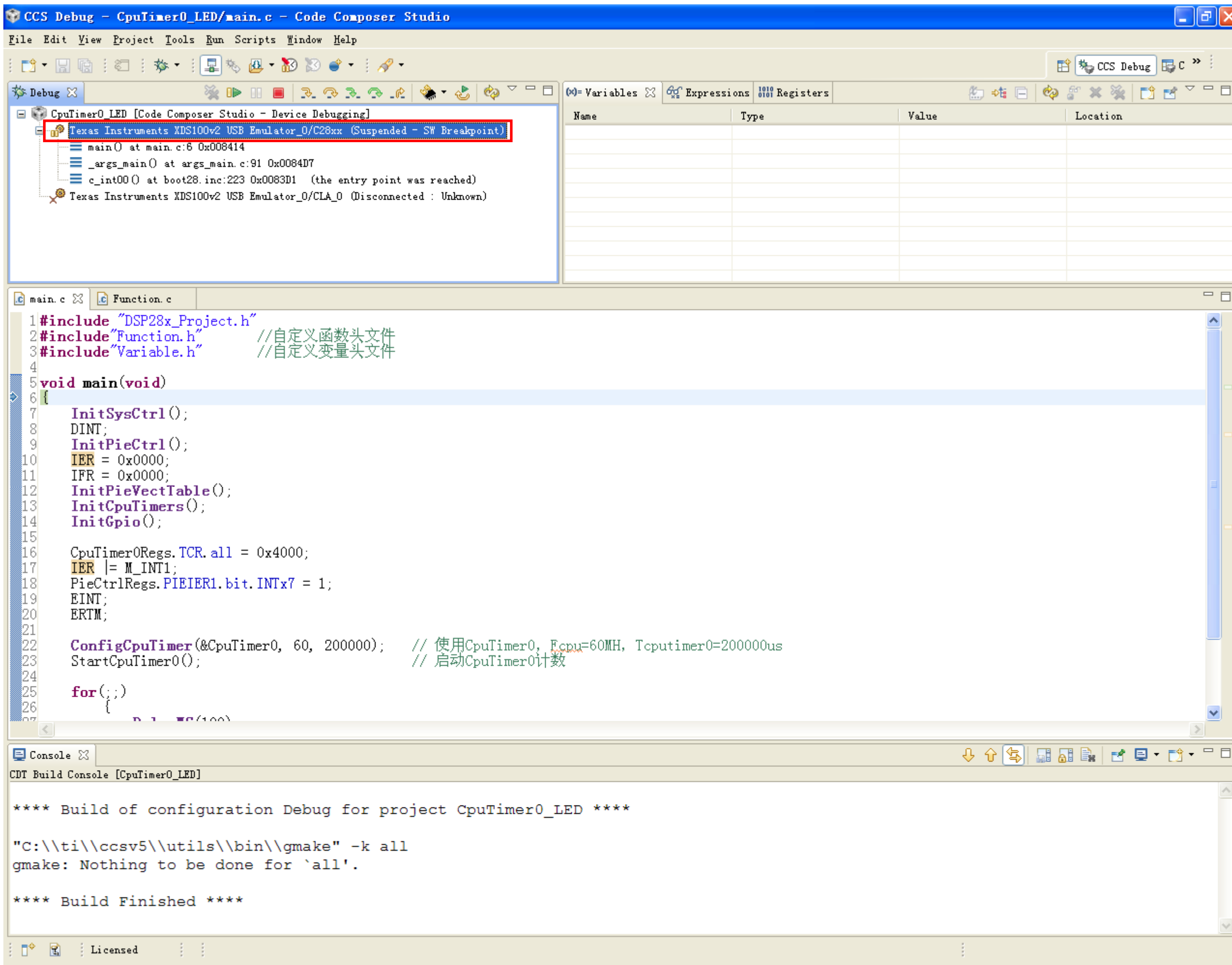
3、下载程序到 RAM（因为在 **CpuTimer0_LED [Active - Debug]** 中配置的是 **28035_RAM_lnk.cmd** 文件），如下图所示：



4、点击 **Load Program...** 后，出现如下对话框：



5、然后点击 ，返回如下界面，然后鼠标点击  **Texas Instruments XDS100v2 USB Emulator_0/C28xx (Suspended - SW Breakpoint)**，使其处于选中状态，如下图：





The screenshot displays the Code Composer Studio (CCS) interface for debugging the CpuTimer0_LED project. The main window shows the source code for main.c, which includes headers for DSP28x_Project.h, Function.h, and Variable.h. The main function initializes system control, PIE controller, CPU timers, and GPIO. It configures CpuTimer0 with a 60MHz frequency and a 200000us period, and starts the timer. The console window shows the build output, indicating that the build was successful.

CCS Debug - CpuTimer0_LED/main.c - Code Composer Studio

File Edit View Project Tools Run Scripts Window Help

Debug X

CpuTimer0_LED [Code Composer Studio - Device Debugging]

-  **Texas Instruments XDS100v2 USB Emulator_0/C28xx (Suspended - SW Breakpoint)**
- main() at main.c:6 0x008414
- _args_main() at args_main.c:91 0x0084D7
- c_int00() at boot28.inc:223 0x0083D1 (the entry point was reached)
-  Texas Instruments XDS100v2 USB Emulator_0/CLA_0 (Disconnected : Unknown)

Variables Expressions Registers

Name	Type	Value	Location

main.c Function.c

```
1 #include "DSP28x_Project.h"
2 #include "Function.h" //自定义函数头文件
3 #include "Variable.h" //自定义变量头文件
4
5 void main(void)
6 {
7     InitSysCtrl();
8     DINT;
9     InitPieCtrl();
10    IER = 0x0000;
11    IFR = 0x0000;
12    InitPieVectTable();
13    InitCpuTimers();
14    InitGpio();
15
16    CpuTimer0Regs.TCR.all = 0x4000;
17    IER |= M_INT1;
18    PieCtrlRegs.PIEIER1.bit.INTx7 = 1;
19    EINT;
20    ERTM;
21
22    ConfigCpuTimer(&CpuTimer0, 60, 200000); // 使用CpuTimer0, Fcpu=60MH, Tcputimer0=200000us
23    StartCpuTimer0(); // 启动CpuTimer0计数
24
25    for(;;)
26    {
27        // ...
28    }
29 }
```

Console X

CDT Build Console [CpuTimer0_LED]

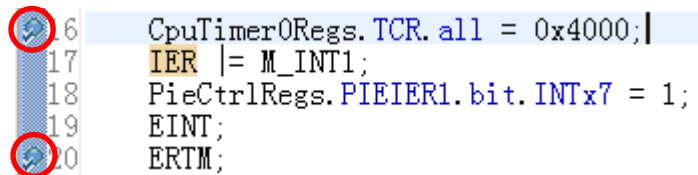
```
**** Build of configuration Debug for project CpuTimer0_LED ****

"C:\ti\ccsv5\utils\bin\gmake" -k all
gmake: Nothing to be done for `all'.

**** Build Finished ****
```

Licensed

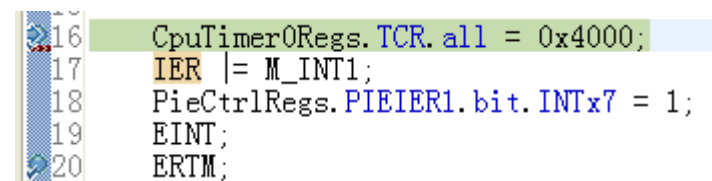
6、在自己需要观察的位置设置断点：鼠标指向行号的左侧，然后双击鼠标，即可设置一个断点，如下图示




```
16 CpuTimer0Regs.TCR.all = 0x4000;
17 IER |= M_INT1;
18 PieCtrlRegs.PIEIER1.bit.INTx7 = 1;
19 EINT;
20 ERTM;
```

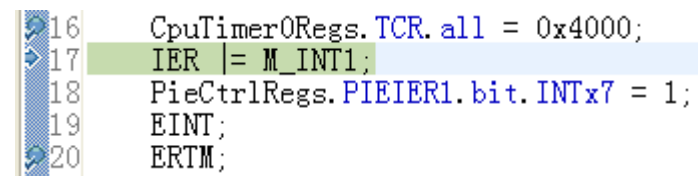
鼠标指向该位置，然后双击左键

7、点击 ，这时调试全速运行，使其执行到断点，如下图：



```
16 CpuTimer0Regs.TCR.all = 0x4000;
17 IER |= M_INT1;
18 PieCtrlRegs.PIEIER1.bit.INTx7 = 1;
19 EINT;
20 ERTM;
```

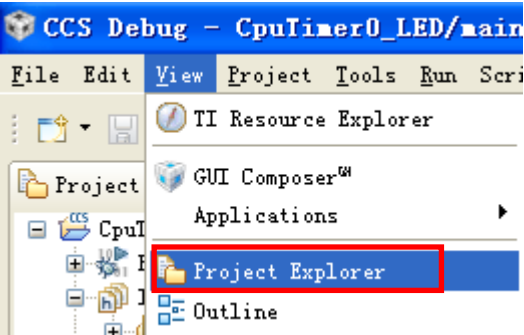
8、点击 （或按 F5）单步运行，如下图：





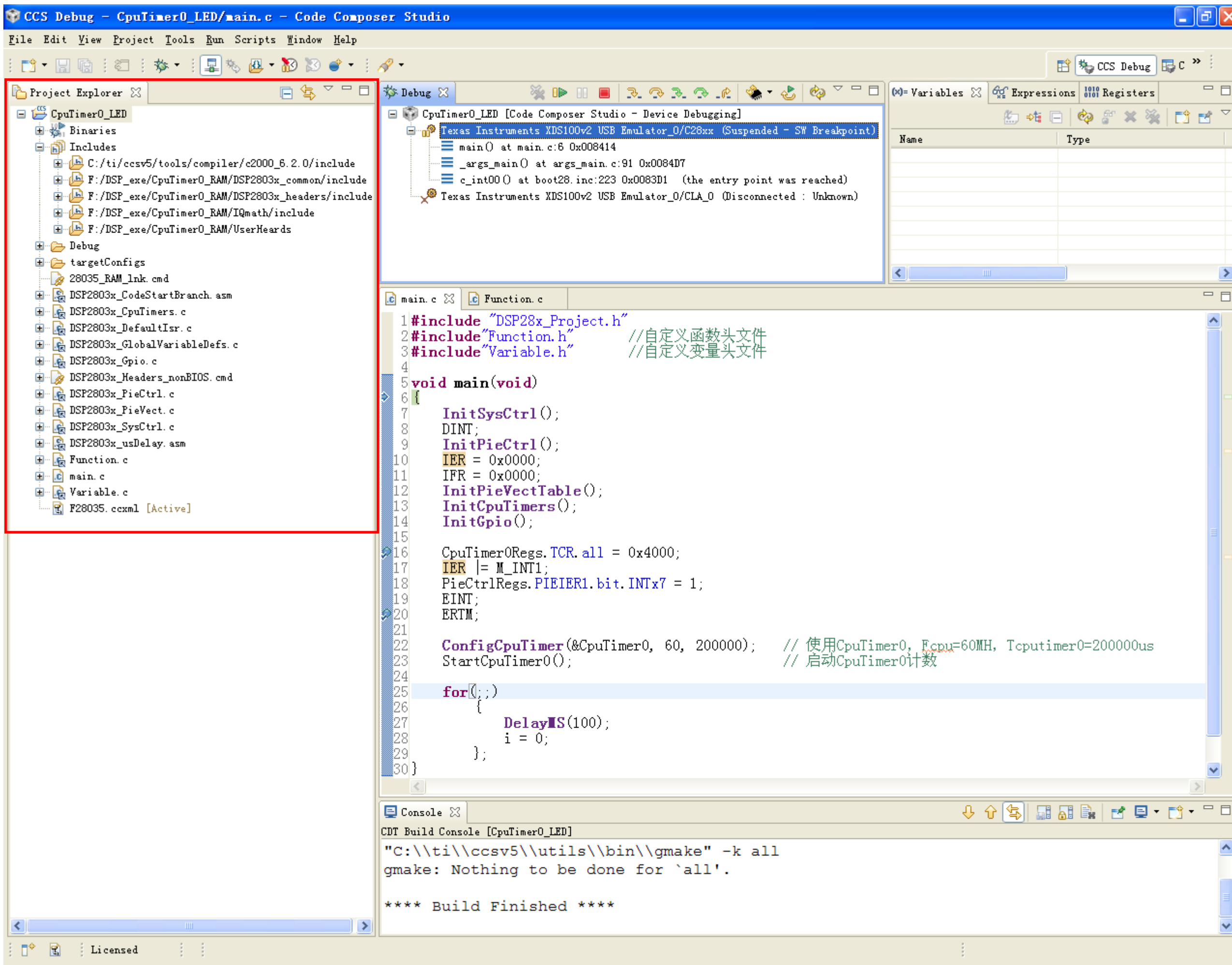
```
16 CpuTimer0Regs.TCR.all = 0x4000;
17 IER |= M_INT1;
18 PieCtrlRegs.PIEIER1.bit.INTx7 = 1;
19 EINT;
20 ERTM;
```

第十二步：在 RAM 中调试完成后，则就要烧录到 flash 中进行实际环境运行了。

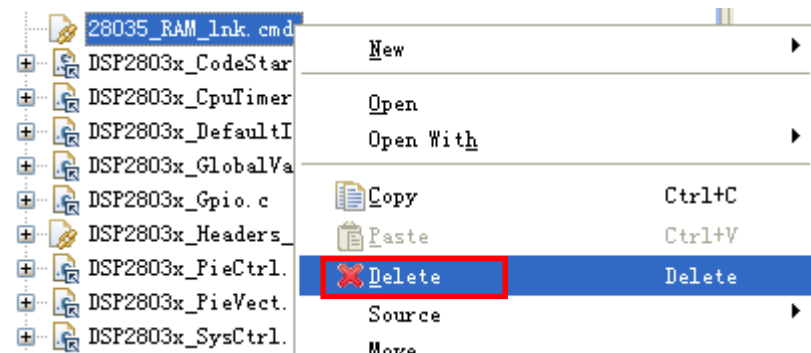
1、View→Project Explorer，如下图：


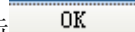


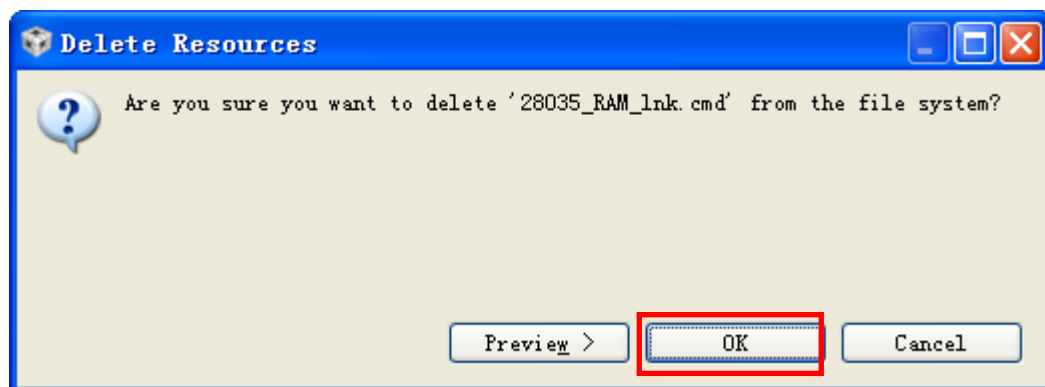
2、点击  Project Explorer 后，显示  Project Explorer 窗口，如下图：



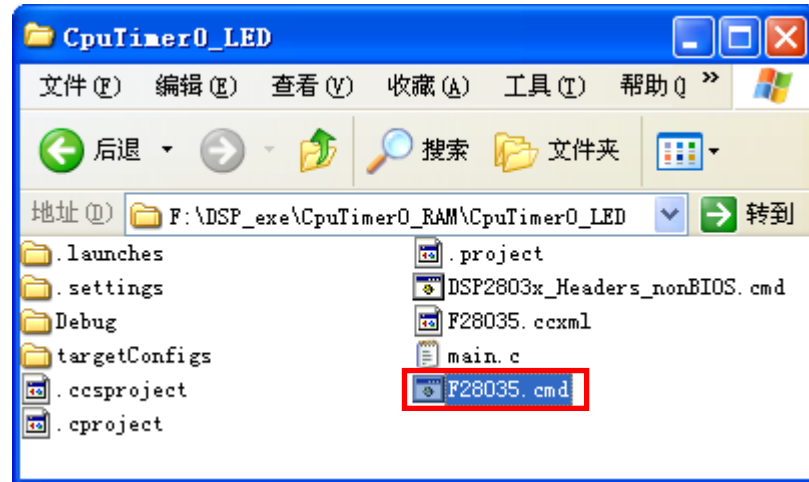
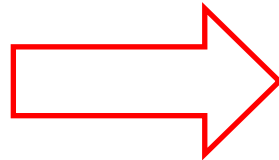
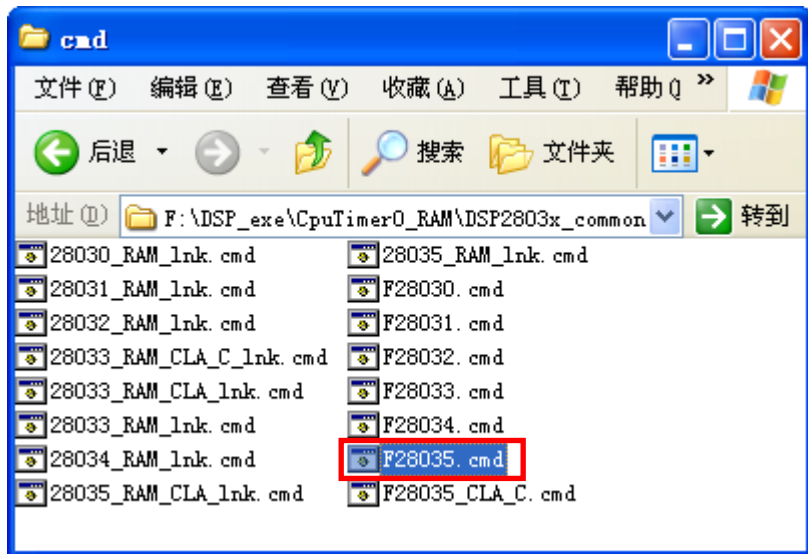
3、删除 28035_RAM_lnk.cmd 文件，如下图：



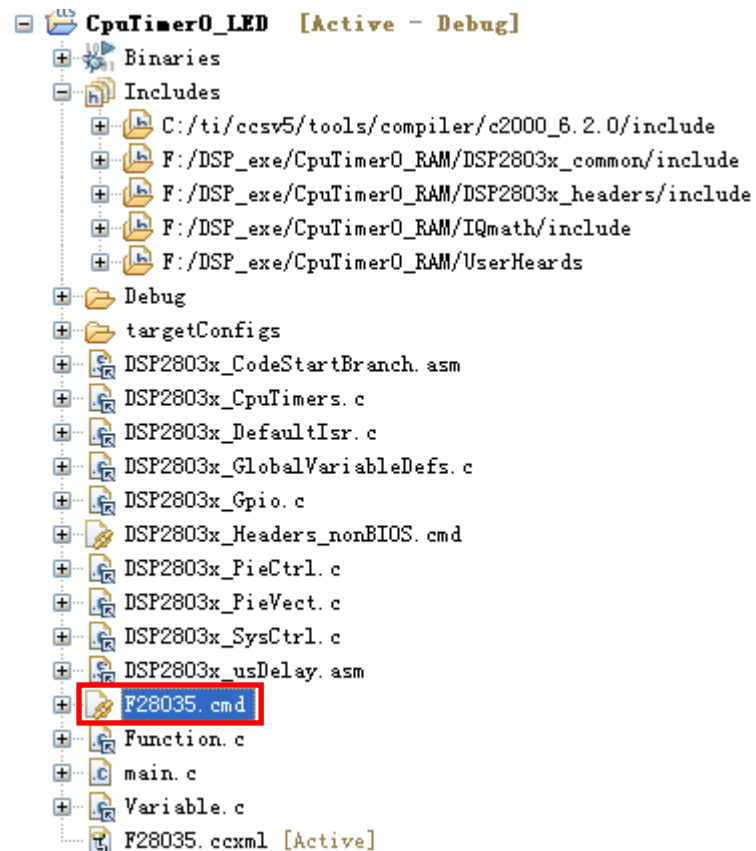
4、点击  后，出现如下对话框，然后点击  即可。



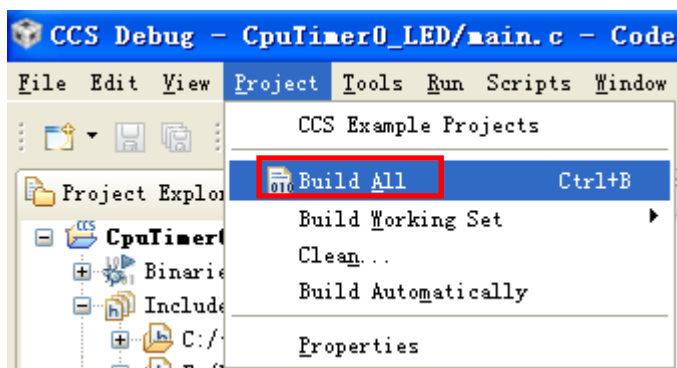
5、把 F:\DSP_exe\CpuTimer0_RAM\DSP2803x_common\cmd 目录下的 F28035.cmd 文件拷贝到 F:\DSP_exe\CpuTimer0_RAM\CpuTimer0_LED 目录下，如下图所示：



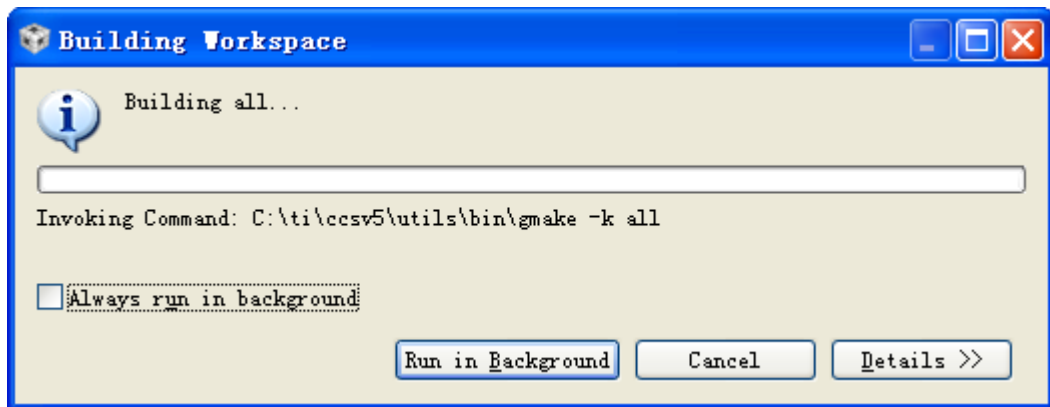
6、拷贝完成后， **CpuTimer0_LED [Active - Debug]** 下框自动出现了添加的“F28035.cmd”文件，如下图：



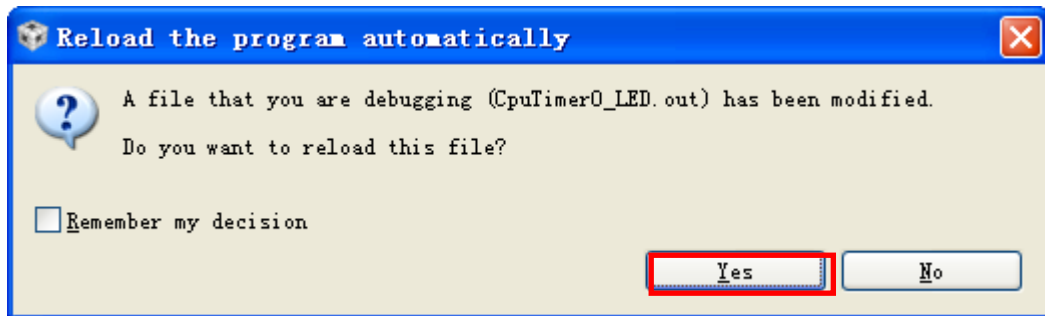
7、重新编译所有源文件，Project→Build All，如下图示：



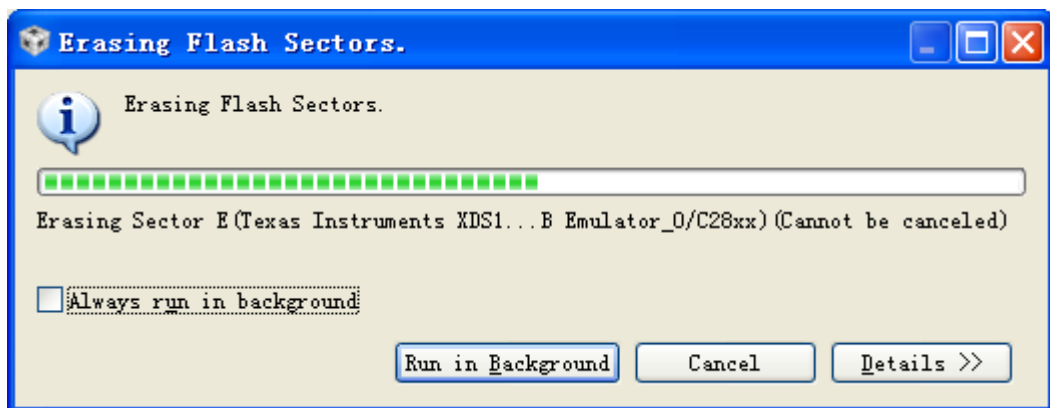
8、点击  Build All，出现如下对话框：



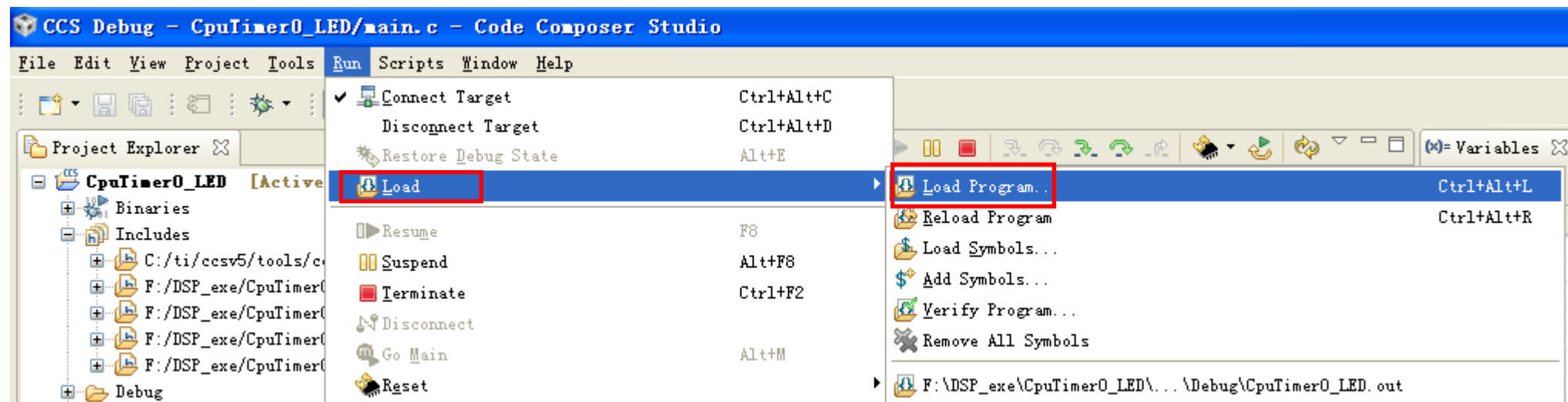
9、完成编译后，可能会出现下对话框，意思是否烧录 flash。



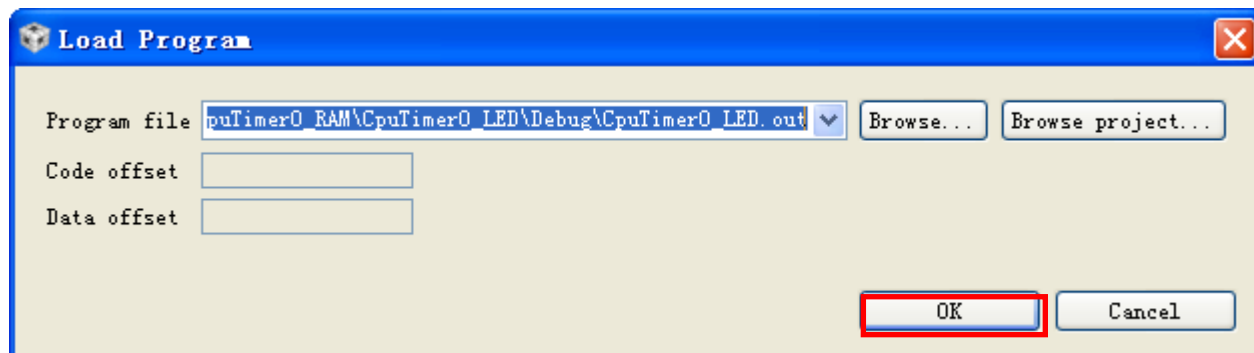
10、点击  Yes 后出现烧录 flash 对话框，如下图：

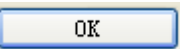


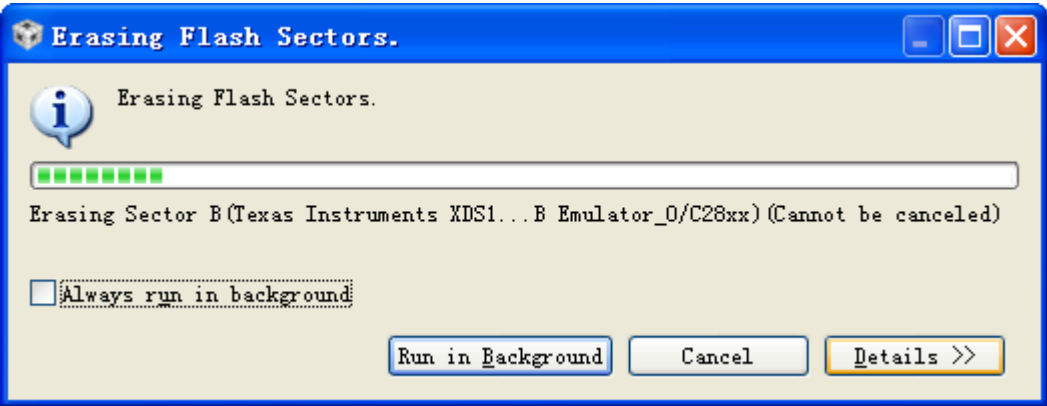
11、若没有出现上面第 9 步的指示，则通过如下图示烧录 flash，Run→Load→Load Program...



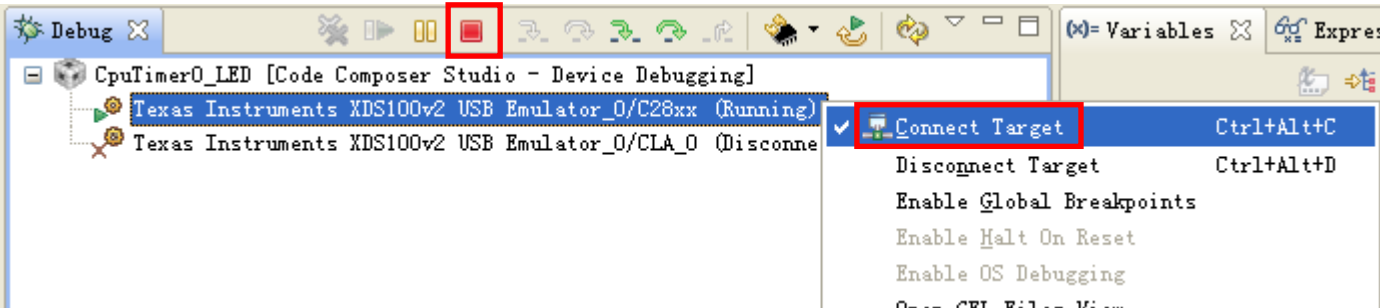
11、点击  Load Program... 出现如下对话框：




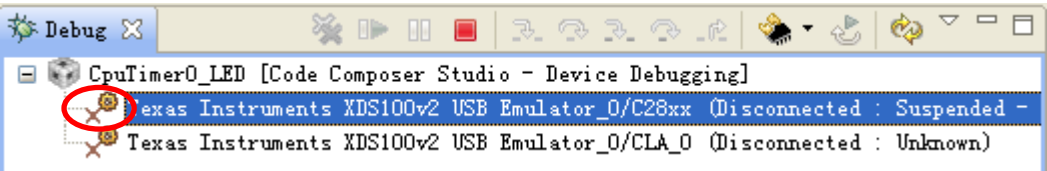
12、点击 ，出现如下对话框：



13、烧录完成后，鼠标指向 **Texas Instruments XDS100v2 USB Emulator_0/C28xx (Running)**，然后点击鼠标右键选择  **Connect Target**，如下图所示：

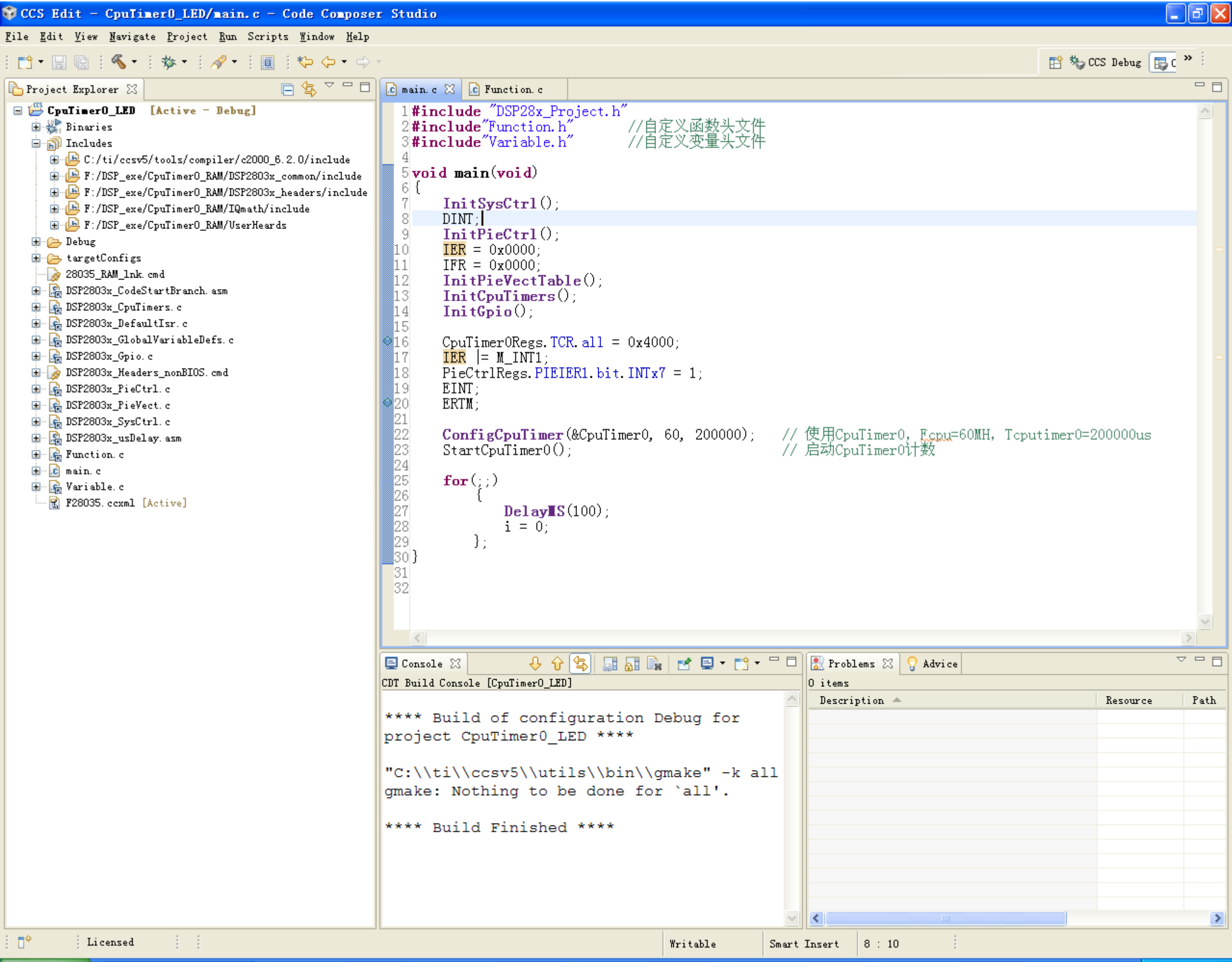


14、点击  **Connect Target** 后，即可断开仿真器 xds100v2 与开发板的连接（若再次点击，则仿真器 xds100v2 连接开发板），如下图所示：

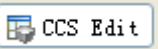


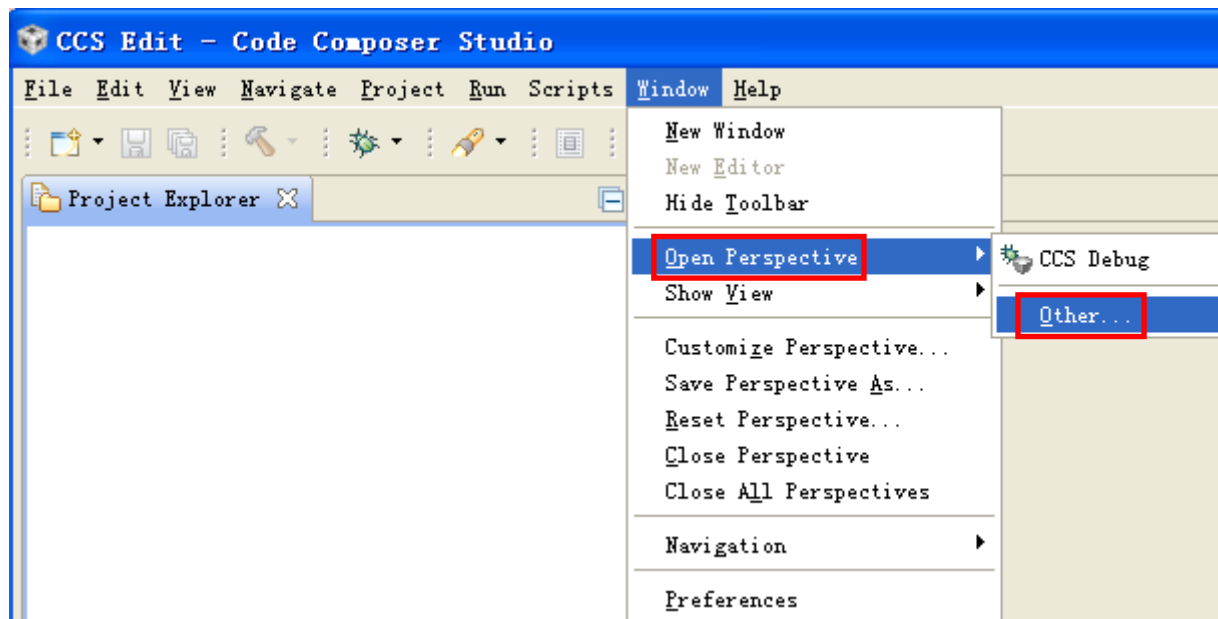
15、关掉开发板的供电（不用拨出 xds100v2 与开发板的 JTAG 接口），然后重新上电，这时开发板处 flash 运行状态了。

第十三步、点击退出调试界面，然后返回正常界面，如下图所示：

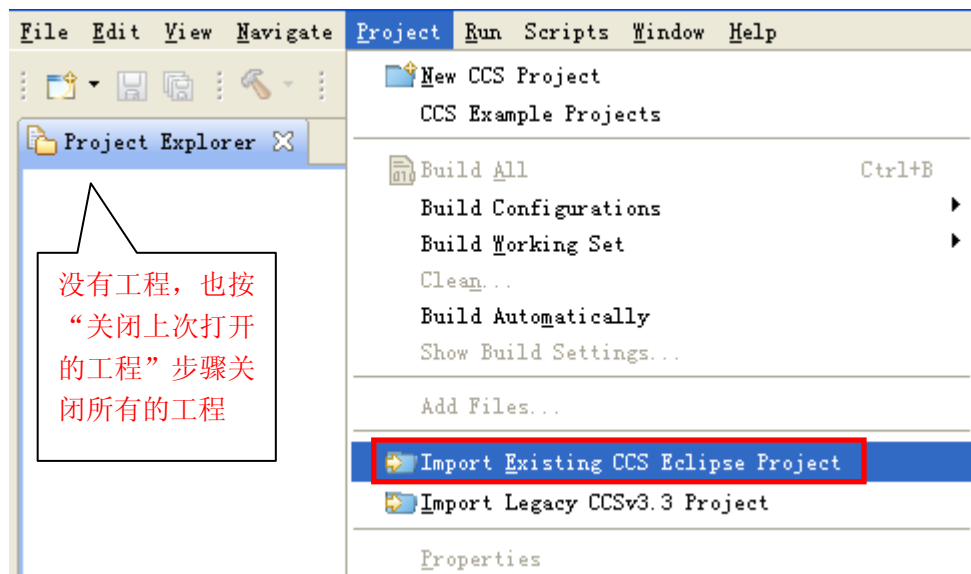


其它的操作：重新打开一个 CCS5.5 的工程：

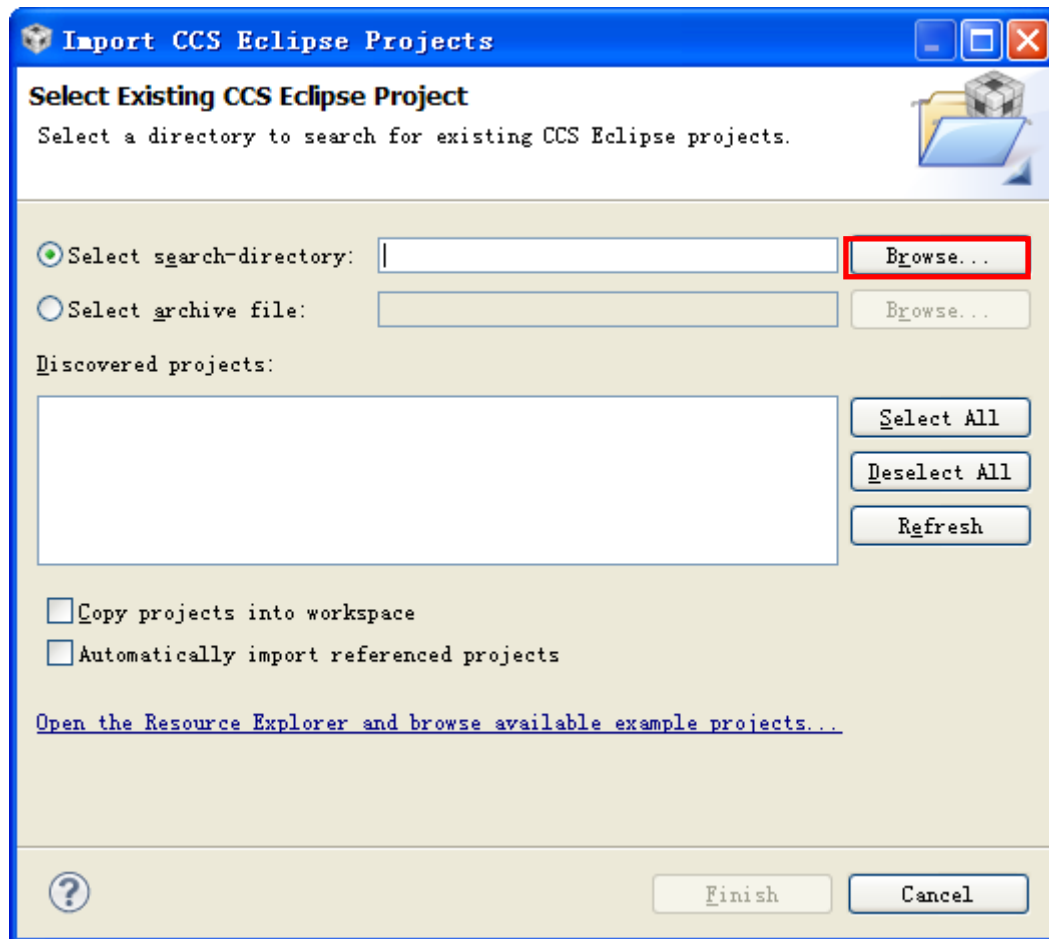
- 1、先设置工作窗口处于  CCS Edit （右上角位置）：Window→Open Perspective→Other...



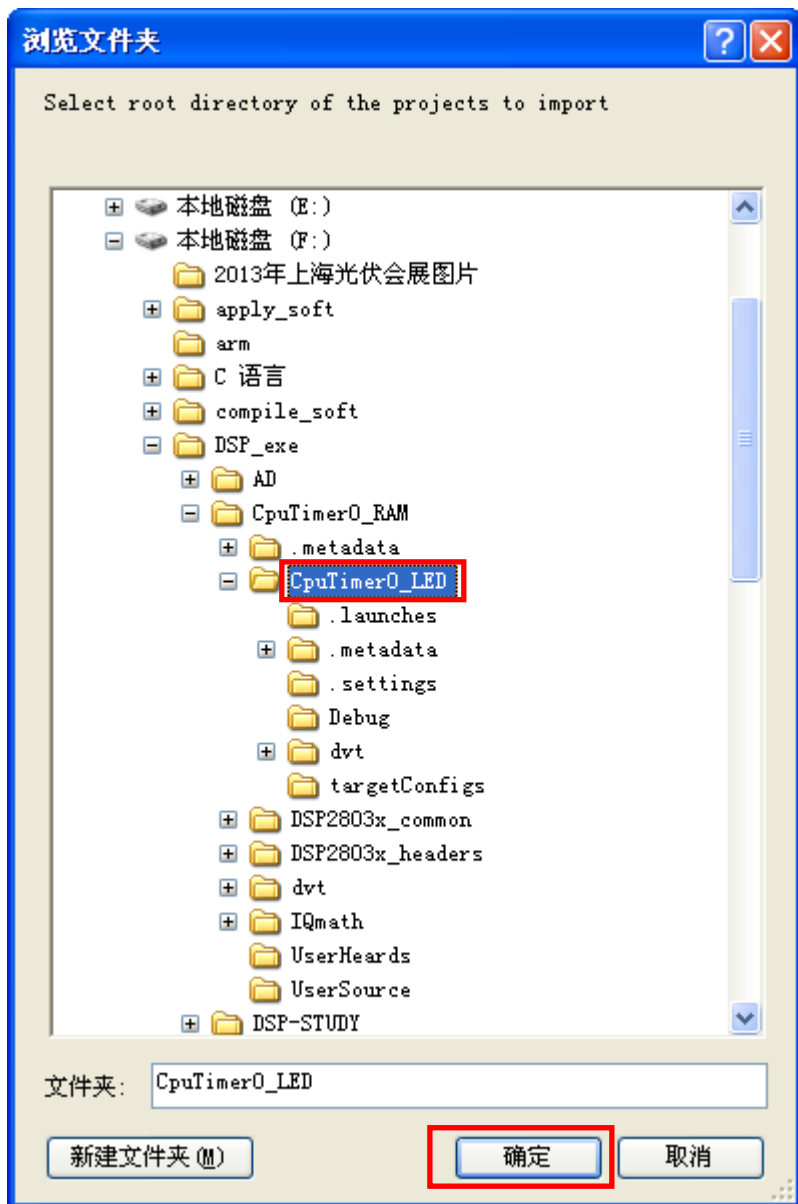
- 2、打开已有的 CCS5.5 工程，Project→Import Existing CCS Eclipse Project，如下图：



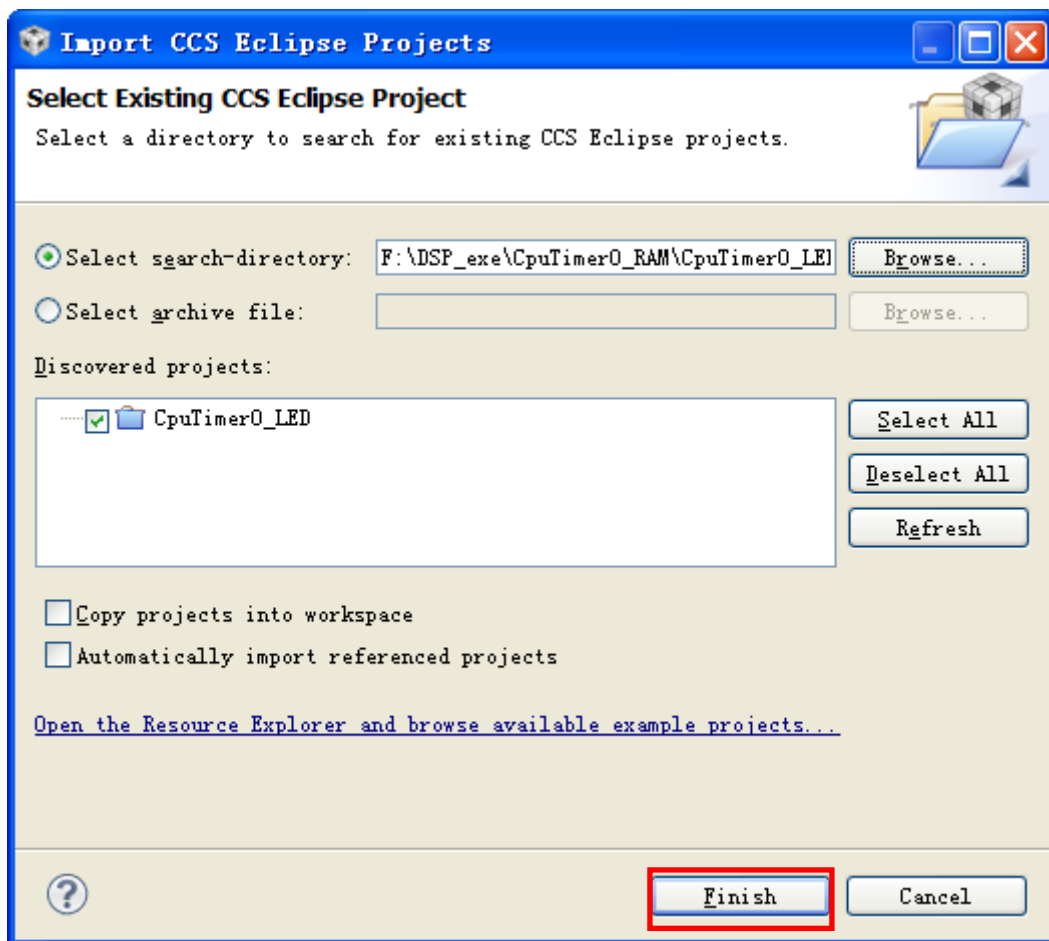
3、点击  Import Existing CCS Eclipse Project，出现如下对话框：



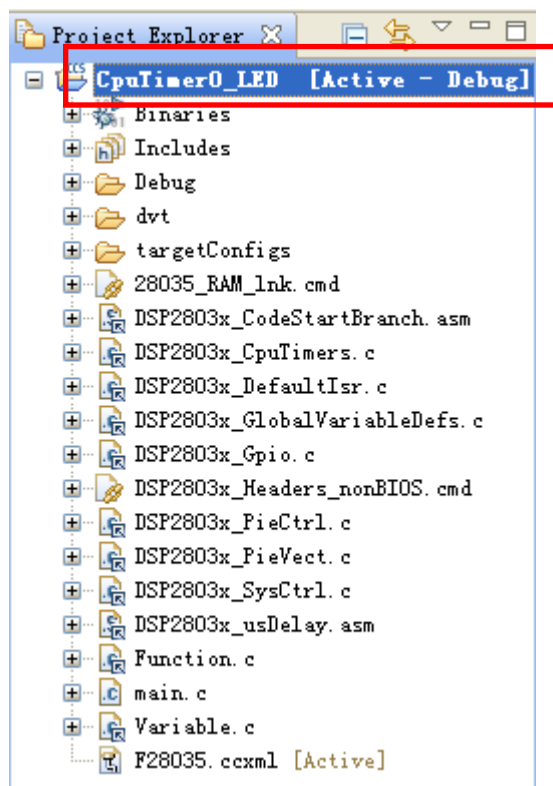
4、点击 **Browse...**，出现如下对话框，并选择要打开的 CCS5.5 工程位置，如下图：



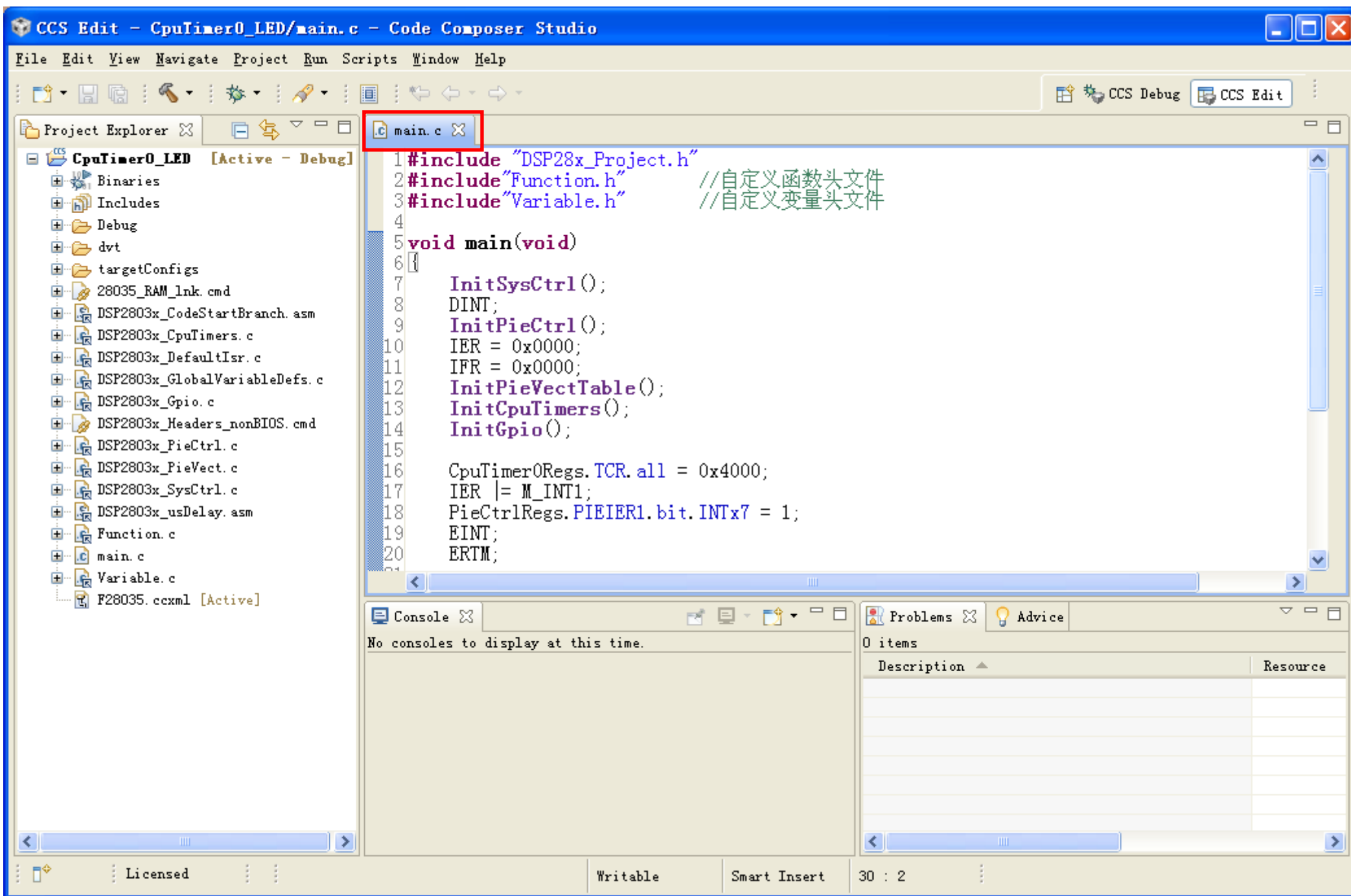
5、点击 ，返回如下界面：



6、点击 **Finish**，在 **Project Explorer** 窗口下出现添加的工程，如下界面：



7、双击需要修改的源文件，打开源文件后进行修改，如下图：



附件

1、main.c 文件程序（自己编写的程序）：

```
#include "DSP28x_Project.h"
#include "Function.h"      //自定义函数头文件
#include "Variable.h"      //自定义变量头文件

void main(void)
{
    InitSysCtrl();
    DINT;
    InitPieCtrl();
    IER = 0x0000;
    IFR = 0x0000;
    InitPieVectTable();
    InitCpuTimers();
    InitGpio();

    CpuTimer0Regs.TCR.all = 0x4000;
    IER |= M_INT1;
    PieCtrlRegs.PIEIER1.bit.INTx7 = 1;
    EINT;
    ERTM;

    ConfigCpuTimer(&CpuTimer0, 60, 200000); // 使用CpuTimer0, Fcpu=60MH, Tcputimer0=200000us
    StartCpuTimer0();                       // 启动CpuTimer0计数

    for(;;)
    {
        DelayMS(100);
        i = 0;
    };
}
```

2、DSP2803x_Gpio.c 文件程序（TI DSP2803x 标准文件）：

```
#include "DSP2803x_Device.h"
#include "DSP2803x_Examples.h"
void InitGpio(void)
{
    EALLOW;
    GpioCtrlRegs.GPAMUX1.bit.GPIO13 = 0;    // 配置GPIO13为数字I/O口
    GpioCtrlRegs.GPADIR.bit.GPIO13 = 1;      // 配置GPIO13为输出
    GpioCtrlRegs.GPAPUD.bit.GPIO13 = 0;      // 启用GPIO13引脚的内部上拉电阻
    EDIS;
}
```

3、DSP2803x_DefaultIsr.c 文件程序（TI DSP2803x 标准文件）:

```
#include "DSP2803x_Device.h"
#include "DSP2803x_Examples.h"

__interrupt void TINT0_ISR(void)      // CPU-Timer 0
{
    GpioDataRegs.GPATOGGLE.bit.GPIO13 = 1;
    CpuTimer0Regs.TCR.bit.TIF = 1;
    PieCtrlRegs.PIEACK.bit.ACK1 = 1;
}
```

4、Function.c程序（自己编写的，可在主函数屏蔽该文件，从而不需要在“Project Explorer”添加该文件）



```
#include "DSP2803x_Device.h"
```

```
void DelayUS(Uint16 USValue)    // Tcpu = 60MHz
```

[illegible]

```
    }  
void DelayMS(uint16 MSValue)    // Tcpu = 60MHz
```

```
{
    Uint16 OneMS;
    while (MSValue--){
        OneMS = 3330;
        while (OneMS--){
            asm(" nop ");
        }
    }
}
```


Variable.c程序（自己定义的变量，可在主函数屏蔽该文件，从而不需要在 Project Explorer  添加该文件）

```
#include"DSP2803x_Device.h"
Uint16 i;
```