第 2 课 一个完整工程的构成(完整版)

本帖最后由 likyo 于 2009-4-15 22:18 编辑

今天开始，我们就要真正开始 2812 的学习了。我们今天的内容比较简单，主要是让没有基础的朋友来了解一下 DSP 开发需要哪些工具，一个完整的 2812 的工程（Project）是由哪些文件组成的，各个文件的主要作用是什么，以及如何在 CCS 里创建一个新的工程。

1.DSP 开发所需要的工具

咱要进行 DSP 的学习和开发了，可能从没接触过 DSP 的朋友就要问了，我们做 DSP 开发首先要哪些开发工具呢？DSP 开发通常需要软件开发环境和硬件平台。软件开发环境为 CCS（Code Composer Studio），是 TI 公司为方便开发人员而设计的软件环境。硬件平台由仿真器和目标板组成。仿真器的作用是将目标板和 PC 机连起来，使得您可以在 CCS 里对目标板上的 DSP 进行编程，烧写和调试等工作，而目标板是指具有 DSP 芯片，上电后能保证 DSP 独立运行电路板，通常为各个公司设计的开发板或者您自己设计的电路板。

2.安装并配置 CCS

首先，让我们来了解一下 CCS 的版本。目前，CCS 常用的版本有 CCS2.2，CCS3.1 以及 CCS3.3，CCS2.2 是一个分立版本，也就是每一个系列的 DSP 都有一个 CCS2.2 的开发软件，分 CCS2.2 for C2000，CCS2.2 for C5000，CCS2.2 for C6000。而 CCS3.1 和 CCS3.3 是一个集成版本，支持全系列的 DSP 开发。我们推荐使用 CCS2.2，因为这是目前最稳定的版本。但是使用最多的是 CCS3.3，因此我们这次学习也以 CCS3.3 为软件开发环境，和大家一起探讨 2812 的软件开发。如果您还没有安装 CCS，请访问下面的地址进行下载。如果您购买了我们 HELLODSP 的相关产品，我们会为您免费提供含有 CCS 开发环境的资料光盘。

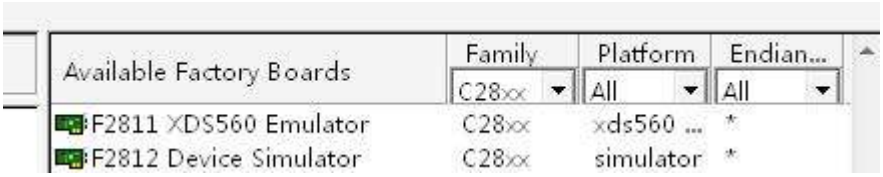
CCS 下载地址：http://www.hellodsp.com/bbs/view ... &extra=page%3D1

CCS 的安装和普通应用软件的安装没有多大区别，在这里就不赘述了。建议大家默认安装路径就行了，如果需要修改安装路径，请确保将 CCS 安装到不含中文字符的路径。

CCS 安装完成之后，桌面上会出现两个图标，一个是 CCS，另外一个是 CCS Setup。在使用 CCS 之前，需要对 CCS 进行一些配置操作，以保证 CCS 支持我们所要开发的 DSP，在这里就是 2812 了。

双击，打开 CCS Setup。如果您具有硬件开发平台，即具有仿真器，那么请您根据仿真器的

生产厂家提供的配置说明进行相应的操作。如果您不具有硬件开发的条件，目前只能软件仿真，那么请您通过中间的筛选框，找到“F2812 Device Simulator”，将其拖入左边的“System Configuration”栏，然后点击“Save and Quit”退出 CCS Setup 的设置，可以进入 CCS 啦。

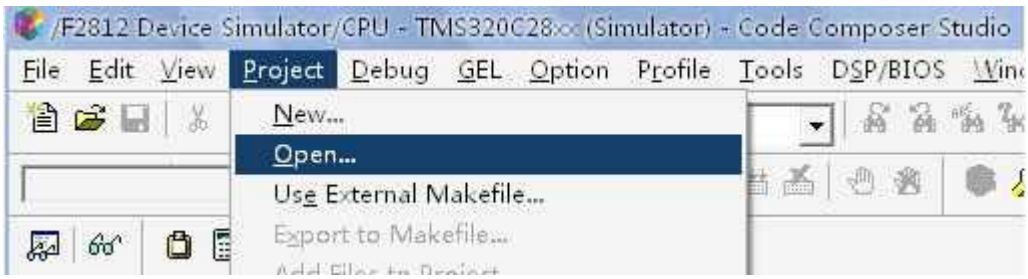


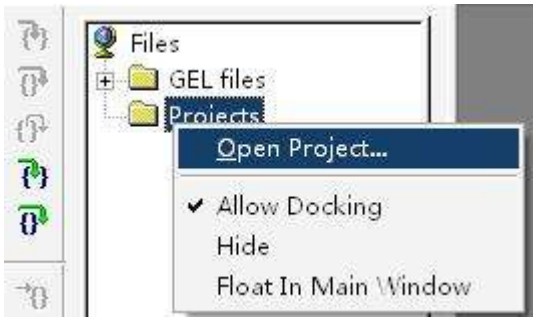
3. 一个完整的工程由哪些文件构成

请下载附件中的例程 gpio，我们将以这个程序为例为您讲解一个完整的工程是由哪些文件构成的，以即这些文件大致的作用。

下载完程序后，请解压缩，然后将其拷贝到 CCS 安装路径下面的 myprojects 文件夹，如果您刚才是默认安装的，那就是 C:\CCStudio_v3.3\myprojects。可能又会有朋友要问了，我只能放到 myprojects 文件夹吗？当然不是的，您可以将工程文件夹放在您喜欢的位置，但是和安装路径一样，请确保访问到这个文件夹的路径里不要出现中文字符。

OK，将 gpio 文件夹放好了吗？放好之后，我们打开 CCS，添加工程。添加工程的方法有两种，一种是点击菜单栏的 Project—>Open，另一种方法是左侧 Files 窗口内，点击“Projects”然后右键，点击“Open Project”。

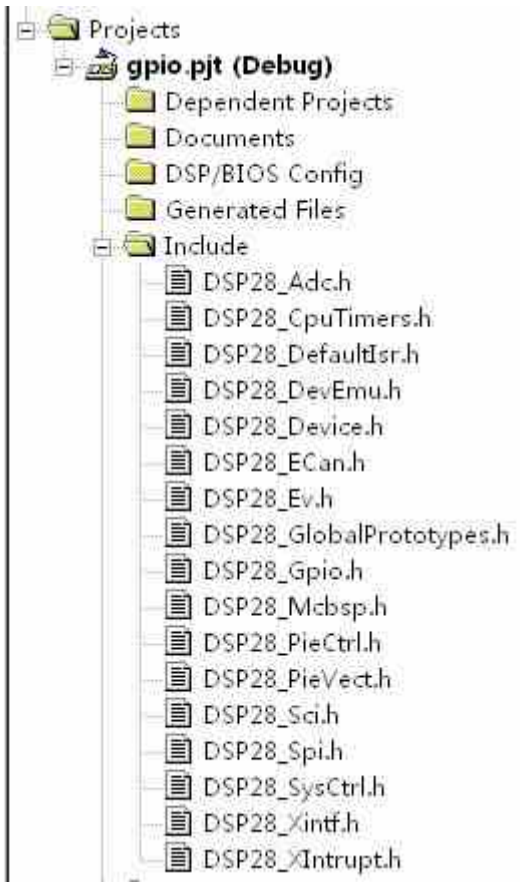




打开 gpio 工程之后，gpio.pjt 工程会显示在左侧 Files 窗口内。

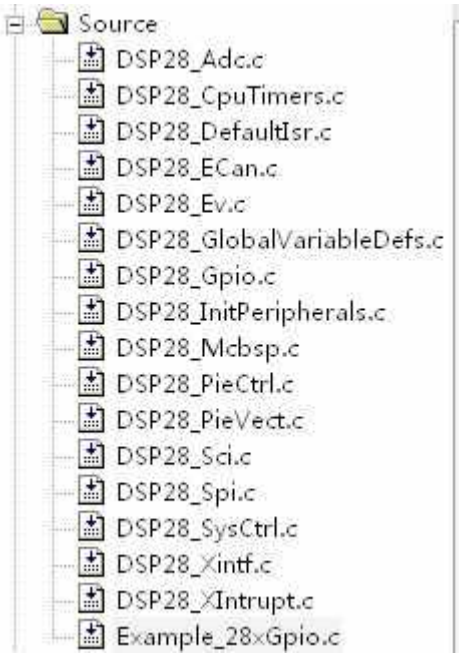
点击 gpio.pjt 左边的加号，工程内部的文件就展现在我们面前了，下面我们来分析一下构成该工程的文件。

首先我们看到的是 Include 文件夹下面有很多后缀是.h 的文件，这就是 2812 的头文件了，头文件的作用是定义了 2812 内部寄存器的数据结构。头文件一般情况下不需要修改，如果你需要定义一些在整个工程内都具有作用域的全局变量的时候，可以在头文件中定义这些变量，具体的方法我们以后在例程或项目实践中应该会有介绍。

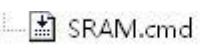


然后是 libraries 下面扩展名为.lib 的库文件，它是 C 语言系统的库文件。

库文件下面是 source 文件夹，里面的文件都是以.c 为扩展名的，顾名思义，就是源文件，也就是我们开发时编写的软件代码都是保存在这些文件中的。



最后是以.CMD 为扩展名的文件，这个文件的作用是用来分配存储空间的。由于 DSP 编译器的编译结果是未定位的，DSP 也没有操作系统来定位执行代码，DSP 系统的配置需求也不尽相同，因此我们根据实际的需求，自己定义代码的存储位置。打个通俗的比喻，就是我们有一个仓库，现在需要把货物存放到仓库里面去，为了便于日后取用货物，我们将货物分门别类，然后把它们存放到指定的位置去。把哪些货物放到哪个位置的规则，就是我们的 CMD 文件的内容。



CMD 文件又分成两种。一种是分配 RAM 空间的，用来将程序 load 到 RAM 内进行调试，因为

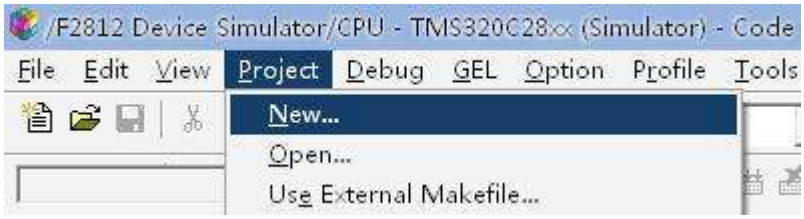
我们大部分时间都是在调试程序，所以多用这类 CMD，gpio 工程中的 sram.cmd 就是用于分配 RAM 空间的，另一种是分配 FLASH 空间的，当程序调试完毕后，需要将其烧写到 FLASH 内部进行固化，这个时侯我们就需要使用这类 CMD 文件了。

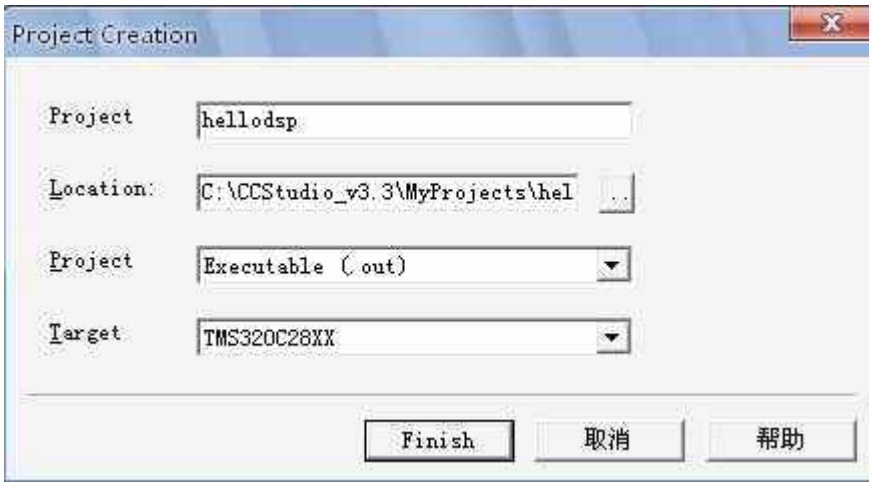
从上面的分析我们可以看出，一个完整的工程需要由库文件（.lib），头文件（.h），源文件（.c）和 CMD 文件组成，缺一不可。

4.如何创建新的工程

下面，跟着我一起来建立个简单的工程吧，主要通过这些步骤来了解一下如何创建新工程，方法有多种多样，我介绍自己常用的，需要用到的文件请下载附件中的 newprojects 文件夹，此文件夹内的文件推荐为建立新工程的素材，请妥善保存。

1.打开 CCS，点击菜单栏里的 Projects，点击 New，会跳出新工程设置的对话框，如下图所示。





在 Projects 栏内我们填写工程的名字：hellodsp。检查 Location，如果您都是按照默认路径来的，那没有问题，如果不是默认路径，您得手工指定一下文件路径，确认没有问题后点击 Finish。我们发现在 myprojects 文件夹下多了 hellodsp 文件夹，hellodsp 文件夹下生成了 hellodsp.pjt 文件，而且在 CCS 左侧的 Files 栏内，出现了我们刚刚创建的

hellodsp.pjt。

2. 根据前面一个完整工程的组成情况的分析，我们首先来准备头文件。由于头文件多数情况下是不需改动的，也就是说大家用的头文件都是一样的，因为是定义 2812 的内部资源，所以可以将 newproject 文件夹内的头文件全部复制到 hellodsp 文件夹。

3. 将 newproject 文件夹内的.lib 文件和.cmd 文件同样复制到 hellodsp 文件夹。

4. 下面就剩下源文件了。我比较喜欢例程中的文件结构，所以建议还没有形成编程风格的朋友也采用这种文件结构。我们先来分析一下 newproject 文件夹下各个源文件的内容，以便于我们更好的理解和采用这种文件结构。

DSP28_ADC.C ——外设 AD 的初始化函数，与外设 AD 相关

DSP28_CpuTimers.C——CPU 定时器的初始化和配置函数，与 CPU 的定时器相关

DSP28_DefaultIsr.C——这个文件很重要，包含了 2812 所有的中断函数，写中断时，只要将程序写在对应的函数内就可以，大大保证了中断的成功率。

DSP28_ECan.C——外设 CAN 的初始化函数，与外设 CAN 相关。

DSP28_Ev.C——外设 EV 的初始化函数，与外设 EV 相关。

DSP28_GlobalVariableDefs.C——全局变量的定义，这个文件也很重要，定义了 2812 的寄存器，中断向量表等内容。

DSP28_Gpio.C——GPIO 的初始化函数，只和 GPIO 相关。

DSP28_InitPeripherals.C——所有外设的初始化函数，函数的内容是调用了 2812 各个外设的初始化函数。

DSP28_Mcbsp.C——Mcbsp 的初始化函数，只和 Mcbsp 相关。

DSP28_PieCtrl.C——PIE 初始化函数，和中断相关，很重要。

DSP28_PieVect.C——PIE 中断向量表定义以及初始化，很重要。

DSP28_Sci.C——外设 SCI 的初始化函数，只和外设 SCI 相关。

DSP28_Spi.C——外设 SPI 的初始化函数，只和外设 SPI 相关。

DSP28_SysCtrl.C——系统初始化，主要对开门狗，时钟等模块进行初始化，以保证 2812 正常工作，非常重要。

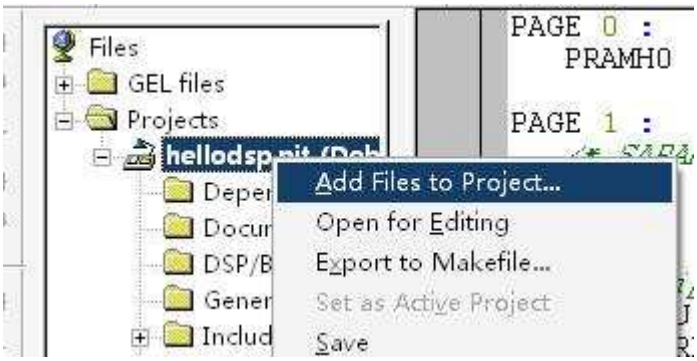
DSP28_Xintf.C——外部接口的初始化函数。

DSP28_XIntrupt.C——外部中断的初始化函数。

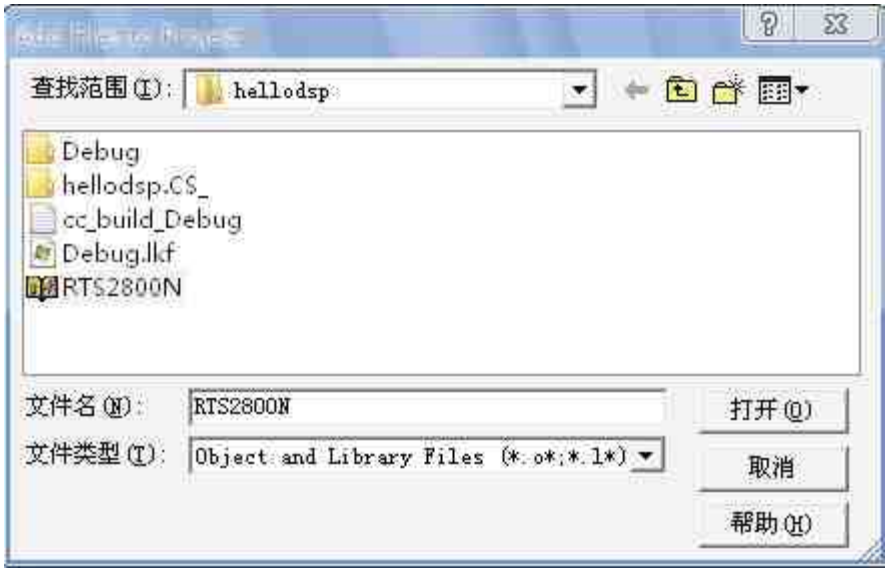
Example_28xGpio.C——main 函数所在的文件，但是各个工程的 Main 函数一般都是不一样的。

通过上面的分析我们可以看到几个文件非常重要，DSP28_DefaultIsr.C，DSP28_GlobalVariableDefs.C，DSP28_PieCtrl.C，DSP28_PieVect.C，DSP28_SysCtrl.C，因此我建议大家每次新建工程的时候，就把这些未编辑过的文件复制过来。其他的外设相关的文件，您这个工程中涉及到哪个外设，您就把这个外设相关的源文件复制过来，一起加入工程。由于 Main 函数所在的文件内容各个工程都不一样，所以建议大家自己创建，顺便来学习一下如何在 CCS 里创建一个.c 的源文件。

此时我们的新工程 hellodsp.pjt 还是空的，里面啥都没有，接下来就是往工程内添加文件了。首先我们来添加库文件。点击“hellodsp.pjt”，右键，点击“Add Files to Project”，跳出了添加文件的对话框。



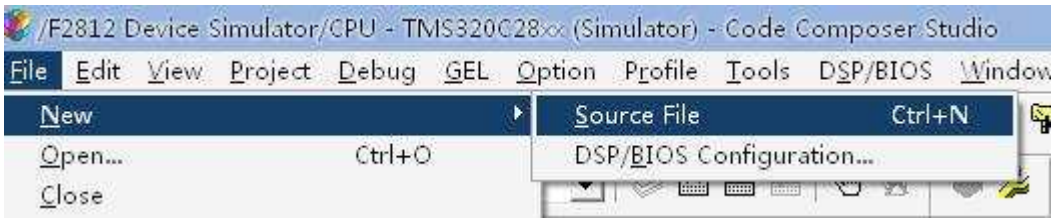
在筛选框栏内选择“Object and Library Files”，就会看到 RTS2800N.lib，点击并打开，便将库文件添加到工程中来了。



接下来，用同样的方法，来添加 sources 文件夹下的源文件和 CMD 文件。只是添加源文件时，筛选框的条件是“C++ Source File(*.cpp;*.cc;*.cxx)”添加 CMD 文件时，筛选框的条件应当选择”Linker Command File（*.cmd;*.lcf）”。

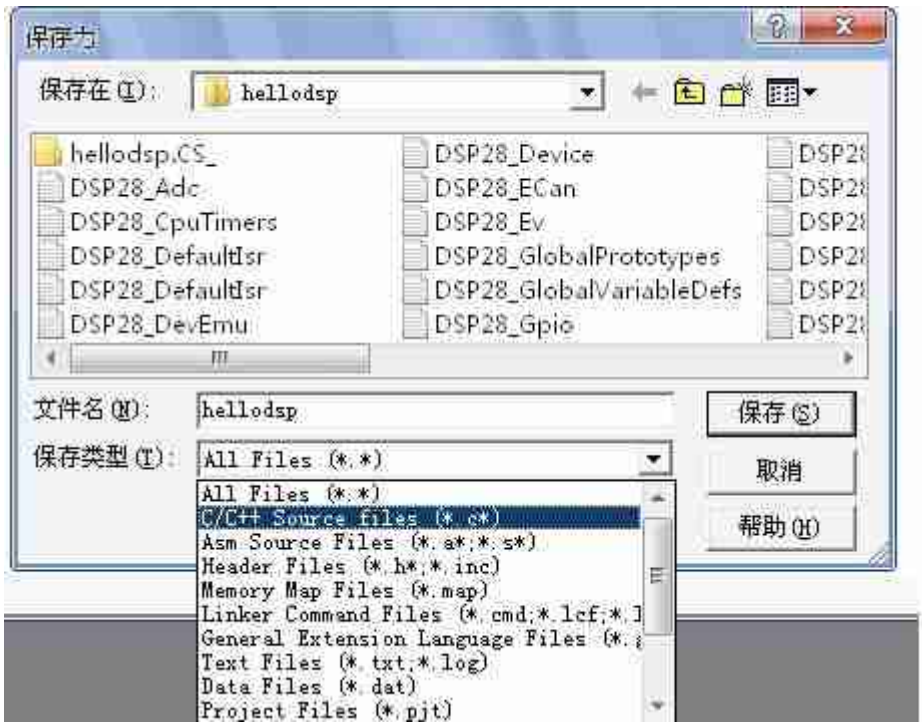
文件添加完成了，是否 hellodsp 这个工程就建好了呢？可能您想起来了，我们还没有添加头文件，也没有主函数呢。对了，我们还缺少主函数，头文件这里先不管，到最后看看是怎么回事。

我们点击菜单栏中的’File’,”New”,”Source File”，在编辑区域内出现了一个新的 Untitled1 文件。



然后，点击“File”，“Save”，会出现保存文件的对话框。文件名填写“hellodsp”，关

键要注意的是保存类型，因为我们现在要建立的是源文件，所以选择“C/C++ Source Flies（*.c*）”，点击保存。

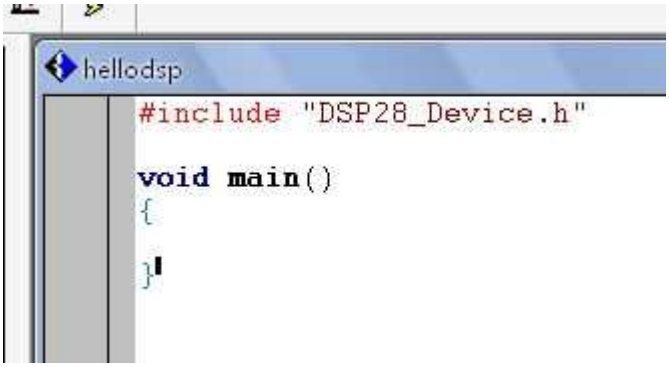


这时，我们看到原来的“Untitled1”变成了“hellodsp”了。我们在 hellodsp 文件内输入以下内容：

```
1.  #include "DSP28_Device.h"
2.
3.  Void main(void)
4.  {
5.
6.  }
```

复制代码

然后点击保存，并关闭 hellodsp.c 文件。



然后用我们刚才添加文件的方法将 hellodsp.c 添加到工程中来。

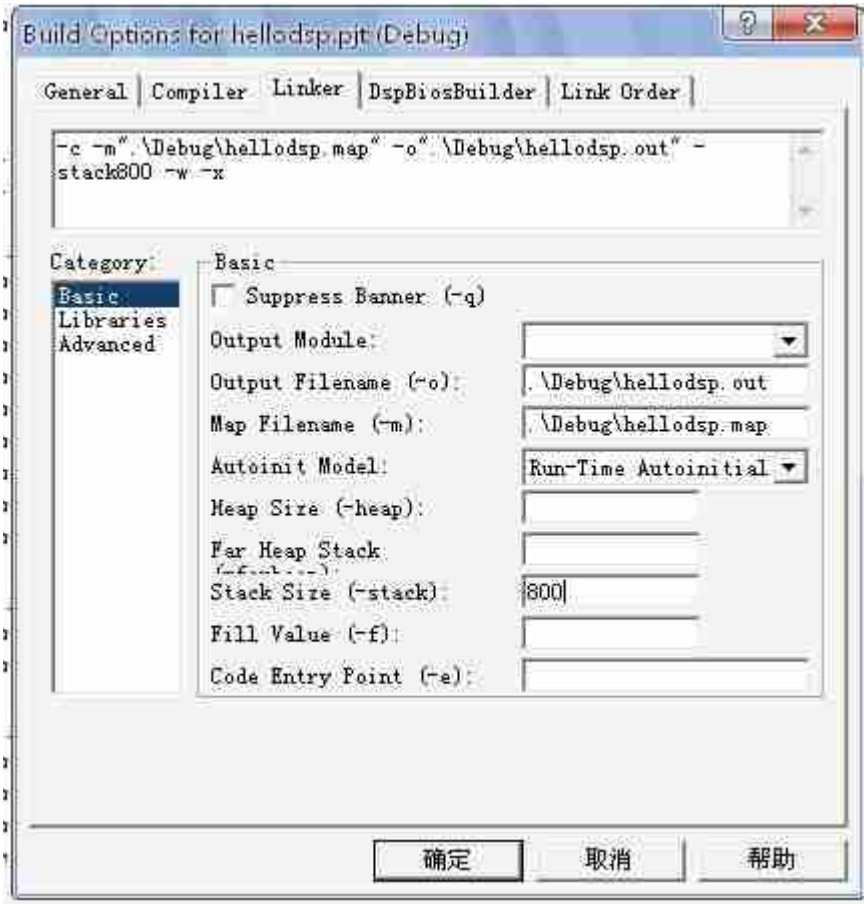
接下来，我们就要编译啦，是不是有些激动？终于把一个新的工程搭建完成啦。点击工具栏上的“Rebuild All”，开始编译咯。

可惜的是，我编译完成时，提示了一个”warning”，内容如下：

```
1.  [hellodsp.c] "C:\CCStudio_v3.3\C2000\cgtools\bin\cl2000" -g -pdsw225
-fr"C:/CCStudio_v3.3/MyProjects/hellodsp/Debug" -d"_DEBUG" -d"LARGE_MODEL" -ml
                    -v28 -@"Debug.lkf" "hellodsp.c"
                                2.
3.  [Linking...] "C:\CCStudio_v3.3\C2000\cgtools\bin\cl2000" -@"Debug.lkf"
                            4.  <Linking>
5.  >> warning: creating .stack section with default size of 1024 bytes.
                              6.     Use
        7.            -stack option to change the default size.
                                8.
                          9.  Build Complete,
                10.   0 Errors, 1 Warnings, 0 Remarks.
```

复制代码

很糟糕啊，呵呵，不过，静下心来先大概分析一下提示 warning 的原因，提示是用默认的 1024bytes 来创建.stack section，就是堆栈段。我在站上搜了一下，发现有朋友也遇到过这样的问题。怎么解决呢？点击菜单栏的”Project”,”build options”，弹出编译选项的对话框。选择“Linker”标签，在 Stack Size(-stack)栏填写“800”，点击确定。



重新编译，哇，通过了。“0 errors ,0 warnings ,0 remarkings”！

不过我想为什么刚才要将 stack section 段改成 800 呢，改成其他的可以吗？我回头又将刚才的 800 改成了 1000，编译也完全正确。所以我觉得，可能改成小于 1024 的值，只要让它不要是默认的 1024 就不会有 warning 了。虽然，有时候 Waring 其实并不会影响程序，但是总会觉得不安，所以还是想办法将其解决吧。

怎么样，心动了吗？赶紧下载附件开始自己搭建新的工程吧。

通过本课时的学习，您应该弄清楚了一个完整的 2812 工程由哪些文件构成了，也应该会搭建属于自己的工程了，但是有些问题我们还没有阐述，留给大家讨论和思考：

1. 什么是 GEL 文件？GEL 文件的作用是什么呢？
2. Lib 文件内部究竟是什么内容，我们自己能编辑 LIB 文件吗？