创建一个新工程

创建一个新工程

众所周知，通常一栋房子是由砖、瓦、钢筋、水泥等材料构建起来的，在准备开工盖房子之前，先得把这些材料准备好。DSP的软件开发就像是在盖一座座的房子，只不过是在创建一个个的工程。本章将以让LED灯闪烁为实例来介绍如何创建一个新工程，从而了解一个工程通常是由哪些文件来构成的，并学习CCS6.x开发环境的一些常用操作。

控制原理分析

如图7-1所示，创建的新工程需要实现的功能是通过GPIO0～GPIO5引脚控制6个LED灯闪烁。

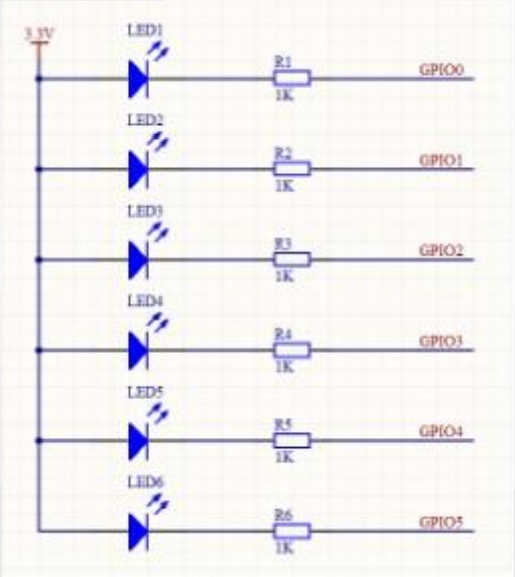

图7-1 GPIO引脚控制LED灯硬件原理图

控制原理分析

由于6个灯的控制原理都一样，所以就以GPIO0控制LED1闪烁为例来分析。F28335的GPIO引脚输出的低电平是0V，输出的高电平是3.3V，所以从图7-1可知，当GPIO0输出高电平时，LED1就会熄灭；当GPIO0输出低电平时，LED1就会被点亮。如果通过程序不断改变GPIO0引脚的输出电平，LED灯就可以实现闪烁了。

控制原理分析

从上一章的学习可知，通过向寄存器GPxSET的位写1，可使相应的GPIO引脚输出高电平；通过向寄存器GPxCLEAR的位写1，可使相应的GPIO引脚输出低电平。如下所示：

```
GpioDataRegs.GPASET.bit.GPIO0=1; //GPIO0输出高电平，LED1灭
GpioDataRegs.GPACLEAR.bit.GPIO0=1; //GPIO0输出低电平，LED1亮
```

创建工程

下面以TI公司的CCS6.x软件为开发环境，介绍如何来创建新工程，创建完工程后又如何编译、下载、运行程序。双击桌面上的CCS6.x图标打开CCS软件，界面如图7-2所示。

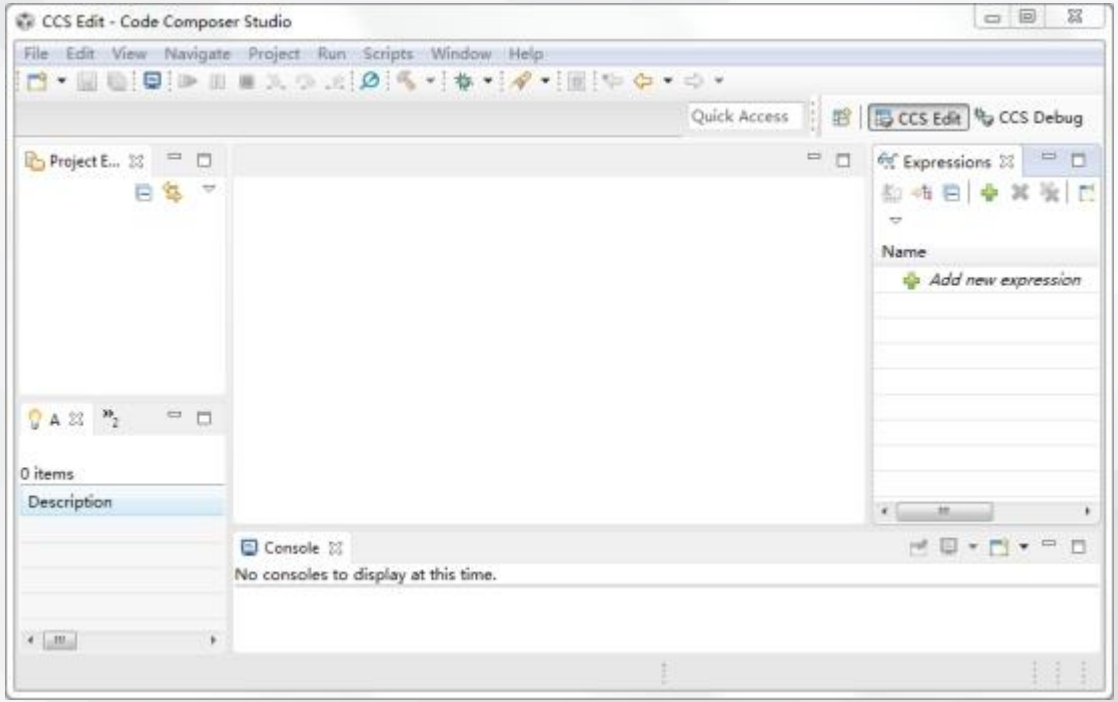

图7-2 CCS6.x软件

创建工程

如图7-3所示，选择ProjectNew CCS Project菜单项，以新建工程，CCS会弹出New CCS Project窗口，如图7-4所示。

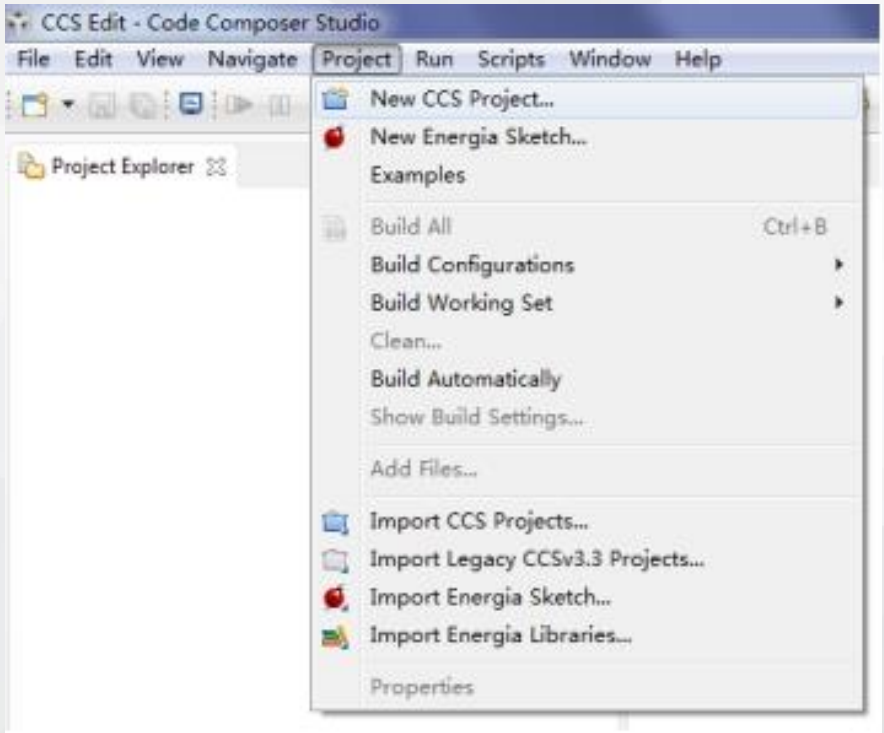

图7-3 新建工程1

创建工程

在New CCS Project窗口中，需要做一些配置。Target选项是选择需要开发的DSP芯片型号，这里选择TMS320F28335。Connection选项是选择手头实际使用的仿真器型号，这里选择Texas Instruments XDS2xx USB Debug Probe,就是使用的是XDS200系列的仿真器，实际使用哪款就选择哪款。接下来，在Project name文本框里输入新建工程的名字，这里输入led，意思是创建一个名为led的工程。最后需要选择新建工程的模板，这里选择Empty Project选项，意思是创建一个空的工程模板。最后单击Finish按钮，CCS便在指定的workspace文件夹下创建了一个led文件夹，工程便在此文件夹内。

创建工程

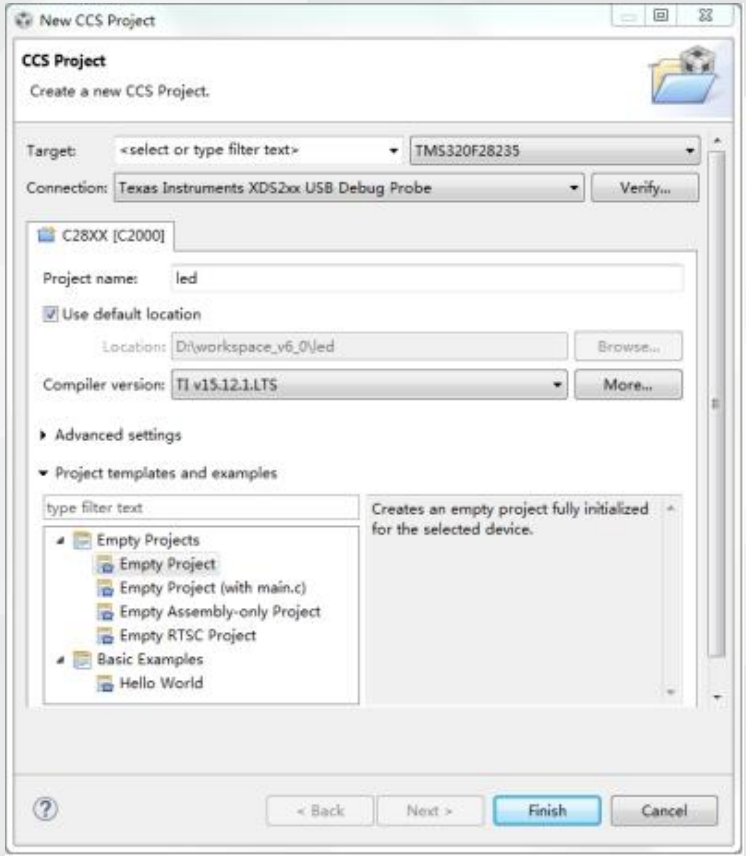

图7-4 新建工程2

（此处可右键选择“放大”功能查看图像）

创建工程

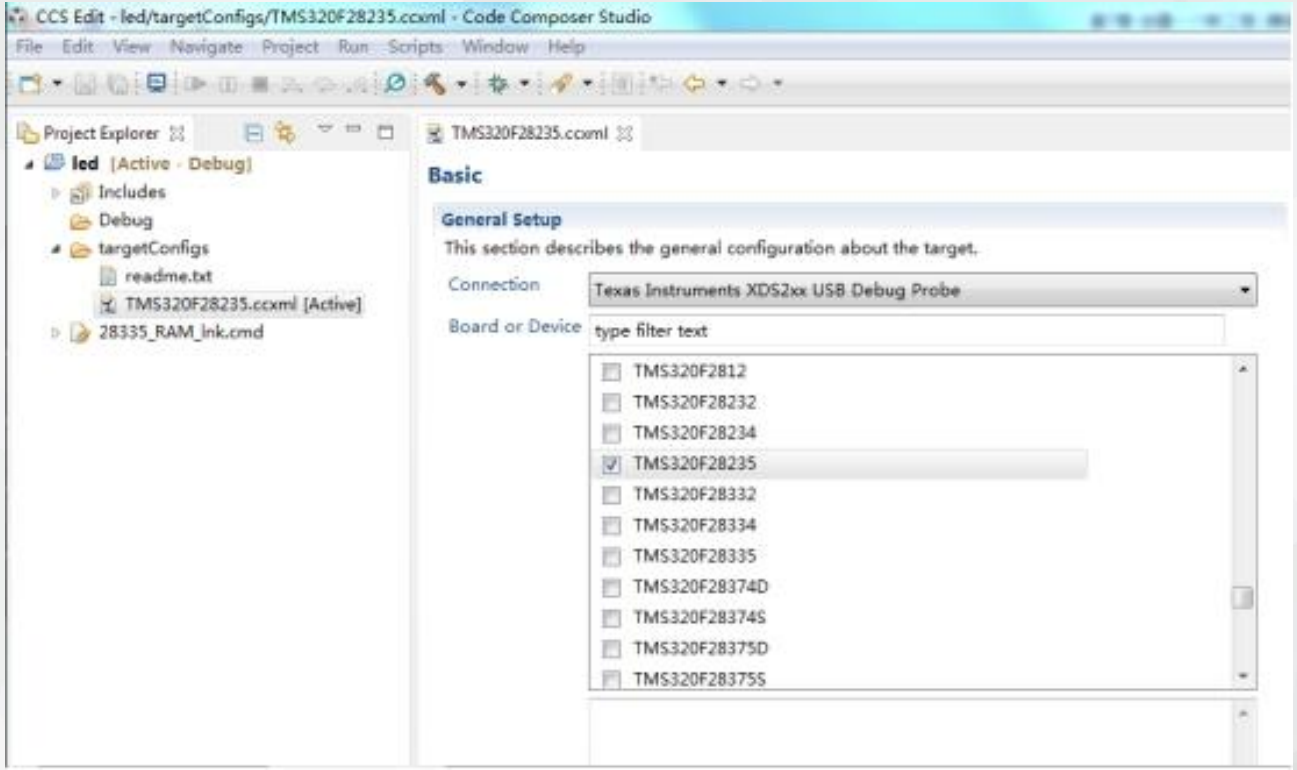

图7-5 创建新工程3

（此处可右键选择“放大”功能查看图像）

创建工程

如图7-5所示，led工程被添加进了CCS的Project Explorer窗口内，这就是新建立的工程模板。从图7-5可以看到，现在led工程内有Includes文件夹、Debug文件夹、TMS320F28335.ccxml和28335_RAM_link.cmd。这里Includes文件夹下面的文件是C语言环境需要用到的一些头文件，比如常用的math.h，string.h等，如图7-6所示。Debug文件夹现在是空的，在工程被成功地编译链接后，所产生的中间文件和可执行文件都会放在Debug文件夹内。TMS320F28335.ccxml是目标链接文件，这里指定了DSP的型号和所使用的仿真器，如果工程没有这个文件，那CCS没有办法和DSP建立连接，也就没有办法下载调试程序了，也可以通过New®Target Configuration File为工程创建一个目标链接文件。

创建工程

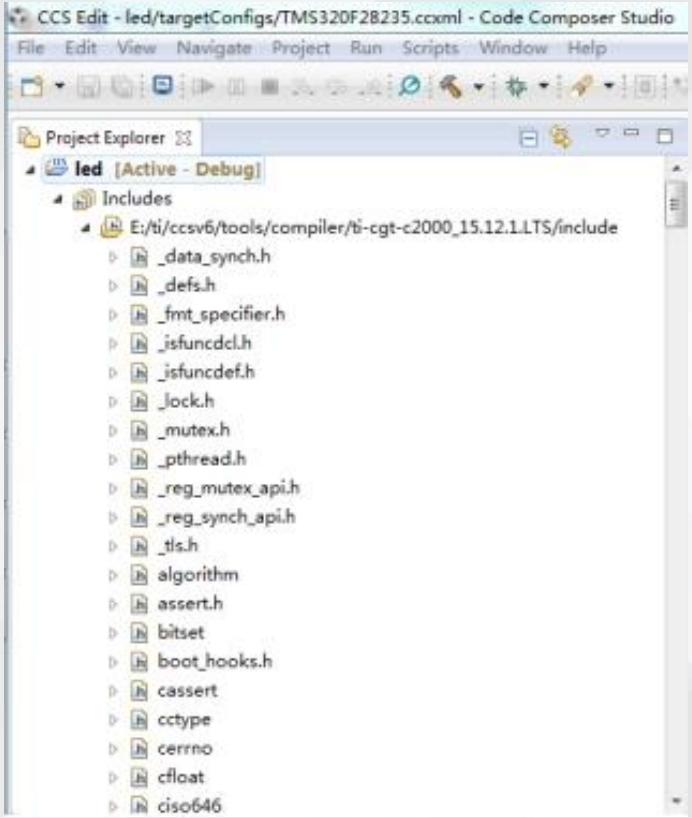

图7-6 系统includes文件夹内的文件

28335_RAM_link.cmd文件定义了用户程序和数据的存储空间及其分配情况，通常不需要做改动了，文件内充分利用了F28335的RAM空间，不过如果当实际工程的存储情况和CMD文件内的分配不符合时，就需要修改CMD文件了。这里只是分配用户数据和程序的，还需要一个CMD文件，用来分配F28335寄存器的空间，这个在下一步操作，向工程添加需要的文件时进行添加。

（此处可右键选择“放大”功能查看图像）

创建工程

接下来就需要向工程添加一些必要的文件了。首先，向工程添加头文件。头文件是以.h为后缀的文件，h即为“head”的缩写。这里的头文件和前面介绍的C语言环境的头文件不同，是F28335自己的头文件，主要定义了芯片内部的寄存器结构、中断服务程序等内容，为F28335的开发提供了很大的便利。F28335工程常用的头文件如表7-1所列。

创建工程

序号	文件名	主要内容
1	DSP2833x_Device.h	包含了其他外设的头文件和大量宏定义
2	DSP2833x_Examples.h	宏定义
3	DSP2833x_Adc.h	模数转换ADC寄存器的相关定义
4	DSP2833x_DevEmu.h	F28335硬件仿真寄存器的相关定义
5	DSP2833x_CpuTimers.h	CPU定时器寄存器的相关定义
6	DSP2833x_DefaultISR.h	F28335中断服务程序的定义
7	DSP2833x_ECan.h	增强型CAN寄存器的相关定义
8	DSP2833x_ECAP.h	增强型CAP寄存器的相关定义
9	DSP2833x_Epwm.h	增强型PWM寄存器的相关定义
10	DSP2833x_EQep.h	增强型光电编码器电路QEP的相关定义
11	DSP2833x_DMA.h	直接存储器访问DMA寄存器的相关定义
12	DSP2833x_Gpio.h	通用输入输出端口GPIO寄存器的相关定义
13	DSP2833x_I2c.h	I2C寄存器的相关定义
14	DSP2833x_McBSP.h	多通道缓冲串口McBSP寄存器的相关定义
15	DSP2833x_PieCtrl.h	PIE控制寄存器的相关定义
16	DSP2833x_PieVect.h	PIE中断向量表的定义
17	DSP2833x_Spi.h	串行外围设备接口SPI寄存器的相关定义
18	DSP2833x_Sci.h	串行通信接口SCI寄存器的相关定义
19	DSP2833x_SysCtrl.h	系统控制寄存器的相关定义
20	DSP2833x_XIntrupt.h	外部中断寄存器的相关定义
21	DSP2833x_Xintf.h	外部接口寄存器的相关定义

表7-1 F28335需要的头文件

（此处可右键选择“放大”功能查看图像）

创建工程

表4-1中所列的头文件构成了F28335寄存器的完整框架，实现了F28335中所有寄存器的C语言的结构体的定义，除了DSP2833x_user.h用户自己编写外，其他的头文件在没有必要的情况下，无需更改其内容。也就是说在创建新工程的时候，不要自己编写这些头文件，只需要将这些具有固定内容的头文件添加到工程中就好。表4-1所列的头文件在配套资源编程素材的include文件夹里，将include文件夹整体复制到led工程文件夹内。此时，CCS已经自动扫描到了工程里新添加的include文件，如图7-7所示。

创建工程

图7-7 添加头文件

虽然头文件已经添加进了led工程，但是CCS的编译器还并不知道这些头文件在哪里，直接进行编译的话会出现错误，因此必须给编译器指定下include文件夹的路径。右击led工程，在所弹出的快捷菜单中选择Properties，打开工程的属性对话框。选择Build→C2000 Compiler→Include Options，如图7-8所示。

创建工程

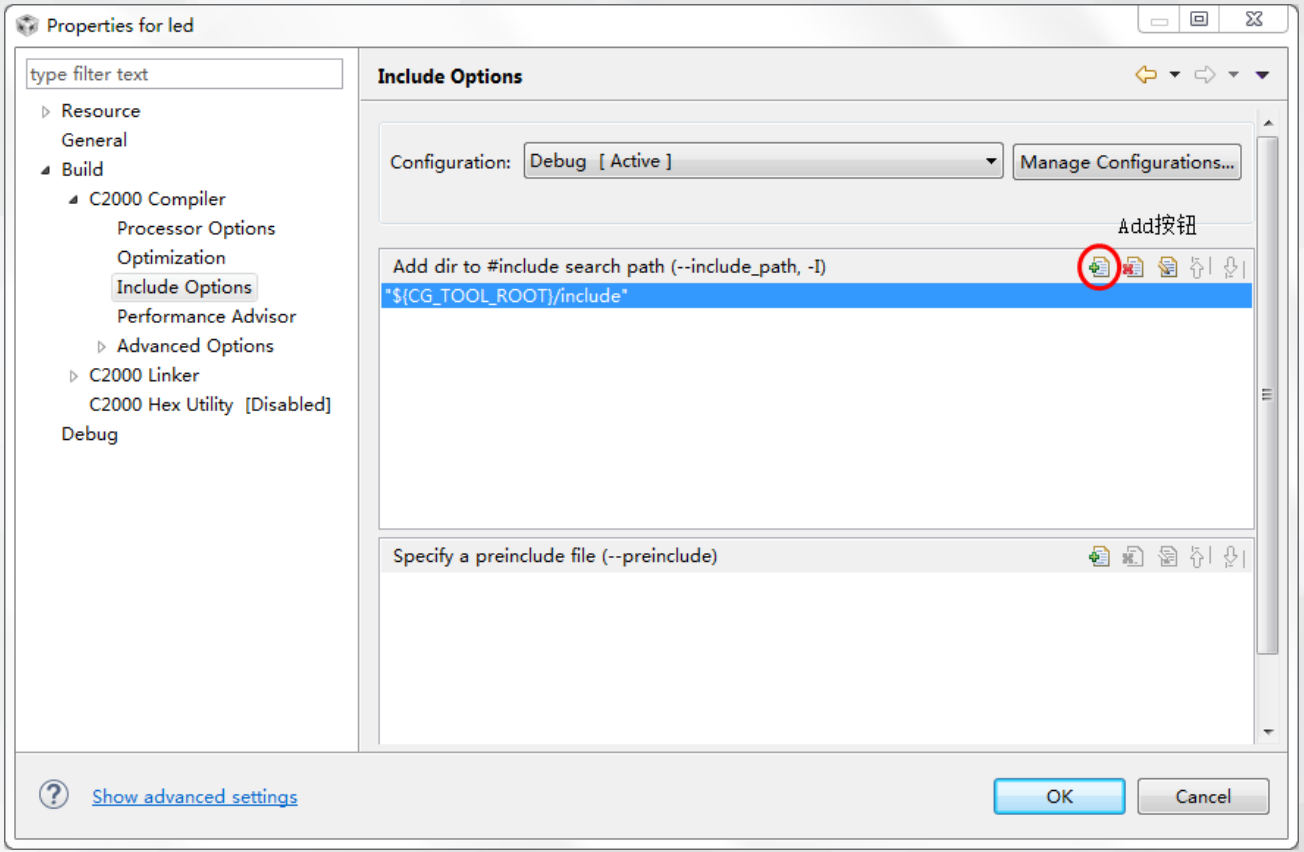

图7-8 设置头文件路径1

创建工程

单击图7-8中的Add按钮，弹出Add directory path对话框，如图7-9所示。单击Workspace按钮，CCS弹出Folder selection，选择led工程下的include文件夹，单击OK按钮，如图7-10所示。界面回到Add directory path对话框，单击OK按钮，如图7-11所示。回到工程的属性设置界面，可以看到头文件的路径已经添加进来了，最后单击OK按钮，完成操作，如图7-12所示。

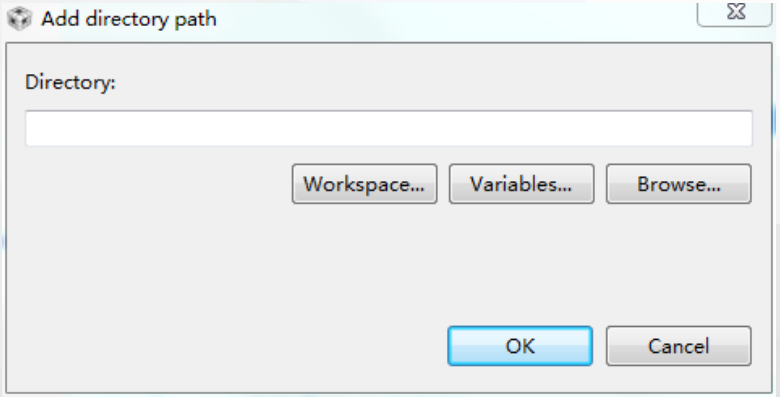

图7-9 设置头文件路径2

创建工程

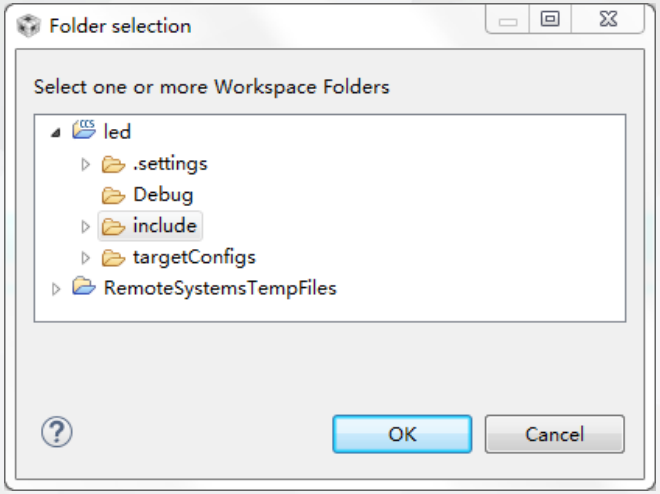

图7-10 设置头文件路径3

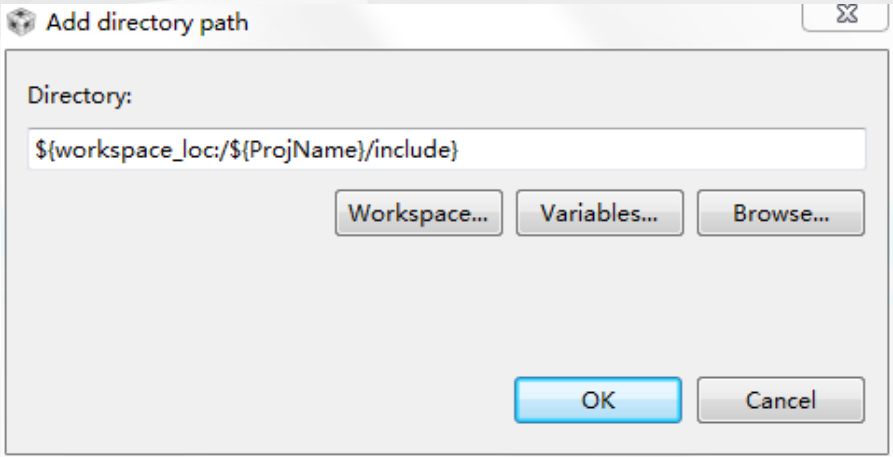

图7-11 设置头文件路径4

创建工程

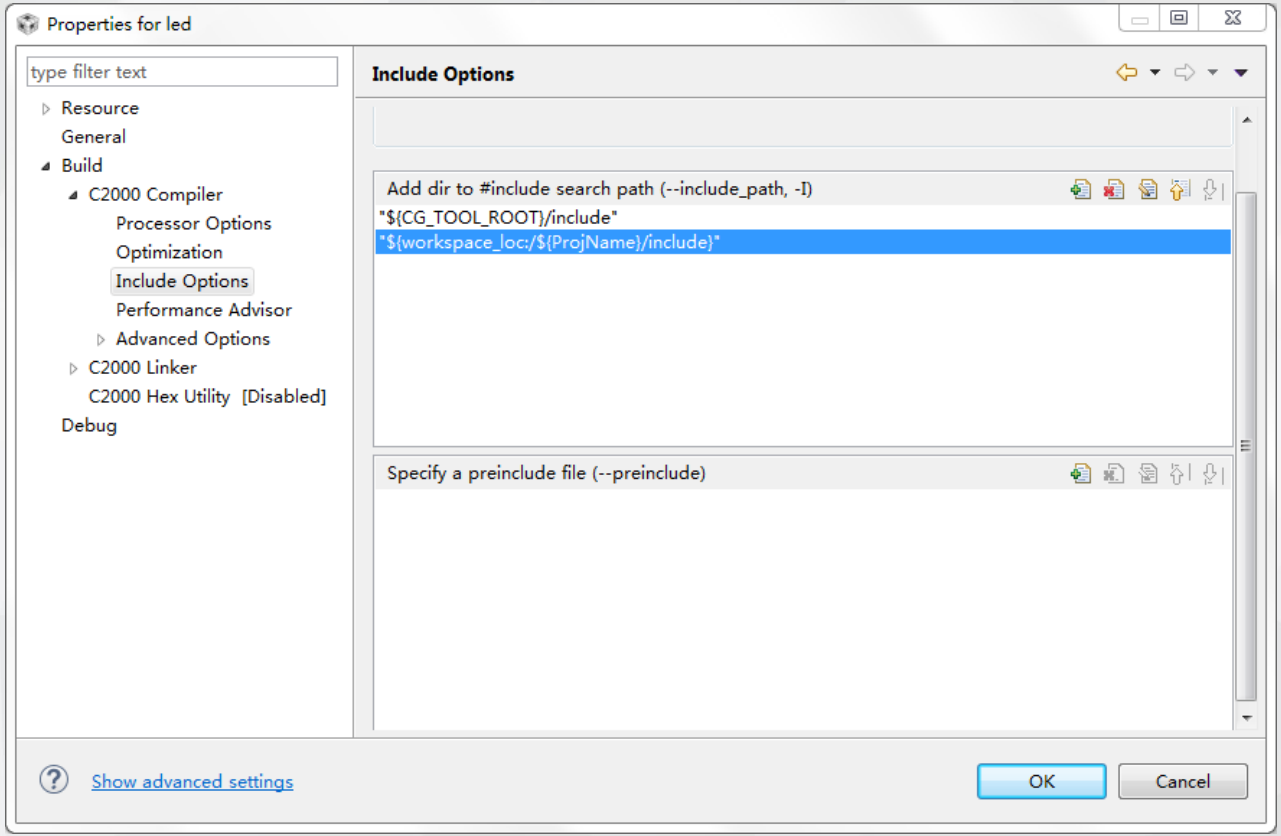

图7-12 设置头文件路径5

创建工程

接下来，需要向工程添加源文件。源文件是以.c为后缀的文件，开发工程时所编写的代码通常都是写在各个源文件中的，也就是说源文件是整个工程的核心部分，包含了所有需要实现的功能的代码。TI为F28335的开发已经准备好了很多源文件，通常只要往这些源文件里添加需要实现功能的代码就行。表7-2列出了F28335工程常用的源文件。

创建工程

序号	文件名	主要内容
1	DSP2833x_Adc.c	ADC初始化函数
2	DSP2833x_CpuTimers.c	CPU定时器初始化函数
3	DSP2833x_DefaultISR.c	包含了F28335所有的外设中断函数
4	DSP2833x_ECan.c	增强型CAN初始化函数
5	DSP2833x_ECAP.c	增强型捕获单元CAP初始化函数
6	DSP2833x_Epwm.c	增强型PWM初始化函数
7	DSP2833x_EQep.c	增强型光电编码器电路QEP初始化函数
8	DSP2833x_DMA.c	直接存储器访问DMA初始化函数
9	DSP2833x_Gpio.c	通用输入输出端口GPIO初始化函数
10	DSP2833x_I2c.c	I2C初始化函数
11	DSP2833x_McBSP.c	多通道缓冲串行口McBSP初始化函数
12	DSP2833x_PieCtrl.c	PIE控制模块初始化函数
13	DSP2833x_PieVect.c	对PIE中断向量进行初始化
14	DSP2833x_Spi.c	串行外围接口SPI初始化函数
15	DSP2833x_Sci.c	串行通信接口SCI初始化函数
16	DSP2833x_SysCtrl.c	系统控制模块初始化函数
17	DSP2833x_Xintf.c	外部接口初始化函数
18	DSP2833x_InitPeripherals.c	包含了需要初始化的外设的初始化函数
19	DSP2833x_GlobalVariableDefs.c	定义了F28335的全局变量和数据段程序
20	main.c	主函数

表7-2 F28335常用的源文件

创建工程

本工程需要添加DSP2833x_Gpio.c、DSP2833x_PieCtrl.c、DSP2833x_PieVect.c、DSP2833x_SysCtrl.c、DSP2833x_GlobalVariableDefs.c、main.c这几个源文件，这也是一般工程都需要的源文件。在led工程文件夹里创建一个source文件夹，然后把上述文件复制到source文件夹内。另外，还需要添加一个DSP2833x_usDelay.asm，这是定义的延时函数。文件复制完成后，ccs软件就自动将这些文件扫描到工程里，如图7-13所示。接着要做的就是往各个源文件里添加实现功能的代码了。

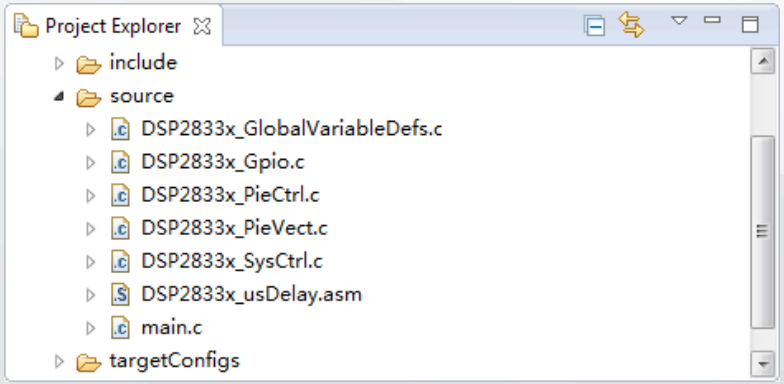

图7-13 添加源文件

编译与调试·编译工程

工程建好后，就可以对其进行编译了。最好的结果就是编译一次通过，但在实际应用中，往往不可能这么理想的，因为在写程序的过程中或多或少地会有疏忽，会犯错，这个也是很正常的，通过编译可以找到问题，然后根据检查出的问题提示来相应的将其解决就好了。

选择Project　Buid ALL，或者右击led工程，在所弹出的快捷菜单中选择Build Project，便可以启动编译，如图7-15、7-16所示。如果编译过程不能顺利进行，请关闭计算机的杀毒软件或者防火墙后重启CCS重新编译试试。

编译与调试·编译工程

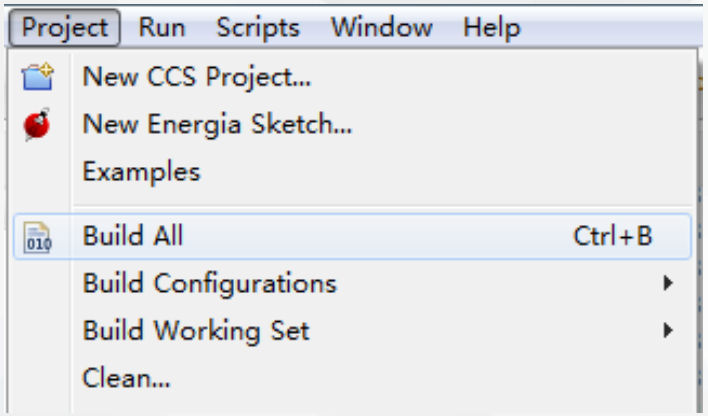

图7-15 编译方法1

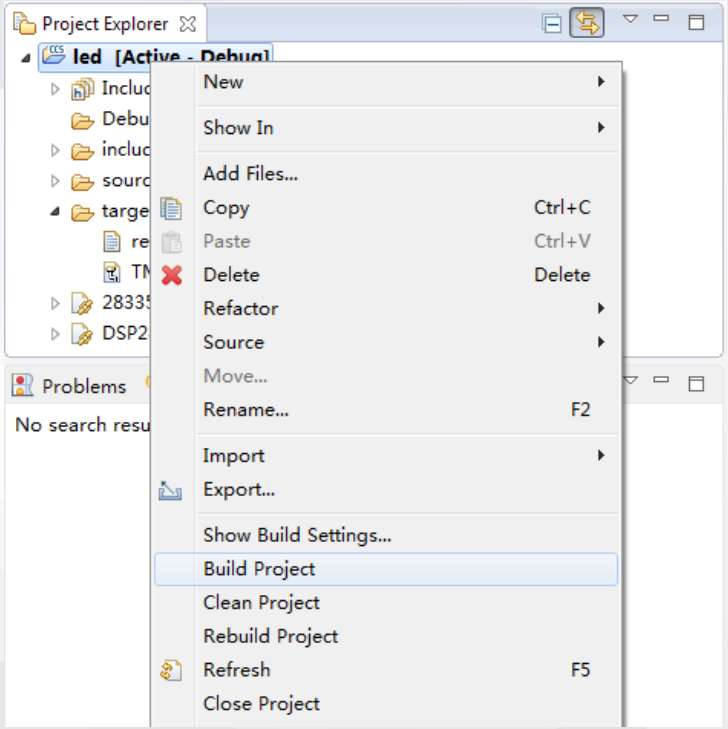

图7-16 编译方法2

编译与调试·编译工程

如果是新建的工程，比如这里的led.pjt，编译完成后如果没有其他语法问题，会有如下所示的一个warning：

```
warning #10210-D: creating ".stack" section with default size of 0x400; use the -stack option to change the default size
```

意思是说工程的.stack段使用的是默认的大小0x400，可以使用-stack选项来改变这个默认大小。首先为什么会有这个提示呢？在28335_RAM_link.cmd文件里，.stack段和.esystem段一起分配给了RAMMM1，而RAMM1空间最大就是0x400，即1K大小。所以如果.stack大小使用的是默认的0x400的话，一旦程序编译后生成的.stack段大小达到了0x400，那么.esystem段就没有存储空间了，这样就会有问题，所以可以给.stack段设置一个小于0x400的数值。右击led工程，在所弹出的快捷菜单中选择Properties，打开属性设置对话框，如图7-17所示。选择Build→C2000 Linker→Basic Options，在Set C system stack size(--stack_size，-stack)一栏填入新的数值，比如0x300，单击OK按钮。当然填入的数值要小于0x400，原因上面已经讲清。

编译与调试·编译工程

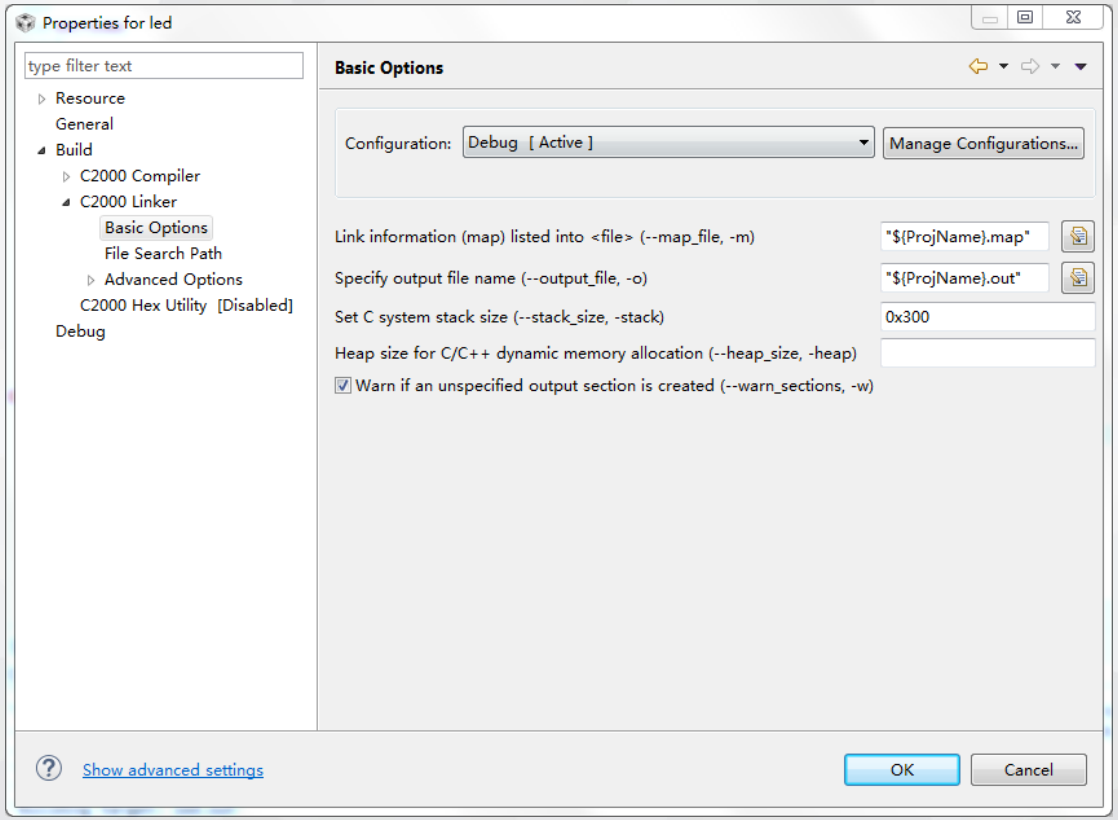

图7-17 设置.stack大小

编译与调试·编译工程

设置好.stack大小后，右击led工程，在所弹出的快捷菜单中选择Rebuild Project，重新编译工程，如图7-18所示。

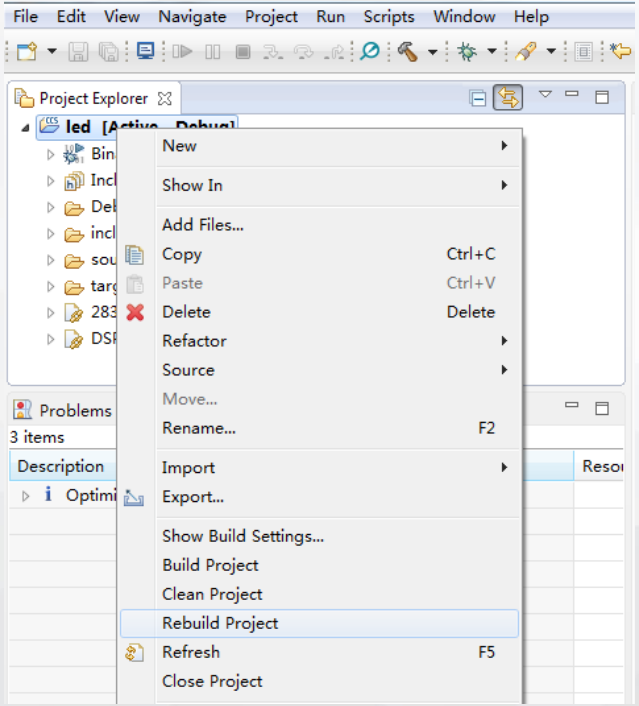

图7-18 重新编译工程

这下编译led工程没有任何问题了，在Debug文件夹里生成了可执行文件led.out，如图7-19所示，这个led.out就是可以下载到F28335里进行运行的文件，可以说是工程的最终结果。不过虽然编译没有任何问题，也生成了可执行文件，只能说明工程里没有语法问题，但是功能是否可以实现现在是无法判断的，只有将可执行文件下载到DSP里，运行调试后才能确定功能是否也是正确的，如果程序功能有问题，那就得具体分析原因，然后去修改代码，再重新编译调试，直到功能也能正确实现。

编译与调试·编译工程

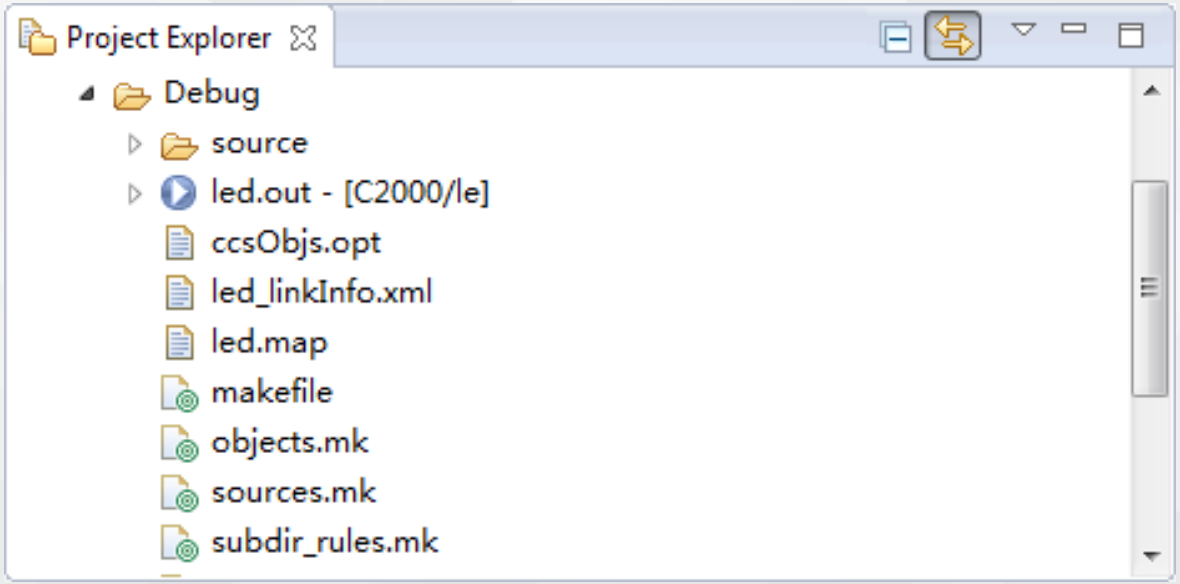

图7-19 生成可执行文件

编译与调试·下载程序

有了可执行的.out文件后，就可以准备把它下载到F28335里运行了。首先，拿出F28335的开发板和仿真器，将开发板和仿真器通过14芯的JTAG口连接好，然后将仿真器插上计算机，如果第一次使用仿真器，还需要给仿真器安装驱动程序，如果已经安装了驱动，计算机插上仿真器后在设备管理器里可以找到相应的仿真器设备。最后，给开发板上电，就是将电源接上开发板。硬件连接如图7-20所示。

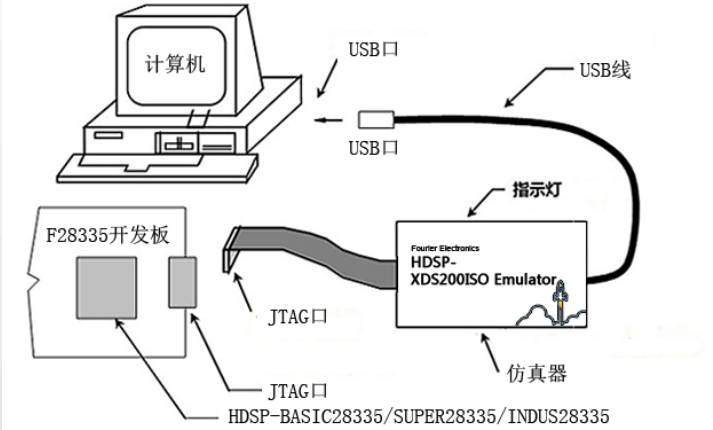

图7-20 硬件连接

接下来，CCS要与DSP建立连接。在CCS的右上角可以看到，CCS6.x主要有两种工作界面，CCS Edit和CCS Debug，之前在建立工程、编写源代码、编译工程等操作都是在CCS Edit下面进行的，如图7-21。接下来的下载、调试等操作就要在CCS Debug下进行了，可以通过鼠标单击图7-21中的两个标签来切换工作界面。当然，当CCS和DSP成功建立连接时，CCS自动会把工作界面从CCS Edit切换到CCS Debug。

图7-21 CCS界面环境

编译与调试·下载程序

选择Run→Debug菜单项，或者单击工具栏上的爬虫图标，然后在弹出的菜单中选择Debug As → Code Composer Debug Session，就可以执行Debug，CCS和DSP开始建立连接，如图7-22和7-23所示。执行Debug还有一种途径，右击led工程，在所弹出的快捷菜单中选择Debug As → Code Composer Debug Session。

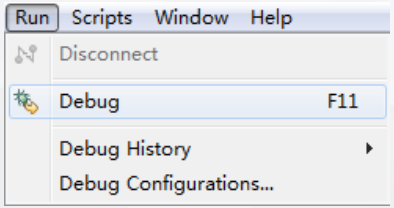

图7-22 执行Debug方法1

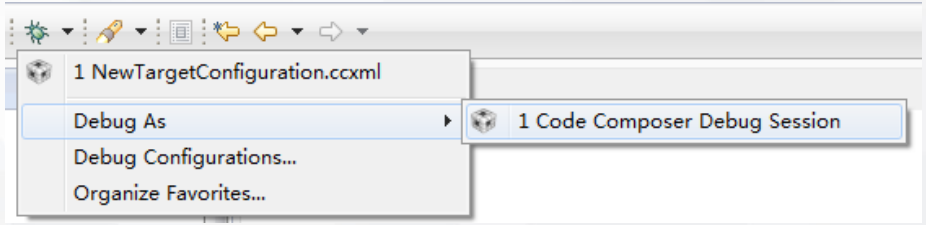

图7-23 执行Debug方法2

如果硬件没有问题，工程中的目标链接配置文件TMS320F28335.ccxml的配置也正确，那CCS应该就会顺利和DSP建立连接，CCS的工作界面切换至CCS Debug，如图7-24所示。

如果CCS和DSP没法建立连接，CCS会弹出错误信息的对话框，可以从配置文件和硬件两个方面去排查问题。图7-24中工具栏上的常用按钮说明如图7-25所示。

编译与调试·下载程序

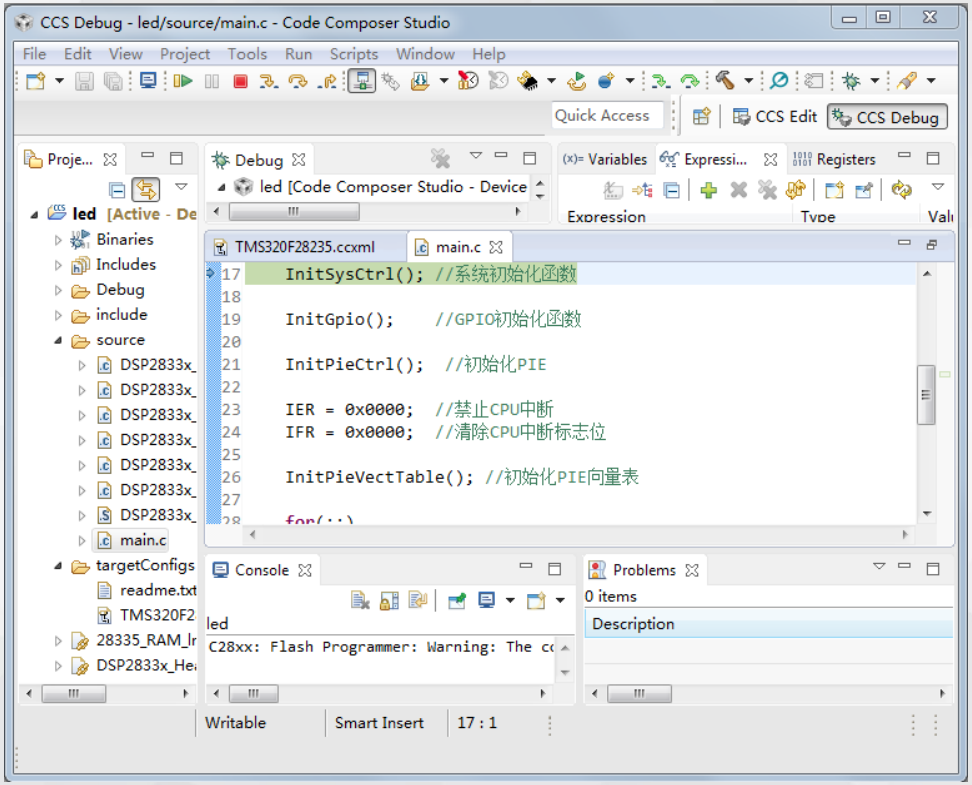

图7-24 CCS Debug界面

编译与调试·下载程序

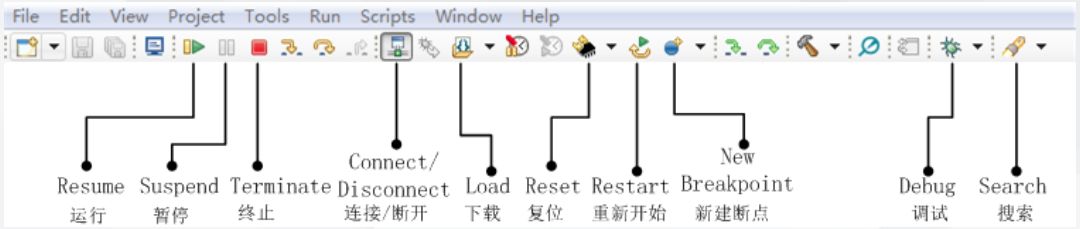

图7-25 工具栏常用按钮说明

在CCS和DSP之间的连接建立成功的时候，CCS已经自动将led.out下载到了F28335中了，单击工具栏上的Resume按钮，程序就开始运行了，如果使用的是HDSP-SUPER28335开发板，就可以看到6个LED灯开始闪烁，说明前面写的程序完全正确，实现了需要的功能。

暂停运行的程序，可单击工具栏上的Suspend按钮；终止调试，退出CCS Debug界面，可单击工具栏上的Terminate按钮；只是断开连接，留在CCS Debug界面，单击工具栏上的Connect/Disconnect按钮。

如果需要单独下载程序的话，该如何操作呢？单击工具栏上的Load按钮，CCS弹出Load Program对话框，如图7-26所示。下载本工程的.out文件，CCS已经把路径设置好了，如果下载的是别的.out文件，可以单击Browser按钮，选择需要下载文件的路径，设置好后单击OK按钮，CCS便会把程序下载到DSP中。

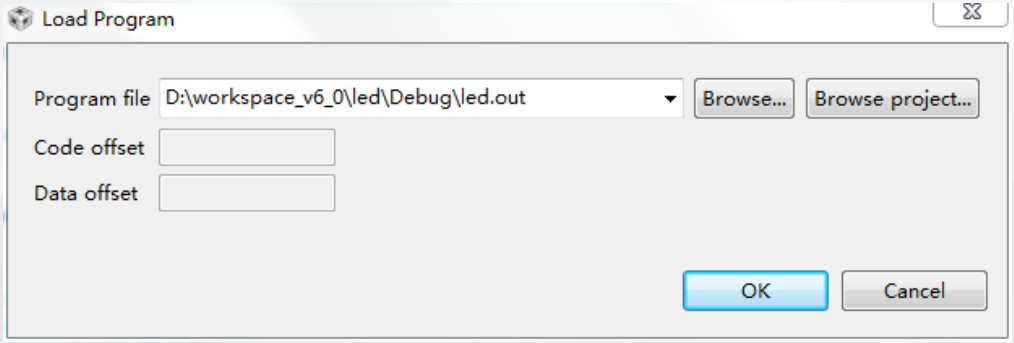

图7-26 Load Program

在调试程序的时候，经常会遇到需要让程序运行时停在某一行代码处的情况，以便于分析程序的功能，分析判断问题，这就是添加断点操作。比如需要在main.c这个文件内的DELAY_US函数处设置断点，只需将鼠标移到这一行，然后在代码编辑窗口上双击，便成功添加了一个断点，如图7-27所示。这样当程序运行到断点处的时候，就会暂停下来。

```
30
31          GpioDataRegs.GPATOGGLE.bit.GPIO0=1; //翻转电平
32          GpioDataRegs.GPATOGGLE.bit.GPIO1=1;
33          GpioDataRegs.GPATOGGLE.bit.GPIO2=1;
34          GpioDataRegs.GPATOGGLE.bit.GPIO3=1;
35          GpioDataRegs.GPATOGGLE.bit.GPIO4=1;
36          GpioDataRegs.GPATOGGLE.bit.GPIO5=1;
37          DELAY_US(1000000); //延时1s
38      }
39  }
```

图7-27 添加断点

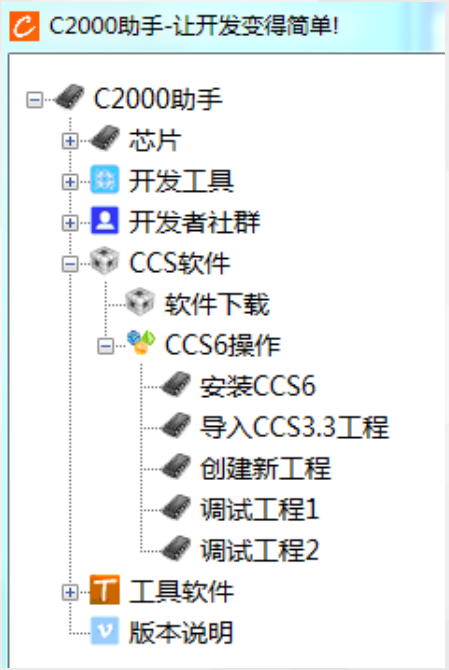

图7-28 CCS6操作视频

除了上面介绍的，常用的调试操作还有一些，放在后面章节中实际遇到的时候进行讲解，也可以通过观看C2000助手里CCS6.x的视频来学习，如图7-28所示。

本章以控制LED灯闪烁为实例，介绍了在CCS6.x开发环境下如何创建一个新的完整的工程，并以此为基础，介绍了下载、调试程序的一些常用操作。