Texas Instruments

TI-EEWORLD

联手打造 越控越有趣

TI C2000 LaunchPad炼成记!

目录

序

看到这本书的第一眼，我就被书名吸引住了——《越控越有趣》。的确，这本书里面的东西非常“有趣”。首先，做的这些东西本身就有趣味性，另外，这些应用的点也是非常有价值的话题。C2000 兼具 DSP 和 MCU 的特性，灵活性非常大，写出高品质的程序不容易，需要历练和付出，所以有趣是件非常重要的事。曾经见过很多书，一上来就讲述控制算法，让我联想到一位老夫子板着面孔讲课，答不上来还要打手板。而这本《越控越有趣》则是用网友分享的方式带领大家入门，无论从语言还是形式都让读者有很强的亲近感。

应该说，这本书让我有些出乎意外。它的概括性非常强，同时又很完整，从刚开始的初识篇、进阶篇，到实战篇，再到相关资源;从网友的体会到 TI 官方的课程，再到网友的实际应用。

对于一个新手而言，读完这本书，就会对 C2000 有一个清晰的整体认识，知道自己该向哪个方向走，而且在走的过程中也有一定的趣味性。大家都知道兴趣是最好的老师，所以 EEWORLD 论坛做的这本电子书是一种非常好的尝试。

还要真心感谢 EEWORLD 的网友们分享了这么多应用，它们涵盖了热点的市场应用，和市场的节奏紧紧相扣。C2000 在工业方面的应用，基本上都触及到了。这些年来我们并没有逐一对这些市场进行推广，但好的产品自己会说话，这些方案充分证明了这一点。虽然 C2000 现在有非常好的市场地位，但我们深知市场一直在向前走。就在写这篇序的前几天，我刚从休斯顿和 C2000 的产品线团队会面回来，那是一直非常值得尊重的团队，他们多年来一直保持创新的动力、在不断地优化产品，包括将 ARM 核与 C2000 平台结合在一起，把二者进行优化整合，这样用户就可以同时享受到 C2000 和 ARM 芯片的特性。

我一直对来自“民间”的应用保持着非常高的兴趣和敏感度。C2000 一开始进入市场就定位于电力电子领域的数字化应用，得到了较好的结果，也就容易将应用限制在这些方面。但是在这个平台上其实可以有更多、更好的应用出现，“民间”的应用会打破这个定势，反过来会影响产品的发展方向。曾经 C2000 发展的前五、六年是由欧洲主导，但是之后中国基本上主导了产品的方向，尤其是 Piccolo 里融入了非常多的中国元素。虽然 TI 是一家典型的美国公司，但 C2000 和中国市场已经建立了非常好的互动关系。

希望网友们在 EEWORLD（www.eeworld.com.cn）平台上多多交流，提出你们的看法，我们都会认真去读，努力在未来 C2000 的产品中涵盖更多中国用户的需求。再次感谢 EEWORLD 的网友们！

谭　徽

德州仪器半导体事业部 MCU 应用部门经理

前言

第一眼看到这本书时，头脑里反复出现着这段话“这是一本活着的书、开放的书，”因为书中的内容源于论坛，而且其中的内容还在不断更新。对于即将使用或正在使用 TI C2000 的工程师，只要顺着书提供的超链接访问 EEWORLD 论坛的相应帖子，即刻进入这些技术话题的最新讨论。

最近几年，因为我本身做的是新能源领域，所以与 TI C2000 打交道比较多，最深的感受就是：C2000 就是为自动控制而生。这款微控制器巧妙地将 DSP 技术和各种高性能外设结合在一起，其内部集成了模 / 数转换器、数字输入 / 输出端口、PWM 发生器、定时器、CAP/QEP、SCI/SPI/CAN 通讯接口等外设，因而让数字化控制变得简单、灵活、高效。

TI-EEWORLD 联手打造的这本《越控越有趣 -TI C2000 LaunchPad 炼成记》汇集了诸多 EEWORLD 资深网友的大量原创作品，这些内容都来自一线工程师的设计经验心得，由浅入深，内容覆盖软件、硬件、工具等各方面。与传统教科书、器件手册相比，这本书显得更鲜活，也许不经意间，读者会发现很多的内容恰恰是为自己量身定制的。

在内容安排上，也足见编者的用心良苦，初识篇、进阶篇和实战篇。这样的安排非常符合学习、掌握一门新技术的客观规律：先将新人手把手带入门，然后参考 TI 资深工程师的讲解、论坛网友的实战经验，尽快上手一个简单的设计项目。其中，初识篇、进阶篇图文并茂，力求完整展示操作过程。实战篇更注重解决实际问题，通常，从自动控制理论到实际编程解决问题，对于新手来说这是一道不小的水沟，而这部分恰恰是一块垫脚石，让新人跨过这道水沟变得更容易。

希望本书的读者能够加入到 EEWORLD C2000 技术专区的讨论中来，分享、讨论、交流、进步。最后，祝大家有一个美妙的 TI C2000 学习之旅！

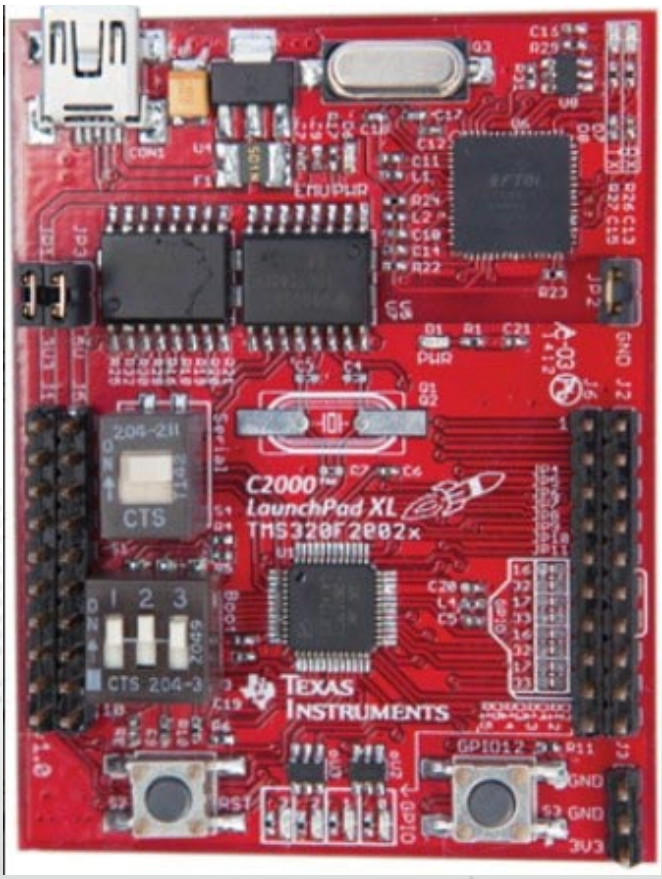
EEWORLD 网友—kata
2013 夏于深圳

第一章 初识篇—
C2000 LaunchPad 你不可不知的事儿

1.1 C2000 LaunchPad 开发板介绍

C2000 ™ Piccolo LaunchPad 是价格低廉的评估平台，旨在帮助您跨入 C2000 Piccolo 微控制器实时控制编程领域。 LaunchPad 基于 Piccolo TMS320F28027，具有 64KB 板载闪存、8 个 PWM 通道、eCAP、12 位 ADC、I2C、SPI、UART 等大量独有特性。 它包含许多板载硬件，例如，集成的隔离式 XDS100 JTAG 仿真器使编程和调试简单易行；采用 40 PCB 引脚，可以方便地连接 F28027 处理器的引脚；具有重置按钮和可编程按钮等。C2000 LaunchPad 不仅有开发所需的硬件，还使用户可以通过免费的 controlSUITE 访问示例代码、库、驱动程序以及大量其他资源。 用户还可以与 controlSUITE 一起下载 Code Composer Studio 集成式开发环境 (IDE) 版本 5 的无限制版。

利用进行开发所需的所有硬件和软件，用户可以集中精力学习或开发在数字照明、电机控制、数字功率转换、精度感应等大量领域中使用的实时控制系统。



特性

✧ 预编程 C2000 Piccolo F28027 MCU。
✧ 利用内置隔离式 XDS100 JTAG 仿真器，可以通过 USB 进行实时系统内编程和调试。
✧ CPU 重置按钮和可编程按钮。
✧ 使您可以为任何 Piccolo F2802x 器件进行开发。
✧ Code Composer Studio 集成式开发环境 (IDE) v5 的免费无限制版。
✧ 免费下载 controlSUITE ™ 软件及示例、库、应用软件等 。

更多详情 : http://www.ti.com.cn/tool/cn/launchxl-f28027

1.2 C2000 LaunchPad 原理图解读

C2000_Launchpad 实验板主要由 MCU 最小系统和 DSP 仿真器两部分电路组成，如图 1 所示，绿色线条以下部分是 MCU 最小系统，绿色线条以上部分是 DSP 仿真器 XDS100V2。

MCU 最小系统部分的供电，可以直接由板载仿真器供电，也可以独立供电实现仿真器和 MCU 最小系统之间电气隔离。如果 JP1、JP2 和 JP3 短路，MCU 最小系统和板载仿真器之间使用相同电源；如果 JP1、JP2 和 JP3 短路跳线断开，需要通过 J3 向最小系统板供电，这时实现仿真器和目标板之间电气隔离。

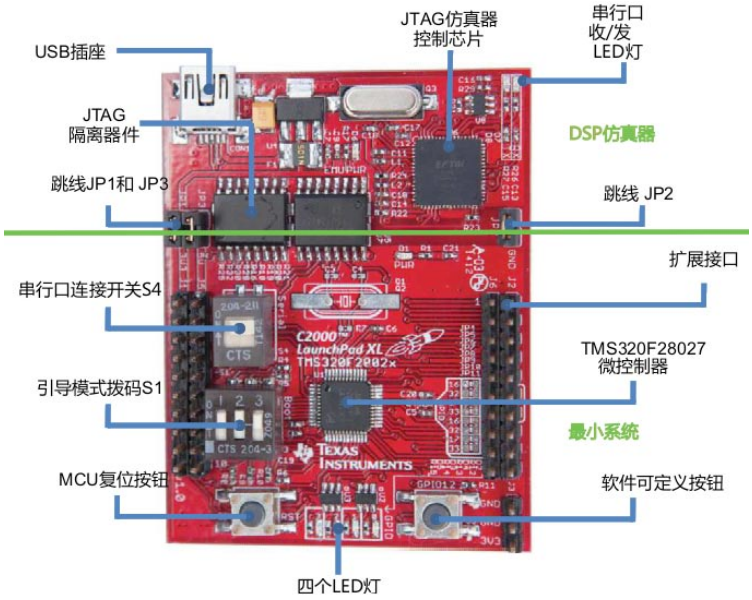


图1 . C2000_launchpad 实验板

一、TMS320F28027 最小系统

1) MCU 供电

TMS320F28027 要求电源电压 3.3V，电源引脚有 4PIN，分别是两路数字核电源，一路模拟核电源，一路 IO 电源，具体电源安排如图 2 所示，电源去藕电容尽量安排在电源引脚和它相邻的地之间，电流环路尽量小。

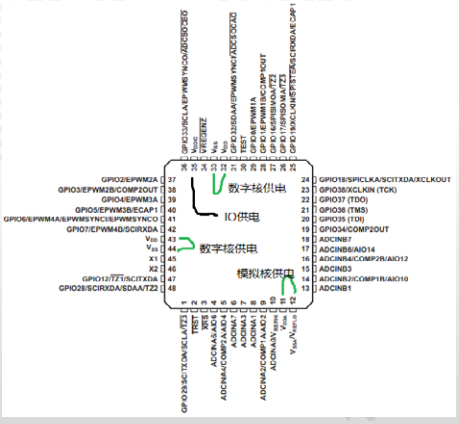


图 2. TMS320F28027 引脚分配图

实验板电源部分的电路如图 3 所示，模拟核要求较干净电源，设计时由 C19、C21 和 L3 对 +3V3 滤波之后再进入 MCU。

两路数字核电源，在 MCU 电源引脚附近分别放上 2.2UF 去藕电容，见图中 C5、C6。这两路电源在画图时有些小失误，从原理图中看不出 VDD1 和 VDD2 是连接到什么网络。

IO 电源则是由 C20、L4 构成 LC 网络对 +3V3 电源滤波得到。

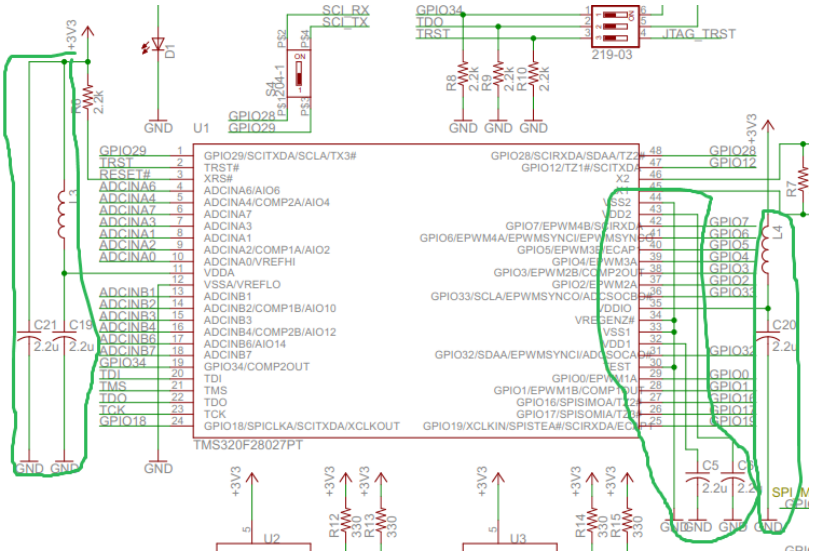


图 3

2) 复位电路

这块实验板关于 MCU 复位的处理非常简单，由 R6 和 S 构成手工复位电路，R6 还为 MCU 提供上电复位信号。

3) 时钟源

如图 4 所示，MCU 选用无源晶体为时钟源，C3、C4 作为晶体负载电容的一部分，需要根据晶体特性选择最佳的容量，R7 用于适当降低时钟电路的 Q 值，保持振荡器处于较合适工作点。

应该是出于安装方便的考虑，晶振 Q1 和 Q2 是一颗晶体的两种封装，实际使用只选用其中一种型号。

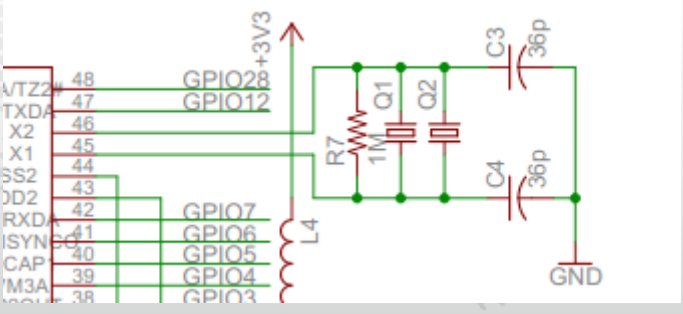


图 4

4) JTAG 调试接口

TMS320F28027 较其他 C2000 系列 MCU 在调试接口上作了一些简化，使用的调试信号减少为 5 个，此外允许将 JTAG 信号作为通用 IO 端口。

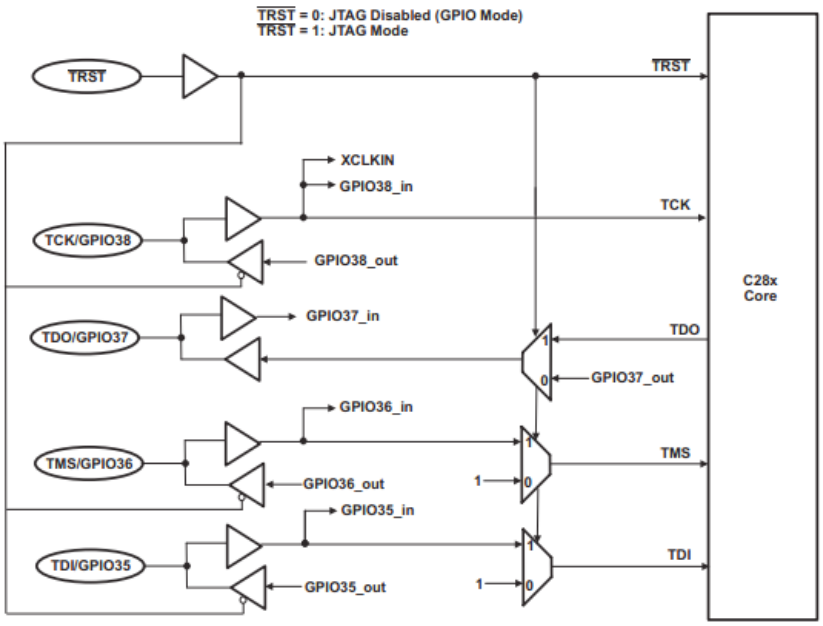


图 5

图 5 JTAG 接口信号定义

4) SCI 接口

实验板将一通道 SCI 接口连接到 XDS100V2 的 USB 虚拟串行口，该串行口直接在 PCB 板上连接，没有使用电平转换电路，直接使用 lvttl 电平进行连接。考虑到 SCI 由扩展接口引出作为其他用途，在 SCI 两根信号中串入带锁的微动开关，SCI 电路见图 6。

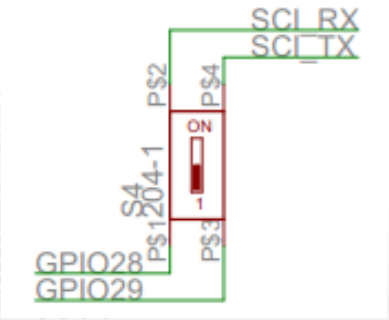


图 6. SCI 连接开关

5) 软件定义 LED

实验板有四个软件定义 LED，分别由 GPIO0、GPIO1、GPIO2 和 GPIO3 四个信号控制，见图7。

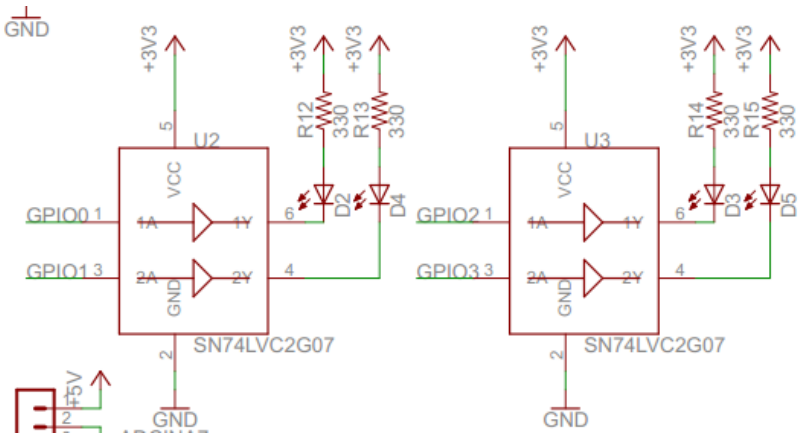


图 7. 软件定义 LED

6) 扩展接口

J1、J2、J5 和 J6 为扩展接口，各引脚定义见表格 1.

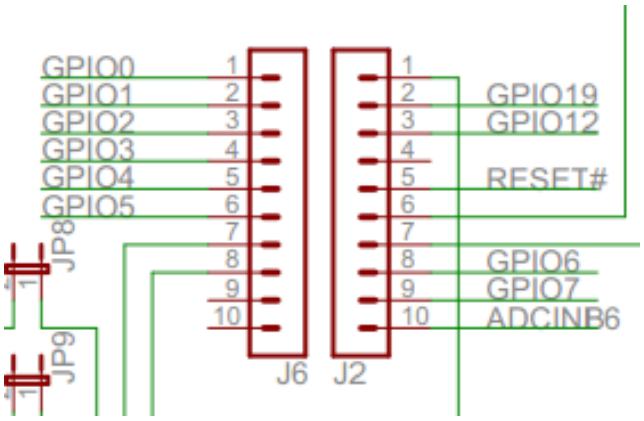


图 8 扩展接口 J2 和 J6

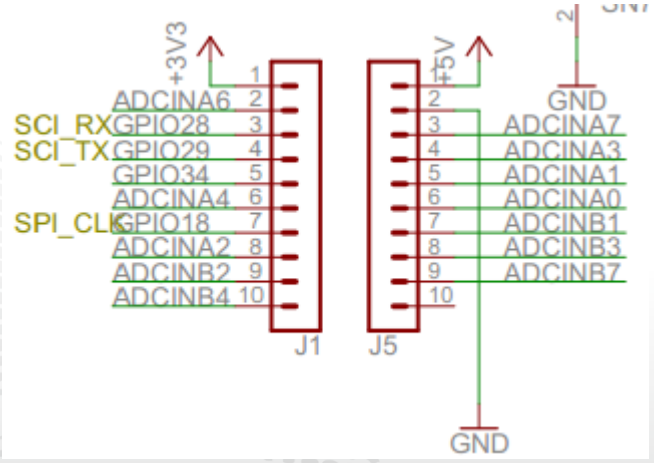


图 9. 扩展接口 J1 和 J5

Table 1. C2000 LaunchPad Pin Out and Pin Mux Options

Mux Value						Mux Value			
3	2	1	0	J1 Pin	J5 Pin	0	1	2	3
			+3.3V	1	1	+5V			
			ADCINA6	2	2	GND			
TZ2	SDAA	SCIRXDA	GPIO28	3	3	ADCINA7			
TZ3	SCLA	SCITXDA	GPIO29	4	4	ADCINA3			
Rsvd	Rsvd	COMP2OUT	GPIO34	5	5	ADCINA1			
			ADCINA4	6	6	ADCINA0			
	SCITXDA	SPICLK	GPIO18	7	7	ADCINB1			
			ADCINA2	8	8	ADCINB3			
			ADCINB2	9	9	ADCINB7			
			ADCINB4	10	10	NC			
3	2	1	0	J6 Pin	J2 Pin	0	1	2	3
Rsvd	Rsvd	EPWM1A	GPIO0	1	1	GND			
COMP1OUT	Rsvd	EPWM1B	GPIO1	2	2	GPIO19	SPISTEA	SCIRXDA	ECAP1
Rsvd	Rsvd	EPWM2A	GPIO2	3	3	GPIO12	TZ1	SCITXDA	Rsvd
COMP2OUT	Rsvd	EPWM2B	GPIO3	4	4	NC			
Rsvd	Rsvd	EPWM3A	GPIO4	5	5	RESET#			
ECAP1	Rsvd	EPWM3B	GPIO5	6	6	GPIO16/32	SPISIMOA/ SDAA	Rsvd/ EPWMSYNCI	TZ2/ ADCSOCA
TZ2/ ADCSOCA	Rsvd/ EPWMSYNCI	SPISIMOA/ SDAA	GPIO16/32	7	7	GPIO17/33	SPISOMIA/ SCLA	Rsvd/ EPWMSYNCO	TZ3/ ADCSOCB
TZ3/ ADCSOCB	Rsvd/ EPWMSYNCO	SPISOMIA/ SCLA	GPIO17/33	8	8	GPIO6	EPWM4A	EPWMSYNCI	EPWMSYNCO
			NC	9	9	GPIO7	EPWM4B	SCIRXDA	Rsvd
			NC	10	10	ADCINB6			

表 1. 扩展接口信号定义

7）其他功能

最小系统外接电源接口，见图 10.

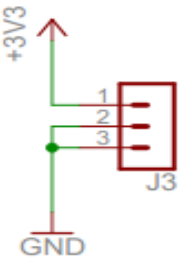


图 10

最小系统电源 LED 指示灯，见图 11.

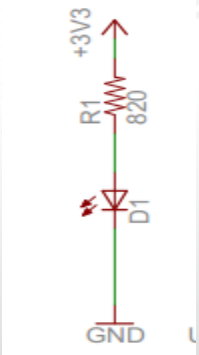


图 11. +3V3 电源 LED

软件定义输入按钮，见图 12。

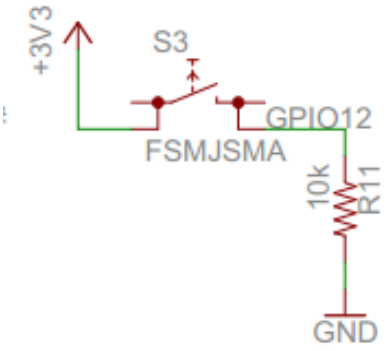


图 12. 软件定义输入按钮

二、XDS100V2 仿真器

1) 仿真器电源

仿真器是由 USB 总线供电，USB 总线 +5V 电源，经过 U4 降压为 +3.3V，为仿真器和目标板提供能量。（注，如果 JP1、JP2 和 JP3 断开，目标板电源需要另外提供）

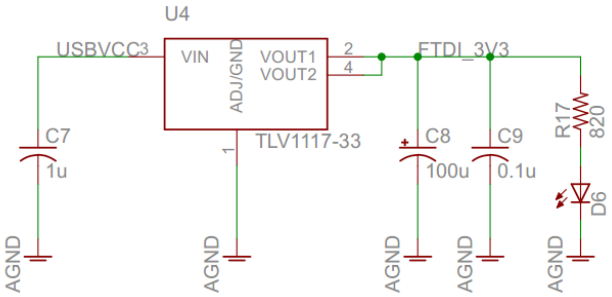


图 13. +3.3 线性电源

该电源电路见图 13 所示，C7 为输入电容、C8 和 C9 为输出电容，D6 作为电源输出指示灯，该电源最大输出电流 800mA。

2) FT2232H 电路

FT2232H 是这款 XDS100V2 仿真器的核心电路，其主要功能是实现 USB 到 UART 通讯协议的转换，FT2232H 有两通道 USB 转 UART，一通道用作仿真器的 JTAG 通讯，另一通道作为虚拟串行口作为自定义用途。

FT2232H 最小系统电路如图 14 所示，电路采用总线供电方式，省去外接电源的麻烦。

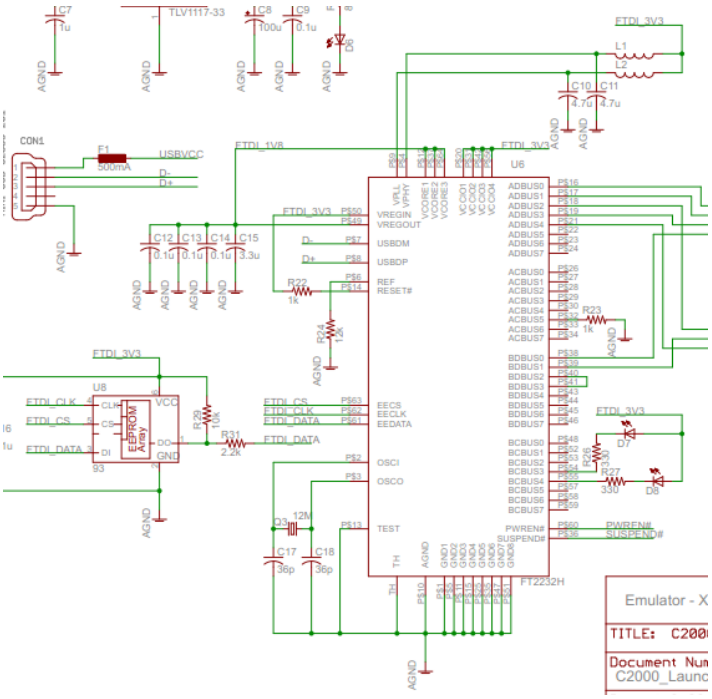


图 14. FT2232H 电路

3) 信号隔离

实验板支持仿真器和目标板之间电气信号的隔离，如图 15 所示，ISO7240 和 ISO7231 为信号隔离器件。

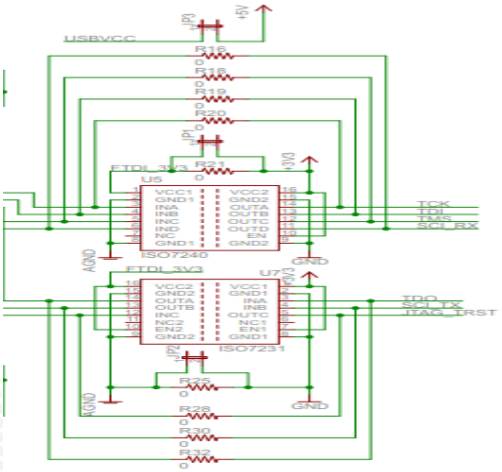


图 15. 仿真器和目标板之间信号隔离

更多详情：http://bbs.eeworld.com.cn/thread-373788-1-1.html

1.3 C2000 LaunchPad 硬件组成及使用

一、概述

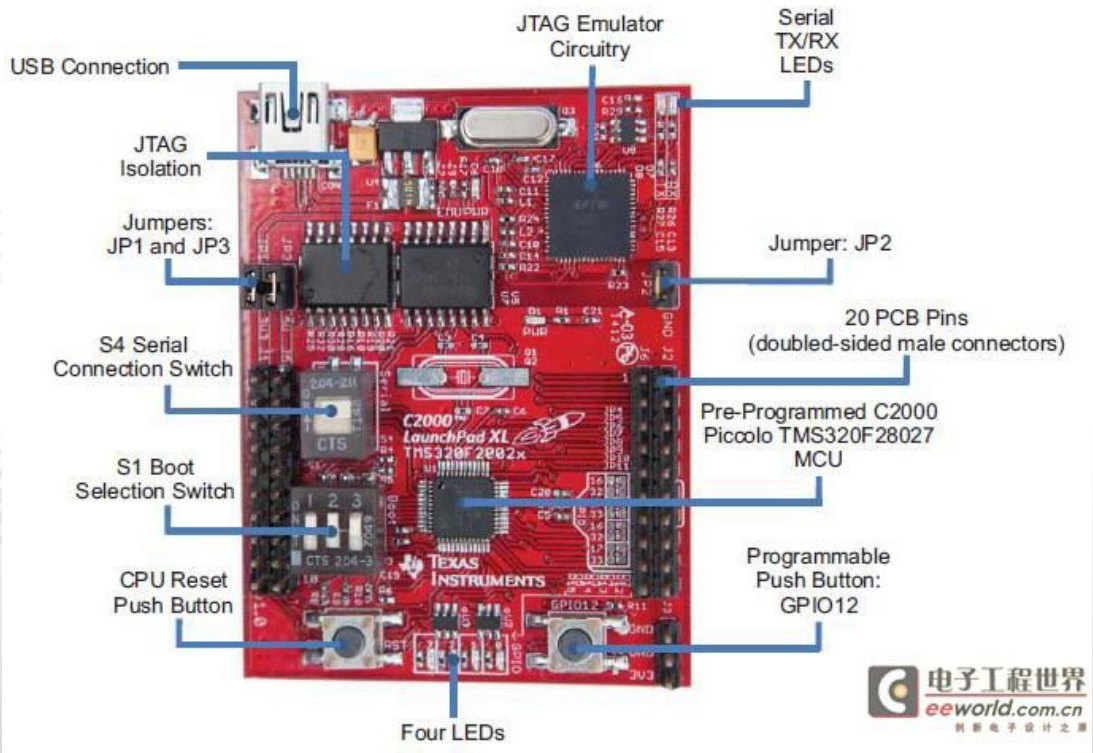
C2000 LaunchPad，LAUNCHXL-F28027，是 TI 使用 Piccolo F2802x 器件制作的一个完备的低成本的实验板。F28027 LAUNCHXL 平台，包括了 F28027 的所需要硬件及软件。 LaunchPad 是基于 F28027 CPU 的，也可以让用户移植为成本较低的其它 F2802x 器件。

更具特色的是,它提供了一个板载 JTAG 仿真工具,可以使用 USB 与电脑相接，易于编程、调试和仿真。除了 JTAG 仿真外，USB 接口还虚拟一 UART 串行口连接到 PC 主机。

用户可以下载无限制版的 Code Composer Studio V5，作为它的编译工具。

C2000 的 LaunchPad 功能包括：

• USB 调试和编程接口，通过一个高速的、带隔离的 XDS100v2 仿真器，USB/ UART 连接。
• F28027 CPU，
• 四个 LED 指示。
• 两个按钮，用户输入和器件复位。
• CPU 引脚均用插针引出，用它可以添加用户定制的扩展板。
• 启动方法选择。USB 和 UART 连接开关



二、C2000 LaunchPad 的安装

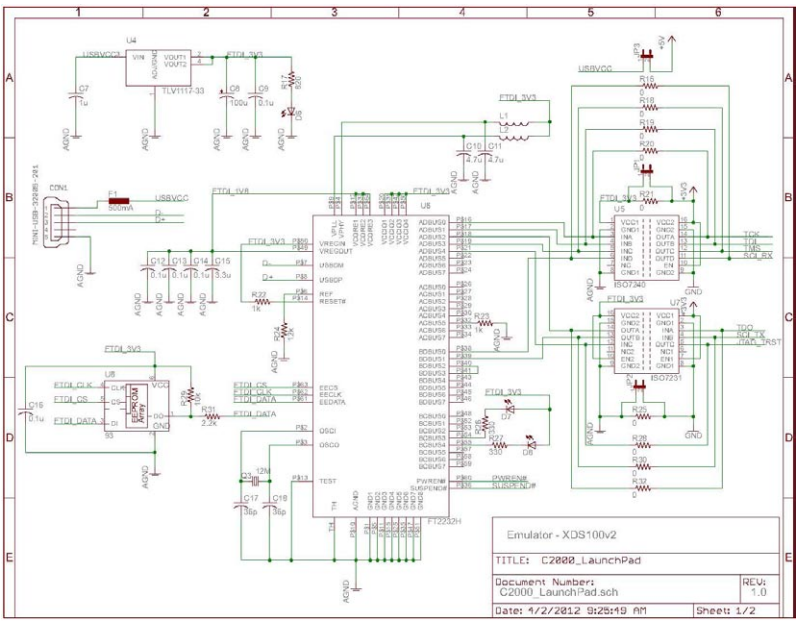
1、下载所需要文档可以在这里 http://www.ti.com/c2000-launchpad 找到所需要的文档。

2、安装软件安装 CCS V5 及 controlSUITE。controlSUITE 包含了 C2000 的帮助文档及例程。

3、硬件安装

当安装安 CCS 后，将 LaunchPad 通过 USB 电缆与电脑相连，windows 系统将自动找到硬件并配置。

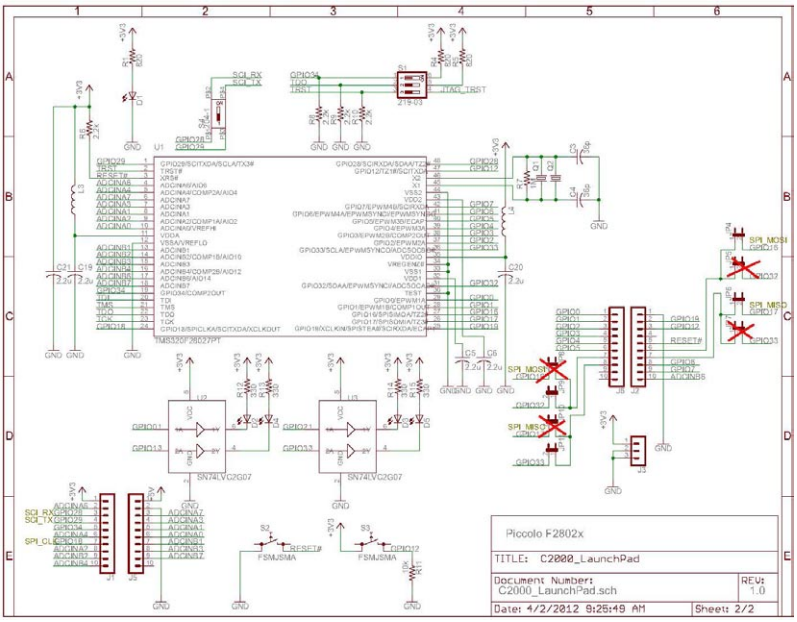
三、电路原理及注意事项仿真器部分



仿真器为 XDS100V2，使用 FT2232H 完成所有功能。电路在使用了 U5、U7 作为隔离，因为 F28027 常作为电源控制使用，TI 的例程中就使用它作双相 PFC 控制，这种情况下就显得特别方便。FT2232H 在作仿真器的同进，还完成一路 UART，也经过 U5、U7 隔离输出。D7 用于 UART 连接指示，D8 用于仿真器的连接指示。

JP1~3 用于供电选择，如果目标板（F28027 部分）有供电电路，则将其断开。

F28027 部分。



图中，J1、J2、J5、J6 把 F28027 的 IO 能功能脚引出，以方便接入其它电路。S1 为启动模式模式选择开关

Switch	Function
1	GPIO34
2	GPIO37
3	TRSTn

一般应用时，将其置 ON 位置。D2~D5 为接于 GPIO0~3 的 LED，并由 U2、U3 驱动。这样，在 GPIO0~3 作其它用途时，D2~D5 的存在不影响端口的功能。J4~J11，为 PCB 上的连线，处于连接状态，用于确定 GPIO16、17、32、33 接于插针上的位置，如果这些 IO 使用时，一定要将 J4~J11 中的某些连接断开，如图中是本人的更改方法。

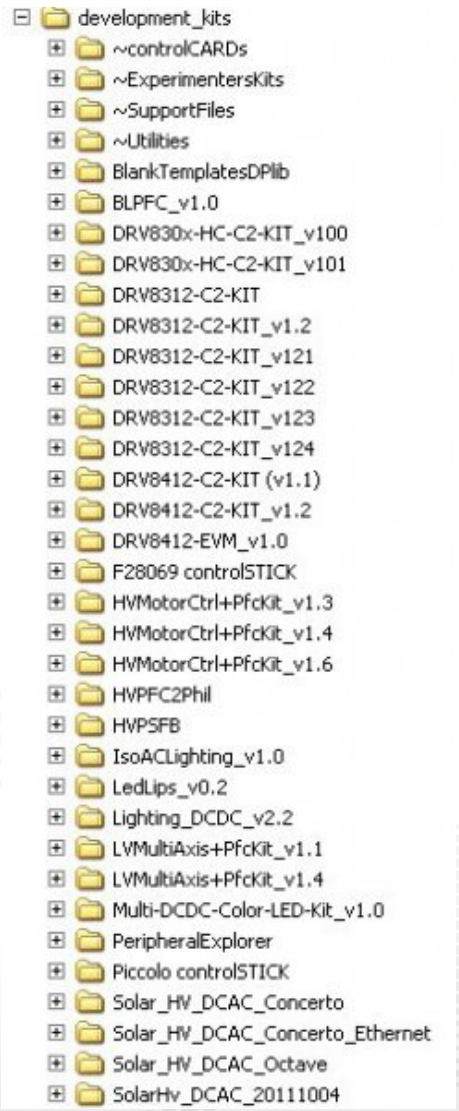
板上的 Q1、Q2 为所留的外接晶体，如果使用外部分时钟时，可以接上晶体谐振器。

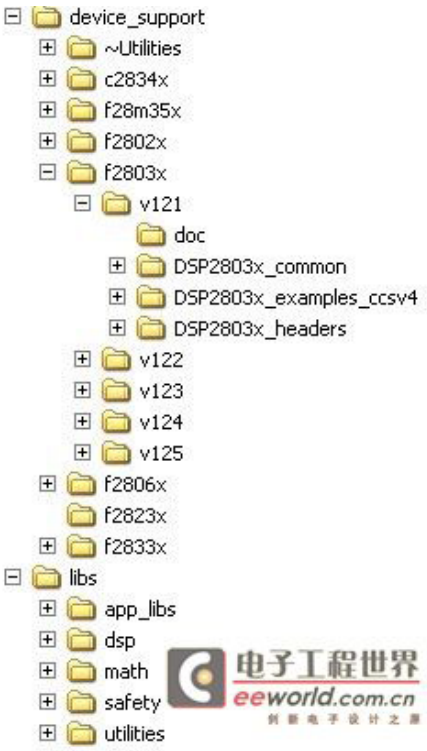
更多详情：http://bbs.eeworld.com.cn/thread-373553-1-1.html

1.4 软件 & 驱动安装之强大的 controlSUITE

controlSUITE 软件是 TI 针对 C2000 推出的可支持实时控制应用的软件，相较于传统的 MCU，controlSUITE 软件可为简化评估、应用调适、除错、测试及重复使用，提供内容及必要的内容管理。除包含一般免费软件产品常见的简单范例外，还为马达控制等应用，提供可做为真正开发系统使用的全面开放原始码数据库与范例。此外，全新安装程序还可消除版本更新与依赖性等问题，并让开发人员能够于同一地方获得完整的软件产品。

通过 controlSUITE，工程师可以获得 C2000 全套的、开源的软硬件资料。以下为升级后的目录参考，大家可以下载此软件后，自动在线升级。





更多详情 : http://bbs.eeworld.com.cn/thread-315773-1-1.html

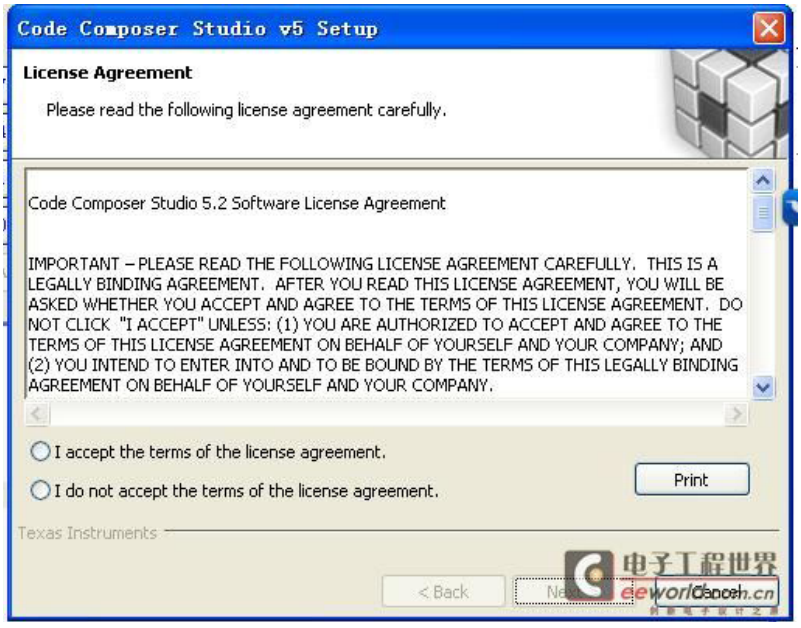
1.5 C2000 LaunchPad 之 CCS5 安装

我电脑本来装了 CCSMSP430 的为了装 C2000 只好，把它先卸了。

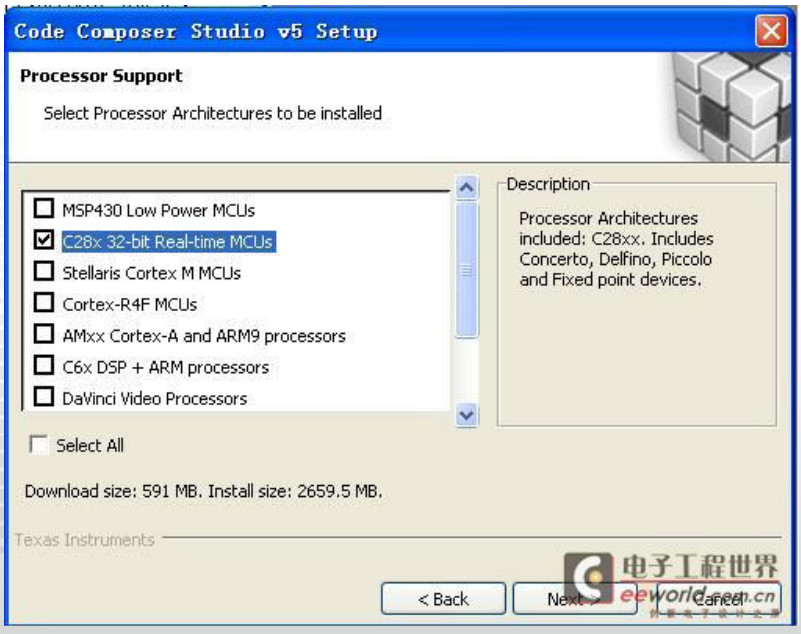
CCS5 是从 TI 网站下的，大小 1.2G 的可本地安装的那种。

安装时，先查一下本地有没有 TI 子目录，有的话，若没有用最好统统删掉，以免发生不必要的错误。再有 C 盘最好用 360 或金山工具之类给清一下垃圾。

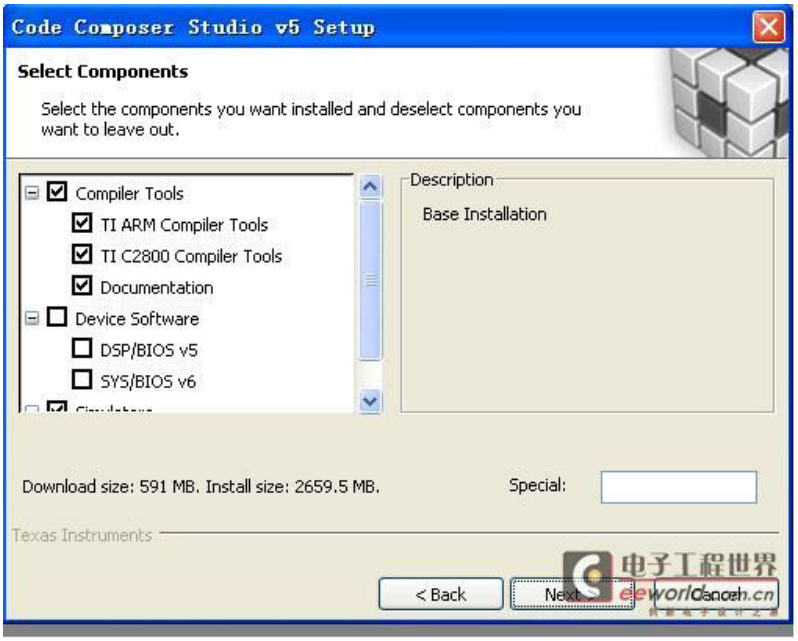
下面开始第一步：



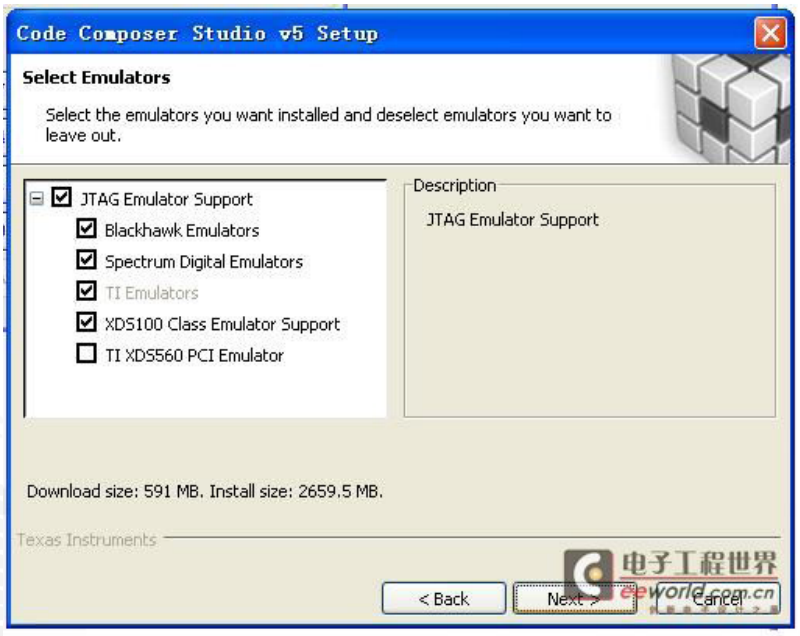
第二步:这步关键装 C2000 时一定要选这项，别的我先不装。因为时间太长。耗不起时间。



第三步：这里默认就行



第四步：也默认



完后等着完成就行。

软件的激活。其实坛子里有破解文件。在这里为了方便再传一下破解文件。

许可证文件，拷贝到 /ccsv5/ccs_base/DebugServer/license 下面，重新运行 CCS 就行。

更多详情：http://bbs.eeworld.com.cn/thread-361212-1-1.html

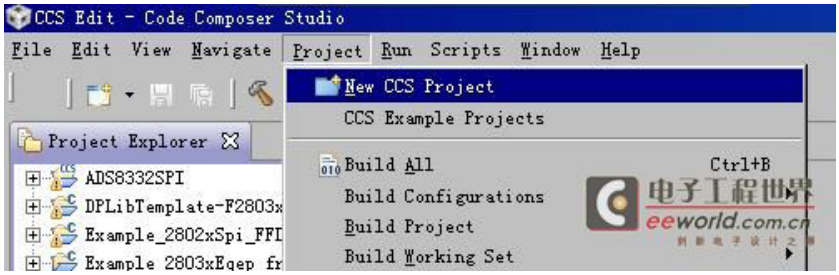
1.6 打造自己的 C2000 LaunchPad 项目

TI 的例程中，V200 以后内部资源的控制，是以库的形式给出：driverlib.lib，对库的使用，有说明文档 f2802x-DRL-UG.pdf。

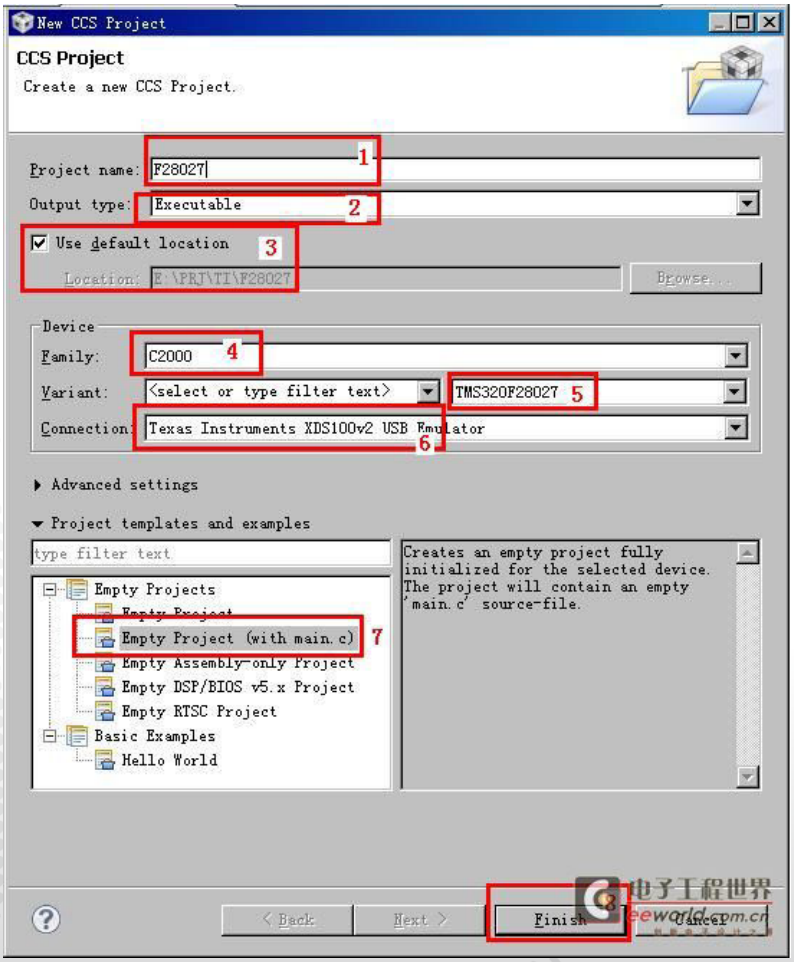
本项目不使用它的 LIB 文件，这样的好处是，对每一个函数的调用直接明了，便于没有经验者学习。

一、新建项目。

点 project 菜单。选择 New CCS Project，



弹出 New CCS Project 对话框：



1）起个项目名称，

2）选择 Executable，

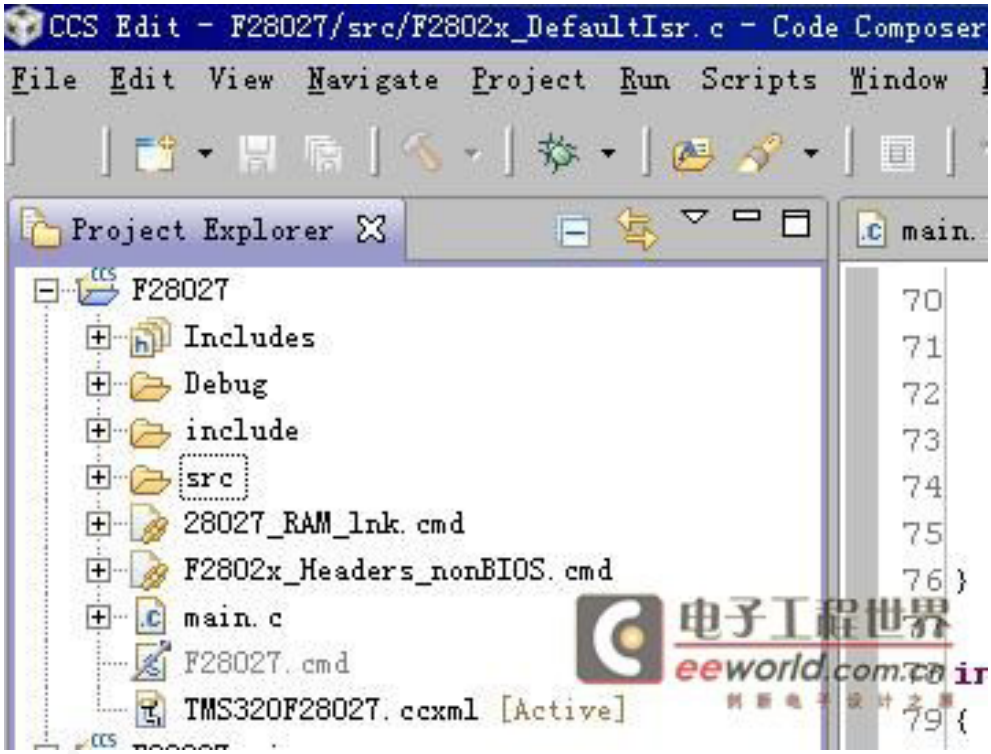
3）选择约定项目目录，或自己选择目录。

4）选择使用的芯片

5）仿真调试器类型

6）项目类型。

此时项目中包含了这几个文件：

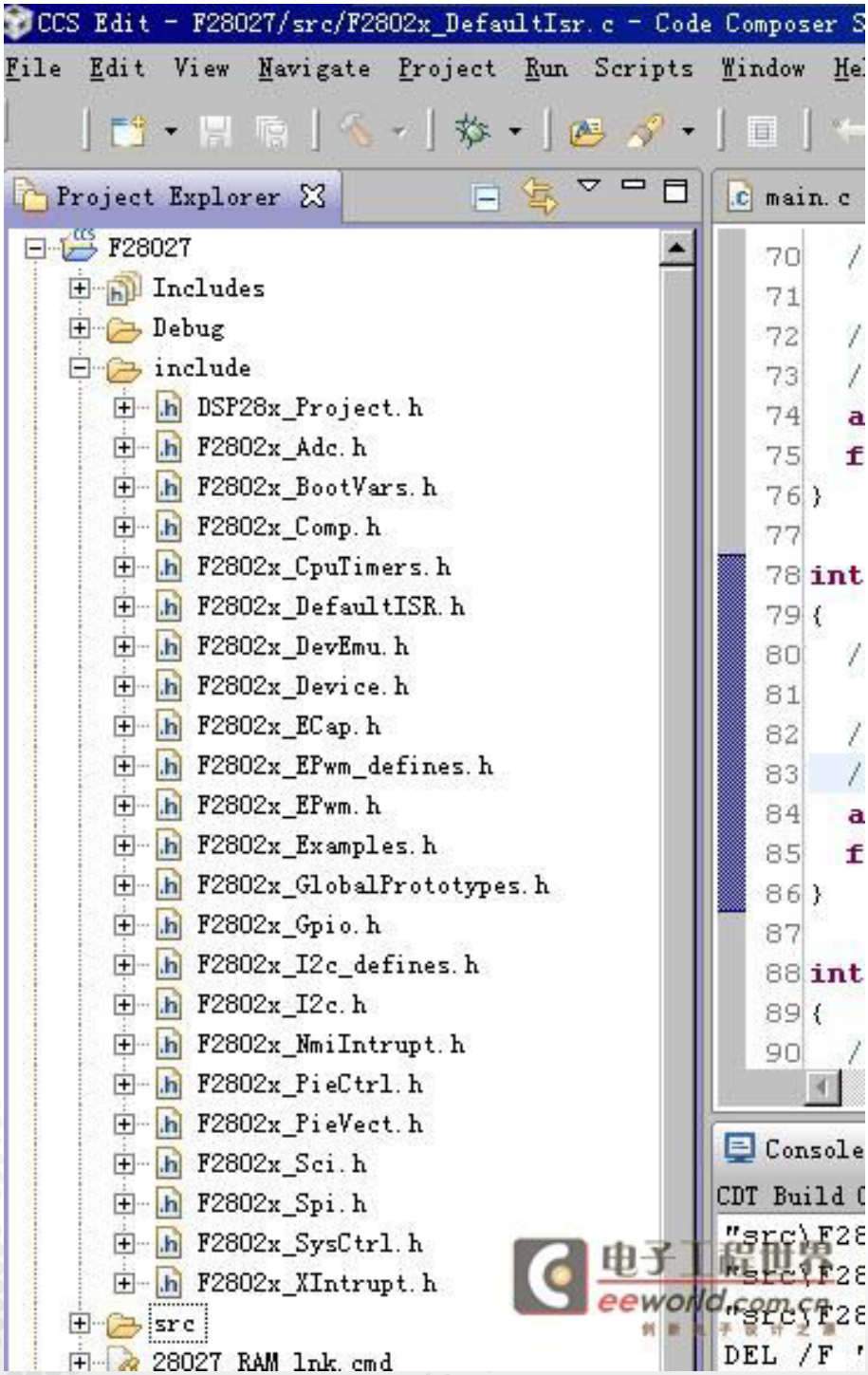


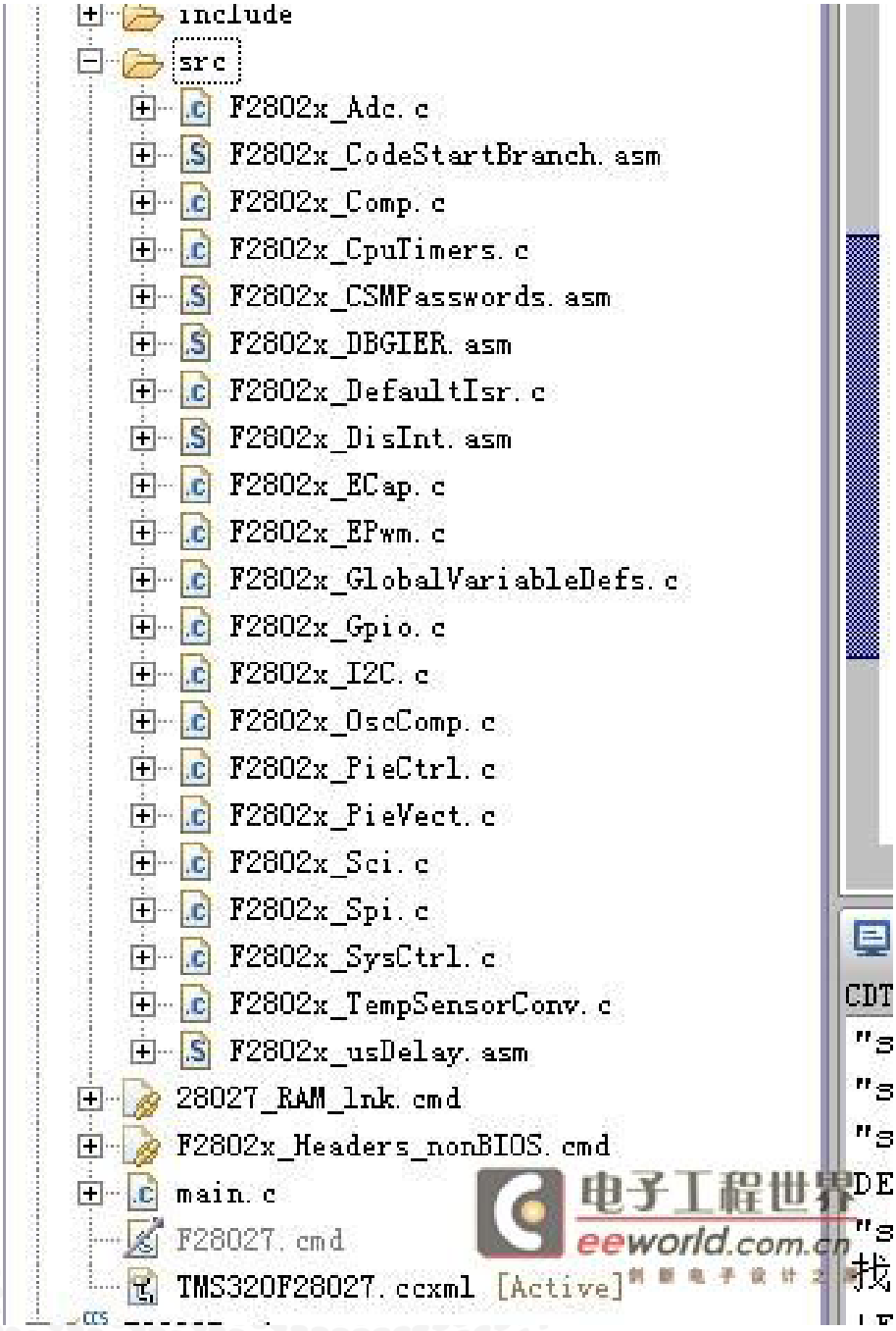
二、拷贝文件

将 C:\ti\controlSUITE\device_support\f2802x\v210 中的部分文件拷贝到项目目录中，此路径是安装 TI 库时的约定目录，如果重新指定，将按指定路径。

为了管理方便，在项目目录下建立 include、src 两个文件夹

拷贝后的文件有：


CCS Edit - F28027/src/F2802x_DefaultIsr.c - Code Composer S
File Edit View Navigate Project Run Scripts Window He
Project Explorer
F28027
Includes
Debug
include
DSP28x_Project.h
F2802x_Adc.h
F2802x_BootVars.h
F2802x_Comp.h
F2802x_CpuTimers.h
F2802x_DefaultISR.h
F2802x_DevEmu.h
F2802x_Device.h
F2802x_ECap.h
F2802x_EPwm_defines.h
F2802x_EPwm.h
F2802x_Examples.h
F2802x_GlobalPrototypes.h
F2802x_Gpio.h
F2802x_I2c_defines.h
F2802x_I2c.h
F2802x_NmiIntrupt.h
F2802x_PieCtrl.h
F2802x_PieVect.h
F2802x_Sci.h
F2802x_Spi.h
F2802x_SysCtrl.h
F2802x_XIntrupt.h
src
28027_RAM_lnk.cmd
main.c
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
Console
CDT Build
DEL /F



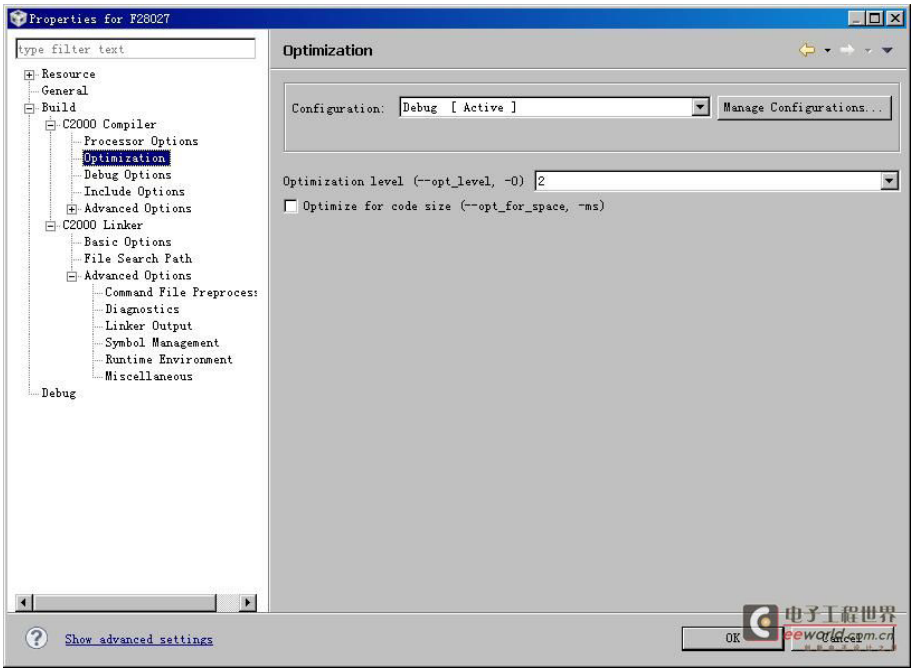
V210 目录下，有带“F2802x_”的文件和不带的。不带“F2802x_”的文件是使用库 driverlib.lib 时所用的。所以，本次拷贝时不拷贝这些文件。

三、设置

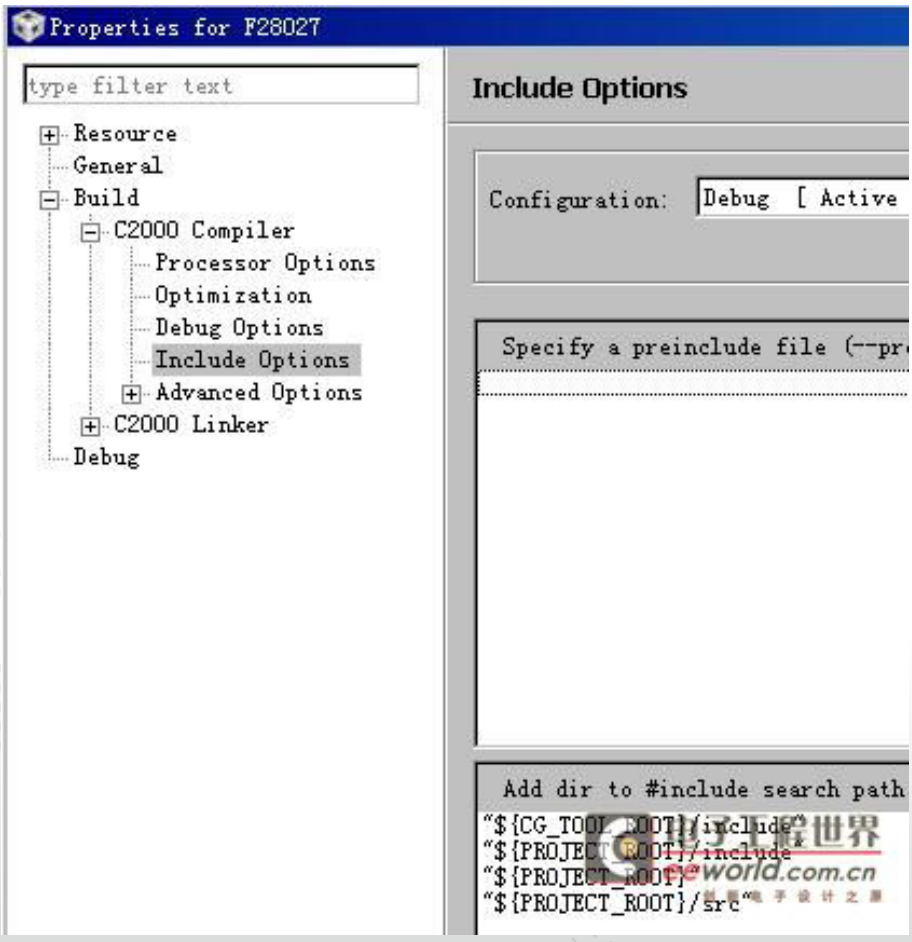
1>、属性对话框设置

右击项目名称，选择 Properties，弹出属性对话框，

设置优化


Properties for F28027
type filter text
Resource
General
Build
C2000 Compiler
Processor Options
Optimization
Debug Options
Include Options
Advanced Options
C2000 Linker
Basic Options
File Search Path
Advanced Options
Command File Preprocess
Diagnostics
Linker Output
Symbol Management
Runtime Environment
Miscellaneous
Debug
Optimization
Configuration: Debug [Active]
Manage Configurations...
Optimization level (--opt_level, -O) 2
Optimize for code size (--opt_for_space, -ms)
Show advanced settings
OK

再设置一下路径：


Properties for F28027
type filter text
Resource
General
Build
C2000 Compiler
Processor Options
Optimization
Debug Options
Include Options
Advanced Options
C2000 Linker
Debug
Include Options
Configuration: Debug [Active
Specify a preinclude file (--pr
Add dir to #include search path
"${CG_TOOL_ROOT}/include"
"${PROJECT_ROOT}/include"
"${PROJECT_ROOT}"
"${PROJECT_ROOT}/src"

剩余的不设置，暂时使用缺省值。

2>、文件设置

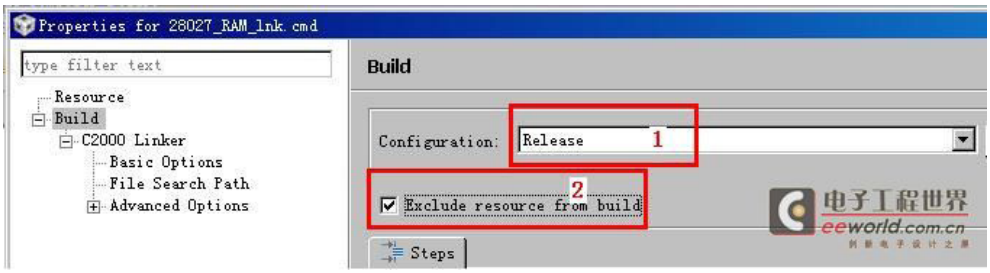
项目中有三个 CMD 文件：F2802x_Headers_nonBIOS.cmd，是非 BIOS 下的寄存器定义文件；28027_RAM_lnk.cmd 是内容 RAM 分配及 IQ 表；F28027.cmd 是 flash 分配文件。

当在 RAM 中调试时，需要用到 F2802x_Headers_nonBIOS.cmd 及 28027_RAM_lnk.cmd

当在 FLASH 中调试或烧写时，需要用到 F2802x_Headers_nonBIOS.cmd 及 F28027.cmd

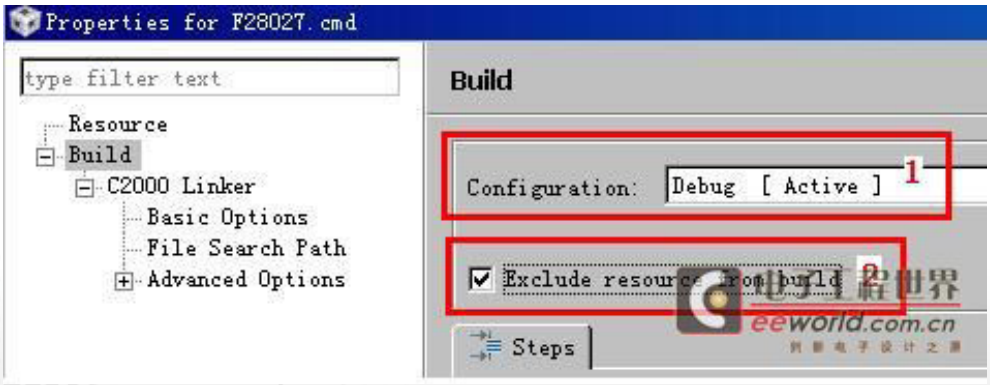
具体设置如下：

1）在项目窗口右击 28027_RAM_lnk.cmd 文件，选择 Properties 弹出 Properties 对话框：



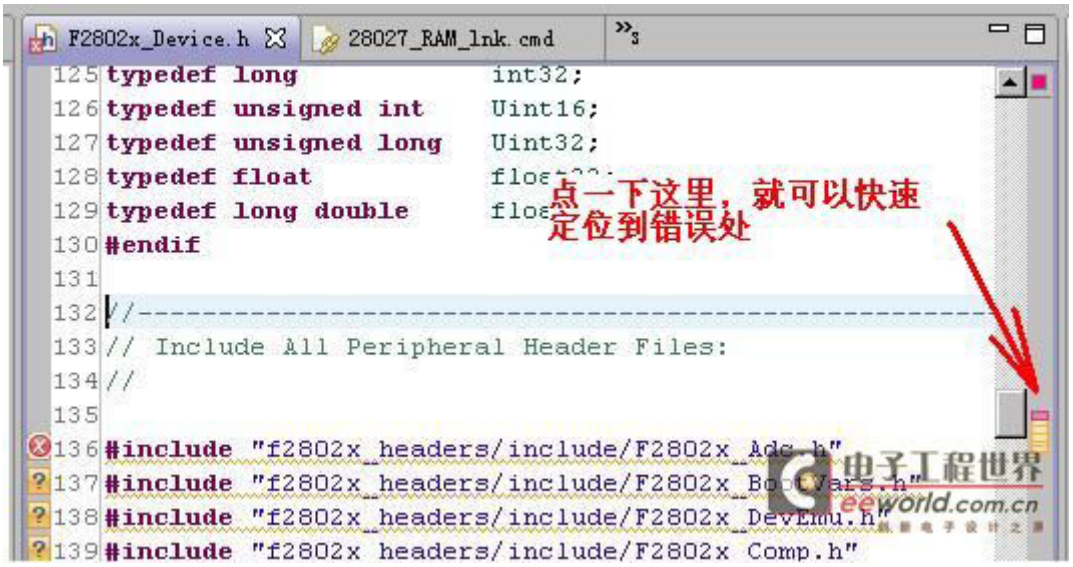
这里约定，在 RAM 中调试是 DEBUG 模式，烧入 FLASH 中是 Release 模式。

2）同样方法设置 F28027.cmd 文件

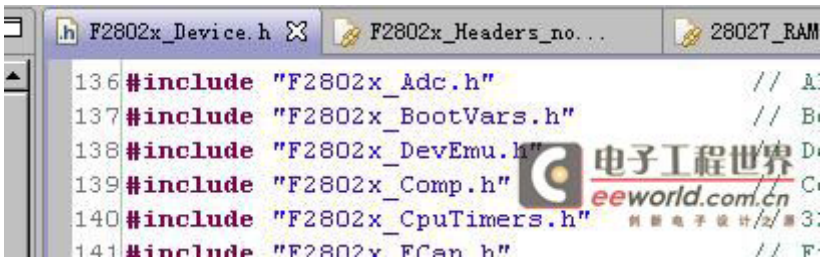


四、文件修改

此时编译一下，发现 F2802x_Device.h 等文件有错误，

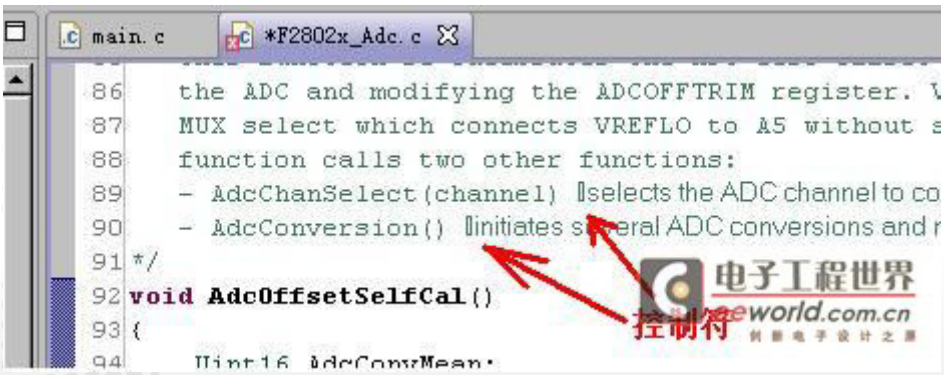


是因为目录指定问题，把它改为：



照此方法，将其它文件中几指定的目录都将它去掉。

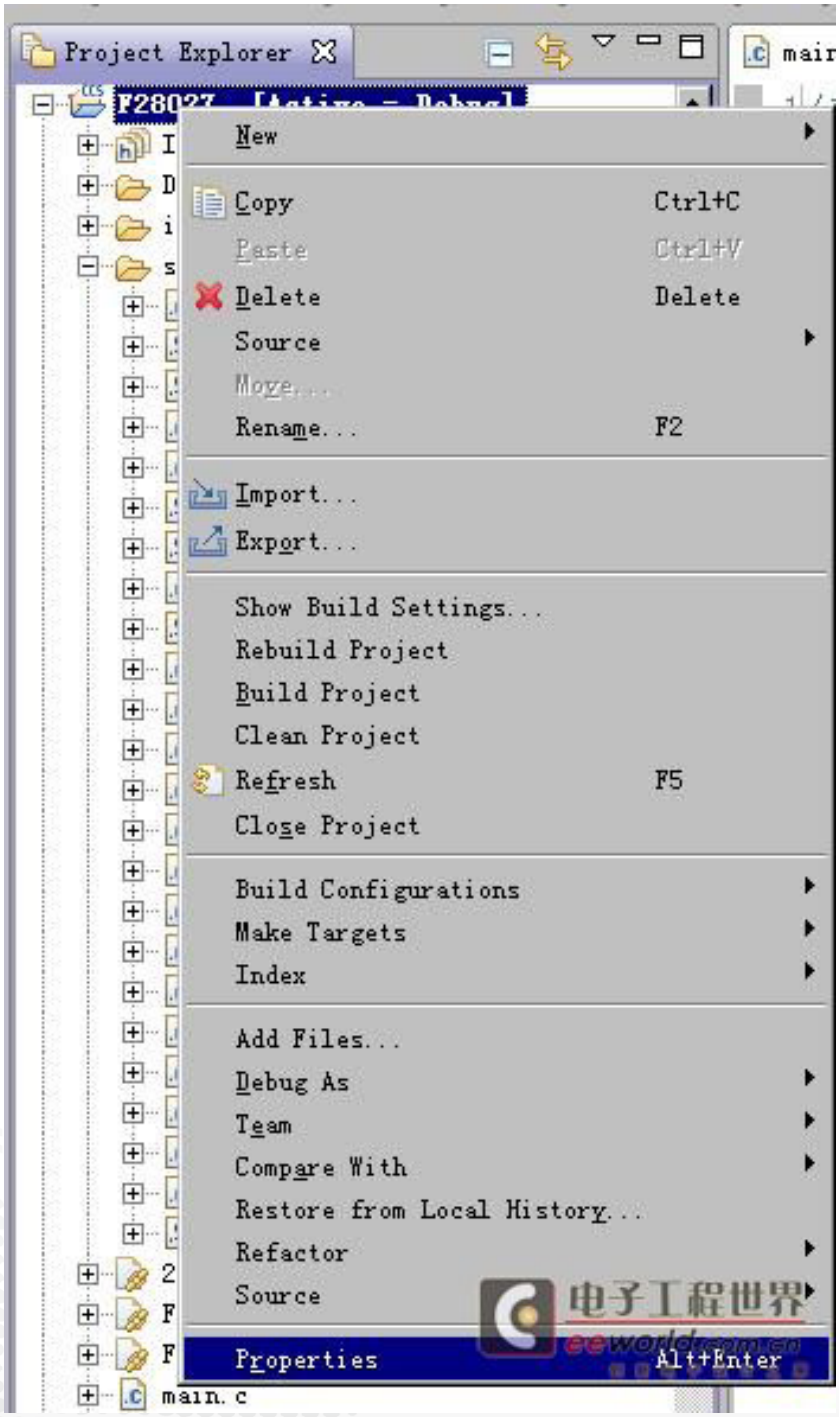
修改中，有的文件修改后就不能存盘，这是由于 TI 专门将文件中插入不可显示的控制符造成的(用 WinHEX 将它打开就可以看到它的 ASCII 值了) 如:



将它删除就可以存盘了。

最后编译一下，通过！

此时的项目还是一个空项目，只是 F28027 内部所有模块的基础文件。



更多详情：http://bbs.eeworld.com.cn/thread-360489-1-1.html

1.7 C2000 LaunchPad 在 FLASH 里运行

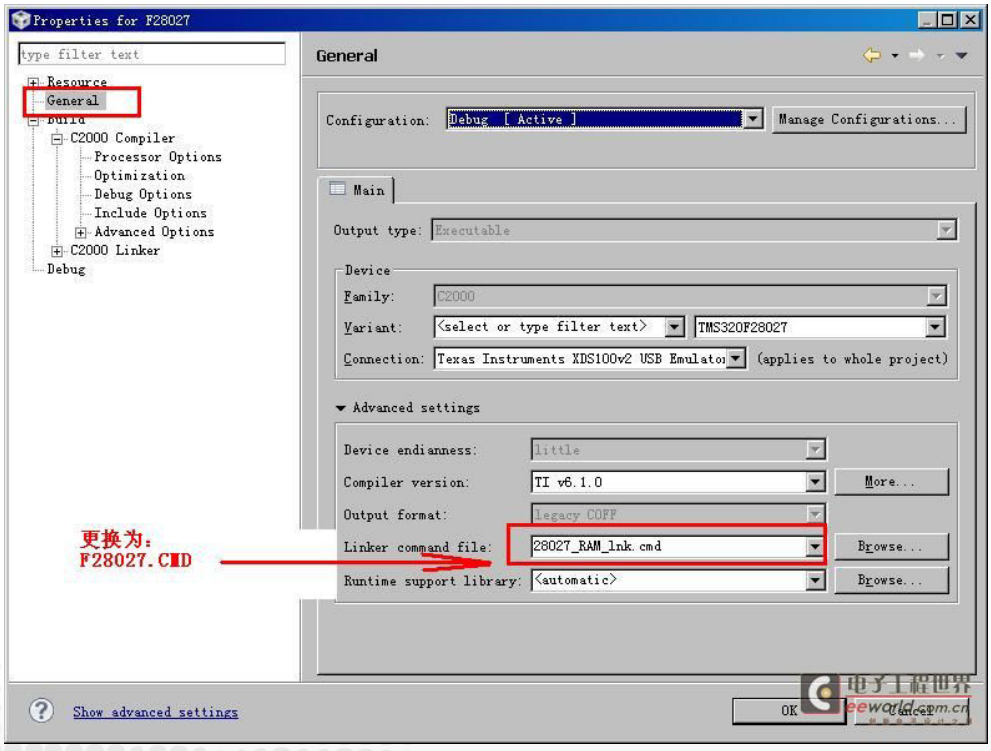
在 FLASH 里运行程序，只是将程序烧写到 FLASH 中，与在 RAM 里运行没有本质的区别。但是在 FLASH 里运行需要设置等待周期，对于对时间要求严格的过程，需要拷贝到 RAM 中运行，以达到最高速度。这样，如果对程序处理不当，就有可能使在 RAM 中运行正常的程序烧入 FLASH 中后就无法正常运行。关于这方面的内容需要看这两个:《TMS320C28x Optimizing C/C++ Compiler User's Guide》、《TMS320C28x Assembly Language Tools User's Guide》及其它相关文件。

下面在前几节实例的基础上，向大家介绍如何在 FLASH 里运行程序。

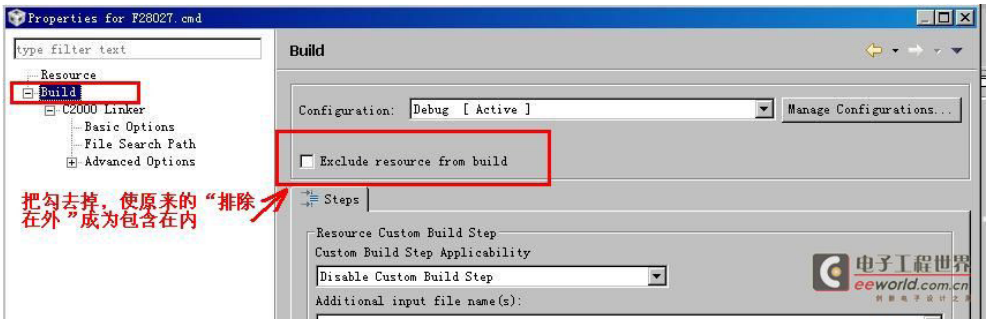
一、更换 CMD 文件

1、使用 F28027.COM 文件。

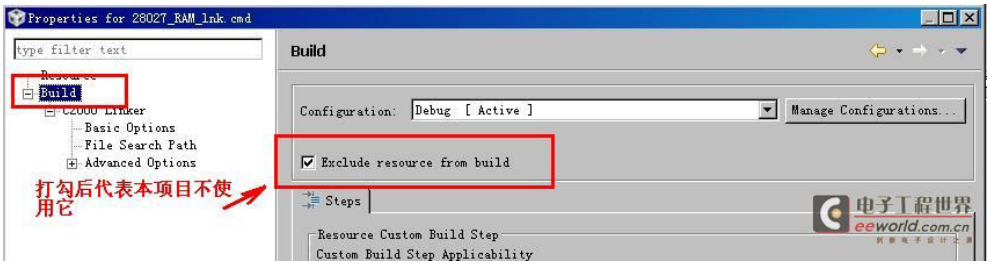
在 Project Explorer 窗口右击项目名称，在出现的右键菜单里选择 Properties，更换 CMD 文件方法如下:



也可以在 Project Explorer 窗口右击 F28027.COM 文件，在出现的右键菜单里选择 Properties，将这个文件包含在工程中:



2、去掉 28027_RAM_lnk.cmd 文件。右击 28027_RAM_lnk.cmd 在出现的右键菜单里选择 Properties，并将它打上勾：



3、F2802x_Headers_nonBIOS.cmd 不管在什么情况下都是要用的。它是定义内部特殊寄存器用的，如 CPU 定时器寄存器。这些是固定不变的。

二、将需要在 RAM 中运行的过程拷贝到 RAM 中。

首先定义哪些过程（函数）需要在 RAM 里运行，然后使用 MemCopy（ ）将它们搬到 RAM 中。

1、定义 ramfuncs 的过程

这里的关键字“ramfuncs”是在 CMD 文件中定义的：

```
ramfuncs            : LOAD = FLASHA,
                      RUN = PRAML0,
                      LOAD_START(_RamfuncsLoadStart),
                      LOAD_END (_RamfuncsLoadEnd),
/*                    LOAD_SIZE(_RamfuncsLoadSize),*/
                      RUN_START(_RamfuncsRunStart),
                      PAGE = 0
```

由上可见，在烧写时，将定义的 ramfuncs 部分写在 FLASHA 里，系统运行时再拷贝到 PRAML0 里运行，这里的 RamfuncsLoadStart 等，是由编译器生成的。在文件的开头或包含的头文件中需要加上声明：

```
extern Uint16 RamfuncsLoadStart;

extern Uint16 RamfuncsLoadEnd;

extern Uint16 RamfuncsRunStart;
```

定义 ramfuncs 的过程是使用 #pragma 宏指令：

```
#pragma CODE_SECTION(SCITXINTA_ISR,"ramfuncs");

#pragma CODE_SECTION(SCIRXINTA_ISR,"ramfuncs");
```

以上两行代表将 SCITXINTA_ISR( ) 及 SCIRXINTA_ISR( ) 作为 ramfuncs。对于其它需要指定 ramfuncs 的过程，参照此方法。

注意：使用 #pragma 宏指令，要在该过程所在的文件中使用，否则无效。可以在 MAP 文件中查看。

以上两行如果不在当前文件中使用，而放在主文件中时，map 文件是这样的

```
ramfuncs    0    003f6112    00000021    RUN ADDR = 00008000

                 003f6112  0000001d F2802x_SysCtrl.obj (ramfuncs)

                 003f612f  00000004   F2802x_usDelay.obj (ramfuncs)
```

如果把这两行放在过程所在的文件中时，MAP 文件是这样的：

```
ramfuncs    0    003f6000    00000061    RUN ADDR = 00008000

                 003f6000  00000040    F2802x_DefaultIsr.obj(ramfuncs:retain)

                 003f6040  0000001d    F2802x_SysCtrl.obj (ramfuncs)

                 003f605d  00000004    F2802x_usDelay.obj (ramfuncs)
```

对于使用汇编语言编制的程序则使用 sect 命令如 F2802x_usDelay( )过程:

```
        .def _DSP28x_usDelay
        .sect" ramfuncs"
        .global  _ _DSP28x_usDelay
_DSP28x_usDelay:
        SUB     ACC,#1
        BF      _DSP28x_usDelay,GEQ   ;; Loop if ACC >= 0
        LRETR
```

此时的 sect 命令的使用，也是在过程所在的当前文件中。

2、将过程从 FLASH 拷贝到 RAM 中。

使用以下方法：

```
MemCopy(&RamfuncsLoadStart, &RamfuncsLoadEnd, &RamfuncsRunStart);
```

原形如下：

```
void MemCopy(Uint16 *SourceAddr, Uint16* SourceEndAddr, Uint16*
DestAddr)
    {
        while(SourceAddr < SourceEndAddr)
         {
            *DestAddr++ = *SourceAddr++;
         }
        return;
    }
```

3、初始化 FLASH，设置等待时钟数。此函数使用 TI 提供的，不需要更改。

```
void InitFlash(void)
{
   EALLOW;
   //Enable Flash Pipeline mode to improve performance
```

```
    //of code executed from Flash.
    FlashRegs.FOPT.bit.ENPIPE = 1;
    //Set the Paged Waitstate for the Flash
    FlashRegs.FBANKWAIT.bit.PAGEWAIT = 2;
    //Set the Random Waitstate for the Flash
    FlashRegs.FBANKWAIT.bit.RANDWAIT = 2;
    //Set the Waitstate for the OTP
    FlashRegs.FOTPWAIT.bit.OTPWAIT = 2;
    FlashRegs.FSTDBYWAIT.bit.STDBYWAIT = 0x01FF;
    FlashRegs.FACTIVEWAIT.bit.ACTIVEWAIT = 0x01FF;
    EDIS;
    asm(" RPT #7 || NOP");
}
```

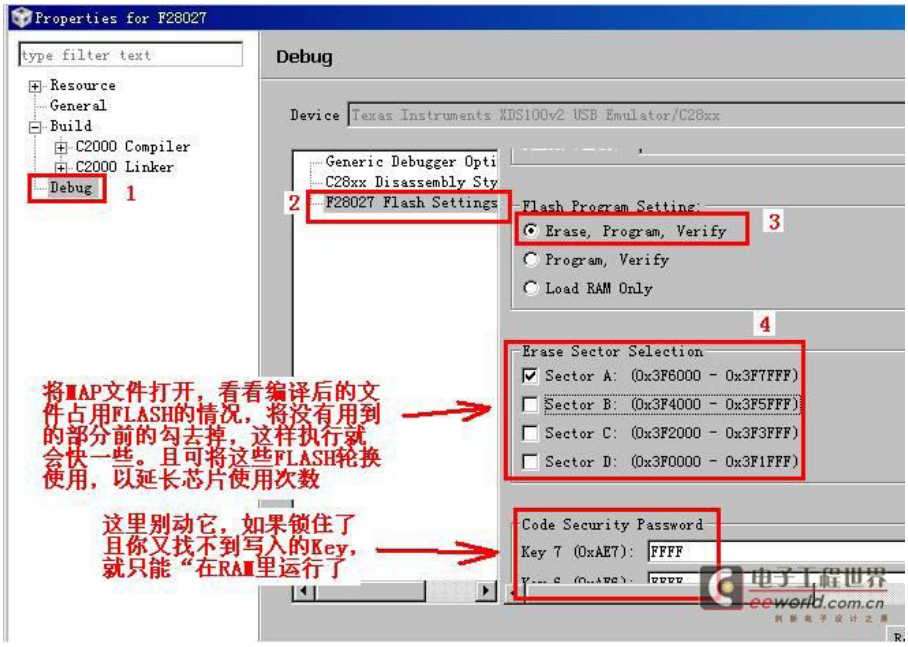
三、两个 CMD 文件比较

28027_RAM_lnk.cmd 文件的 PAGE0 是由 RAM 组成的。PAGE0 用来放置 .cinit、.pinit、 .text 以及固定表格等，就象一般单片机的 ROM 一样。在这个 CMD 文件里，把 RAM 看作是 ROM。

F28027.cmd 的 PAGE0 是这样的：

四、烧写

在 Project Explorer 窗口右击项目名称，在出现的右键菜单里选择 Properties ，



将项目重新编译，并进入 DEBUG，此时即将程序写入内部的 FLASH。

如果退出 DEBUG，板子重新加电后，仍能按照调试的结果运行。

更多详情：http://bbs.eeworld.com.cn/thread-362543-1-1.html

第二章 进阶篇—TI FAE 与你面对面

2.1 课程前言

“带有 DSP 灵魂的 MCU 系列产品”——这句话用来形容 TI C2000 似乎再形象不过，既具有 DSP 强大的处理能力，又具有 MCU 系统集成的优势，也难怪很多工程师观念中仍然把它归结为 DSP。

C2000 系列产品很多被广泛用于数字电机驱动、数字电源控制、新能源、等工业、家电、汽车方面的应用。C2000 Launchpad 开发板是 TI 2012 年推出的一款非常超值的针对 TMS320F2802x0 系列 MCU 的开发板。这一次，由 TI 资深工程师罗焰明亲自操刀，讲述 C2000 应用入门技术，并有丰富的针对基于 Launchpad 的 C2000 例程设计实验讲解及互动。

本培训课程包含六个部分，

1、TI C2000 C28X 架构概述
2、编程开发环境
3、外设寄存器头文件
4、复位、中断和系统初始化的介绍。
5、控制外设
6、闪存编程

2.2 C2000 入门培训介绍

TI 嵌入式处理器产品分为三种：微控制器（MCU）、基于 ARM 的处理器和数字信号处理器（DSP），其中微控制器（MCU）产品包括三大产品系列： 16 位超低功耗 MCU MSP430、用于实时控制的 32 位 MCU C2000 以及有基于 ARM 内核的 MCU。

TI C2000 是用于实时控制的 MCU，最高主频达 300MHz，主要应用于电机控制、数字电源、照明、可再生能源（风能、太阳能）等领域。TI C2000 包括三个系列：

1、Piccolo 系列：包括定点和浮点两类。
2、Delfino 系列：浮点 MCU。
3、Concerto 系列：基于 Cortex M3 和 C2000 双内核的 MCU。

Piccolo 系列 MCU 的主频范围从 40 MHz 到 80MHz。其中 F2806x 片上有浮点 FPU 和 VCU 加速器 F2803x 片上有 CLA 协处理器。该系列 MCU 具有 16KB 至 128KB 的闪存，6KB 至 100KB 的 SRAM，以及 ADC、PWM、QEP、DMA、SPI、UART、I2C、CAN 和 USB 等主要外设，封装管脚数为从 38 pin 到 100pin。

Delfino 系列 MCU 是主频在 100 至 300MHz 之间的浮点 MCU，其存储器外设具有高达 512KB 的闪存，以及高达 516KB 的 SRAM。Delfino 系列片上外设和 Piccolo 相似，但比 Piccolo 系列多了外设存储器的接口，即 EMIF 或 MC BSP，封装主要有 176PinQFP、176PinBGA, 179Pinu*BGA,256PinBGA。

Concerto 系列 MCU 是双核 MCU，片上具有高达 100MHz 的 Cortex M3 的内核和高达 150M 的 C28x 的内核，此外还包括浮点 FPU 单元和 VCU 加速器。存储器外设为高达 1MB 的闪存以及高达 132KB 的 SRAM，外设除了拥有 Piccolo 和 Delfino 外设外，还有 USB 以及以太网接口。

TI C2000 微控制器拥有广泛的控制应用基础，应用范围从电机控制、可再生能源、数字电源、照明、汽车到智能电网无所不包。其中电机控制主要包括白色家电、工业驱动器和

电机控制、电动工具和电动自行车等；可再生能源主要包括太阳能逆变器、风能逆变器等；数字电源主要包括通信的 AC/DC 整流器模块、不可间断电源 UPS 以及 DC/DC 模块等。

	MHz	闪存 (x16)	RAM (x16)	CLA	模拟比较器*	ADC* (通道)	PWM / (HR)*	CAP	QEP	通信端口
F28020	40	16Kw	3Kw	No	1 / 2	7 / 13	8 (0)	1	0	SPI, SCI, I2C
F28021	40	32Kw	5Kw	No	1 / 2	7 / 13	8 (0)	1	0	SPI, SCI, I2C
F28022	40	16Kw	6Kw	No	1 / 2	7 / 13	8 (4)	1	0	SPI, SCI, I2C
F28023	40	32Kw	6Kw	No	1 / 2	7 / 13	8 (4)	1	0	SPI, SCI, I2C
F28026	60	16Kw	6Kw	No	1 / 2	7 / 13	8 (4)	1	0	SPI, SCI, I2C
F28027	60	32Kw	6Kw	No	1 / 2	7 / 13	8 (4)	1	0	SPI,SCI, I2C
F28032	60	32Kw	10Kw	No	3	14 / 16	12(6) / 14(7)	1	1	SPI,SCI,I2C,LIN,eCAN
F28033	60	32Kw	10Kw	Yes	3	14 / 16	12(6) / 14(7)	1	1	SPI,SCI,I2C,LIN,eCAN
F28034	60	64Kw	10Kw	No	3	14 / 16	12(6) / 14(7)	1	1	SPI,SCI,I2C,LIN,eCAN
F28035	60	64Kw	10Kw	Yes	3	14 / 16	12(6) / 14(7)	1	1	SPI,SCI,I2C,LIN,eCAN

** 不同外设数量取决于封装类型：F2802x – 38/48 引脚，F2803x – 64/80 引脚*

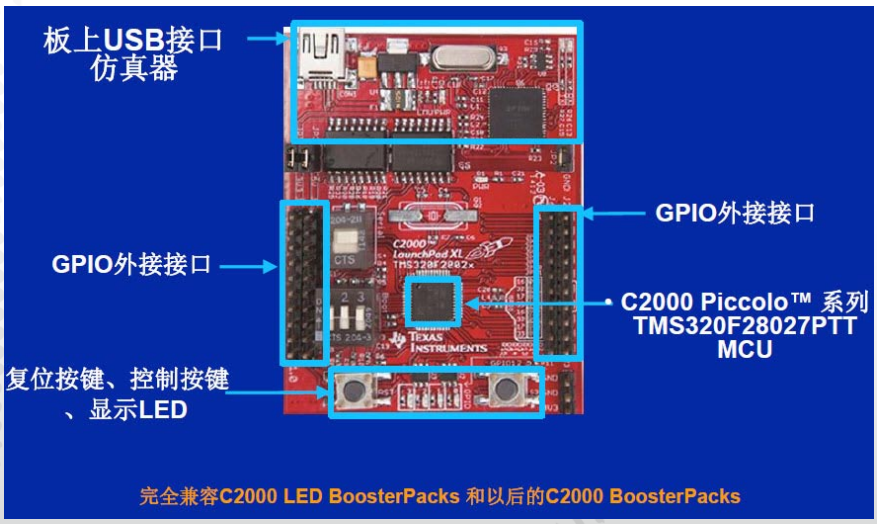
上图说明了 F2802x 系列和 F2803x 系列协处理器之间的差异。F2802x 系列片上闪存大小在 16Kw 到 32Kw 之间，CPU 主频大小在 40 到 60MHz 之间，不带 CLA。从上图还可以看出模拟比较器、ADC（通道）、PWM 数量以及通信端口的数值。F2803x 系列中，F28033 和 F28035 包括片上 CLA。不同外设数量取决于封装类型：F2802x – 38/48 引脚，F2803x – 64/80 引脚。所有 F2802x / F2803x 系列都具有 VREG、POR/BOR、看门狗、OTP、CPU 定时器。如欲了解 TI C2000 系列产品更多细节，请去 TI 官网下载相应的数据手册和用户指南 (http://www.ti.com/lit/sg/sprb176p/sprb176p.pdf)。

现在来对 Delfino/Piccolo 进行比较， Delfino 系列以 F2833x 系列为代表，Piccolo 系列则以 F2803x 和 F2806x 系列为代表。其中 Delfino 系列最高主频可达 150MHz，Piccolo 系列 F2806x 的主频只到 80MHz。片上 12 位 ADC 也有较大差别，Delfino 系列基于 SEQ 的触发方式，Piccolo 系列则基于 SOC 的触发方式。此外，Delfino 系列全部包括片上的 FPU 协处理器，而 Piccolo 系列只有 F2806x 具备和 Delfino 系列相同的片上 FPU。在片上通信接口方面也有一些差异，详见截图。

	F2833x	F2803x	F2806x
时钟频率	150 MHz	60 MHz	80 MHz
闪存/ RAM	128Kw / 34Kw	64Kw / 10Kw	128Kw / 50Kw
片内振荡器	-	2	2
VREG / POR / BOR	-	✓	✓
看门狗定时器	✓	✓	✓
片上12 位 ADC	基于 SEQ	基于 SOC	基于 SOC
带 DAC 的模拟比较器	-	✓	✓
浮点单元 (FPU)	✓	-	✓
6 通道 DMA	✓	-	✓
控制律加速器 (CLA)	-	✓	✓
矢量运算单元 (VCU)	-	-	✓
ePWM / HR ePWM	✓ / ✓	✓ / ✓	✓ / ✓
eCAP / HR eCAP	✓ / -	✓ / -	✓ / ✓
eQEP	✓	✓	✓
SCI / SPI / I2C	✓	✓	✓
LIN	-	✓	-
McBSP	✓	-	✓
USB	-	-	✓
外部接口	✓	-	-

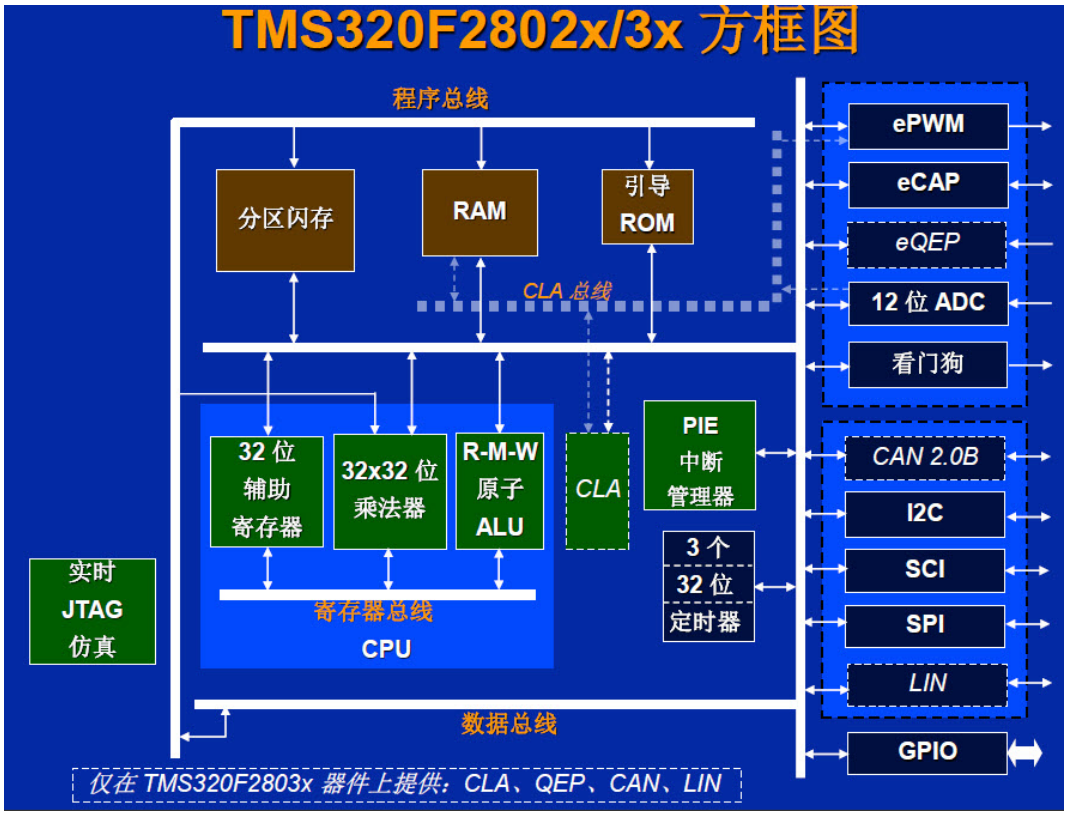
Piccolo F2802x 应用入门包括三个实验，本次讲座实验采用的是 TMS320F280270 LaunchPad 开发工具，该开发工具包括以下四部分：

1、板上 USB 接口仿真器，该仿真接口除了 SDS100 仿真器外，还包括隔离部分。
2、C2000 Piccolo ™ 系列 TMS320F28027PTT MCU。
3、GPIO 等外接接口。
4、LED 显示、控制按键、复位按键。



2.3 TI C2000 C28X 架构概述

经过了对 TI C2000 整体性的认识，下面将介绍 C28x 架构。



TMS320F2802x/3x 架构主要包括 CPU、内存、外设和定时器四部分。

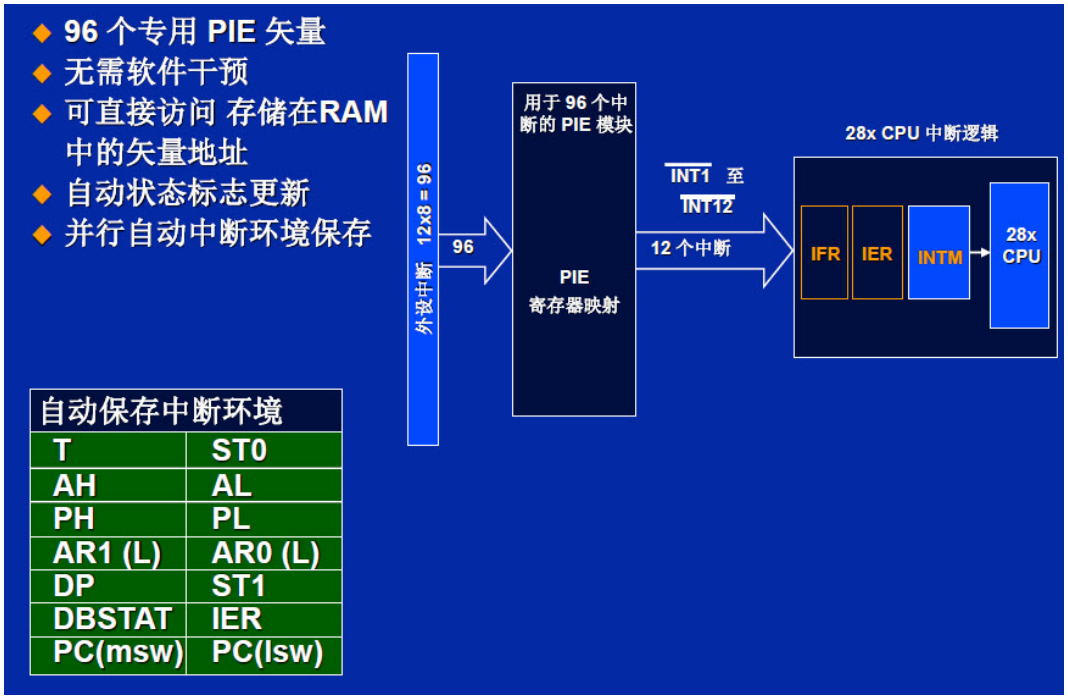
CPU 由片上 32 位辅助寄存器，32×32 位乘法器，逻辑运算单元组成。

内存包括片上闪存，RAM 和应用程序引导的 ROM 部分。

外设包括 ePWM、eCAP、eQEP、12 位 ADC、看门狗以及 CAN、I2C、SCI 等通信接口。其中只有在 F2803x 上有 CLA、QEP、CAN 和 LIN。

所有 Piccolo 系列都有片上的实时 JTAG 仿真接口，可以进行应用程序调试以及闪存编写。

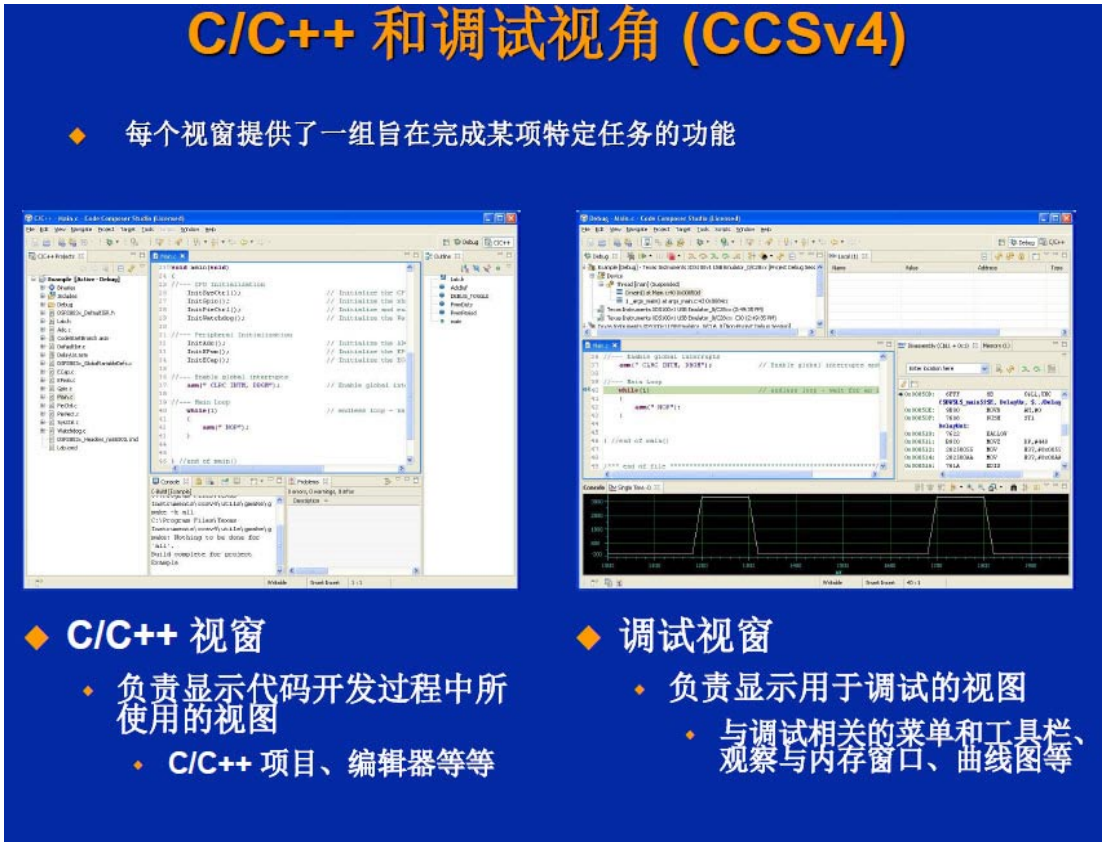
TMS320F2802x 系列存储器结构由 RAM、外设寄存器映射、片上 OTP、Flash 和应用程序引导的 ROM 部分组成。其中 L0 RAM、OTP、片上闪存、ADC 校准寄存器和 Flash 寄存器部分可以通过密码对其进行保护。



F28x 拥有快速中断响应管理器 ,F28x 的中断包括 96 个专用 PIE 的矢量、且中断响应不需额外软件干预，中断可以直接访问存储在 RAM 中的矢量地址，可以自动完成状态标志的更新，并可以自动进行相应的寄存器保存，保存的寄存器包括乘法接口寄存器、状态寄存器、累加器，辅助寄存器、PC 指针等。其中外设中断包括 12 组 96 个中断，这 12 组中断从 INT1 到 INT12，并由 IER 使能寄存器进行控制，IFR 寄存器显示 12 个中断状态，INTM 是全局中断使能的位置，通过 IER、INTM 来管理 96 个中断，给 28x CPU 发出对应的中断请求。

2.4 编程开发环境

C2000 采用 CCS 集成开发环境，集成了编辑、代码生成和调试功能，可以通过对应的菜单进入相应的功能。此外，CCS 还包括功能强大的绘图 / 分析工具，使用脚本的自动化任务,内置的 BIOS 功能访问,且CCS基于Eclipse开源软件框架，能允许用户完成对应功能的嵌入。



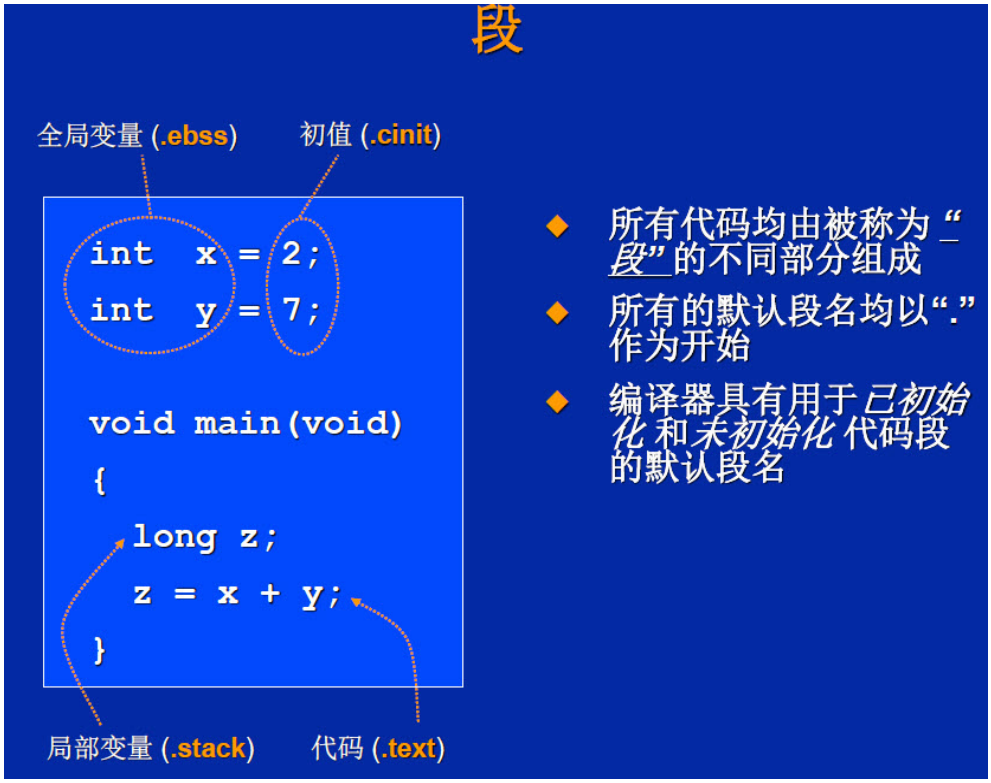
CCS 针对不同功能具有不同的视窗， CCS 编辑视窗负责 CCS 项目的建立、管理和代码的编辑， CCS 调试视窗用于芯片的调试、仿真等，在显示结果中可以观测到变量、内存大小和对应的图形。

CCS 项目文件包括文件清单和项目设置 。文件清单中包括源文件，源文件编程语言为 C 语言、C++ 语言、汇编语言,也可以通过这三种语言进行混合编程;此外，文件清单还包括库文件（例如 C28x 的 Lib 文件），DSP/BIOS 配置文件，链接器命令文件。项目配置可以设置生成选项（编译器、汇编器、链接器和 DSP/BIOS），还可以通过生成配置中配置优化的级别、程序的入口以及配置 Lib 件的链接。

那么，如何创建一个新的 CCS 项目呢？

点击菜单栏中的 File 选项，在其下拉菜单中点击 New，生成对应的 CCS Project。设计者可以命名 CCS Project 的名字，选择相应的存储路径，选择不同的 CCS 的配置。还可以选择 Debug 的配置，也可以选择 Release 的配置，或者两个都选可以选择，也可增加对应的新配置。通常默认的 Debug 配置是没有 CCS 生成配置的优化，Release 配置则是对对应的代码进行优化。设计者可以增加 CCS 的附属配置，还可以进行芯片的选择、编译器版本的选择、链接头文件的选择和对应的库文件选择。

CCS 生成选项包括编译器选项和链接器选项两部分。编译器选项主要用代码生成工具的配置，例如优化级别的选择，目标器件的选择，编译器、汇编、链接选项的设置。链接器选项主要规定各种不同的链接选项，例如文件的链接顺序，当前项目目录、生成目标文件名字定义以及生成目标 .map 的定义等。



下面介绍链接文件的构成。链接文件中所有代码均由被称为“段”的不同部分组成，所有默认段名均以“·”作为开始，编译期具有用于已初始化和未初始化代码段的默认段名。例如“·ebss”用于全局变量的定义，“·cinit”用于已初始化变量的分配，“·stack”用于局部变量的分配，“·text”用于程序代码的分配。

编译器中的段名

已初始化段

名称	说明	链接内存位置
.text	代码	闪存
.cinit	用于全局和静态变量的初始化值	闪存
.econst	常数 (例如：整数常数 k = 3；)	闪存
.switch	分支语句表	闪存
.pinit	全局构造子表 (C++)	闪存

未初始化段

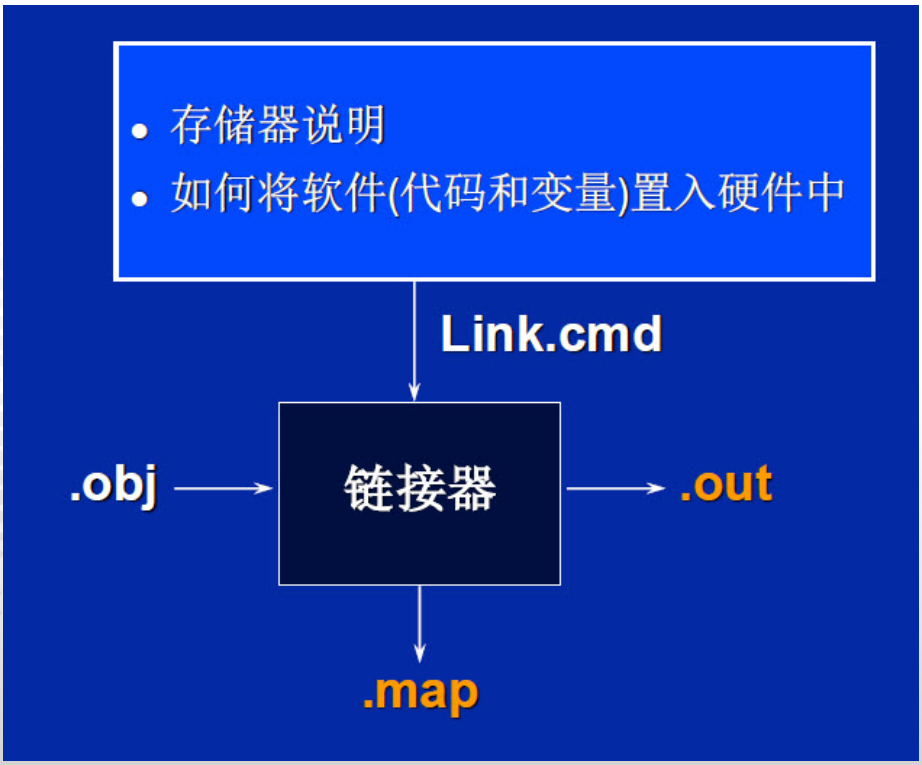
名称	说明	链接内存位置
.ebss	全局和静态变量	RAM
.stack	堆栈空间	低地址的 64Kw RAM
.esysmem	内存分配函数 far malloc	RAM

注：由于仿真器可用于 RAM 加载，因此在开发过程中可将已初始化段链接 至 RAM

常用的编译器主要包括已初始化段和未初始化段两部分，已初始化段有“·text”，主要用于代码的分配，一般它的链接位置是闪存；“·cinit”用于全局和静态变量的初始化值的分配，也同样是链接在闪存中；“·econst”主要用于常数的定义，例如某一个常数等于多少，同样也用于闪存中；“·switch”主要用于 C 语言中分支语句的定义，同样链接在闪存中；“·Pinit”主要用于 C++ 中全局构造子表的分配，同样链接在闪存中。未初始化段主要用于分配全局变量、堆栈和相应的内存分配函数，要注意的是“·stack”通常定义在低地址的 64Kw 的 RAM 中，其他的可以任意分配到相应的 RAM 中。

在存储器中不同段的分配。Memory 映射包括 M0 的 RAM、M1 的 RAM 和闪存，通常我们把“·ebss”、“·stack”映射在相应的 RAM 中，例如把“·ebss”映射到 M0 部分，“·stack”映射到 M1 的 RAM 部分，而“·cinit”、“·text”则映射到闪存 Flash 部分。链接文件主要由两部分文件组成，一是存储器说明，例如要说明 RAM、Flash 映射到芯片的哪部分，它的地址、长度；二是规定如何将软件（代码和变量）置入硬件中，例如要把变量映射到对应的 RAM 中，代码映射到对应的闪存中，通过链接文件和链接器把源文件生成的“·obj”文件生成对应的目标文件“·out”文件，另外生成对应的“·map”文件，通过“·map”文件可以看出变量分配的位置，闪存、RAM 的使用，每一个函数入口地址位置，每一个变量存储位置等等。

下图展示了链接器命令文件例子，链接器文件主要有两大部分， Memory 部分和 Section 部分。其中 Memory 部分就是芯片内存相对应的部分，包括 PAGE 0——默认的是程序代码的内存部分，PAGE 1 是变量代码的内存部分。选址可以自定义，比如定义 Flash 是定义片上的闪存部分，起始地址、长度也是和对应的芯片相对应。Section 部分有通常的链接器默认名，“·text”、“·ebss”等等，要指定这个代码映射到某一个位置所对应的 PAGE 是 0 还是 1。



2.5 外设寄存器头文件

```
#define ADCTRL1       (volatile unsigned  int *)0x00007100
                      ...
void main(void)
{
    *ADCTRL1 = 0x1234;            //write entire register
    *ADCTRL1 |= 0x4000;           //enable ADC module
}
```

优势
- 简单、快速且易于键入
- 变量名称与寄存器名称精确匹配 (易于记忆)

劣势
- 需要生成专门的掩码以操控个别位
- 不能很容易地在观察窗口中显示位字段
- 在许多场合中将生成效率低下的代码

下面介绍一个 C 语言编码的传统方法，从深蓝色框图可见，通过定义控制寄存器地址的方式来对寄存器进行赋值，这样能够对整个寄存器赋值。其优势就是简单、快速，且易于键入;此外，变量名称与寄存器名称能够精确匹配，易于记忆。但劣势是，需要生成专门的掩码以操控个别位，比如要实现对控制寄存器中一个位进行赋值，就比较复杂；此外，不不能很容易地在观察窗口中显示位字段，比如要看某一位的状态；在很多场合下，这种命名方式的代码效率非常低。

C2000 头文件定义是通过结构体的方法，既可以对整个变量进行赋值，也可以对单独的位进行赋值，所以 C2000 的优点比较明显，操控个别的位会在调试窗口中容易观测寄存器位的值，通过 C2000 也会生成比较高效的代码，但其缺点是结构体的名称较难记忆和需要输入更多的内容，但这两个缺点会通过 CCS 编译器的自动完成功能来进行弥补，因为 C2000 自动完成功能会自动列出结构体的名称、成员等等。

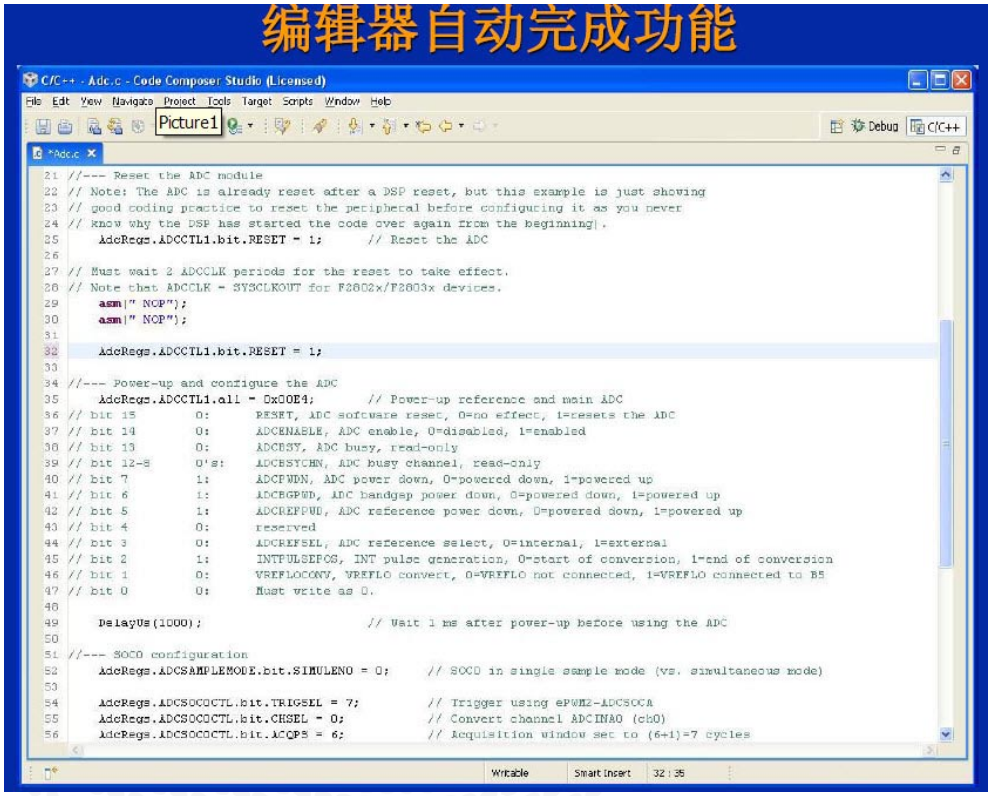
观测寄存器的值可以通过两种方式，内置的 CCS 寄存器窗口直接观测寄存器的内容，和运用结构体的方式观测寄存器的值。结构体的方式既可以看寄存器的值，也可以看寄存器中每一位的值。

下面介绍结构体的定义规则，F2802x 头文件定义包括：所有的外设结构体，所有的寄存器名称，所有的位字段名称，所有的寄存器地址，可以通过访问整个 16 位或 32 位寄存器来对寄存器进行赋值或者读取，可以通过 32 位寄存器的 16 个高位或 16 个低位，对 16 位部分进行访问，或者访问寄存器的指定位字段。

结构定义规则包括：

“PeripheralName” 是由大小写字母组合定义结构体的名称，该名称是和 F2802x 的外设相对应，由 TI 指定并载于 F2802x 标头文件（如：CpuTimer0Regs）。“RegisterName” 与数据表中使用的名称相同，通过大写字母来定义对应的寄存器名称，比如 TCR 寄存器、TIM 寄存器、TPR 寄存器等。同时，在数据表中对位字段的定义，也通过大写字母来完成，例如 POL、TOG、TSS 对应的位。

下图演示的是利用编辑器自动完成功能，来实现对结构体中某一个位进行赋值。例如对 ADC 寄存器的控制寄存器中的 Reset 位来进行赋值，F2802x 头文件包会通过 TI 对应网站直接下载，文件编号是 SPRC832。头文件包中包含了使用结构体法所需的一切资源，定义了所有的外设寄存器位和寄存器地址；此外，头文件包中还包括头文件的例程和对应说明文档。



外设结构体头（.h）文件包括每个外设寄存器的位字段结构体的定义，例如 DSP2802x_Adc.hADC 头文件包括 ADC 所有寄存器的定义。那么，如何用 ADC 寄存器呢？通过其源文件可以看到，我们可以直接对对应的 ADC 寄存器（或 ADC 的位）进行赋值。外设头文件中头文件包含有一个用于器件中每个外设的 .h 文件，例如 ADC 头文件、CPU 头文件等，只要通过一个 “DSP2802x_Device.h” 文件，就可以把所有的外设头文件包含进来。

那么，如何在项目中利用外设头文件呢？这时就需要通过全局变量定义文件，从而实现对每个外设寄存器的结构体变量进行定义，对每一个结构体使用一个

DATA_SECTION 宏指令来强制结构体的存储位置，以实现至正确的地址链接。首先，我们需要将该文件添加到 CCS 项目中，即将 DSP2802x_GlobalVariableDefs.c 文件添加到对应的项目中。

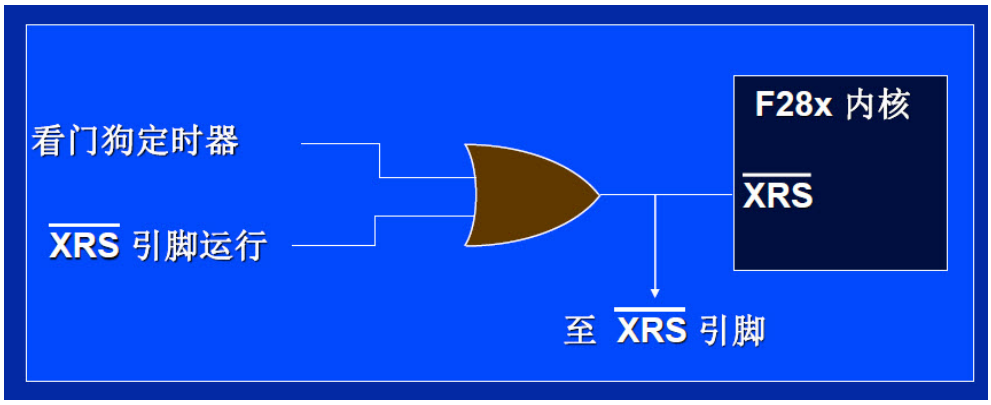
此外，我们还要添加用于结构体的链接器命令文件，这样的文件包括两种，nonBIOS 和 BIOS 的链接文件。其中，nonBIOS 用于不带 DSP BIOS 的项目，BIOS.cmd 主要用于带 DSP BIOS 的项目文件。首先在 DSP2802x_GlobalVariableDefs.c 文件中定义每个结构体的名称及其映射地址，通过链接文件规定对应的寄存器位段，比如定义 ADC 寄存器在什么段，起始地址和长度等，这个起始地址和长度是根据 CPU 和外设定义的地址和长度一一对应的。然后将定义的 .cmd 文件或 .c 文件添加到 CCS 项目中。

TI 为每个外设提供了对应的外设应用范例，通过外设应用范例可以帮助您快速启动开发工作，例如 TI 提供了 ADC、CPU、PWM、SCI 等所有外设的开发应用范例，如果用到 SCI 打开其范例，就会看到 SCI 的寄存器配置方式，把对应配置的范例文件直接添加到你的目标项目中去，就可以完成对 SCI 的应用。

至此，我们对外设寄存器头文件进行一个总结：首先，通过外设寄存器头文件可以更方便地进行代码开发，简单易用，可以生成最高效的代码，提高了 CCS 调试窗口观察寄存器的有效性， TI 帮你完成了所有的基础工作，给你提供正确的头文件包。2802x、2803x 等所有 C28 系列的头文件包，电子工程师都可以通过访问 http://www.ti.com 网址并在关键字搜索框内输入文献编号进行下载。

2.6 复位、中断和系统初始化的介绍

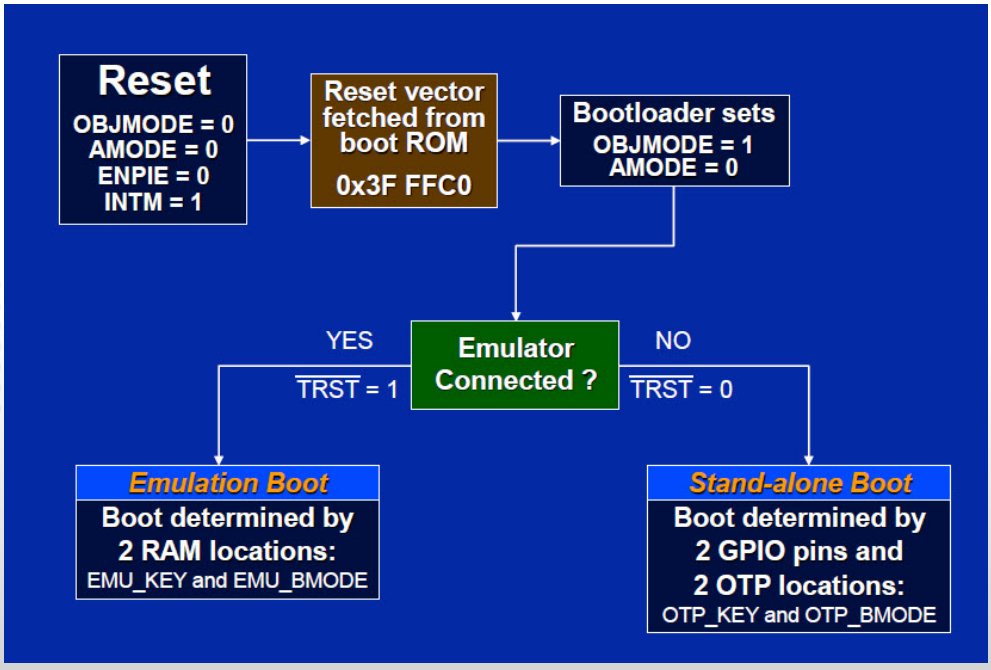
✧ 实验 1：看门狗和中断



CPU 复位源有内部和外部，内部是通过看门狗定时器的复位和 XRS 引脚运行状态的复位，外部就直接由 XRS 引脚给一个低电平 CPU 的复位。其中，看门狗定时器的复位，是指看门狗能使以后溢出造成看门狗的复位。XRS 内部引脚运行引起的复位主要包括 POR、BOR，POR 是指上电复位功能，在上电的条件下会产生一个器件复位信号给 CPU；BOR 是指欠压复位功能，可在电源电压降低到器件电源规定的阀值时产生一个器件复位。所有 Piccolo 系列器件上都支持一个片上稳定器生成的内核电压，实现芯片的单电源供电。

CPU 复位流程

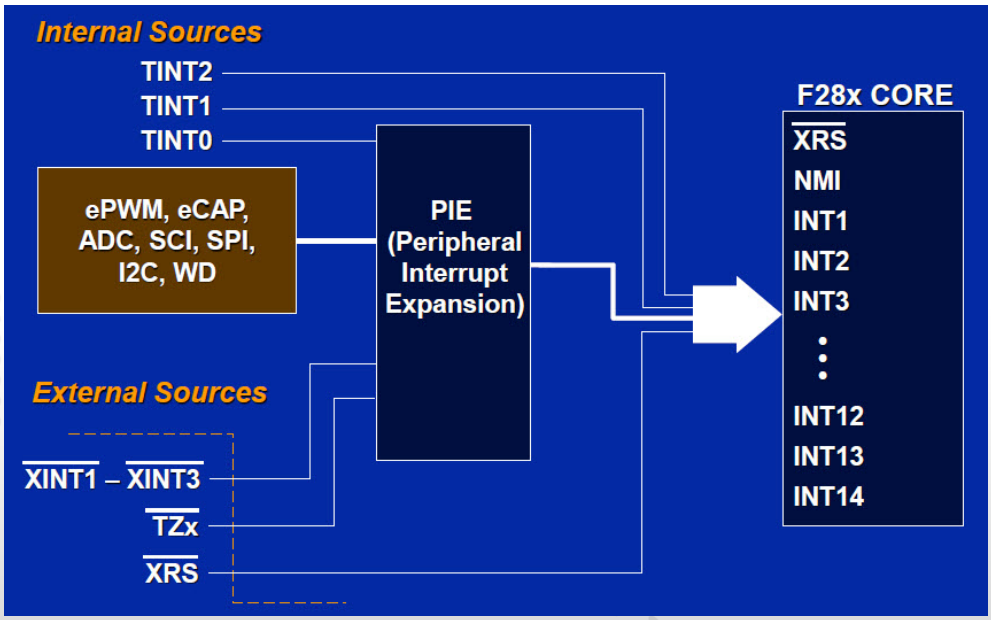
CPU 复位后，PC 指针会指向 0x3FFFC0 的位置，在 0x3FFFC0 中存储一个 PC 地址指向 Boot ROM，通过 Boot ROM 引导程序检查 PC 指针下一步会指向什么位置，可以指向 FLASH、OTP、RAM 或指向外设引导的 ROM 地址。



那么，CPU 如何进行启动加载呢？首先启动后会对相应的寄存器设置为默认值，跳到 0x3FFFC0 部分，Boot ROM 引导程序进行查询，根据访问器接口 TRST=0 还是 TRST=1 进入仿真模式或单机运行模式。仿真模式会检测两个寄存器的值，EMU_KEY 和 EMU_BMODE 两个寄存器的内容，来决定 PC 指针的指向位置。单机运行模式会检查 GPIO 口的状态，来决定 CPU 的指针下一步会指向什么位置。仿真模式是指 TRST 等于 1 的模式中的引导流程，首先会检查 EMU_KEY 寄存器的内容，如果 EMU_KEY 寄存器等于 0x55AA，那它会继续检查 EMU_BMODE 寄存器的内容，来决定 PC 引导位置。如果 EMU_KEY 寄存器不是等于 0x55AA，那它一直都会在等待模式。在引导模式中会查 BMODE 里的内容，例如内容等于 3，它会去查 OTP_KEY 里的值，如果 OTP_KEY 里的值是等于 0x55AA ,它会进入不同模式的查询 ,查 OTP_BMODE 里的值。从这张表里（PPT: 仿真器模式 TRST=1 的引导流程）可以看出，如果 BMODE 里的值是等于不同的值以后，PC 会指向闪存、OTP、SCI 等等。

单机模式，就是 TRST=0 模式的引导流程。单机模式时，首先 PC 会跳到复位这个指针部分，之后它会跳到 Boot ROM 的内容中，Boot ROM 通过检查 GPIO 口 37 或 34 的状态来决定下一步的模式。比如说等于 01，那它就会到 SCI 模式，如果等于 10，那它会进入 Wait（等待）模式，等于 11，那它会到下一个 Check 的模式（GetMode），在下一个 Check 模式的时候，它会查 OTP_KEY 寄存器的值，如果是等于 0x55AA，它会来查 OTP_BMODE 寄存器里的值，如果等于 01 那它从 SCI 引导 等于 03 它从 FLASH 引导 等于其他它从不同位置进行引导。如果等于 Wait 模式的话它会进入 FLASH 模式，就是 GPIO 等于 10 默认我们进入 FLASH 模式。

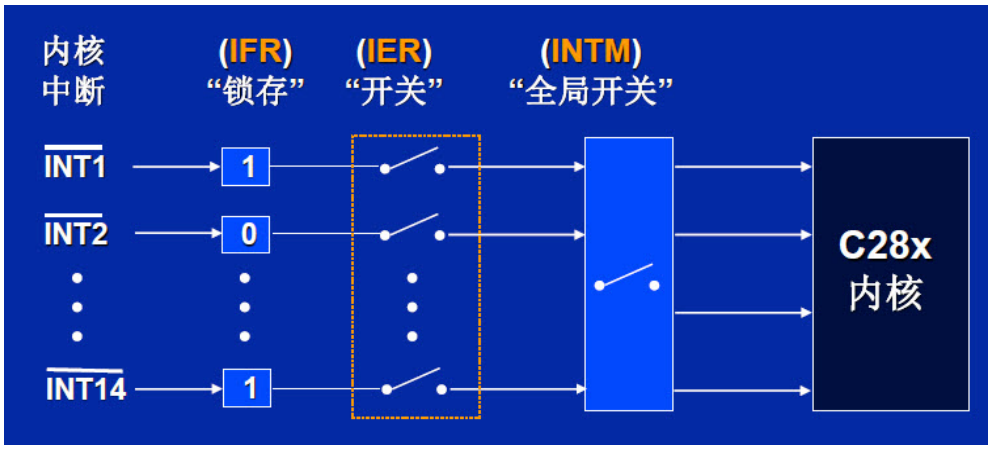
28x 系列的中断源。



28x 系列中断源有两个来源，内部和外部，其中内部是 CPU 的定时器、外设等等，外部是由外部中断，即 GPIO 口可以配置外部中断 TZx 信号或 XRS。外设中断主要有 PWM、CAP、ADC、SCI、SPI 等等都是形成外设中断。F28x CORE 有很多的中断有复位,有不可屏蔽的 NMI 中断,有 INT1—INT14 的外设扩展中断。

外设扩展中断分 12 组，每组里有 8 个中断，每个中断对应不同的外设中断源，此外，还包括 TINT1 和 TINT2 的 CPU 定时器中断，所有中断通过 IER 寄存器进行能使，IFR 指示对应中断的状态，INTM 寄存器的开关来能使全局的一个中断，当这个关闭以后，所有的中断都不能被 CPU 响应，只有开 INTM 这个位所有的中断才可能被 CPU 响应。

可屏蔽中断处理。



一个有效的内核中断的产生会导致中断标志寄存器在适当的位中显示“1”，比如 INT1 外设扩展中断进来以后，如果产生了，就会在 IFR 对应的位置一个“1”，如果这时候 IER 寄存器对应的位设成能使，那这个中断源就会直接进到全局开关部分，如果全局开关在能使，那 C28x 的核就会响应这个中断源。

内核中断寄存器主要包括状态寄存器 IFR、中断能使寄存器 IER 或状态寄存器里的 INTM 位。通常，我们通过 IFR 寄存器指示中断的状态，如果有一个中断被触发了，它对应的状态寄存器的位会被置“1”，IER 寄存器主要是应用中断能使的控制，当对应的位置“1”我们就能使这一个中断。状态寄存器 STE 中的 INTM 位主要用于全局中断的开通或关闭，我们可以配置 IAR 对应的位是“1”，全局中断进行打开，从下面几个例程可以看到，在对应的位设置成“1”，那这个中断或能使了,比如 INT4 ,当我们把全局中断打开以后,所有的中断都可以进到 CPU 的核里。

外设中断总共有 12 组，每一组对应一个状态寄存器和一个能使寄存器，比如 PIEIFR1—12，状态寄存器同前面的 IFR 类似。当对应的一个外设扩展中断被触发以后，相应的位同样也会被置“1”，同样只有对应的 PIEIFR 相应的中断能使位被置“1”，那这个中断才可能被后面的 CPU 所响应。PIEACK 这个寄存器主要应用于中断下次进入的设置，当我这一组中断被响应以后，PIEIER、PIEACK 必

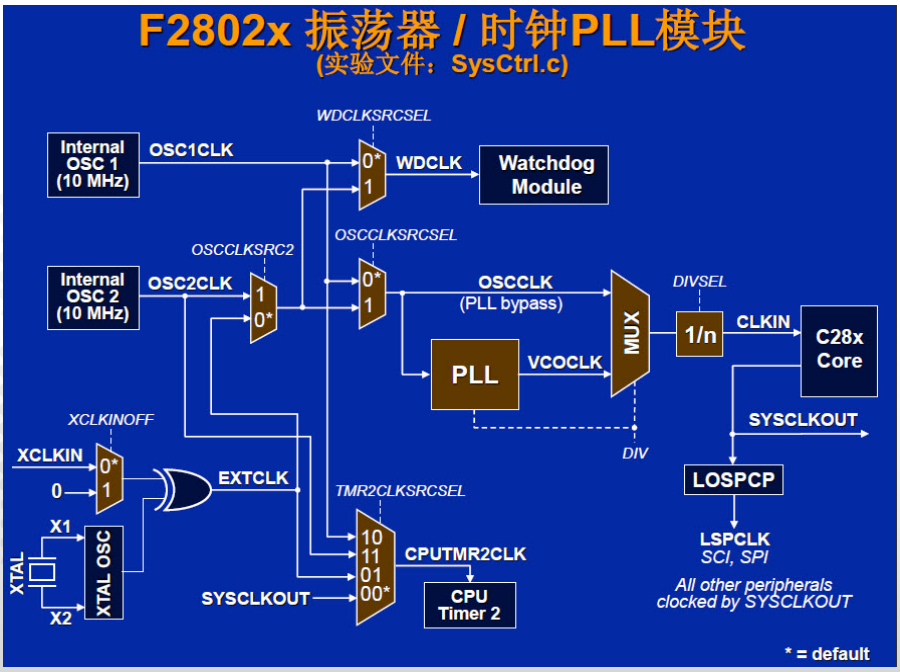
须在中断服务子程序里被清（0）一下才能让下一个中断进入。

F2802x PIE 中断分配表

	INTx.8	INTx.7	INTx.6	INTx.5	INTx.4	INTx.3	INTx.2	INTx.1
INT1	WAKEINT	TINT0	ADCINT9	XINT2	XINT1		ADCINT2	ADCINT1
INT2					EPWM4_TZINT	EPWM3_TZINT	EPWM2_TZINT	EPWM1_TZINT
INT3					EPWM4_INT	EPWM3_INT	EPWM2_INT	EPWM1_INT
INT4								ECAP1_INT
INT5								
INT6							SPITX INTA	SPIRX INTA
INT7								
INT8							I2CINT2A	I2CINT1A
INT9							SCITX INTA	SCIRX INTA
INT10	ADCINT8	ADCINT7	ADCINT6	ADCINT5	ADCINT4	ADCINT3	ADCINT2	ADCINT1
INT11								
INT12								XINT3

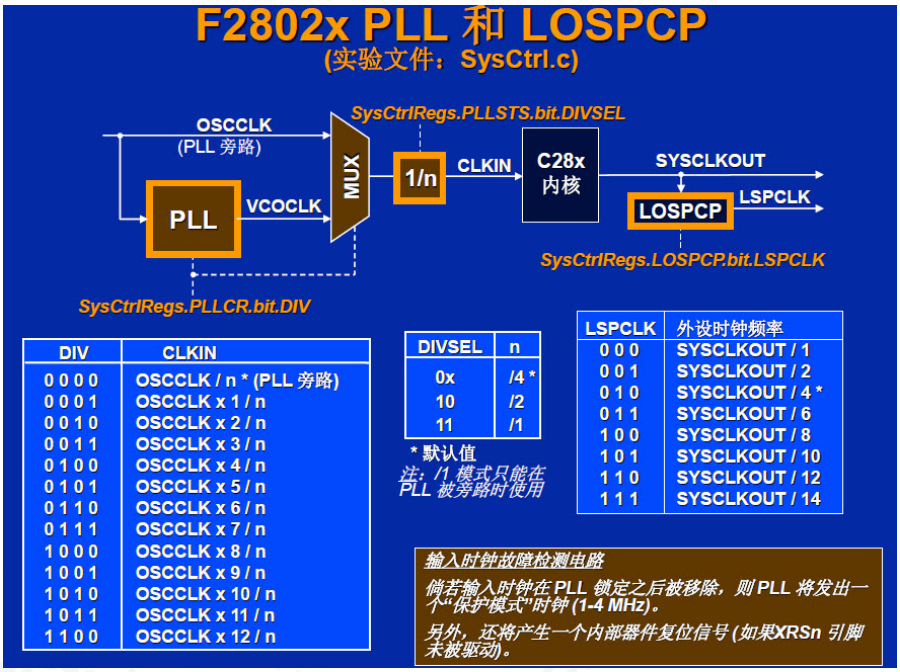
IECTRL 中的 ENPIE 位主要用于中断向量分配的能使。上表总共可以看到，每组里有 8 个中断，每一些中断对应哪一个中断向量可以从这张表（PPT：在复位时中断矢量表的默认设置）可以看到，比如 ADC、PWM、CAP 等所有映射的位置。在复位时，中断矢量表的默认位置由 ENPIE 来决定，当 EMPIE 等于 0 的时候中断向量表的位置会在 BROM 中，当 ENPIE=1 中断向量表的位置，映射到 PIE 的中断向量表的 RAM 中。

F2802x 振荡器 / 时钟锁相 (PLL) 模块。



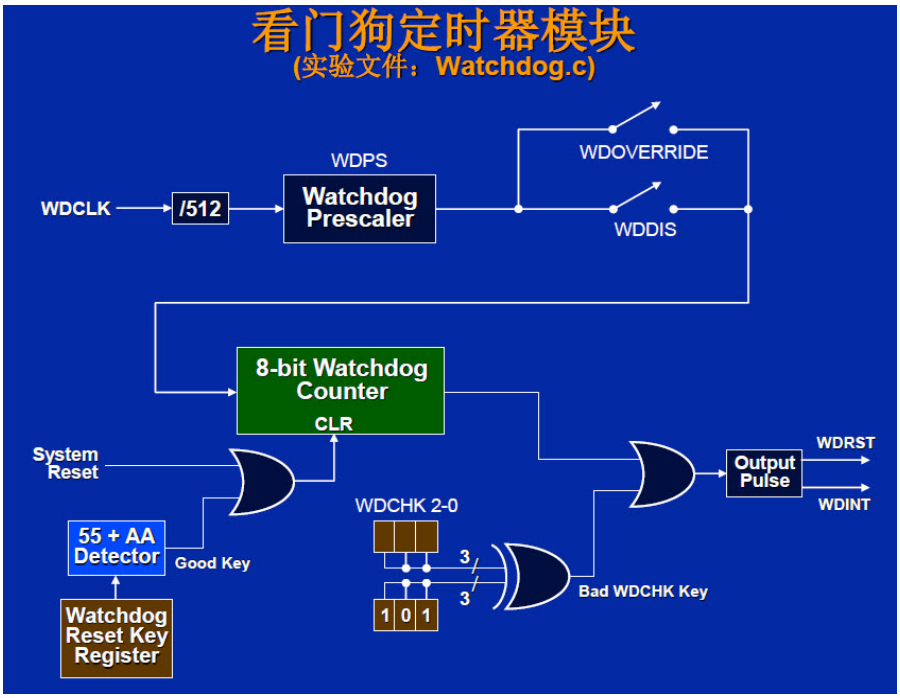
F2802x 的时钟源可以由内部提供，也可以由外部提供，内部的时钟源包括两个振荡器，OSC1 和 OSC2，上电默认时，我们通常采用 OSC1 的内部时钟源 . 外部时钟源可以通过无源晶振或有源的时钟信号提供时钟源。无源晶振我们通过 X1 和 X2 两个管脚提供，有源的时钟源通过 CLKIN 管脚提供，可以通过内部控制寄存器来选择时钟源，可以选择内部的 OSC1 和 OSC2 和外部的，再通过对应的锁相来把时钟源进行倍频，提供系统的时钟。比如说外部时钟源是 10M 或内部时钟晶振固定是 10M，可以通过锁相模块（PPL）把时钟倍频到 60M，来提供给 CPU 的系统时钟。CPU 的系统时钟又可以通过 LSPCLK 控制寄存器来进行分频，提供更低频率的时钟源给 SCI、SPI 等外设。

CPU Time2 的时钟源可以通过 OSC1、OSC2 或外部时钟来提供，它可以独立系统时钟，也可以直接由系统时钟来提供。Watch Dog 看门狗的时钟源同样选择是用内部或是外部的来提供，它也可以独立 CPU 的系统时钟。Watch Dog 的时钟同样会通过对应的寄存器进行分频控制。



从上图可以看到 F2802x 锁相模块的工作情况或 LOSPCP 低速时钟配置。锁相模块通过寄存器设置其为能使的或旁路的，如果是能使状态可以设置倍频的倍数，如果是旁路状态可以把这个倍数设置成 0，它就直接把锁相模块给旁路了，此外还可以设置倍频以后的分频倍数。比方说让它 12 倍频，在 2 分频，通过 10M 时钟源可以提供 60M 的系统时钟给 CPU。低速时钟同样可以通过分频，从 1 分频到 14 分频给系统外设，主要是一些低速的外设，比如 SCI、SPI。C28 内核还提供了一个时钟检测机制，当外部时钟失效、锁相环被移除之后，PLL 模块将会提供一个模糊模式的时钟，主频在 1MHz—4MHz，同时它还会产生一个内部器件的复位信号，通过 SRS 管脚给外部提供一个复位信号给 CPU。

看门狗模块的作用。假如 CPU 崩溃，看门狗会使 CPU 复位，看门狗计数器的运行与 CPU 无关，如果看门狗计数器溢出则触发一个复位或中断信号（可由用户选择）提供给 CPU。CPU 必须写入正确的数据密钥序列，才能使计数器在溢出之前复位。看门狗必须在复位之后的 131,072 个指令周期之内完成维护或停用，当采用一个 10MHz 看门狗定时器时，其复位产生时间是 13.11ms，我们可以通过对应的控制寄存器来设置不同的复位产生时间。



看门狗定时器模块，首先选择看门狗定时器的时钟源，通过 512 分频提供给看门狗，在看门狗寄存器中设置它的预分频，控制看门狗能使或关闭。如果能使以后，看门狗计数器会一直计数，当它计数溢出会产生对应的中断或复位。CPU 必须在它溢出之前写入 55 再写入 AA，来使计数器清零。

F2802x 的 GPIO 可以配置成不同的功能——外设的功能和通用的 GPIO 口功能，这个配置通过 MUX 寄存器来实现。比如 GPAMUX1、GPBMUX2 分配对应不同的 GPIO 口。通过寄存器我们来设置它是外设寄存器，例如 SCI 寄存器或 SPI 功能，或 PWM 功能，或者是通用的 GPIO 口功能。如果是通用的 GPIO 口功能，我们可以再通过方向寄存器来设置它是输入还是输出。另外，我们可以通过输入鉴定功能来设置它的输入滤波功能。F2802x 部分模拟通道输入可以配置成通用输入口，我们可以配置成它是输入还是输出。

F2802x GPIO 口首先通过 MUX 寄存器来配置成不同的功能，通常一个 GPIO 口会支持四种功能，默认是 GPIO 口功能，还可以通过配置来把它配置成不同的外设功能，01、10、11 可能分别对应不同的外设功能，比如有的口可以同时支持 SCI 功能或 PWM 功能。如果配置成通用 GPIO 口以后，Input Qualification 这个模块可以设置它的输入功能的输入鉴定，GPxPUD，可以配置

它是能使还是关断，GPIO 方向寄存器用来配置它是输入还是输出，如果是输入，外部 I/O 口状态的改变会存到 GPx 的 Data 寄存器里，如果做输出，可以通过三种方式设置对应的 GPIO 口的输出状态，可以通过 GPxSET 寄存器、GPxCLEAR 寄存器、GPxTOGGLE 来设置外部输出的状态。如果通过 SET 寄存器对应的位置成“1”，那 GPIO 口输出状态就是高。如果把 CLEAR 寄存器对应的位置成“1”，GPIO 口就输出一个低电平。TOGGLE 寄存器的功能是设置成“1”，对应的 GPIO 口的状态会翻转，如果当前状态是“0”，它就翻转成“1”，如果当前状态是“1”，它就翻转成“0”。

当 GPIO 口设置成输入时，我们可以配置输入鉴定功能，输入鉴定功能主要用于 GPIO 口输入滤波，每个鉴定输入功能都可以配置在对应的端口上，GPIO 口 0—38 都提供对应的输入鉴定功能。每一个引脚会单独选择，会配置成它是非同步输入，这时候输入鉴定功能是不能使的，配置成同步的 CLK 功能，就是和系统时钟同步的输入功能，可以配置成不同的鉴定样本，可以配置成 3 个样本或 6 个样本，样本的意思是只有连续输入 3 个同样信号，才认为它是一个有效信号，连续输入 3 个“1”，我认为它是一个高电平；连续输入 3 个低电平，那认为它是一个低电平，如果是 6 个样本，同样它是要连续 6 个信号才认为是一个有效信号。AIO 口，就是模拟的输入口是不能被配置成鉴定功能的，它是固定和系统时钟是同步的。

下面我们通过一个实验来对系统初始化进行熟悉。实验文件会包括两个部分，第一部分通过测试看门狗在停用和启动时的运行状态；第二部分通过预置外设扩展矢量，来使看门狗生成一个中断，并进行响应。通过这个实验我们同熟悉如何运用 CCS 修改、编译和测试代码。

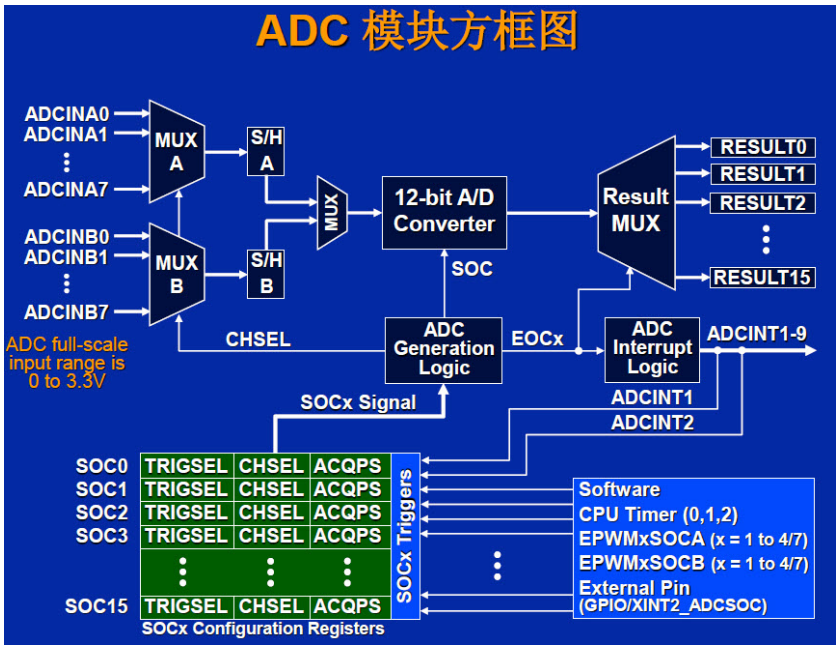
下面我们给大家介绍一下本次入门培训中的实验。

首先，从 TI 网站上下载 CCS 的安装软件，安装完后会有一个 CCS 快捷键，点击快捷键会启动 CCS。这里我已经建立了一个 CCS C2000 的 LauchPad Worksbase，大家可以自己建一个 Worksbase，指定对应的一个目录，写入这个 Worksbase 的名字。点击 OK 开始，会启动 CCS。本次实验我们用的是 LauchPad 的开发套件，这个 LauchPad 套件包括一个 TMS320F28070 auchPad 和 USB 线，片上有一个 USB 仿真器，连接后，直接连接到电脑的 USB 口。如果大家有 LauchPad 板子需要注意一下，可以通过 USB 直接给芯片供电，也可以通过外部给芯片供电，如要通过 USB 给 F28027 供电的话，要把这两个跳线连上，才能把 USB 的电源连到 F28027 部分上面，实现对 F28027 供电。

CCS 启动完成后，可以通过菜单栏的 File，点击 New 来进行 CCS 项目的建立，这里省略若干步骤，等 CCS 启动以后，然后在 Project 里直接把我们实验的目标代码项目工程 Load 进来，通过 Input Setting CCS Project 进行我们的实验工程。

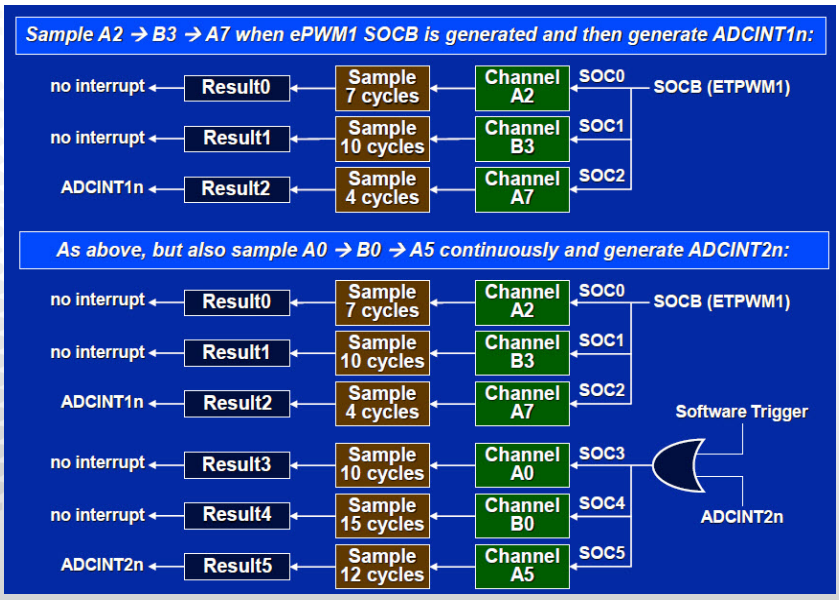
2.7 控制外设

✧ 实验 2 : 如何利用 PWM 生成 波形



Piccolo 系列的 ADC 模块有一个 12 位的 AD 转换器，2 个片上采样保持电路和 2 个选择开关组成，每个选择开关支持 8 个通道，最多可以支持到 16 个通道，所有 ADC 通道输入支持 0—3.3V 的量程输入。ADC 通道的触发采样时间的设置或通道触发事件的设置是由 ADC 控制寄存器完成，触发事件可以由软件、CPU 的 Timer，EPW 的 SOC 事件和外部管脚来触发 ADC 通道采样。每次触发一个通道采样完成以后，采样的结果会存入结果寄存器，同时产生对应的 EOC 事件，设置相应的 EOC 事件会产生对应的中断，ADC 的中断会被 CPU 响应。

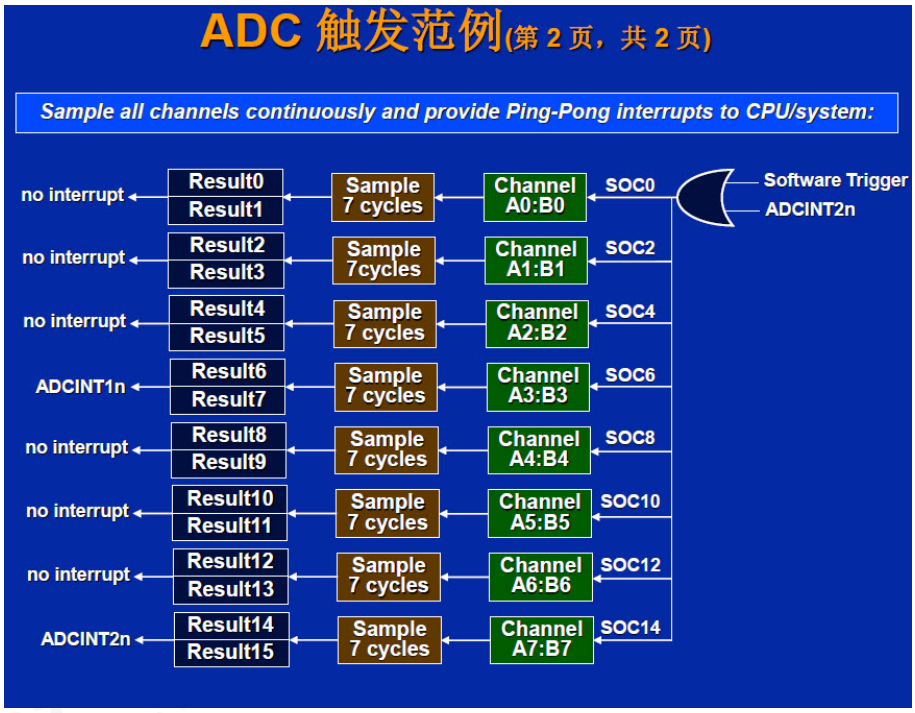
通过三个范例介绍 ADC 触发设置。



第一个范例，想触发 A2、B3、A7 三个通道，触发事件用 ePWM1

的 SOCB 同时产生 ADC 中断 1，如下面图所示，A2 采样时间设置 7，B3 设为 10，A7 设为 4，当接收到 ETPWM1 的 SOC 触发以后，A2、B3、A7 顺序完成采样或转换，将结果存于寄存器 0、1、2，当整个三个通道的触发和采样完成以后，产生中断 1。

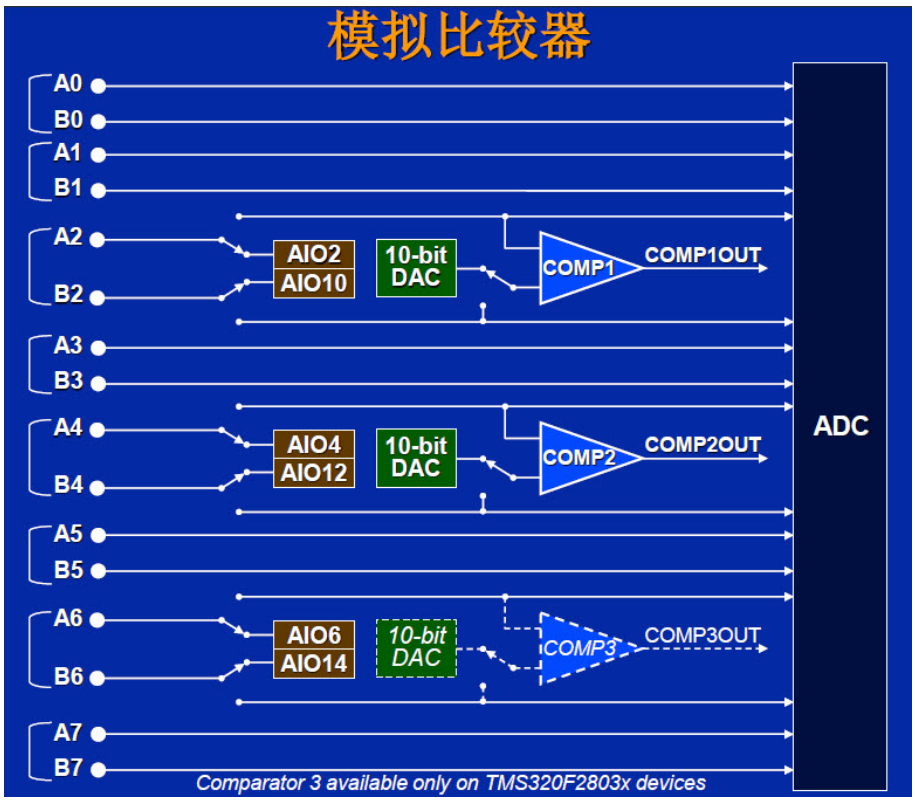
第二个范例，除了要完成 A2、B3、A7 的转换以外，还希望完成 A0、B0、A5 的连续转换，同时产生 ADC 中断 2。A2、B3、A7 的设置和第一个范例相同，A0、B0、A5 的触发第一次由软件触发来完成，当第一次软件触发完成以后，产生对应的 ADC 中断 2，之后由 ADC 中断 2 来继续触发 A0、B0、A5 的转换，最后放入结果寄存器 3、4 和 5，这样同时又产生 ADC 中断 2，再回过来，ADC 中断 2 又会继续触发 A0、B0、A5 的转换，这样会形成 A0、B0、A5 的连续转换。



第三个范例，同步采样的设置，也是同样完成 A0、B0 到 A7、B7 顺序的连续转换。首先设置它的采样模式是同时采样模式，第一次触发也是由软件触发来完成，同时产生 ADC 中断 2，当 ADC 中断 2 完成以后，由 ADC 中断 2 来触发整个 ADC 通道转换。A0 和 B0 是同时采样，A1、B1 是同时采样。如此类推，到 A7、B7 也是同时采样，这样所有的采样完成以后，顺序放入结果寄存器 0—15 里。

在 Piccolo 系列的 ADC 模块里，除了支持 AD 转换以外，还支持模拟比较器功能。从下图可以看到（PPT：模拟比较器），片上有三个模拟比较器，其中 A2、B2 是一对，A4、B4 是一对；A6、B6 是一对，构成三个模拟比较器，每个模拟比较器的正向输入直接接到 A2 通道，反向输入可以接到 B2 或内部 DAC 上，可以

通过控制寄存器来设置，这时候反向输入是否接到外面的 B2 通道或内部 10 位 DAC 来设置对应的比较参考。当正向输入或反向输入进行比较以后，根据值来输出对应的“1”或“0”提供给比较器的输入。

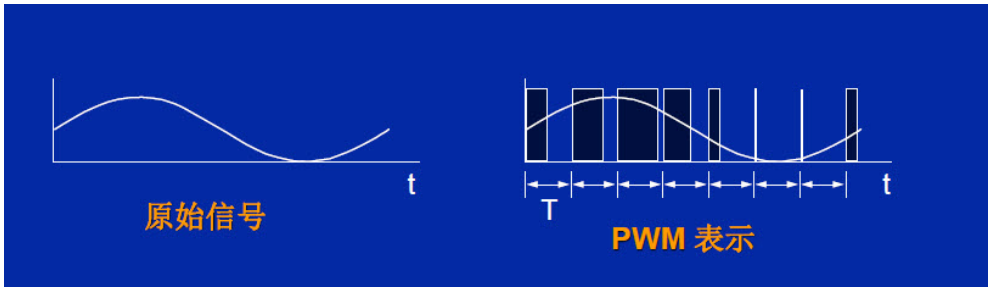


ADC 寄存器主要包括控制寄存器、中断 SOC 选择寄存器和中断选择寄存器采样模式设置寄存器。中断控制寄存器主要功能是完成模块和复位 ADC 使能，通道状态显示、基准选择（可以选择是内部基准还是外部基准），以及控制中断生成。ADC SOC 控制寄存器主要是设置每个通道触发源，在 SOC 里触发哪个通道及每次采样时间；中断 SOC 选择寄存器主要是选择 ADC 中断 1 或 2 是否作用触发器用于 SOC；采样模式寄存器是设置 ADC 是做顺序采样还是同时采样，中断选择寄存器设置一个中断数目，比如说每次 ADC 触发以后可能会产生 EOC0—15，那么 ADC 中断 1—9 选择哪一个 EOC 作为源。结果寄存器主要是完成每次 ADC 的转换，其结果放在对应的结果寄存器里。SOC0 对应结果寄存器 0，SOC1 对应结果寄存器 1，依此类推。

下面介绍 PWM 模块。

PWM（脉宽调制）是一种用连续脉冲序列产生固定信号的方案，其特征包括固定的斩波频率，固定的脉冲幅度，脉冲宽度和瞬时信号振幅成正比，所以整个周期下来 PWM 能量要和原始信号的能量是相同的。比如我们要产生图中左下方的原始信号，通常的数字信号是无法输出的，通过右边 PWM 方式来产生对应的原始信号。它通过固定的载波频率来调节它的宽度，最后它的能量是和右边的原

始信号是相等的。



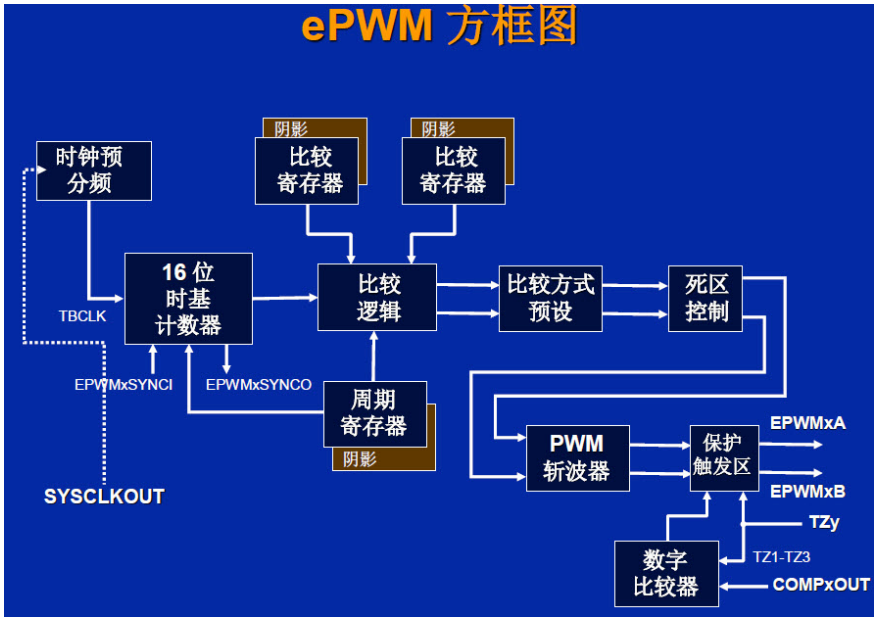
PWM 通常和输入功率开关器件配合使用，通过控制功率开关的开通或关断来得到所需要输入输出的电流或电压。功率开关器件是晶体管，其特性是在放大区难以控制，但在饱和区中易于控制其开通或者关断。PWM 信号是一个数字信号，易于用 DSP 输出。

从左图可见，比如有一个直流源，通过 PWM MOSFET 开关想输出一个期望的 Sin 或 Cos 的信号，但是很难从理论上来实现这种输出。通常通过 PWM 调制的方式来得到对应的近似期望的信号。

ePWM 模块信号和连接。

Piccolo 系列芯片每个芯片有多个 ePWM 模块，有输入输出信号和它进行连接。ePWM 模块可以接收模拟比较器的输入，可以接收其他模块同步信号的输入，可以接收外部 Trip Zone 保护触发信号输入，同时，ePWM 模块可以输出 ePWM 的 TZ 中断、ePWM 的 SOC 中断 还可以输出 SOC 对应 ADC 模块的触发。它有 ePWMxA 和 ePWMxB 作为 ePWM 信号输出，通过 GPIO 口输出到外部来开通和关断对应的功率开关，通过 ePWM 输入输出来同步上一个模块和控制下一个模块。

ePWM 模块方框图。



每个 ePWM 模块由几部分组成，时基计数器，通过时钟预分频来设置输入时钟基准，也可以通过 ePWM 同步输入来设置计数器和其他 ePWM 模块技术同步。系统时钟通过时钟预分频可以配置 1 分频、2 分频或 4 分频等提供它对应的时钟源给 16 位的时基计数器。比较寄存器分 A 和 B，每个通道有自己的比较寄存器，有 Compare A 和 Compare B，比较寄存器都是带阴影寄存器的，我可以设置比较寄存器的值，同时可以设置比较寄存器在什么时候有效。如果设置是立即有效就直接写入比较寄存器，如果写成当计数器 =0 或者计数器等于周期的时候有效，那它就先存在对应的阴影寄存器中，当满足计数器 =0 的时候，比较寄存器从阴影寄存器中读入对应的值。比较逻辑模块是控制比较寄存器，当满足比较寄存器的值和计数器的值相等的时候，PWM 通道输出的事件，周期寄存器设置 PWM 的周期或 PWM 的频率,它和比较寄存器相同,也是有对应的阴影寄存器。比较方式的预设模块，是设置比较产生以后，它是翻转还是设置成高，还是设置成低，此处控制寄存器部分主要是用于控制两个设置为互补通道的死区插入时间，或死区能使或不能使。PWM 载波寄存器主要用于在对应的 AD 通道上载入一个更高的频率，保护触发功能模块用于当接受到外部的 TZ 信号或接受到数字比较器的信号以后来关断或不关断对应的 PWM 通道的输出。数字比较器是接收外部的 TZ 信号和对应模拟比较器输入，来进行逻辑控制，当两个输入同时有效或一个有效一个无效时对应输出状态，由数字比较器来完成。

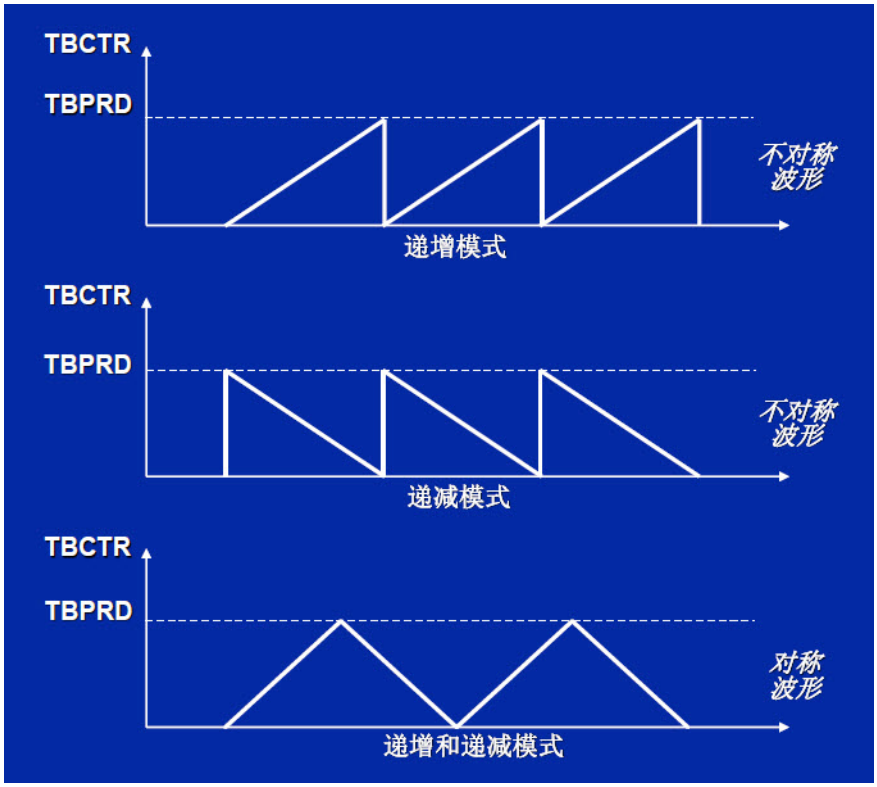
ePWM 时基子模块，主要由三部分组成：

1、周期寄存器,由周期寄存器设置 PWM 周期或频率。时基计数器就是计数，当接收到一个时钟是增加或减的时钟计数。

2、时钟前置分频，由系统时钟过来，通过时钟前置分频来进行分频，2 分频或 4 分频作为时基提供给计数器。

3、控制寄存器，同时有对应的控制寄存器来设置计数模式是上升计数还是

下降计数，或者上升 / 下降计数。



ePWM 时基计数模式，主要有三种：

1、递增方式，从 0 计到周期，再从 0 计到周期。
2、递减方式，先从周期往下计，计到 0 以后，再从周期往下计到 0。
这两种方式都是不对称的计数方式，生成的波形是不对称的。

3、递增和递减方式，对称波形的产生方式，递增和递减是先从 0 计到周期，再从周期减计数，一直计到 0，再回过头来往复计数方式。

ePWM 相位同步。

每个芯片内部都有多个 ePWM 模块，每个模块之间可以通过对应的信号来实现相位的同步或移相。一般相位同步或移相的模块必须要顺序连接，比如说 ePWM 模块 1 和 2、3 要顺序连下去，不能 ePWM1 和 3 连接。首先可以设置 ePWM1 为一个主模块，它会发出对应的同步信号给 ePWM2 的模块，ePWM2 再发出信号给 ePWM3 模块，这样实现 ePWM 模块 1、2、3 的同步或移相。如果在移相寄存器里设置移相寄存器 =0 也是同步的功能。

如果在移相寄存器设置对应的一个值和实际相应的移相控制，比如说周期寄存器 =600，移相寄存器里设置不同的值会实现相应的移相。比如在 PWM 模块 2 和 1 有一个 120 度的移相，我们可以把移相寄存器设置成 200，相当于它有一个移相 120 度。如果在模块 3 里设置成 400，那相当于有一个 240 度的移相。

ePWM 比较子模块。

通过 ePWM 比较子模块可以实现 ePWM 通道 A 和 B 的输出。

ePWM 比较事件波形。有几种不同的方式，当递增模式时，ePWM 比较事件等于 0，A 的比较和 B 的比较，等于周期和等于 0 是相同的。递减模式的时候也一样，有等于 0 或周期，这两个是相同的，等于比较 A、等于比较 B，总共有三个事件。

在递增和递减模式下总有 ePWM=0，ePWM 上升 A 的比较，ePWM 上升 B 的比较，等于周期，ePWM 下降计数时等于 A 的比较，下降等于 B 的比较，所以总共会看到 6 种比较事件。

ePWM 比较方式设置子模块。

从前面 ePWM 比较子模块中可以看到，当计数器和比较寄存器相等时，会产生对应的比较事件，当等于 0 或等于周期的时候也会产生相应的比较事件。比较方式设置子模块就是指来设置当我们产生对应的比较事件以后，对应的 A 通道和 B 通道的表现行为，当产生对应的比较事件以后，我们可以设置 A 通道等于高，等于低，或者是不做任何动作或切换。

比较方式预设动作针对每一个 ePWM 模块，其 A 通道和 B 通道有不同的预设动作。在实际计数器可以等于 0、A 比较、B 比较、周期的时候都会产生对应的动作。我们还有软件强制方式来产生对应的动作。下表中打 “X” 的方式就表示不执行任何动作。向下的箭头表示 A 通道或 B 通道输出低电平，向上的箭头表示我们在 A 通道和 B 通道上输出高电平，在方框中有 “T” 的表示在 A 通道或 B 通道做触发翻转，每次比较事件进来以后都会在 A 通道或 B 通道上产生表中所有的这种动作。

ePWM 比较方式预设动作
(针对 EPWMA 和 EPWMB)

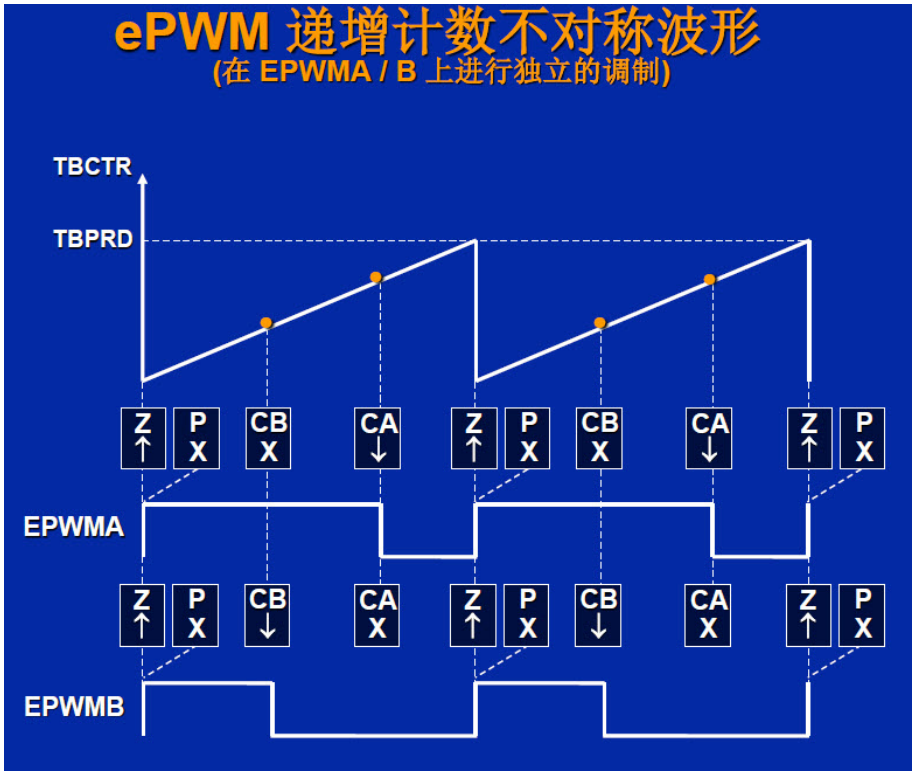
软件强制	时基计数器等于：				EPWM 输出动作
	零	CMPA	CMPB	TBPRD	
SWX	ZX	CAX	CBX	PX	不执行动作
SW↓	Z↓	CA↓	CB↓	P↓	清除低电平
SW↑	Z↑	CA↑	CB↑	P↑	设定高电平
SWT	ZT	CAT	CBT	PT	触发

ePWM 递增计数不对称波形。

下面我们通过几个例子来说明如何在 ePWMA 或 ePWMB 通道上进行调制。

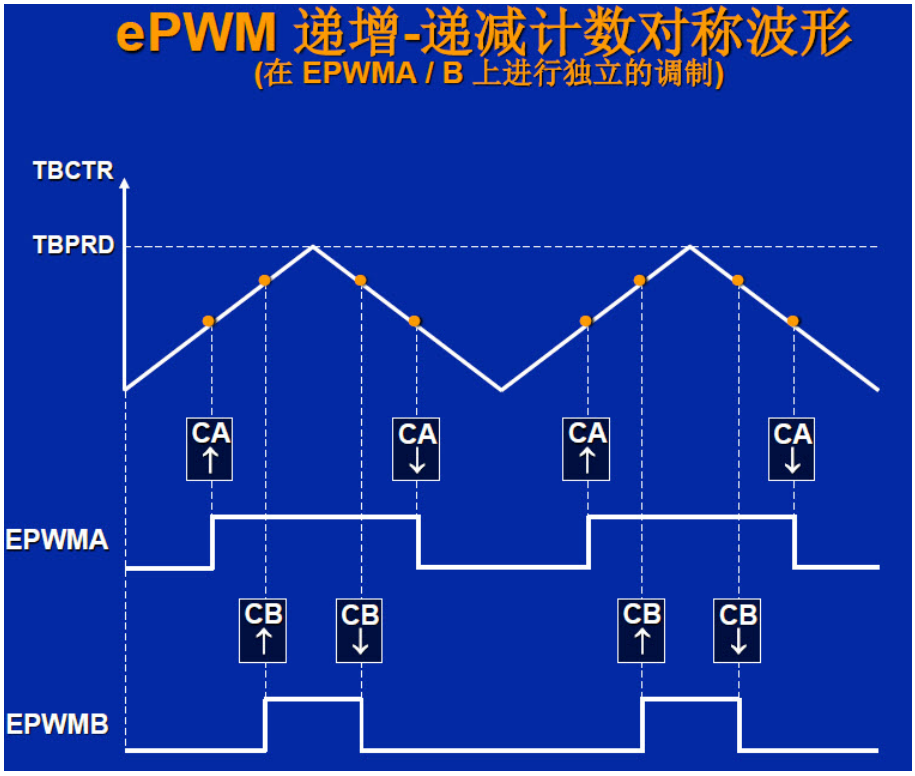
第一个例子，在 ePWM A 和 B 上进行独立调制，计数方式是递增的方式，有几个不同的比较事件,等于 0、A 的比较、B 的比较的时候,我们希望 A 通道 =0 时，它是制成高电平,当Compare A产生的时候 就是计数器等于Compare A的时候，就是 A 比较事件产生的时候，A 通道变成低，当等于 0 的时候 A 通道变成高，A 的比较产生以后变成低，如此每一个周期会循环下去，在 B 通道时，我们希望当等于 0 的时候同样变成高，B 的比较产生的时候变成低。B 通道同样也会产生对应的，等于 0 变成高，B 通道产生的时候变成低，如周期，一个周期循环下去。

ePWM 递增计数不对称波形。



怎么在 A 通道上进行独立的调制，就是说 A 或 B 的比较同时来控制 A 通道波形的产生，在 A 通道上设置等于比较 A 产生的时候或比较 B 产生的时候，在 A 通道进行对应的输出，A 比较产生的时候让 PWM 输出是高，B 比较产生的时候让 PWM 输出是低。这是 A 通道上的波形,在 B 通道上等于 0 的时候进行翻转，第一个周期等于 0 的时候，B 通道设置成高；第二个周期等于 0 的时候，PWM 的输出变成低，同样是一个周期一个周期循环下去。

ePWM 递增 - 递减计数对称波形。



我们再继续演示 ePWM A、B 在递增、递减方式下如何进行独立地调制。它的计数方式是递增 - 递减，有几种事件，有等于 0、A 的上升、B 的上升、A 的下降、B 的下降和周期几种比较事件。来设置 A 通道在 A 比较上升时变成高，同时 A 下降的时候变成低；B 通道是在 B 上升的时候设置通道的输出变成高，B 下降时通道输出变成低，从 A 通道和 B 通道波形可以看出，他们分别进行独立的控制。

ePWM 递增 - 递减计数对称波形，这一页给大家演示在 ePWM 上进行独立的调制。

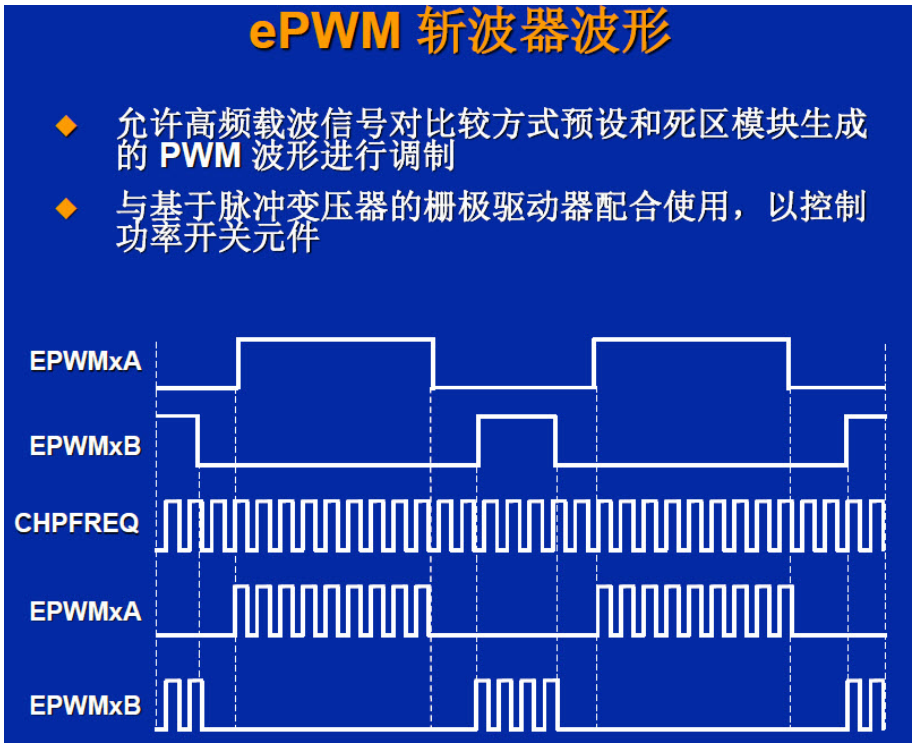
同样采用递增 - 递减计数方式，A 通道它既受 Compare A 的控制，同时又受 Compare B 的控制，就是当 Compare 上升比较事件产生的时候，设置 A 通道的输出为高，当 Compare B 下降比较事件产生的时候设置 PWM 通道 A 为低的输出，这样一个周期一个周期下去。B 通道设置在等于 0 的时候变成低，等于周期变成高，就是说它是在一个递增 - 递减的方式下是个 50% 的 Duty（占空比）输出方式。

ePWM 死区子模块。

前面我们介绍了 ePWM 两个通道输出控制，在输出控制的时候，如果设置成互补方式我们需要在两个通道之间插入对应的死区。为什么需要在 ePWM 模块中加入死区模块？主要是在功率器件中对功率器件进行保护，比如逆变桥中有上桥和下桥，通常情况下功率器件开通速度和关断速度是不相等的，接通速度通常比关断速度要快，如果两个功率器件同时导通的话会使电路短路，这样会把功率

器件给烧坏。所以，我们要在功率器件电路中让一个器件开通会比关断要晚，就是保持两个器件要有同时关断一段时间。

这是 ePWM 斩波器波形示意图



从中可以看出 ePWM 斩波器的作用。第一，它能使高频斩波信号对比较方式预设和死区模块生成的 PWM 波形进行调制，主要目的是为了应用进入脉冲变压器的隔离驱动的使用，来控制功率开关元件的开通和关断。通常低频信号很难通过脉冲变压器或者需要把脉冲变压器的体积设置得非常大。我们通过高频信号能够减少脉冲变压器的体积。下面的图可以看到，上面两个 A、B 是它的基波，下面是 PWM 斩波信号，比如说上面的 A、B PWM 信号是 1K，我们会把下面 1M 或更高频率的斩波在 A、B 两个通道上面，最后形成两个通道的波形，在 A、B 上有对应的斩波信号。

ePWM 斩波频率设置可以从几兆到十几兆，它的 Duty(占空比) 也可以设置，从 25%、50%、75% 来进行设置，比如我们设置 50%，最后在 A 和 B 上的 Duty(占空比) 会比原来你所设置的 Duty(占空比) 要小一倍。

ePWM 保护触发子模块。

ePWM 保护触发子模块的主要作用是实现对 ePWM 输出的控制，接收对应的触发保护信号以后，要把 A 通道或 B 通道来进行屏蔽输出或不做任何动作，它可以接收的信号有来自外部的，Trip Zone 就是 TZ1—TZ3 的信号，也可以接受数字比较器里的信号。

保护触发特性，ePWM 子模块中保护触发子模块主要用于实现对 ePWM 通道 A、B 引脚的快速、时钟无关逻辑路径的保护。通常我们也可以通过中断服务子程序来响应过流或短路的故障，但中断延时有可能会起不到保护硬件的作用，因为中断的进入需要很多的 SYNC 来完成，所以我们会通过中断触发子模块来实现针对短路或过流状况的快速触发，比较适合应用限流操作的逐周期的控制。

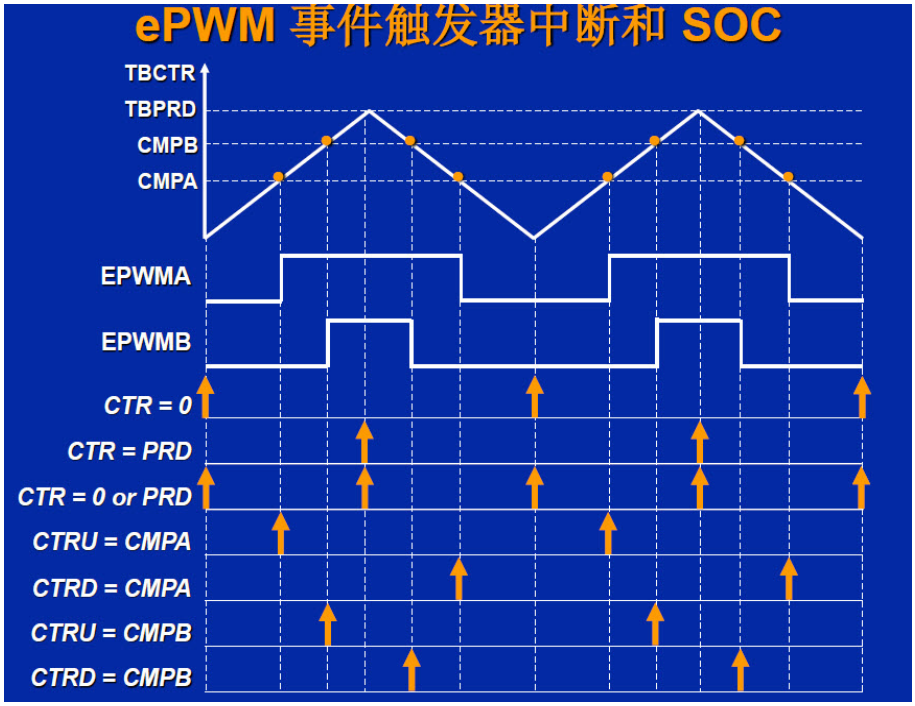
所有 Piccolo 系列保护触发子模块支持两种模式，适合限流操作的逐周期跳变和针对短路或过流状况的单触发跳变。逐周期模式是当每一次保护触发产生以后，在当周期内会快速把 PWM 的输出关断。当下一个周期如果触发条件解除，它会继续按照设定的 PWM 设置来输出 PWM 波形。单触发模型是指一旦接收到外部的保护触发信号，会把对应的 A 或 B 通道输出永久关掉，只有软件继续干预才会允许 PWM A 或 B 的输出。所有 Piccolo A 或 Piccolo B 系列会支持 6 个 TZ 信号的输入，其中 TZ1、TZ2、TZ3 有对应的外部引脚，可以接到对应的过流传感器、过压传感器上面，来实现过流、过压或温度保护，TZ4、TZ5、TZ6 是接到内部的保护事件，TZ4 用于 QEP、TZ5 用于时钟控制，TZ6 用于 CPU 控制，当 QEP 没有对应信号输入时会关断 PWM 输出。TZ4 可以允许或屏蔽，当模块没有 QEP 的时候会把 TZ4 给屏蔽掉;TZ5 和 TZ6 主要用于当时钟丢失或 CPU 不工作时，也会把 PWM 模块输出，所以，当你在做实时调试的时候，如果仿真器断掉的时候，我们会把 PWM 通道关掉。

当每次接收到外部 TZ 信号时或数字比较器的输入信号，会触发对应的 ePWM 的 TZ 中断，ePWM 的 TZ 中断会提供给 CPU

保护触发特性，ePWM 保护触发子模块是一种提供快速对 ePWM A、B 的输出进行关断的一种方式，当接收到外部的保护触发信号以后，会立即关断 A 或 B 的输出，当然我们可以通过中断服务子程序，采用软件响应的方式来实现这种过流、短路等故障的保护，但是中断延时会造成硬件保护的作用降低，可能对硬件保护不能起到很好的快速响应，所以保护触发子模块它会支持针对短路或过流状况的快速触发响应。第二，它可以提供给做限流操作的逐周期控制。

ePWM 事件触发器子模块， ePWM 可以产生很多种事件，有比较的，等于 0 的，等于周期的事件，每个事件产生以后，都可以触发对应的 SOC，这个 SOC 可以提供给 CPU、ADC 模块。

ePWM 事件触发器中断和 SOC。



从上图见，在上升或下降计数模式下，它有多少事件可以触发，等于 0，等于周期，等于 0 或者等于周期，上升 A 的时候，上升 B 的时候，下降 A 和下降 B 的时候，都会触发对应的事件，也可以作为一个 SOC 提供给 CPU。

ePWM 数字比较子模块，在 F2802x 和 F2803x 新一代 C2000 的芯片中都提供一个数字比较模块。数字比较模块的功能主要是对外部 TZ 信号或模拟比较器的比较输出信号进行数字逻辑控制，来提供给 PWM 模块的保护触发子模块。数字比较器子模块的输入信号可以为模拟比较器的输出，可以为 TZ1、TZ2、TZ3 的输入信号。每一个数字比较器对这几个输入信号进行对应的比较操作，输出相应的事件，这些产生的事件可以提供给时基子模块、保护触发子模块和事件触发器子模块，它来产生对应的同步信号，产生对应的中断和强制触发、SOC，提供给 ADC 等等。

每个输出事件可以由数字比较器中的一个控制寄存器来设置其比较器输入有效信号是哪一个信号，比如设置成 DCAEVT1、DCAEVT2 或者 DCBEVT1、DCBEVT2 这几个输出事件 我们会分别对每一个事件设置有效的输入信号。例如，我们设置 DCAEVT1 的有效输入信号时，TZ1 和 DCMP1OUT，就是模拟比较器 1 的输入信号；当 TZ1 和模拟比较器输出 1 同时有效时才产生 DCAEVT1 的触发事件。

高分辨率 PWM。

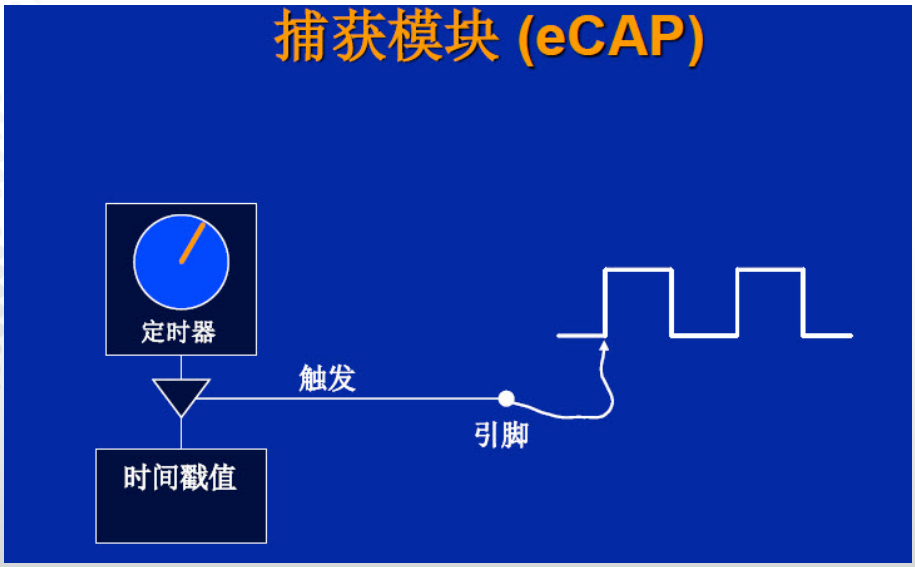
在 F2802x 和 F2803x 和以后所有 C2000 的 PWM 中，TI 都提供高分辨率 PWM，与常规 PWM 最大的差异是，常规 PWM 精度与时钟频率有关系，当时

钟频率是 60MHz 时，PWM 时间间隔是 16.67ns。高精度的 PWM 是一个和时钟无关的 PWM 信号，可以提供给一个固定的 PWM 精度，在 F2802x 中，最高的精度可以到 150ps。PWM 每变化一个间隔，它的分辨就是 16.67ns，下面对每一个 16.67ns 时间间隔分隔成更小的微间隔，它的精度到 150ps。通过这种方式，显著地提高了传统 PWM 方式的分辨率。高精度 PWM 支持比较周期和相移，能够实现 Duty（占空比 ）的高精度，实现频率的高精度，实现移相控制的高精度。在通常情况下，当 PWM 分辨率降到 9—10 位时通常会采用高精的 PWM，例如在 60M 时钟时，如果用 100K 以上的 PWM 频率，这时候 PWM 精度就会降到 10 位，传统的 PWM 就无法支持更高精度的 PWM 输出，这时我采用高精的 PWM 输出就会显著提高 PWM 输出精度来满足输出的控制要求。

ePWM 控制寄存器。

和 ePWM 相关的常用控制寄存器，时基控制寄存器，通过时基控制寄存器我们可以设置寄存器的计数模式（递增、递减或递增 - 递减等方式），可以设置 PWM 分频，通过控制比较寄存器来设置比较寄存器的加载模式或操作模式是立即有效还是在某一个时刻才 Load 进来，通过比较方式预设控制寄存器来控制 PWM 比较事件产生以后，在 PWM 通道上的动作。死区控制寄存器可以来配置死区是否能使它的极性选择或死区的设置方式，PWM 斩波寄存器可以配置是否能使斩波频率和占空比。通过 TZ 控制寄存器来控制保护跳变控制的设置，是启动还是屏蔽，当出现跳变时它的动作是强制高电平、低电平还是不执行任何动作。通过事件触发寄存器可以设置 PWM 产生对应的事件时是否能使产生对应的 SOC，比如我们可以选择 SOC A 是哪一个触发事件来产生，SOC B 是哪一个触发事件来产生，中断是哪一个触发事件来产生，对应的触发事件是否启用或屏蔽也可以由这个寄存器来设置。数字比较跳变选择寄存器中，可以选择数字比较器 A 或 B 是高电平还是低电平输入，就是输入的时候是高电平有效还是低电平有效，同时可以选择它是作为哪一个数字比较触发输出。

捕获模块 eCAP。



捕获模块主要作用是用来捕获输入引脚上面两个有效电平的时间间隔，eCAP 模块方框捕获模式。C2000 中的 eCAP 模块有两种方式，捕获模式和辅助 PWM 模式。

捕获模式——当设置为捕获模式时，可以通过事件预分频或极性选择，来配置捕获模式的工作方式。例如，我在事件预分频中设置预分频为 4 时，它就把 4 个寄存器都利用上，我们会设置极性选择器寄存器的 1、2、3、4 有效极性什么样。例如我们设置有效极性是高电平，第一个高电平进来以后会触发对应的事件，把事件计数器里的值放到捕获寄存器 1；当第二个有效的高电平进来以后会把对应的计数器值放到寄存器 2 里；第三个进来放到寄存器 3；第四个进来放到寄存器 4 中，这是事件预分频为 4 的时候。如果我们设置事件预分频为 1 的时候，第一个进来以后会把对应的计数器放在寄存器 1 中，当第二个进来它会把前面放的覆盖掉，这是事件预分频设置为 1 的时候，如果是 2 的时候，那就是第一个放在 1 里，第二个放在 2 里，第三个就会把第一个再覆盖掉，这样依次会循环下去。极性选择里可以选择极性的有效电平是高电平或低电平，高电平或低电平都是有效电平。我们可以通过放置在寄存器里的值来计算出两个延口之间的间隔时间。

eCAP 模块还支持另外一种功能叫辅助 PWM 功能 ,这时候前面有 4 个寄存器，就是 eCAP1—eCAP4 寄存器 ,当作为辅助 PWM 模式时 ,eCAP1 作为周期寄存器，eCAP3 作为周期寄存器的阴影寄存器，eCAP2 作为比较寄存器，eCAP4 作为比较寄存器的阴影寄存器，这样它的工作原理就和 ePWM 是类似的。当计数器的值和周期寄存器的值相等时，会产生对应的一个频率的 PWM，当和比较寄存器的值相等时，它会触发对应的事件，根据你设置，这时候的电平是高或是低做一个翻转。

什么是增量式正交编码器？

C2000 很多系列上有 QEP 模块，即增量是正交编码器的接口模块，QEP 是一个数字位置传感器，增量运用正交编码器的原理，它由光源、光电传感器和安装在传动轴的码盘组成。正交编码器的分辨率是由码盘上的线速决定，线速就是指码盘上刻的缝隙个数，每个缝隙之间间隔就是它的分辨角度 è 角。

光电传感器安装的位置是固定它的间隔是 1/4 分辨率角度，当 LED 光源发出光，被光电传感器检测到以后，输出右边的 A、B 通道信号，A、B 通道是一个正交信号，他们俩的相移是 90 度，每个信号 A 和 B 的周期分辨率是光电码盘的分辨率，就是 è 角。通过 A、B 两个通道来实现四分频的进度，就可以提供 1/4 的 è 角分辨率。

如何利用正交信号确定位置？

从前面增量式正交位置编码器原理能看到，位置的分辨率为 1/4 的 è，它有两个输出信号，A 通道和 B 通道的频率是相同的，只是相位相差 90 度，这样我们可以通过正交解码器的状态机来确定旋转轴的旋转方向和它的旋转速度，当每

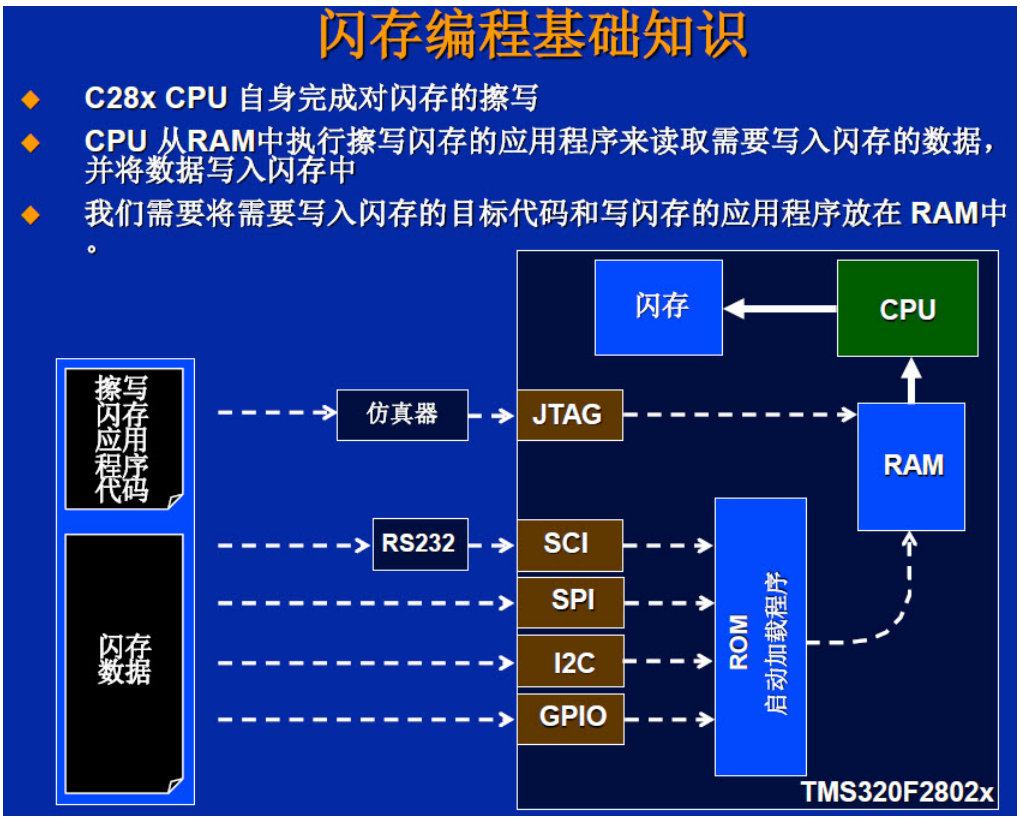
次得到一个有效信号，比如说 00—10 的变化，我们认为这光电码盘转动了 1/4 的 è 角，如果是 00—10 我们就认为是个增量计数器，10—00 我们认为它是一个减计数器，这样它才在计数器里实现加 1 或减 1 的动作。

eQEP 模块的连接。

eQEP 模块主要由几部分组成:时基 ,eQEP 模块配置部分 ,eQEP 模块解码部分，通常 eQEP 模块有四个外部信号的输入，A、B 输入，Index 信号输入、Strobe 信号输入，通常 A、B 两个信号就可以来实现 eQEP 模块的方向、判断和速度的计数或位置的计数，Index 和 Strobe 是一个辅助的功能，当 eQEP 模块旋转一圈，很多 eQEP 模块（正交编码器）会输出一个 Index 信号提供给 QEP 模块。在某些伺服应用控制上，会设置一个形成开关，它会触发一个 Strobe 信号提供给 eQEP 模块。

2.8 闪存编程基础知识

C28x 的 CPU 是通过自身来完成对闪存的擦写。CPU 首先从 RAM 中执行擦写闪存的应用程序，来读取写入闪存的数据，然后擦除闪存，并把读入的目标代码写入闪存中，所以我们需要将写入闪存的目标代码和写入闪存的应用程序放到片上的 RAM 中来执行。



从上图可见，我们借助仿真器和对应的外设接口来实现对片上闪存的编程，不论通过仿真器，还是通过对应的外设接口来实现片上闪存的擦写，其操作原理都是相同的。首先我们需要将擦写闪存的应用程序代码通过对应的接口放置到片上 RAM 中，然后通过对应 RAM 中执行擦写闪存的程序来读取目标代码，通过 RAM 或 CPU 来实现对闪存的编程。如果想通过外部接口 SCI、SPI、I2C、GPIO 或 CAN 来实现对片上闪存读写，首先要执行一个 ROM 启动加载程序，所有 C2000 芯片通过 TI 工厂出来以后，都在 ROM 中放置启动加载程序，通过该程序实现 SCI、SPI、I2C、上位机或其他 MCU 进行通讯，把擦写闪存应用程序或闪存数据加载到 RAM 中去实现闪存的编程。

闪存编程通常分为三步：

第一步擦除，将所有闪存的位先全部设为“0”，然后再全部设为“1”，通过这种方式可以把所有片上闪存进行擦除。

第二步写入，根据目标代码，如果目标代码是“0”，我们就把对应的这一位从“1”

变成“0”，如果目标代码对应的位是“1”，那么这一位就保持不变，同时完成所有闪存的写入。

第三步是验证，这是可选步，对闪存中写入的数据和目标代码进行比较验证，看看是否写入的是正确的目标代码。

需要注意的是，所有闪存最小的擦除容量就是一个扇区（(4Kw 或 8Kw)），扇区大小因不同的芯片规格而不同。在编程时，最小的编程写入长度为一个位！所以如果想对片上闪存进行编程，首先要擦除一个扇区，然后根据目标代码对每一个位分别进行操作。重要的是在擦除步骤中不要掉电：假如 CSM 密码碰巧全部为“0”，则 CSM 将被永久锁定！当然，这种事件发生的概率非常小，因为擦除步骤是逐个扇区进行的。

闪存编程例程。

C2000 可以通过多种方式实现片上闪存的编程：

1、通过 CCS 嵌入到闪存编程应用程序实现闪存的编程，通常这种方式用 JTAG 实现。

2、通过第三方提供对应的闪存编程的应用程序来实现对片上闪存的编程，例如来自 Spectrum Digital 公司的 SDFlash JTAG，可以通过 SDFlash 来实现对片上闪存的编程，但利用 SDFlash，必须使用 Spectrum Digital 仿真器；还有 Signum System 闪存编程应用程序同样也要应用其相应的仿真器；此外，Black Hawk 闪存编程上位机的应用程序，也需要用到 Blackhawk 的仿真器。

3、通过 Spectrum Digital 提供的 SDFlash 的串行应用软件实现片上闪存编程，可以通过 C2000 的 SCI 引导实现片上闪存的编程。

4、一些公司提供专用的编程器，比如 BP Micro 编程器、Data I/O 编程器等。BP Micro 有一种编程器能提供同时写两片、四片或八片的专用编程器。

上述方式都是提供 JTAG 对应的接口来实现对片上闪存的编程。TI 还提供对应的 Flash API，允许创建您自己的定制例程。运用由 TI 提供的闪存 API 算法来创建项目，可以通过 SCI ROM 的加载方式或者 SPI 等方式启动加载方法去实现对片上闪存的编程，同时还可以将闪存编程的程序嵌入到您的应用程序中。

TI 网站提供了所有例程的链接 (http://www.ti.com/c2000)， 电子工程师可以到 TI C2000 网站下载不同芯片的 Flash API 的算法，在 Flash API 的算法中，TI 会提供一个例程来告诉您怎么将 Flash API 算法嵌入到您的应用程序中，以实现对片上闪存的擦除和编程。

CCS 片上闪存编程器

如何应用 CCS 嵌入片上闪存编程器，左图显示我们可以设置它的时钟频率，可以设置对哪一个扇区进行闪存的编程。闪存分很多个扇区，例如有 8 个扇区，你可以设置只要编 A、B、C 或任何扇区进行擦除或编程。右图显示我们可以可以设置通过 K0—K6 设置闪存密码，进行加密，通过设置的密码来对已经加密的芯片进行解密。

闪存加密模块（CSM）

我们通过闪存加密模块对片上存储器的访问进行加密。通常下面所有闪存，RAM、OTP、ADC/OSC（晶振）校准数据、存储器都可以被加密模块加密。仅加密存储系中运行的代码能够读写同样加密存储器中的数据，例如 Flash 被加密了，我们只能通过 Flash 来读写 L0 里的 RAM 值。同样我们只能通过 L0 来读写 Flash 里的值。如果未加密 RAM 的执行代码，它是不能读写加密区的 RAM 和 Flash 内容。

一个加密程序或加密芯片，我们只能先把对应的 Flash API 放到未加密的 RAM 里，对这个芯片先进行解密，才能进行下一步的擦除或编程动作。所以，所有其他方式来读写加密存储器的数据都不会被屏蔽掉，不管通过 JTAG 仿真器、通过 ROM 里加载应用程序、通过外部不受加密 RAM 来读写，还是执行代码来读写闪存等加密里的程序是没办法来做到的。

CSM 密码。

闪存加密模块是由128位密码构成,位置放在0x3F7FF8—0x3F7FFF8个字中,共128位的内存中。128位的用户定义密码存储在闪存中(就是闪存最后的128位中),128位密钥(KEY)寄存器主要用于器件的锁定和解锁。这个密钥寄存器的位置是在AE0—AE7中,这个寄存器是受到保护的,每次想去操作这个寄存器要首先执行“EALLOW”动作,才能对密钥寄存器进行读写。

CSS密码匹配流程。

那么,如何对片上闪存进行解密?当一个闪存芯片复位以后,如果有密码就处于安全的加密状态。我们首先对密码进行解读,就是对0x3F7FF8—0x3F7FFF这8个字进行虚读,然后判断这个密码是否全部为0,如果全部为0就不执行下一步动作,因为这个芯片是被永久锁住的,没有办法对其进行任何擦写。如果不是全0的话,我们可以进行下一步的动作,看这个密码是不是全F,如果是全F的话,说明这个芯片是未加密的,那么可以对这个芯片进行任何的动作,这个芯片可以读、写、擦除。

如果密码既不是全0也不是全F的话,那就说明这个芯片被加密了,我们首先要把密码写入对应的密钥寄存器里,就是AE0—AE7这8个寄存器中。这8个寄存器是受到保护的。如果我们写到这8个密钥寄存器的值正好和存在密码寄存器里0x3F7FF8—0x3F7FFF这8个字的值完全相同,那就说明密码是正确,把芯片解锁,然后可以执行下一步的动作,会对芯片上的闪存或RAM进行擦除或读写。

实验3:闪存的编程。

我们通过一个实验来了解闪存编程的过程。首先我们会熟悉在闪存中运行目标程序,然后再熟悉如何通过CCS嵌入闪存擦写程序来进行片上闪存的编程,这个闪存编程的例程是我们提供的一个PWM触发ADC或读ADC寄存器的例程,与我们实验2的例程是完全相同的,只不过实验2的例程是在RAM中运行的,实验3的例程可以在闪存中运行。当我们把程序写在闪存中以后,可以把仿真器断开,这个程序会继续运行。

闪存扇区块。

下面我们对实验中的闪存定义和连接命令文件进行解释说明。首先闪存分为几部分:第一部分是代码写入部分,就是定义闪存的起始长度和地址;第二部分是密码位置部分,就是3F7FF8开始,长度是一个8的闪存位置;第三部分是要定义在闪存中执行的代码起始地址,就是0x3F7FF6,你会看到右面链接命令文件,Codestart就是我们在闪存中运行的代码起始位置,当每次复位以后,PC指针从复位位置跳到ROM,再跳到Codestart的位置来执行你的目标代码。Password这部分要和你对应的闪存上的位置对应,“csm_rsvd”这个段是要和片上闪存的一个段对应的,它是一个预留的。当你想使用闪存加密功能时,必须预留0x76

的长度，它是不能写入任何数据，我们要直接往预留的位置里写入 0，才能让闪存加密模块完全正常工作。

从闪存启动程序的流程。

复位以后，PC 指针会指向 0x3FFFC0，从这个位置跳转到 Boot ROM 的引导程序中，在这个引导程序中会执行一个查询程序，查询对应的 GPIO 口 34 和 37 的状态，如果这个 GPIO 口 34 和 37 是确定从 Flash 闪存中执行程序的话，那么 PC 指针会跳到对应的 0x3F7FF6 的位置，在 3F7FF6 的位置我们会存储你的 C 程序的入口代码地址。通常我们用 C_int00，这会在我们“rts2800_ml.lib”里定义为用户代码的入口程序，在这部分我们会完成一些变量的初始，完成引导程序的定义，然后再跳到“用户”主函数的入口，就是 Main 函数的入口。

前面我们给大家介绍了 C28x 的架构和一些片上外设、闪存的编程内容，下面我们给大家介绍一下 C2000 的开发环境和应用套件，如果大家想了解更多的芯片应用，C2000 套件和对应的应用程序可以到 TI 网上下载 “ControlSUITE”开发套件，通过 ControlSUITE 可以了解并下载到更多的最新芯片数据手册、应用指南和开发套件，其中开发套件包括硬件电路图、应用程序，以及一些实用库文件，其中包括电机控制的库文件，有关数字电源的，PMBus、CLA、FPU、VCU 的应用库文件，以及 Flash API 的函数等。

C2000 Workshop Download Wiki

大家可以通过 C2000 Workshop 的 Wiki 网下载 C2000 入门技术研讨会的内容（英文版本），同时您还可以通过 Wiki 网下载其他讲座的内容，包括 PPT 文件、PDF 文件，以及 C2000 的实验例程文件。网址为：http://processors.wiki.ti.com/index.php/Training，你可以通过这个地址登录到 TI 处理器的 Wiki 网来下载我们培训的材料。

感谢大家学习 C2000 入门培训，如果大家对 C2000 感兴趣想了解到更多的信息，可以登录到 TI 的官方网站或者联系当地的销售办事处。

第三章 实战篇—TI C2000 inside！

3.1 单相交流电压 + 电流表

“研究傅里叶算法已经不是一两天的事，理论已经搞通了，但却从未验证过，一直想做个电能质量分析仪来验证一下。但人的惰性是很可怕的，我们总能找好多理由推脱，为了断了我的惰性，事实上 3 月末我已经把 3 相交流采样板打样好了。没想赶上了 4 月 TI 的“春暖花开，我为 TI C2000 LaunchPad ‘画’ 外围！”的活动。

在电力监控行业，傅里叶算法应用是很普遍的，大家的做法基本相同 [(硬件：DSP)+(软件：傅里叶)]，C2000 做这个在合适不过了。针对“C2000 LaunchPad”板卡尺寸小，如果继续坚持做 3 相采样，会造成采样板过大，“C2000 LaunchPad”过小（另一方面与 28027 的资源也有关），两个配在一块有点不搭。果断改为单相，难度其实并没有明显下降，只是功能有些缩减。就这样一路做了下来还算顺利，中间也卡过几次壳，好在有惊无险最后都突围了。做出来的整体性能还不错，验证了我对傅里叶算法的理解。我很享受在整个过程中突破每个难点时的激动。

感谢 TI 和 EEWORLD 组织的这次活动，感谢他们给予我挑战自我的机会。”

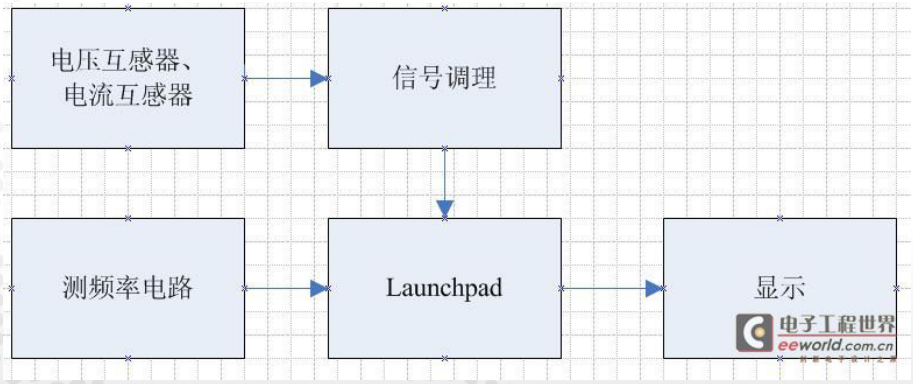
——ltbytyn

3.1.1 单相交流电压 + 电流表项目描述

【项目名称】：单相交流电压、电流表

【功能描述】：测量某一相交流支路的电压、电流、频率、功率因数

【实施方案描述】:互感器采样信号，经运放调理进 AD，通过算法（fft 或者均方根）计算交流电压、电流、频率、功率因数。



【预期测量范围】：电压 0~300V，电流 0~10A。具体与选用互感器有关。

更多详情：http://bbs.eeworld.com.cn/thread-369113-1-1.html

3.1.2 单相交流电压 + 电流表 __ 方案篇

直流信号采样比较简单。若信号电压超出 AD 采样范围可以分压或者用其他方式衰减信号进 AD 采样，读取 AD 转化的结果。再进行一个简单换算就能得到信号大小。

相对于直流信号，交流信号是一个正弦波信号，在一个周期内幅值是不段变化的。所以要计算交流信号也就没有直流那么简单了。

交流信号有三要素：幅度、频率、角度。如何去计算交流信号呢？

方法 1：使用专用芯片。比如真有效值芯片、电能芯片等。如果只想计算交流信号有效值可以使用真有效值芯片就可以了。如果除了计算有效值外，还想计算有功、无功、功率因数、频率就需要选用电能芯片。优点：电路简单，软件处理简单。缺点：成本高，实时性差（受限于芯片自身性能）。多用电阻分压采样，故抗扰性差。

方法 2：均方根计算。将周波等间隔采样 N 次，通过软件计算均方根可计算出有效值。周波等间隔采样前提是要知道当前波形的频率（周期），所以需加测频电路。计算有效值的精度与当前 AD 精度和采样 N 次数有关。优点：软件处理简单。缺点：只能计算有效值。

方法3:傅里叶计算。电路与方法2电路差不多。区别在于处理数据的算法不同。优点:傅里叶算法可以称为交流信号的"照妖镜"，所以它的强大我就不描述了。缺点:算法比较复杂。对 MCU 要求比前两种方法苛刻。

场合、要求不同，选用的方法也就不同。在电力方面（继电保护）采样是必不可少。一直想做做傅里叶，没有想到刚好碰到 LAUNCHXL- C2000 的活动。TI 的 DSP（数字信号处理器）分 2000、5000、6000 三个系列。而 C2000 的市场定位就是工业控制，做复杂算法那是它的特长。

方案 1、2 用 51、PIC 这些低端的单片机就能搞定。方案 3 用低端单片机是搞不定的。为了充分发挥 C2000 在数据处理方面的优势，选方案 3 是必须的。至于交流变直流在计算有效值过于简单。且误差较上面方法三种方法都大就不说了。

更多详情：http://bbs.eeworld.com.cn/thread-370599-1-1.html

3.1.3 单相交流电压 + 电流表 __ 工程创建篇

以前一直用 CCS3.3。最近装了个 CCS5.3，想创建个工程，结果把我难住了。网上查了些资料看了一下大复杂了，自己琢磨了好几天。下面图解新工程创建步骤（以 C2000 Piccolo LaunchPad Evaluation Kit 为基板创建实例）：

第一步：创建带主函数的项目工程

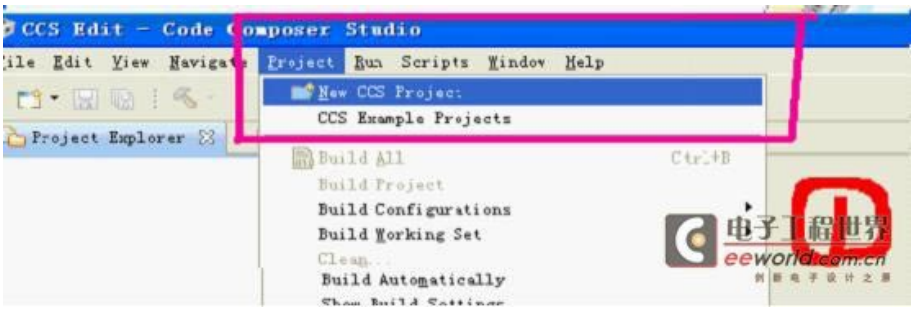


图 1

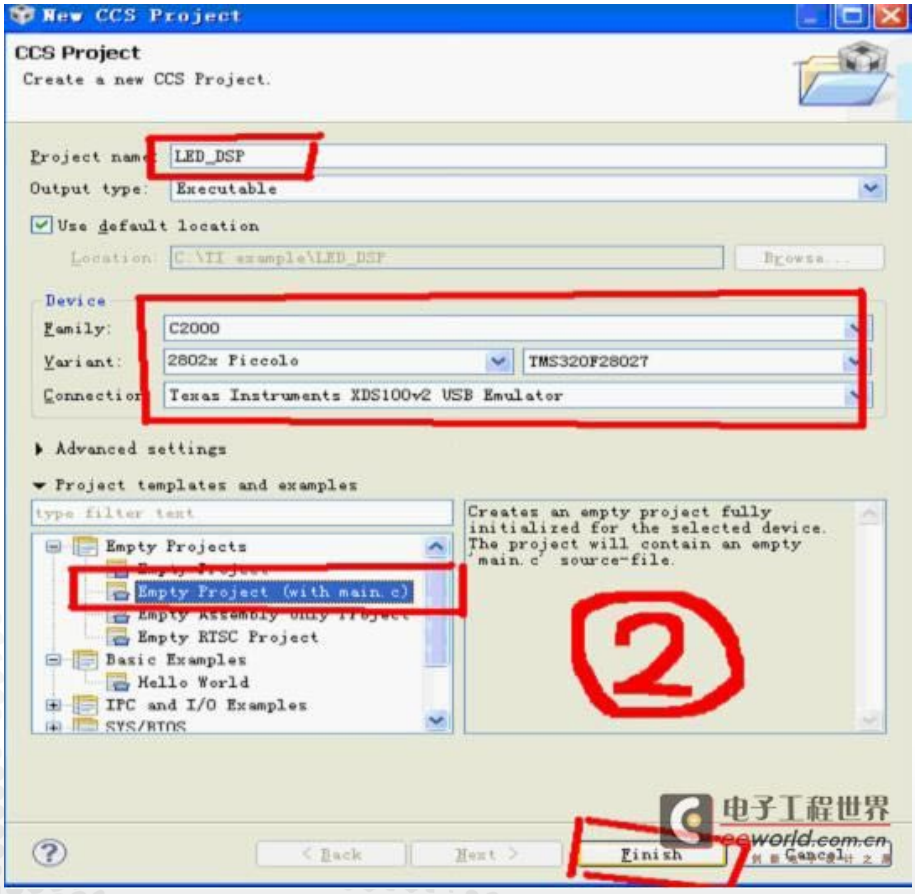


图 2

图 3

图 3 为创建好的工程项目。

第二步：添加库文件。先安装“controlSUITE”，然后找到 28027 的相应文件夹，复制下图 4 个文件。

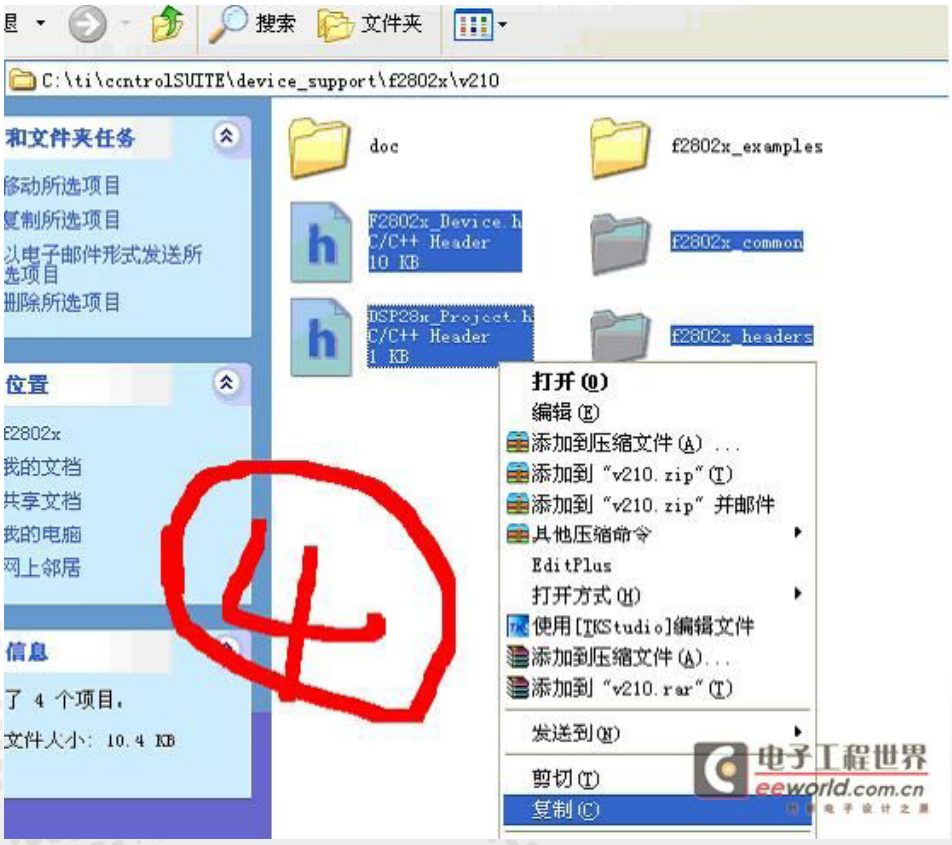


图 4

右键单击“项目工程”，选择粘贴。将上图复制的库文件复制到工程。见图 5

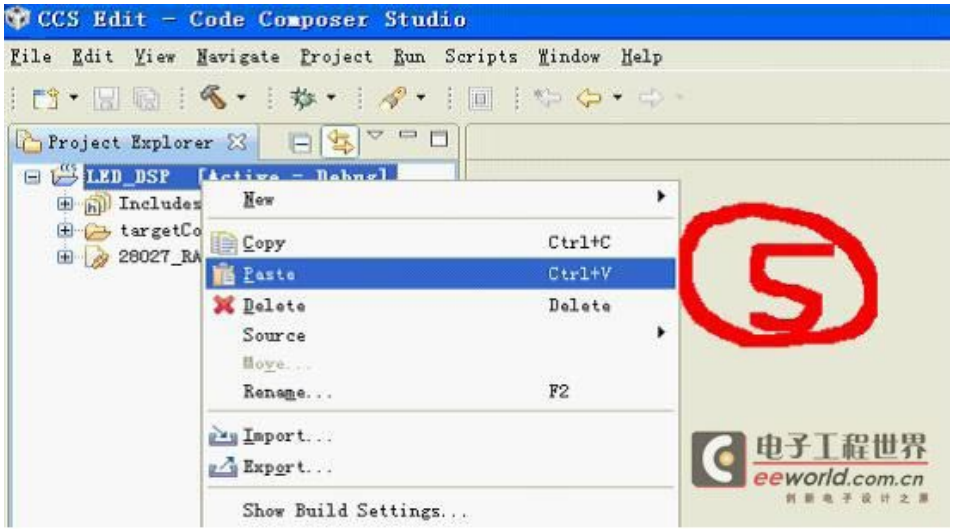


图 5

添加完库文件的工程如图 6 所示

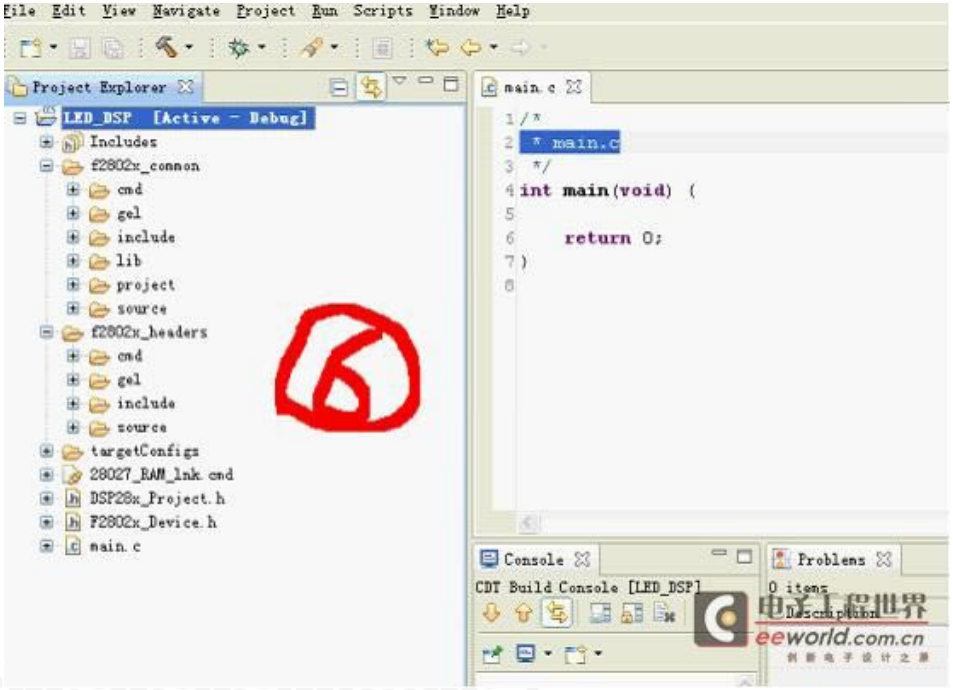


图 6

第三步：修改项目配置

右键单击“项目工程”，添加库文件路径，见图 7——图 10 所示

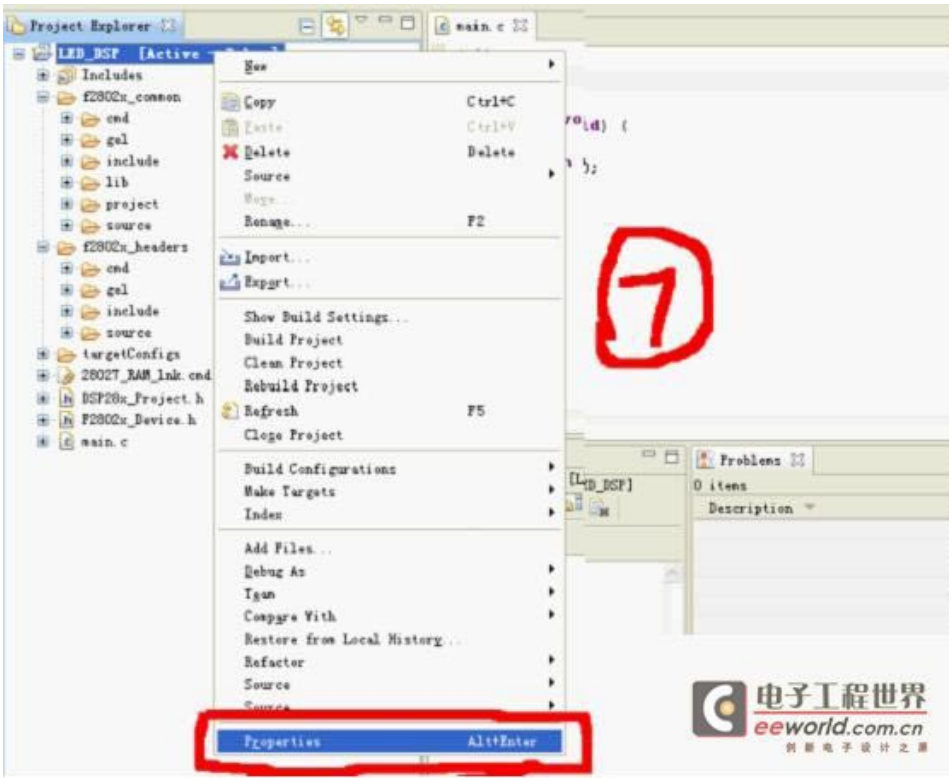

Project Explorer
main.c
LED_DSP [Active
Includes
f2802x_common
cmd
gel
include
lib
project
source
f2802x_headers
cmd
gel
include
source
targetConfigs
28027_RAM_lnk.cmd
DSP28x_Project.h
F2802x_Device.h
main.c
New
Copy Ctrl+C
Paste Ctrl+V
Delete Delete
Source
Move...
Rename... F2
Import...
Export...
Show Build Settings...
Build Project
Clean Project
Rebuild Project
Refresh F5
Close Project
Build Configurations
Make Targets
Index
Add Files...
Debug As
Team
Compare With
Restore from Local History...
Refactor
Source
Properties Alt+Enter
7
Problems
0 items
Description
电子工程世界
eeworld.com.cn

图 7

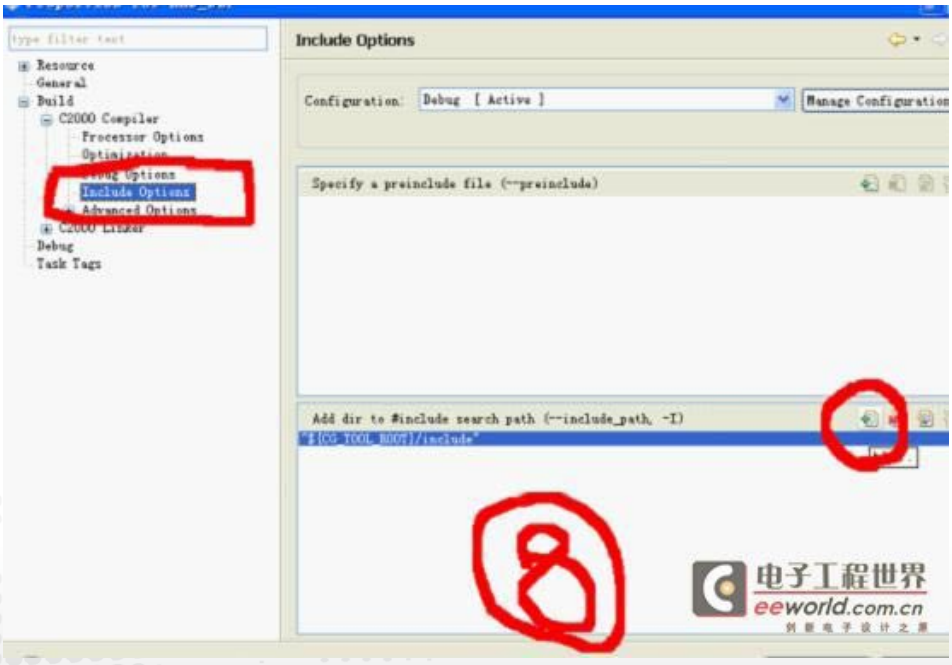

Include Options
Resource
General
Build
C2000 Compiler
Processor Options
Optimization
Include Options
Advanced Options
Debug
Task Tags
Configuration: Debug [Active]
Manage Configuration...
Specify a preinclude file (--preinclude)
Add dir to #include search path (--include_path, -I)
"${CG_TOOL_ROOT}/include"
8
电子工程世界
eeworld.com.cn

图 8

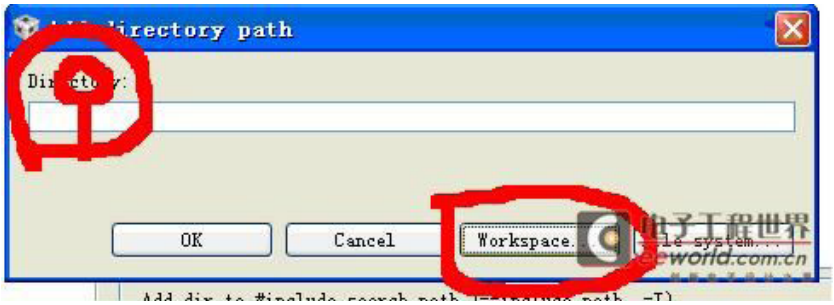


图 9

选择 f2802x_common 和 f2802x_headers 文件夹下的“include”，点击“OK”。

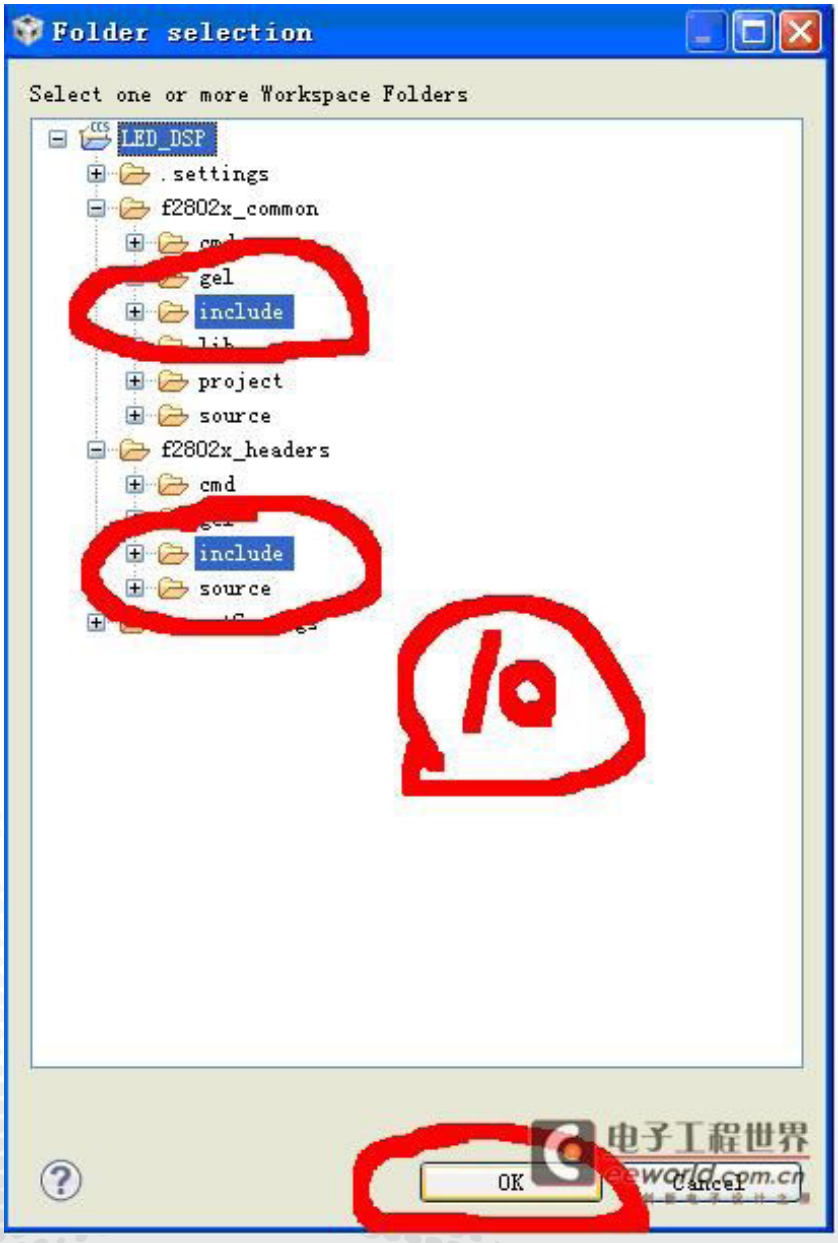


图 10

屏蔽（或者删除）项目中为未用到的文件。不屏蔽会出现编译错误和警告。

选中 f2802x_common 文件夹下的“cmd” 和“gel” 文件夹和 f2802x_headers 文件夹下的“gel”文件夹右键进行屏蔽。

见图 11——图 1 所示

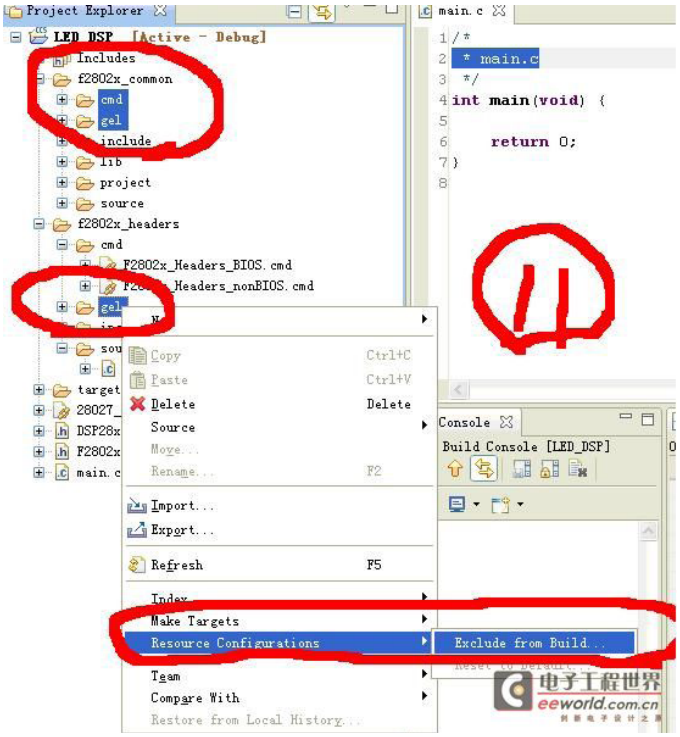


图 11

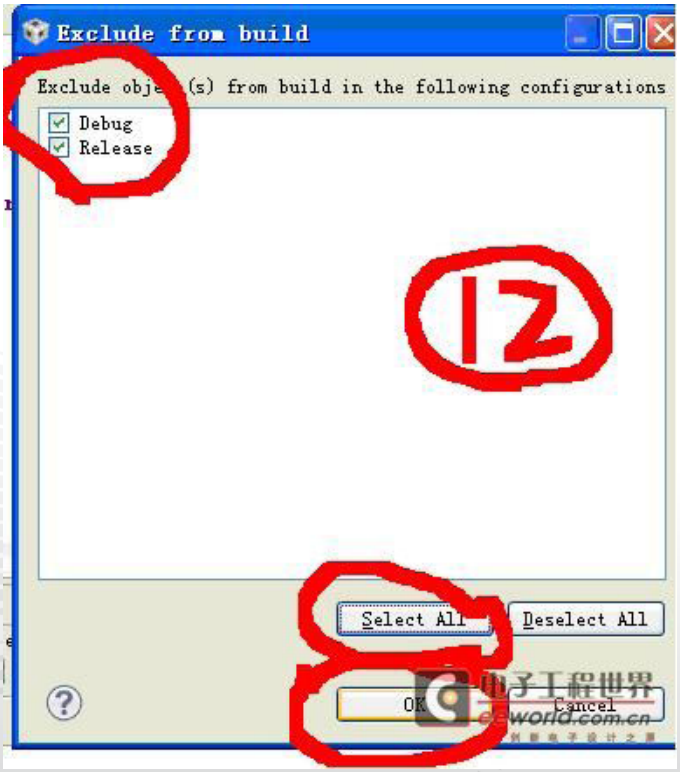


图 12

屏蔽 f2802x_headers 文件夹下的“cmd”文件夹下的“F2802x_Headers_BIOS.cmd”文件，方法如图 11——图 12 所示。屏蔽后的结果如图 13 所示。

图 13

屏蔽 f2802x_common\source 文件夹下的无用文件，一部分文件需要保留。屏蔽后的结果如图 14。

自此，项目文件配置完毕。接下来修改 “main.c”。见下图 15

图 15

编译见图 16

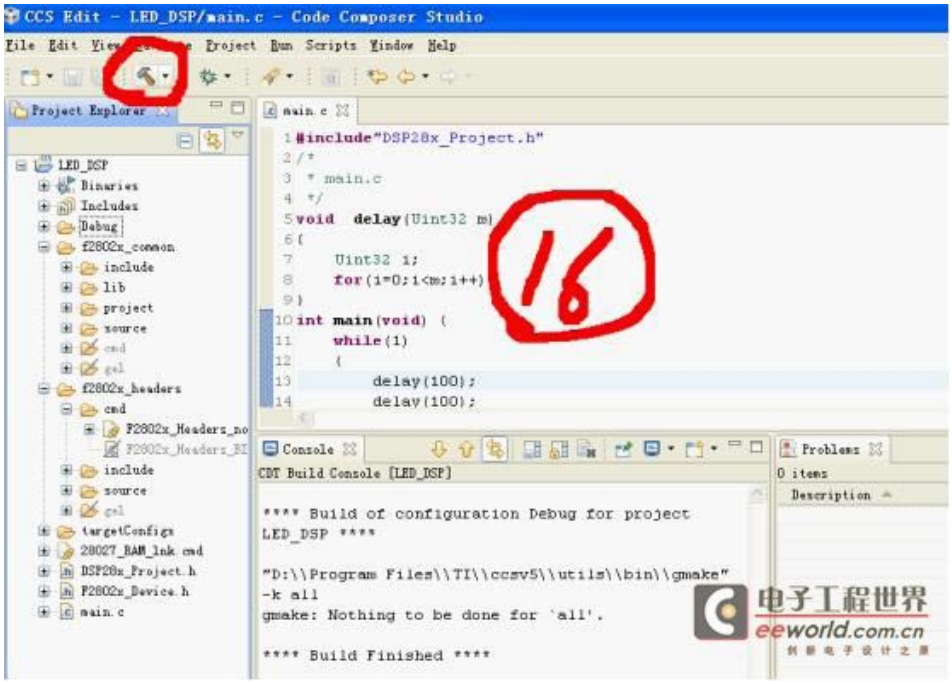


图 16

仿真见图 17

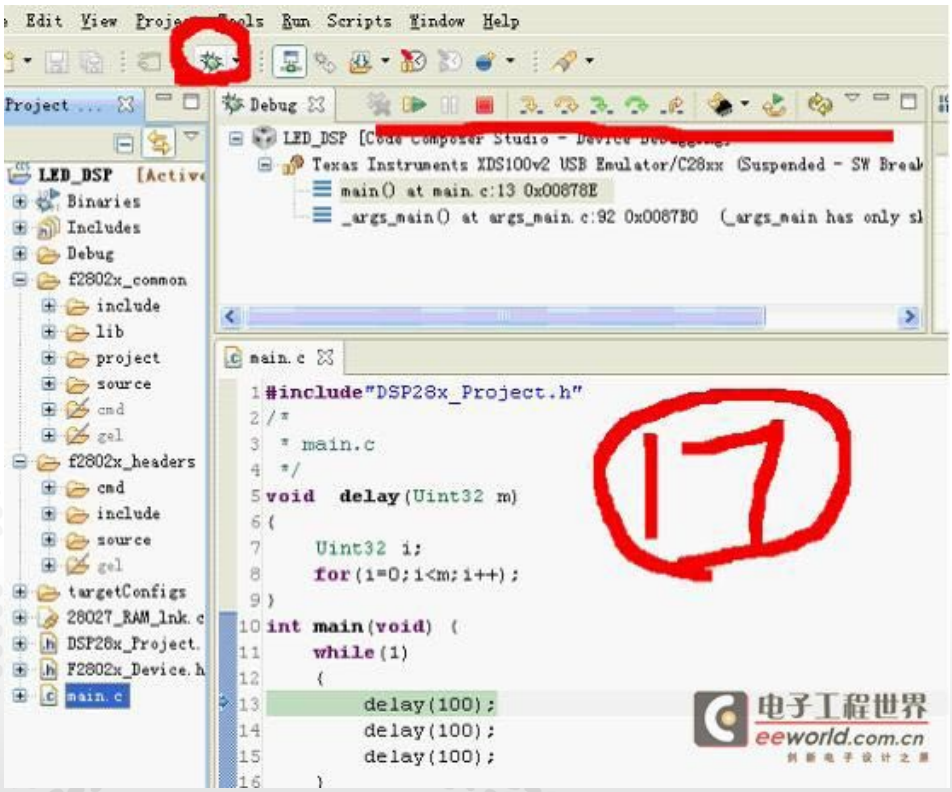
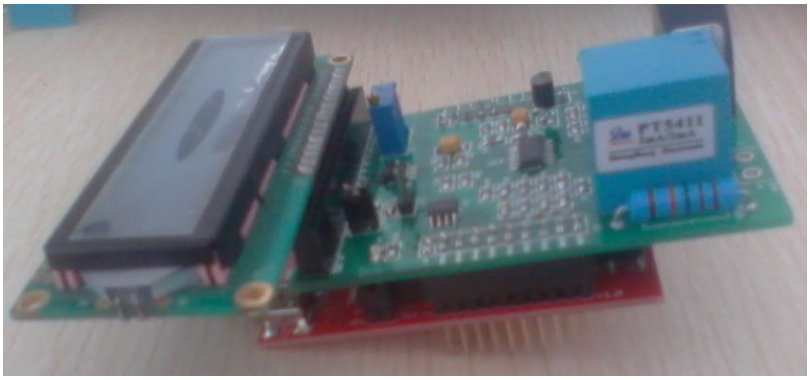


图 17

工程模板创建成功。以后在这基础上添加、修改程序即可。

更多详情：http://bbs.eeworld.com.cn/thread-370600-1-1.html

3.1.4 单相交流电压 + 电流表 __ 显示篇

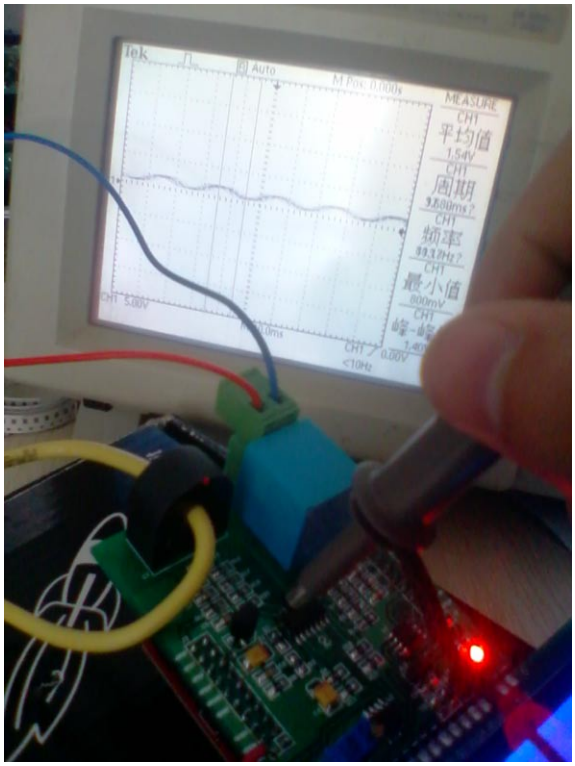


显示部分兼容 1602 及 12864（控制器 T7290，带字库）。
管脚定义：
RS————GPIO17
RW————GPIO16
VO————GPIO12
D0————GPIO0
. .
. .
. .
D7————GPIO7

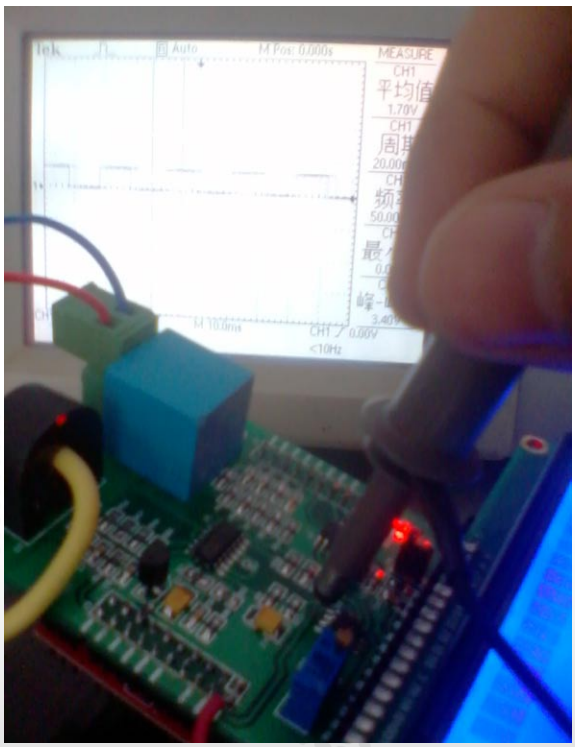
图中值为调试时填充值。计划用这个界面来显示参数，如果还有时间会将 2——16 次谐波也显示出来。
需要注意的是做显示时必须先将“C2000 Launchpad 板”的 GPIO16 与 GPIO32、GPIO17 与 GPIO33 的线割开。参看原理图 那一块线不好割 ,大家小心点。搞不清楚当初为什么要把那两个脚连在一块。
中间调 1602 出现过一些问题，抽时间在讨论。总之比较怪异。

更多详情：http://bbs.eeworld.com.cn/thread-371180-1-1.html

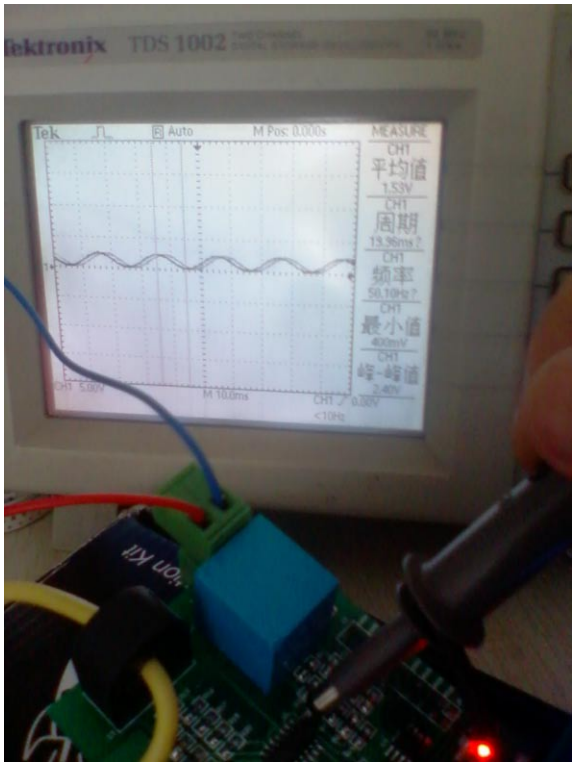
3.1.5 单相交流电压 + 电流表 __ 均方根法测电压电流



电流信号检测



频率信号检测



电压信号检测

软件使用均方根法计算电压电流。视频中的功率是缓冲区中填充值 非计算得到)。频率采集软件暂时未作。标准源的输出为 50hz 的交流信号，故默认交流信号 50hz。采用 32 点同步采样，采样间隔 625us。 video-2013-05-03-13-19-48.3gp (2.28 MB) 视频为单相交流电压 + 电流表的测量。使用 3 相功率源加入标准的 200V 电压 (50hz)、5A 电流对采样模块进行进行校对，校对完成后即可测量。电流测量精度尚可。电压的偏差略大些，比如实际 120V 测量最大在 121.2v。原因有两点：

1、硬件：电压在 200v 时测量信号已趋于饱和。需调整采样电路相应相应电阻。

2、软件：未对采样数据做优化处理，后面加入数据优化，精度自然就提上去了。

更多详情：http://bbs.eeworld.com.cn/thread-371417-1-1.html

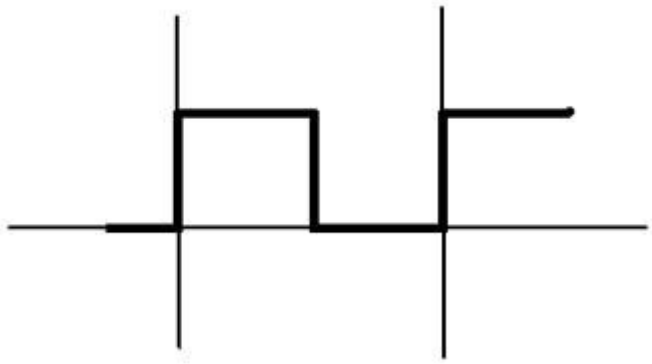
3.1.6 单相交流电压 + 电流表 __ 捕获篇（测频率）

28027 Ecap 的 CAP1、CAP2、CAP3 和 CAP4 具有捕获边沿信号的功能。将 CAPx 设置为上升沿捕获或下降沿捕获，当外部信号和当前 CAPx 设置一致时，并将当前 TSCTR 值赋给 CAPx 寄存器。根据 CAPx 之间的差值计算捕获的方波信号周期和占空比。TSCTR 的变化模式取决 ECap1Regs.ECCTL1.bit.CTRRSTx 的设置，如果为‘0’，则动作匹配时，不复位 TSCTR，将一直往上增加，直至溢出；如果为‘1’，则动作匹配时，复位 TSCTR。

举例：在计算一个方波的周期时，只需要 CAP1 和 CAP2 即可：

CPA1：上升沿，ECap1Regs.ECCTL1.bit.CTRRST1=1(复位 CTSTR)

CPA2：上升沿，设置 CEVT2 中断



在CAP1(上升沿)时复位TSCTR, 在CAP2(上升沿)时设置中断记录的TSCTR即为方波的周期。

第一上升沿时，CPA1 动作，复位 TSCTR。当第二个上降沿时，CPA2 动作，并且产生 CEVT2 事件。同时将 TSCTR 数加载到 CPA2 中。这时可以利用 CAP2 里面的值计算出方波的周期。

如果计算方波的占空比，大家根据计算周期的方法想想就能找到方法。

以 28027 的 ECAP 计算方波周期为例。

第一步：初始化 ECAP 引脚（28027 的 GPIO5 和 GPIO19 均可配置为 ECAP 口，本例中将 GPIO19 配置为 ECAP 引脚。可参见 F2802x_ECap.c 中的 InitECap1Gpio 函数）

```
void InitECap1Gpio(void)
{
   EALLOW;
// GpioCtrlRegs.GPAPUD.bit.GPIO5=0;  //
Enable pull-up on GPIO5 (CAP1)
   GpioCtrlRegs.GPAPUD.bit.GPIO19=0; //
Enable pull-up on GPIO19 (CAP1)
// GpioCtrlRegs.GPAQSEL1.bit.GPIO5=0; //
Synch to SYSCLKOUT GPIO5 (CAP1)
   GpioCtrlRegs.GPAQSEL2.bit.GPIO19=0;//
Synch to SYSCLKOUT GPIO19(CAP1)
// GpioCtrlRegs.GPAMUX1.bit.GPIO5=3; // Configure GPIO5 as CAP1
   GpioCtrlRegs.GPAMUX2.bit.GPIO19=3; // Configure GPIO19 as CAP1
   EDIS;
}
```

第二步：设置 ECap 寄存器：

```
void InitECapture()
{
   ECap1Regs.ECEINT.all = 0x0000; // Disable all capture interrupts
   ECap1Regs.ECCLR.all = 0xFFFF;  // Clear all CAP interrupt flags
   ECap1Regs.ECCTL1.bit.CAPLDEN = 0; //
Disable CAP1-CAP4 register loads
   ECap1Regs.ECCTL2.bit.TSCTRSTOP = 0; //
Make sure the counter is stopped
   // Configure peripheral registers
   ECap1Regs.ECCTL2.bit.CAP_APWM = 0;  //CAP mode
   ECap1Regs.ECCTL2.bit.CONT_ONESHT = 0; // continuous
mode   ECap1Regs.ECCTL2.bit.STOP_WRAP = 1;   // Stop at 2 events
```

```
    ECap1Regs.ECCTL1.bit.PRESCALE = 0;    // Divide by 1
    ECap1Regs.ECCTL1.bit.CAP1POL = 0;        // Rising edge
    ECap1Regs.ECCTL1.bit.CAP2POL = 0;        // Rising edge
    ECap1Regs.ECCTL1.bit.CTRRST1 = 1;        // Difference operation
    ECap1Regs.ECCTL1.bit.CTRRST2 = 0;        // Difference operation
    ECap1Regs.ECCTL2.bit.SYNCI_EN = 0;       // Enable sync in
    ECap1Regs.ECCTL2.bit.SYNCO_SEL = 3;      //  Disable sync out signal
    ECap1Regs.ECCTL1.bit.CAPLDEN = 1;        // Enable capture units
    ECap1Regs.ECCTL2.bit.TSCTRSTOP = 1;      // Start Counter
    ECap1Regs.ECEINT.bit.CEVT2 = 1;          // 2 events = interrupt
}
```

第三步：外部 PIE 和 CPU 中断使能。

```
EALLOW;  // This is needed to write to EALLOW protected registers
PieVectTable.ECAP1_INT = &ecap1_isr;
EDIS; // This is needed to disable write to EALLOW protected
registers  // Enable CPU INT4 which is connected to ECAP1-4 INT:
IER |= M_INT4;
PieCtrlRegs.PIEIER4.bit.INTx1 = 1;
// Enable eCAP INTn in the PIE: Group 3 interrupt 1-6
EINT;   // Enable Global interrupt INTM
ERTM;   // Enable Global realtime interrupt DBGM
interrupt void ecap1_isr(void)
{
//ECap1Count = ECap1Regs.CAP1;
   ECap2Count = ECap1Regs.CAP2;
//ECap1Regs.ECCLR.all = 0xFFFF;
   ECap1Regs.ECCLR.bit.CEVT2 = 1;
```

```
    ECap1Regs.ECCLR.bit.CEVT1 = 1;
    ECap1Regs.ECCLR.bit.INT = 1;
    PieCtrlRegs.PIEACK.all = PIEACK_GROUP4;
}
```

注：当使用 One-shot mode 方式时，若需要不断的捕获方波，需在中断程序中需加入“Cap1Regs.ECCTL2.bit.REARM = 1;”，否则只能捕获一次。以后即使有方波信号也不会进入捕获中断处理函数。

如上即可利用 ECAP 的捕获功能来计算方波周期。当然灵活利用 ECAP 功能可以实现很多，这些需要大家去摸索。

测试时发现了一个问题，用新的就是项目文件单纯的覆盖旧的项目文件。仿真可能有问题。比如我碰到的问题，在家里程序测试正常，在公司测试时不进中断。一样的程序在家里和公司两种情况。我后来直接在 CCS 中将就旧项目文件完全删除，再在 CCS 中重新添加新项目文件，测试 OK。所以建议大家做的不要单纯的去覆盖旧文件。

更多详情：http://bbs.eeworld.com.cn/thread-371885-1-1.html

3.1.7 单相交流电压 + 电流表 __ 算法篇

FFT 是离散傅立叶变换的快速算法，可以将一个信号变换到频域。有些信号在时域上是很难看出什么特征的，但是如果变换到频域之后，就很容易看出特征了。这就是很多信号分析采用 FFT 变换的原因。另外，FFT 可以将一个信号的频谱提取出来，这在频谱分析方面也是经常用的。虽然很多人都知道 FFT 是什么，可以用来做什么，怎么去做，但是却不知道 FFT 之后的结果是什意思、如何决定要使用多少点来做 FFT。

现在圈圈就根据实际经验来说说 FFT 结果的具体物理意义。一个模拟信号，经过 ADC 采样之后，就变成了数字信号。采样定理告诉我们，采样频率要大于信号频率的两倍，这些我就不在此罗嗦了。

采样得到的数字信号，就可以做 FFT 变换了。N 个采样点，经过 FFT 之后，就可以得到 N 个点的 FFT 结果。为了方便进行 FFT 运算，通常 N 取 2 的整数次方。

假设采样频率为 Fs，信号频率 F，采样点数为 N。那么 FFT 之后结果就是一个为 N 点的复数。每一个点就对应着一个频率点。这个点的模值，就是该频率值下的幅度特性。具体跟原始信号的幅度有什么关系呢？假设原始信号的峰值为 A，那么 FFT 的结果的每个点（除了第一个点直流分量之外）的模值就是 A 的 N/2 倍。而第一个点就是直流分量，它的模值就是直流分量的 N 倍。而每个点的相位呢，就是在该频率下的信号的相位。第一个点表示直流分量（即 0Hz），而最后一个点 N 的再下一个点（实际上这个点是不存在的，这里是假设的第 N+1 个点，也可以看做是将第一个点分做两半分，另一半移到最后）则表示采样频率 Fs，这中间被 N-1 个点平均分成 N 等份，每个点的频率依次增加。例如某点 n 所表示的频率为：Fn=(n-1)*Fs/N。由上面的公式可以看出，Fn 所能分辨到频率为为 Fs/N，如果采样频率 Fs 为 1024Hz，采样点数为 1024 点，则可以分辨到 1Hz。1024Hz 的采样率采样 1024 点，刚好是 1 秒，也就是说，采样 1 秒时间的信号并做 FFT，则结果可以分析到 1Hz，如果采样 2 秒时间的信号并做 FFT，则结果可以分析到 0.5Hz。

如果要提高频率分辨力，则必须增加采样点数，也即采样时间。频率分辨率和采样时间是倒数关系。

假设 FFT 之后某点 n 用复数 a+bi 表示，那么这个复数的模就是 An= 根号 a*a+b*b，相位就是 Pn=atan2(b,a)。根据以上的结果，就可以计算出 n 点（n ≠ 1，且 n<=N/2）对应的信号的表达式为：An/(N/2)*cos(2*pi*Fn*t+Pn)，即 2*An/N*cos(2*pi*Fn*t+Pn)。对于 n=1 点的信号，是直流分量，幅度即为 A1/N。

由于 FFT 结果的对称性，通常我们只使用前半部分的结果，即小于采样频率一半的结果。好了，说了半天，看着公式也晕，下面圈圈以一个实际的信号来做说明。

假设我们有一个信号，它含有 2V 的直流分量，频率为 50Hz、相位为 -30 度、幅度为 3V 的交流信号，以及一个频率为 75Hz、相位为 90 度、幅度为 1.5V 的交

流信号。用数学表达式就是如下：

S=2+3*cos(2*pi*50*t-pi*30/180)+1.5*cos(2*pi*75*t+pi*90/180) 式中 cos 参数为弧度，所以 -30 度和 90 度要分别换算成弧度。我们以 256Hz 的采样率对这个信号进行采样，总共采样 256 点。按照我们上面的分析，Fn=(n-1)*Fs/N，我们可以知道，每两个点之间的间距就是 1Hz，第 n 个点的频率就是 n-1。我们的信号有 3 个频率：0Hz、50Hz、75Hz，应该分别在第 1 个点、第 51 个点、第 76 个点上出现峰值，其它各点应该接近 0。实际情况如何呢？我们来看看 FFT 的结果的模值如图所示。

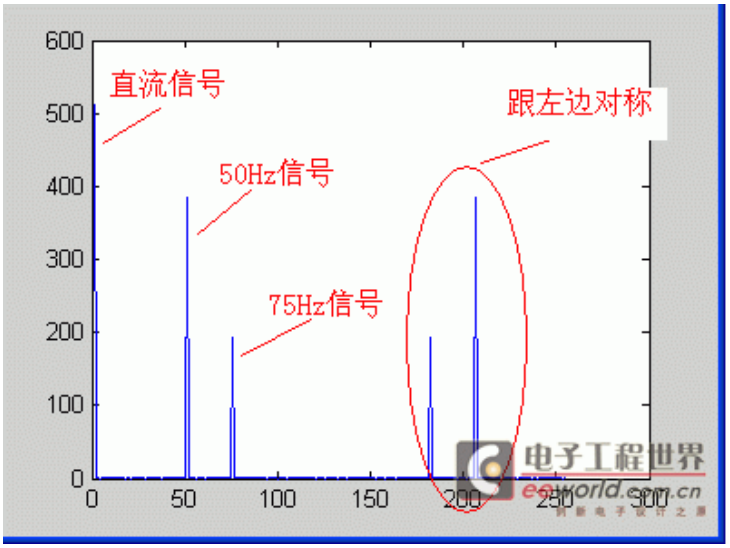


图 1 FFT 结果

从图中我们可以看到，在第 1 点、第 51 点、和第 76 点附近有比较大的值。我们分别将这三个点附近的数据拿上来细看：

1 点： 512+0i

2 点： -2.6195E-14 - 1.4162E-13i

3 点： -2.8586E-14 - 1.1898E-13i

50 点：-6.2076E-13 - 2.1713E-12i

51 点：332.55 - 192i

52 点：-1.6707E-12 - 1.5241E-12i

75 点：-2.2199E-13 -1.0076E-12i

76 点：3.4315E-12 + 192i

77 点：-3.0263E-14 +7.5609E-13i

很明显，1 点、51 点、76 点的值都比较大，它附近的点值都很小，可以认为是 0，即在那些频率点上的信号幅度为 0。接着，我们来计算各点的幅度值。分别计算这三个点的模值，

结果如下：

1 点： 512

51 点 : 384

76 点 : 192

按照公式，可以计算出直流分量为：512/N=512/256=2；50Hz 信号的幅度为：384/(N/2)=384/(256/2)=3；75Hz 信号的幅度为 192/(N/2)=192/(256/2)=1.5。可见，从频谱分析出来的幅度是正确的。

然后再来计算相位信息。直流信号没有相位可言，不用管它。先计算 50Hz 信号的相位，atan2(-192, 332.55)=-0.5236, 结果是弧度，换算为角度就是 180*(-0.5236)/pi=-30.0001。再计算 75Hz 信号的相位，atan2(192, 3.4315E-12)=1.5708 弧度，换算成角度就是 180*1.5708/pi=90.0002。可见，相位也是对的。根据 FFT 结果以及上面的分析计算，我们就可以写出信号的表达式了，它就是我们开始提供的信号。

总结：假设采样频率为 Fs，采样点数为 N，做 FFT 之后，某一点 n（n 从 1 开始）表示的频率为：Fn=(n-1)*Fs/N；该点的模值除以 N/2 就是对应该频率下的信号的幅度（对于直流信号是除以 N）；该点的相位即是对应该频率下的信号的相位。相位的计算可用函数 atan2(b,a) 计算。atan2(b,a) 是求坐标为 (a,b) 点的角度值，范围从 -pi 到 pi。要精确到 xHz，则需要采样长度为 1/x 秒的信号，并做 FFT。要提高频率分辨率 就需要增加采样点数 这在一些实际的应用中是不现实的，需要在较短的时间内完成分析。解决这个问题的方法有频率细分法，比较简单的方法是采样比较短时间的信号，然后在后面补充一定数量的 0，使其长度达到需要的点数，再做 FFT，这在一定程度上能够提高频率分辨力。具体的频率细分法可参考相关文献。

[附录：本测试数据使用的 matlab 程序]

```
close all; % 先关闭所有图片
Adc=2;  % 直流分量幅度
A1=3;   % 频率 F1 信号的幅度
A2=1.5; % 频率 F2 信号的幅度
F1=50;  % 信号 1 频率 (Hz)
F2=75;  % 信号 2 频率 (Hz)
```

```
Fs=256; % 采样频率 (Hz)
P1=-30; % 信号 1 相位 ( 度 )
P2=90;  % 信号相位 ( 度 )
N=256;  % 采样点数
t=[0:1/Fs:N/Fs]; % 采样时刻
% 信号
S=Adc+A1*cos(2*pi*F1*t+pi*P1/180)+A2*cos(2*pi*F2*t+pi*P2/180);
% 显示原始信号
plot(S);
title( '原始信号' );
figure;
Y = fft(S,N); % 做 FFT 变换
Ayy = (abs(Y)); % 取模
plot(Ayy(1:N)); % 显示原始的 FFT 模值结果
title( 'FFT 模值' );
figure;
Ayy=Ayy/(N/2);  % 换算成实际的幅度
Ayy(1)=Ayy(1)/2;
F=([1:N]-1)*Fs/N; % 换算成实际的频率值
plot(F(1:N/2),Ayy(1:N/2));  % 显示换算后的 FFT 模值结果
title( '幅度 - 频率曲线图' );
figure;
Pyy=[1:N/2];
for i=1:N/2
Pyy(i)=phase(Y(i)); % 计算相位
Pyy(i)=Pyy(i)*180/pi; % 换算为角度
end;
plot(F(1:N/2),Pyy(1:N/2));  % 显示相位图
title( '相位 - 频率曲线图' );
```

更多详情 http://bbs.eeworld.com.cn/thread-372320-1-1.html

3.1.8 单相交流电压 + 电流表 __ 演示篇

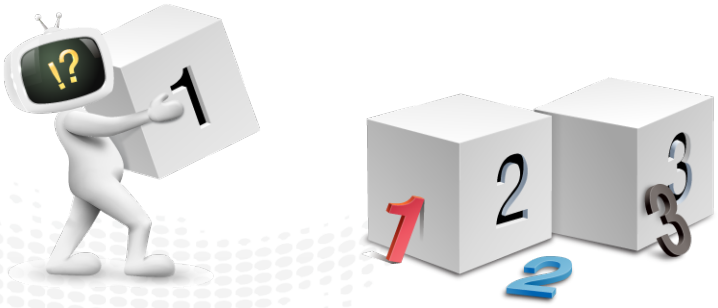
刚开始把这次要 DIY 的东东叫“单相交流电压 + 电流表”，事实上它的功能远比单纯的交流电表功能更加强大，称呼它“单相电能分析仪”更加合适一些。功能的强大在于采用了 FFT 算法，它能把波形分析的通通透透。显示画面解析：

U xxx.xv I x.xxA

F xx.xHZ N xx.xX（X 为 C 时电压滞后电流，X 为 L 时电压超前电流）

除了所演示的计算实时电压、电流、功率因数，它还能分析各次谐波含量（与采用点数有关，采样点数越多，能分析出【N 次谐波】N 越大），有功，无功，视在功率等。受限于 1602 的屏幕过小，只能演示有限的参数。

视频演示：http://bbs.eeworld.com.cn/thread-372485-1-1.html

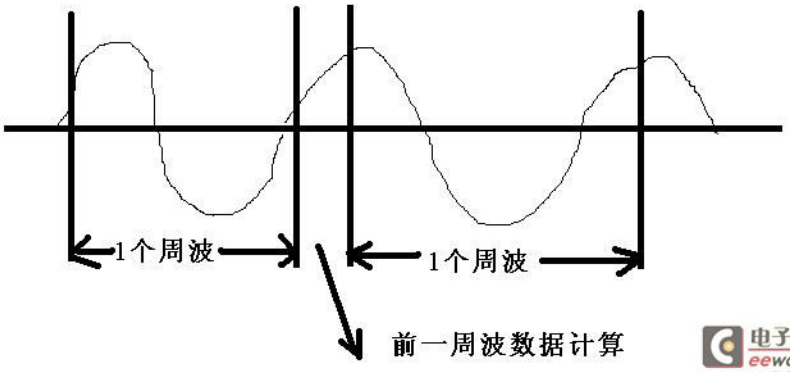


3.1.9 单相交流电压 + 电流表 __ 技巧篇

1 数据处理

交流信号的计算过程：周波数据采样——(完成采样)——> 执行 FFT 计算——(完成采样)——> 下一周波采样

这里有个问题：上一周波刚采样完成，缓冲里面的数据即将面临旧数据计算又要面临新一轮采样数据更新缓冲中的旧数据。如果旧数据未计算完而被新采用数据破坏，会造成错误的计算结果。当然我们可以等旧数据计算完成后在开始新采样，但是造成的结果是计算的不连续性。



采用双缓冲，一个缓冲接收数据，一个缓冲处理数据。

BUF1 接收数据——(接收完成)——>BUF1 处理数据 ,BUF2 接收数据——(接收完成)——>BUF2 处理数据 ,BUF1 接收数据，当然我们也可以根据情况使用多缓冲。

2、当程序过大或者 RAM 使用过多时 ,CMD 文件的修改时必不可少。有可能我们多添加一个函数就可能引起错误。一方面大家需多关注编译生成的“***.map”文件，搞清楚划分的段是否够用 (一般正确的时候，可以不要关注)。另一方面修改 CMD 文件需要好好看看相应 DSP 内部的“Memory Map”，不能凭空去修改 CMD 文件。我认为会写 DSP 程序不一定是掌握了 DSP，会熟练修改 CMD 文件的才算掌握了 DSP。

3、不要观看资料，也要注重实践。比如定时器里面“ ConfigCpuTimer(&CpuTimer0, 60, 20000)”我们利用中断反转IO用示波器测一下是否与理论值相否，如果有偏差，会差多少。还有捕获一个标准 20ms 方波，计数器里面值是否与理论值相否，如果有偏差，会差多少。等等你会发现好多好多东西。

4、在做 FFT 时，AD 采样频率是比较重要参数。远大于信号的采样频率。信号的采样频率由信号的频率和采样点数决定。RAM 的使用率和采样点数几乎是成几何方式增长。采样的点数也决定了能计算出的最高多少次谐波的相应参数。采样的点数不是凭空想出来的，要根据 MCU 的处理速度、AD 的采样频率、MCU 的 RAM 使用情况、自身对数据的需求多方面因数决定的。

更多详情：http://bbs.eeworld.com.cn/thread-372617-1-1.html

3.2 基于 C2000 LaunchPad 的电子负载

> “C2000 是 TI 推出的实时控制 CPU，在实时控制方面有其独特的优势。F28027 控制完备、实时性强，性能可靠，价格低廉，用于电子负载控制是最恰当的选择。
>
> 本次设计的电子负载是初稿，在调试时发现硬件上还有些不当的地方，软件方面也仅是初步调试。当然性能还没有达到设计要求，有待进一步改进。”
>
> ——dontium

3.2.1 基于 C2000 LaunchPad 的电子负载项目描述

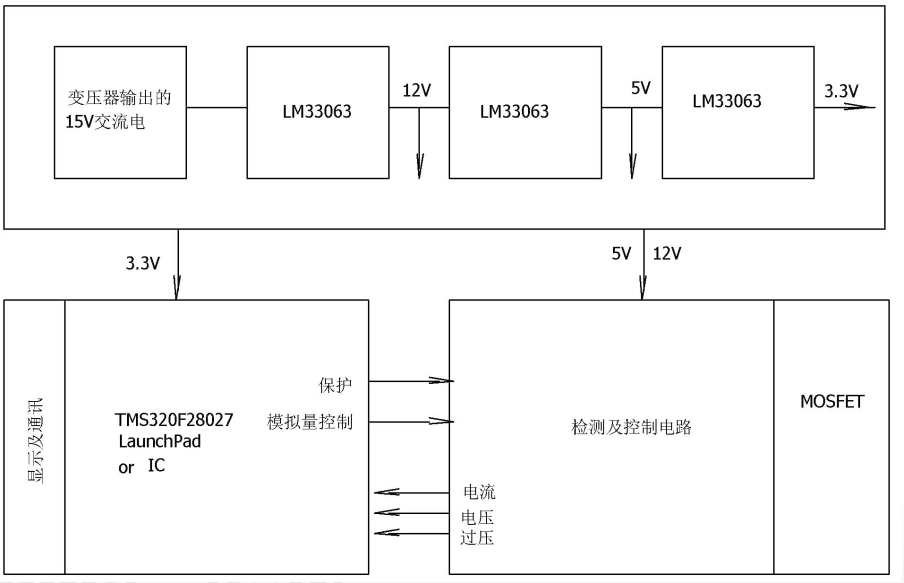
用 F28027 做个电子负载是非常恰当的，可以充分发挥其实时控制的特点。做出的电子负载将是反应灵敏、控制精确、保护完善的高档产品

[项目名称] : 基于 C2000 的电子负载

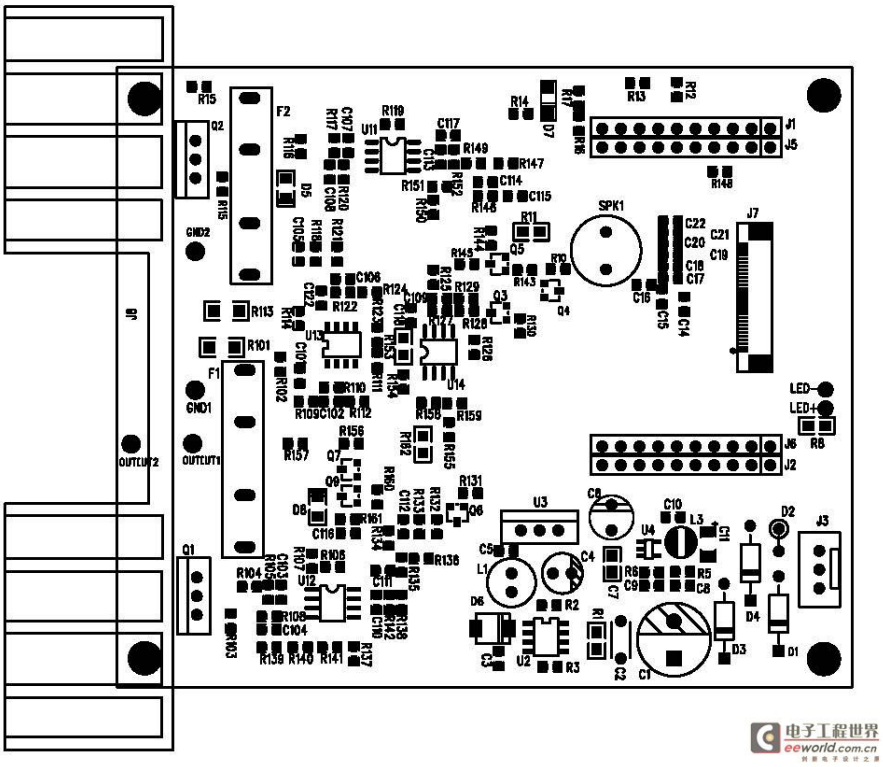
[功能描述]:多达 8 个控制点 (PWM) , 及多达 13 个 ADC 输入端的 F28027, 可以很好地控制大功率电子负载，实现负载的电流、电压的精确控制。拟定制作的负载功率 : 100W

[电路框图]

整 机 框 图



PCB 出来了，有点大，100mm x 85mm，为的是加个散热器。

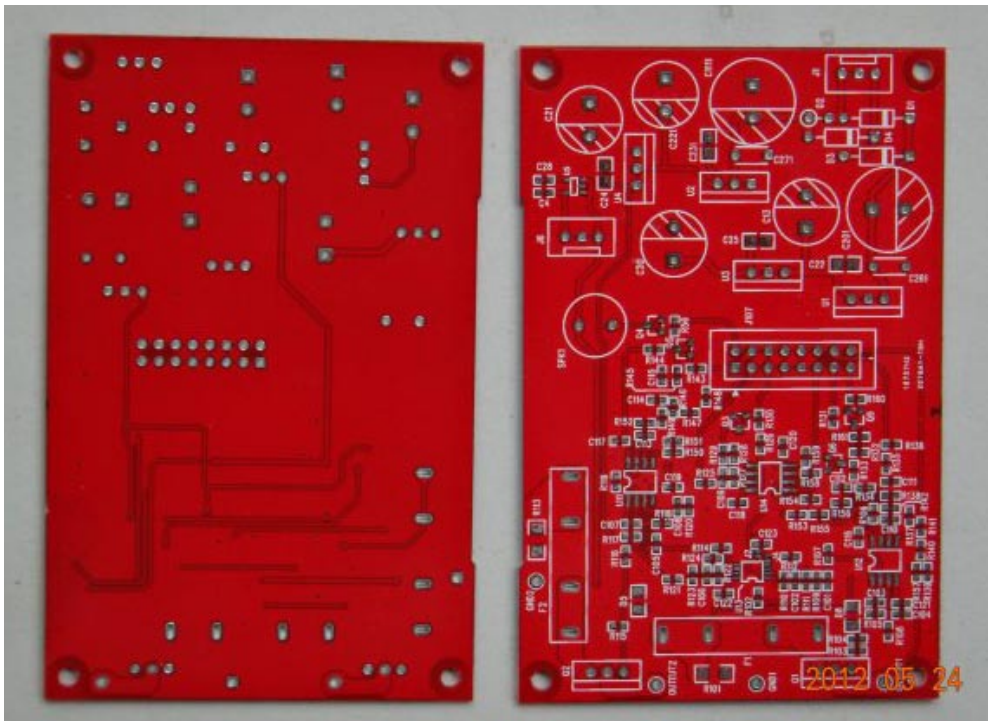


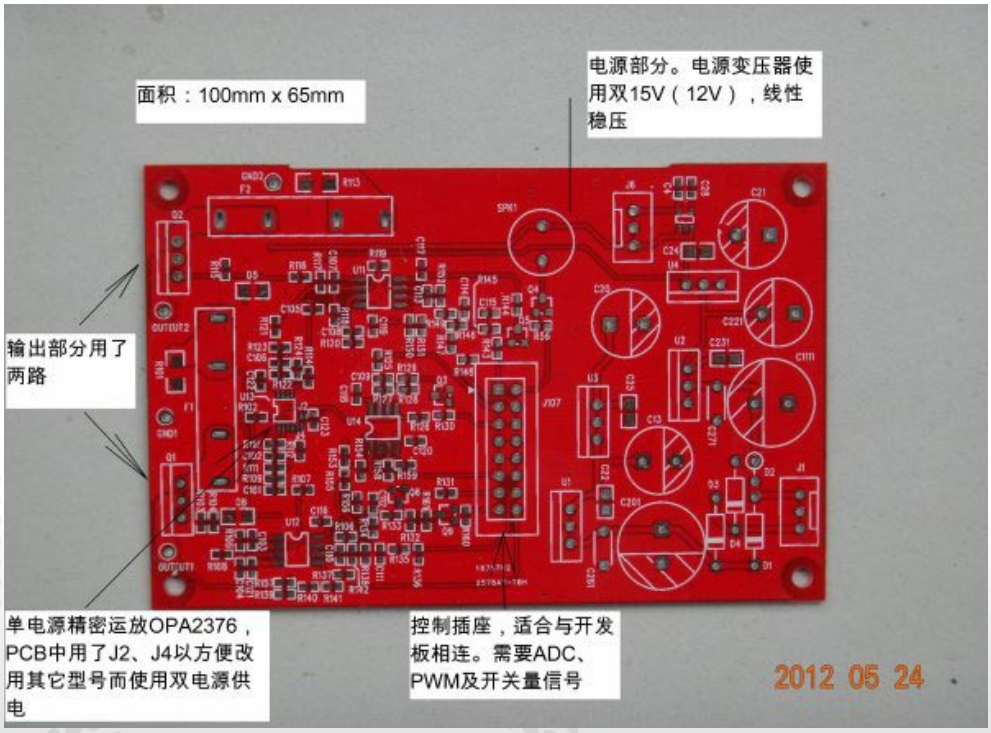
电子负载开环调试通过，未加入 ADC 采样对输出的控制，运放部分、控制部分均正常工作。

更多详情：http://bbs.eeworld.com.cn/thread-369622-1-1.html

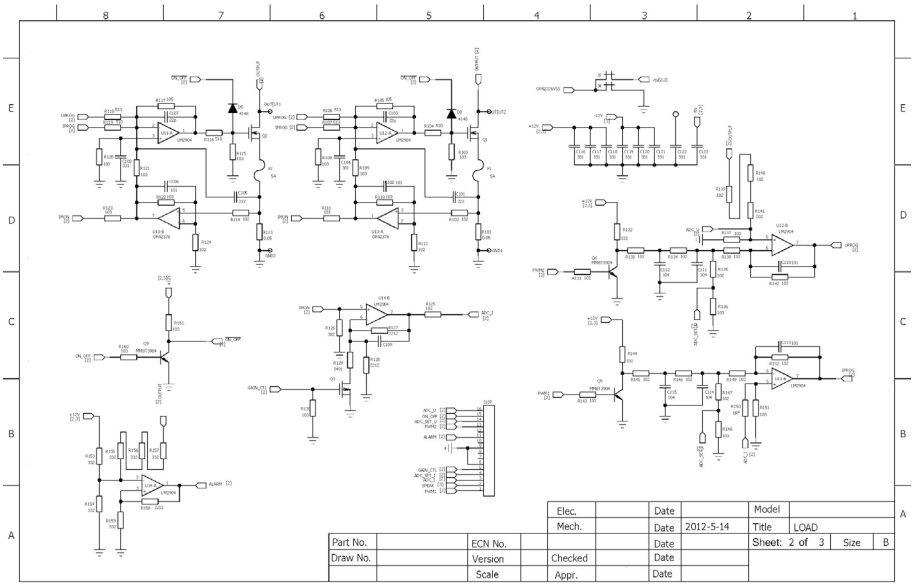
3.2.2 基于 C2000 LaunchPad 的电子负载——第一次 PCB 打板

1、先上个 PCB





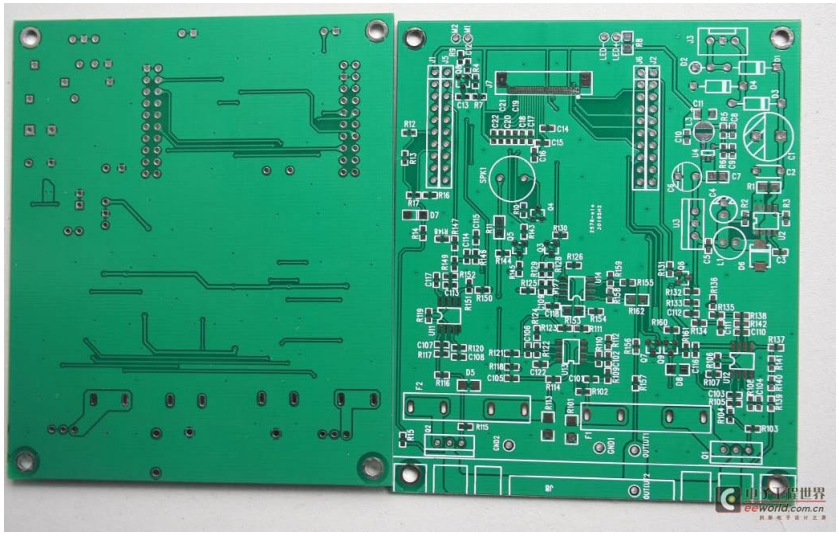
2、这是电子负载的电路图，图中有不妥或错误。

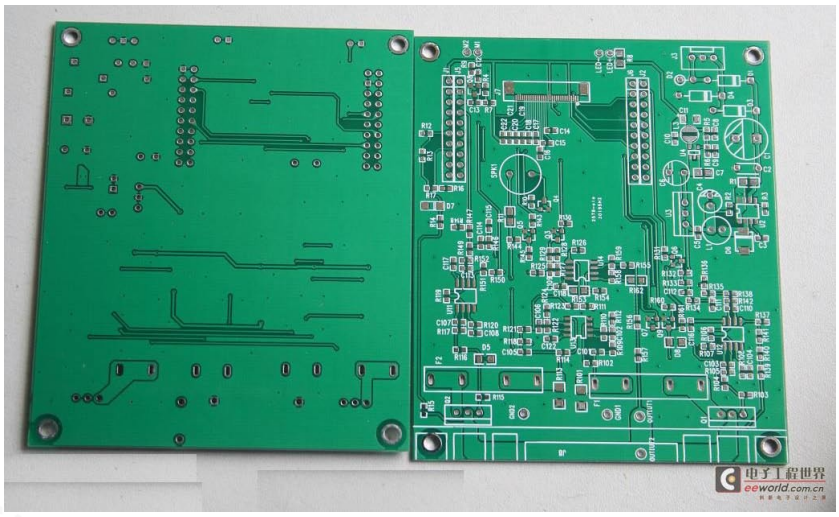


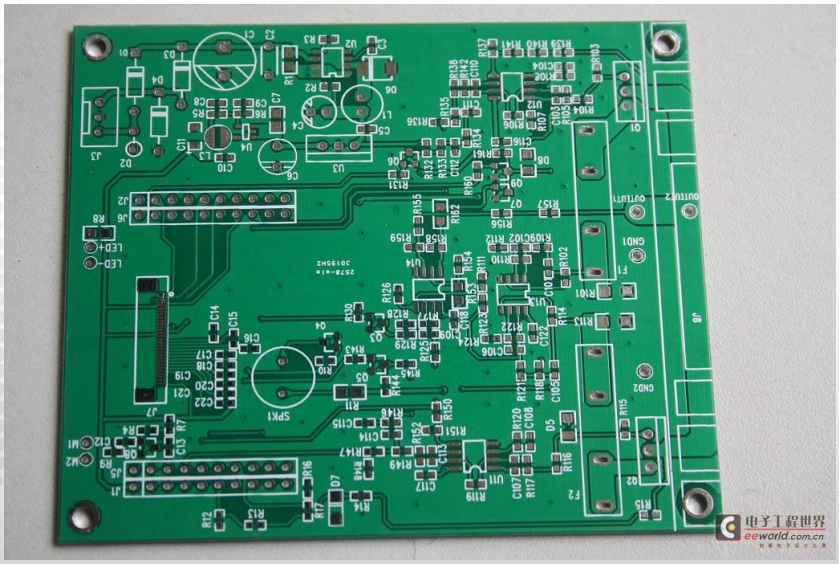
更多详情：http://bbs.eeworld.com.cn/thread-330629-1-1.html

3.2.3 基于 C2000 LaunchPad 的电子负载——电子负载 PCB 回来了

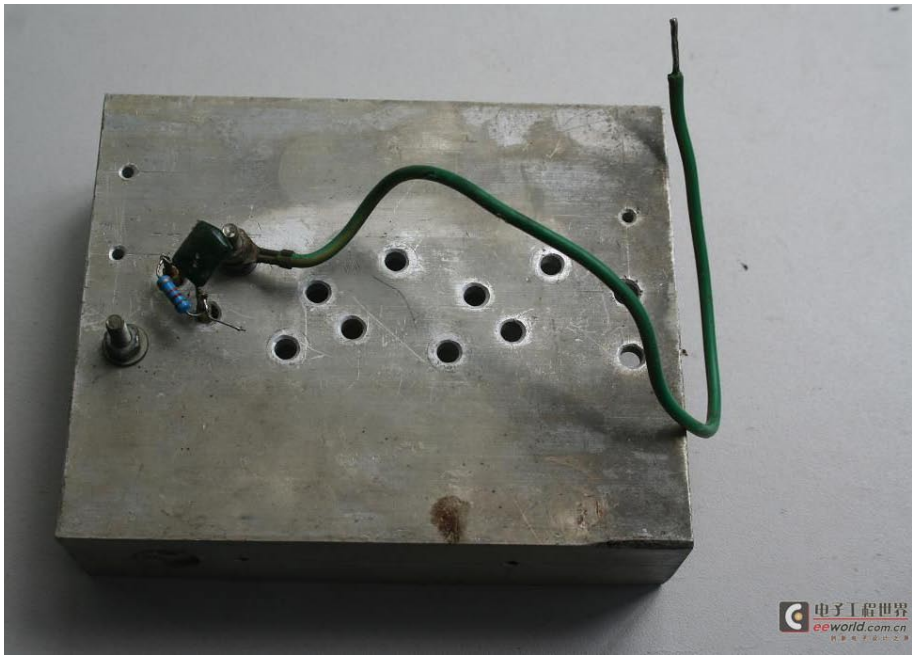
PCB 靓照：



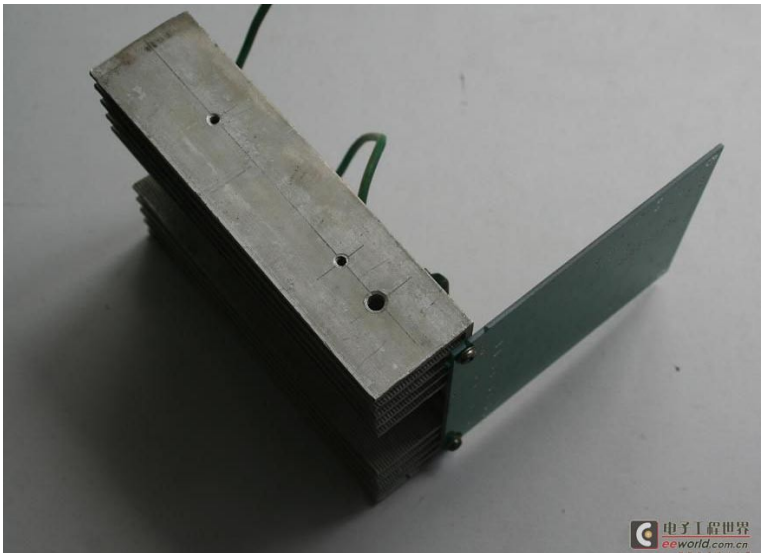




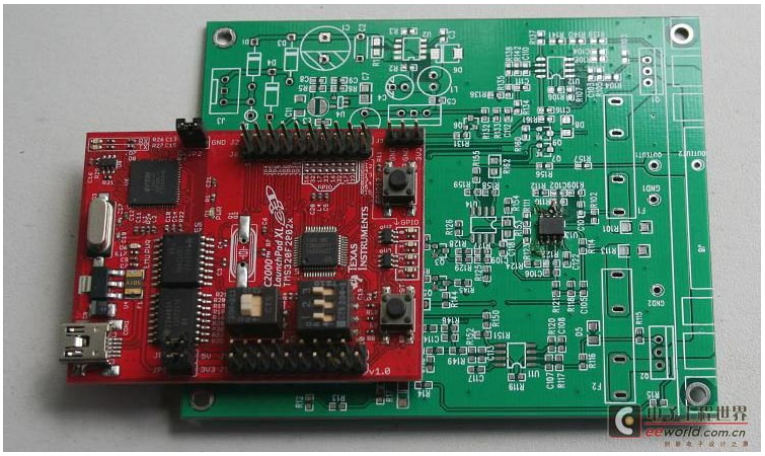
和它相配的散热器：

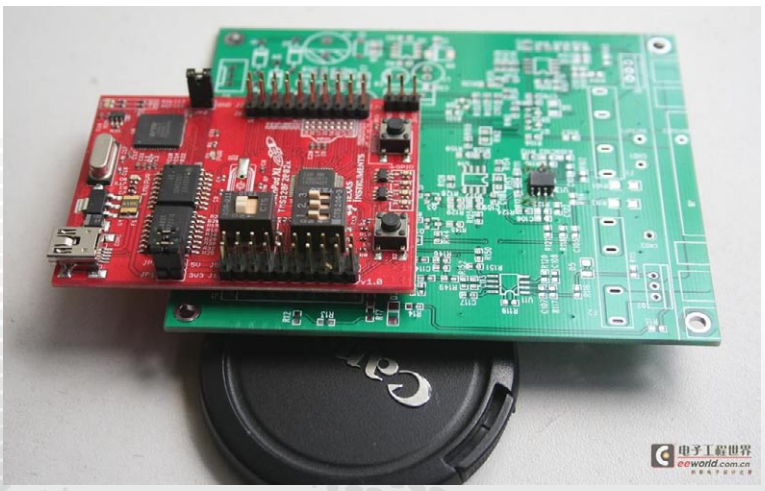


安装在一起的样子：



和它相配的开发板，感谢 wudayongnb 及 jishuaihu 网友给的结构尺寸。插座位置对得非常准。



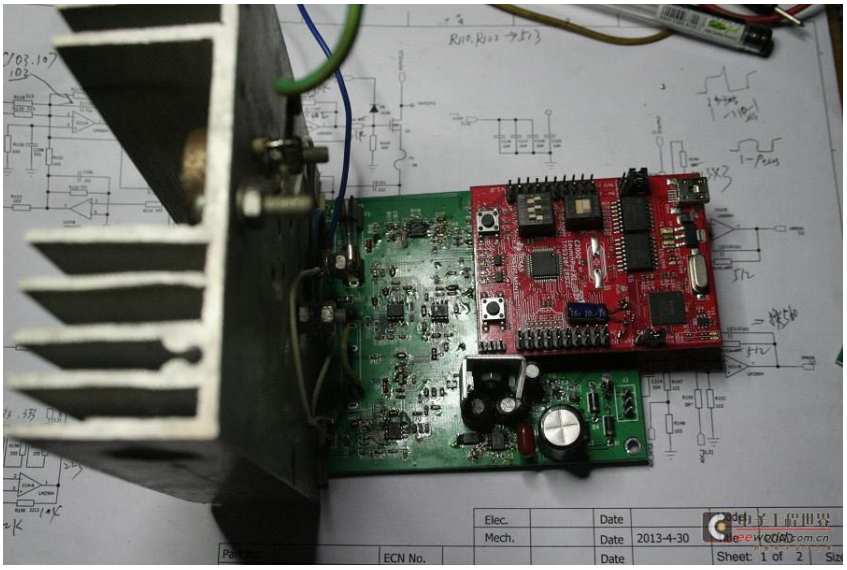


更多详情：http://bbs.eeworld.com.cn/thread-371260-1-1.html

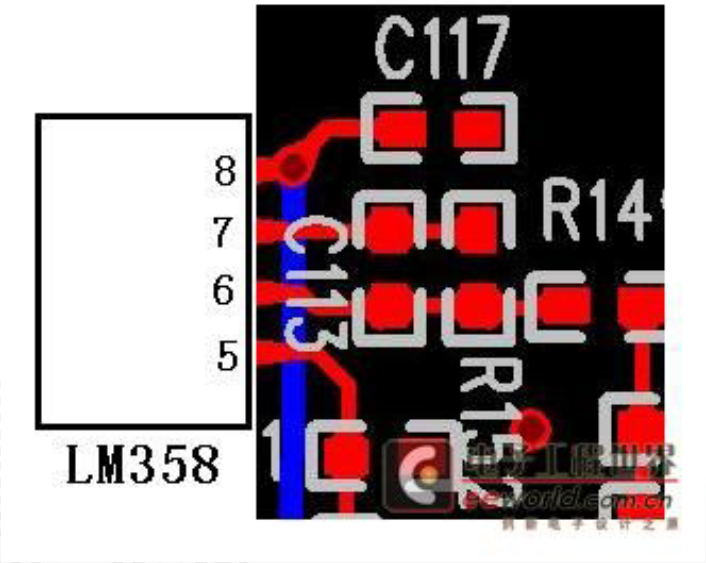
3.2.4 基于 C2000 LaunchPad 的电子负载——电子负载电路调试完成

之前把程序做得差不多了，今天把就把板子在程序指挥下工作，遇到了不少问题，现已基本解决，功能性调试完成！

下一步该调整性能了。



在调试时，由于马虎，走了些弯路，如：



358 的反馈电阻及电容 R152、C113，应该按照上图的接法，上下竖着焊接，站立着才对。可是我把它们俩象 C117 那样横着，让它躺着，结果是怎么调整也不能使之不自激。

更多详情：http://bbs.eeworld.com.cn/thread-372351-1-1.html

3.3 基于 C2000 LanuchPad 的其它应用设计方案

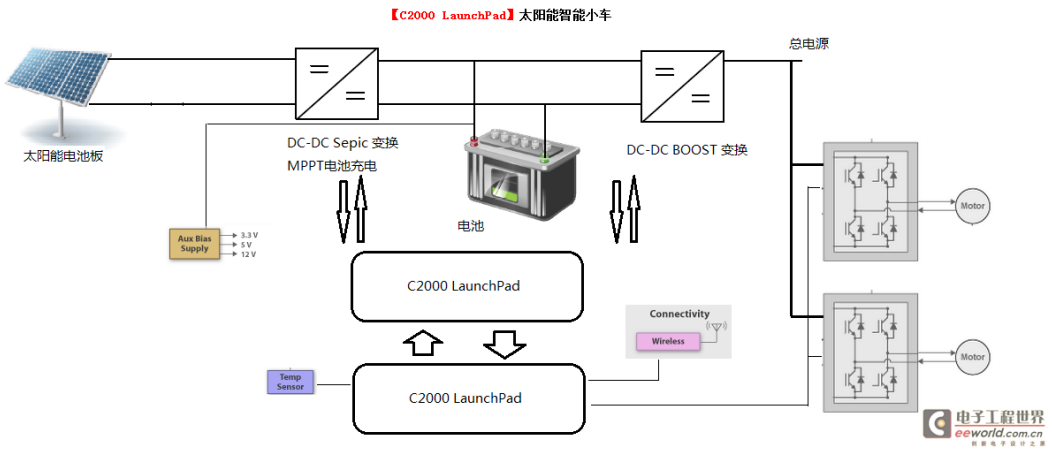
3.3.1 太阳能智能小车

一、【功能描述】

TMS320F28027 通过太阳能电池板收集能源，然后通过外围电路把电能储存到电池中，同时直流变换输出给整机其他外设工作供电．同时控制直流有刷电机对小车行使控制，充分体现 28027 在新能源，数字电源控制，电机驱动等方面的应用．

二、【系统结构框】

如图所示



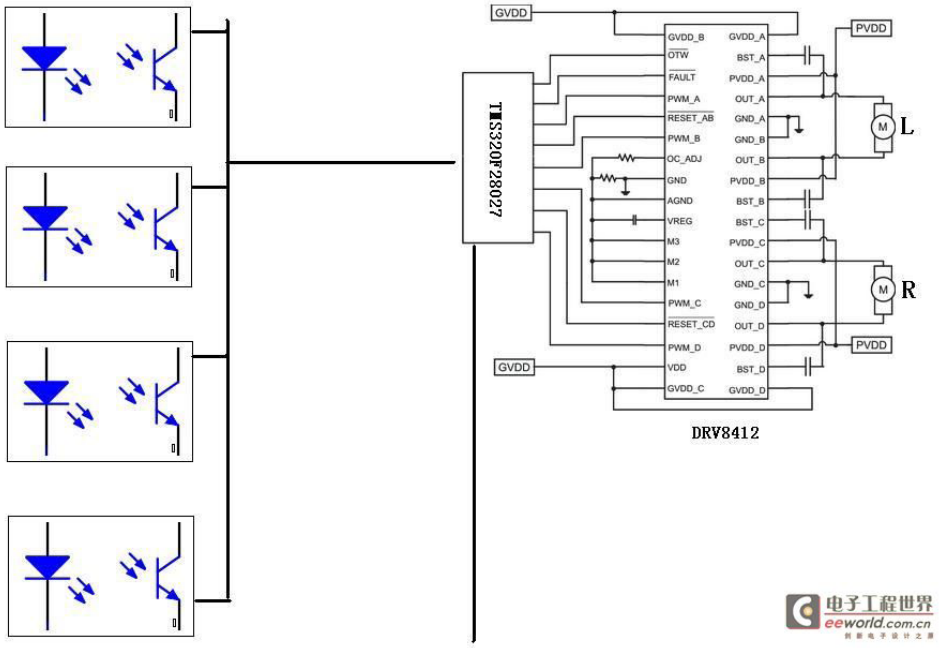
TI 相关资源

1，安装 controlSUITE
2，安装路径 :\ti\controlSUITE\libs\app_libs

相关 TI 库文件资源

太阳能相关 : 安装路径 :ti\controlSUITE\libs\app_libs\solar
数字电源： 安装路径 :\ti\controlSUITE\libs\app_libs\digital_power
电机控制： 安装路径 :\ti\controlSUITE\libs\app_libs\motor_control

小车驱动部分，基本结构



更多详情 : http://bbs.eeworld.com.cn/thread-369064-1-1.html

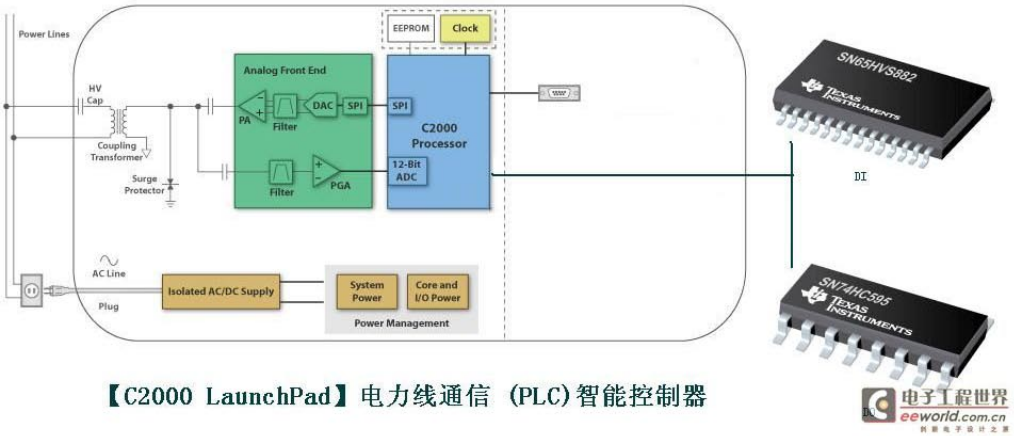
3.3.2 电力线通信 (PLC) 智能控制器

一、【功能描述】

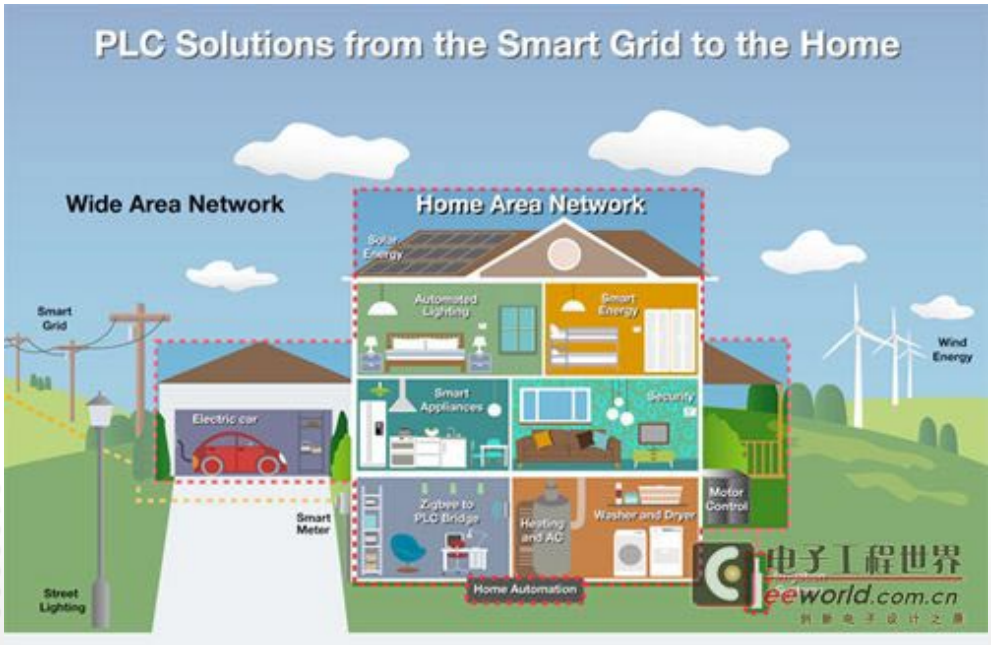
采用 TMS320F28027 实现最小的电力线通讯智能控制器，作为智能家具的一小控制单元，通过现成的家庭供电线路进行电力线载波通讯，实现电灯，电视，其他家用电器进行控制，监测。

二、【系统结构框】

如图所示



【C2000 LaunchPad】电力线通信 (PLC) 智能控制器



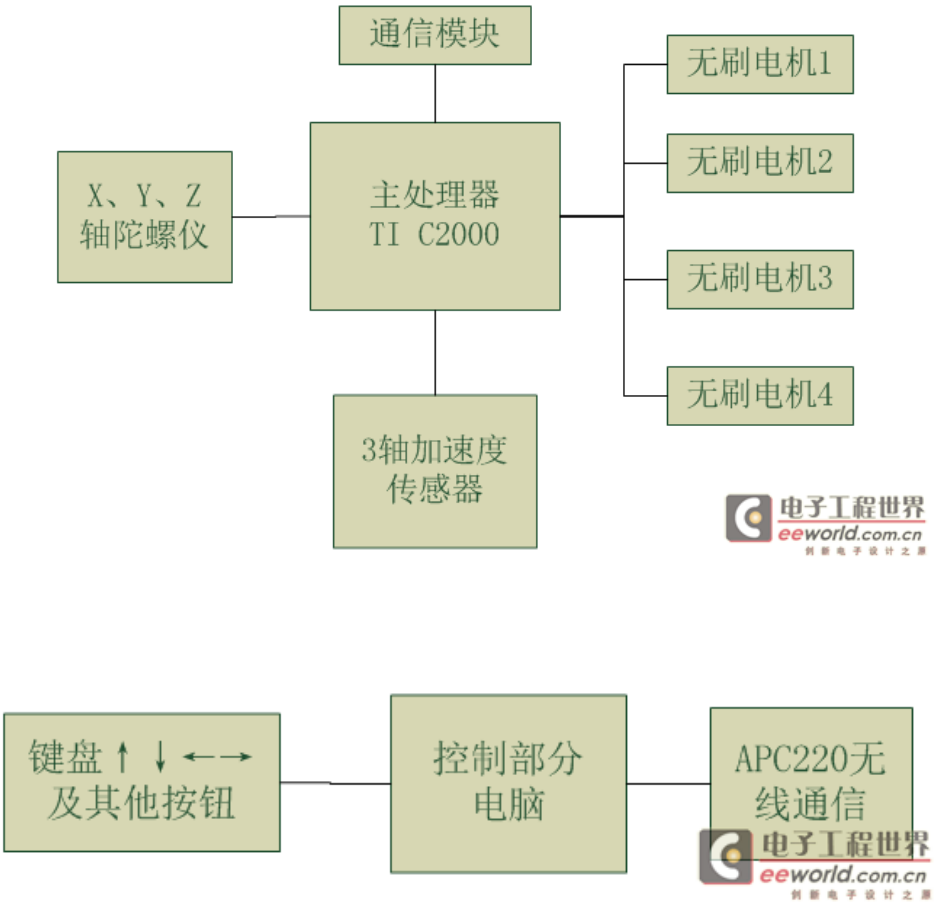
更多详情：http://bbs.eeworld.com.cn/thread-369086-1-1.html

3.3.3 小型四轴飞行器

[项目名称]、基于 TI C2000 LaunchPad 的小型四轴飞行器

[功能描述]、以 C2000 为主控制作小型四轴飞行器

[软硬件框图]、

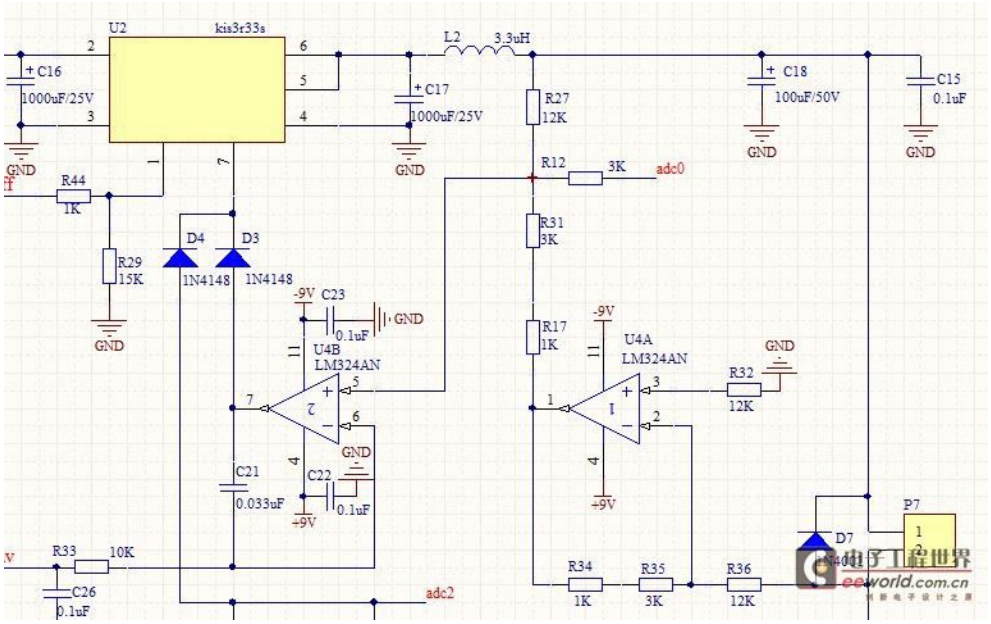


更多详情 : http://bbs.eeworld.com.cn/thread-369075-1-1.html

3.3.4 高精度数控恒流恒压同步整流电源

【功能描述】：使用 C2000 LaunchPad 作为控制板，使用降压型 DC-DC 转换器 kis3r33s 电源模块和 LM324 运放组成稳压电路，调整 28027 的 PWM 输出控制输出电压，输出电压的范围控制在 0~20V。通过运放和二极管组成稳流电路，调整电流的控制 PWM，使得电流的调整范围控制在 0~3A

【电源的实现原理】：



电压的调整精度为 0.02v，电流的调整精度为 0.005A。这个电源可以满足大部分的实验电源需求，鉴于这个电源的调整精度比较高，又具有恒流恒压的功能，在这个设计中加入了充电的功能，在单片机中存入多组常用的使用电压电流数据方便调取，提供充电时间定时关断的功能。

电源的结构设计打算使用“塔”状的结构，底层为电源的输入和输出电路，中间层为 C2000 LaunchPad，最上面一层为液晶显示和按键调整等功能，这三块板之间通过排针很排母连接。

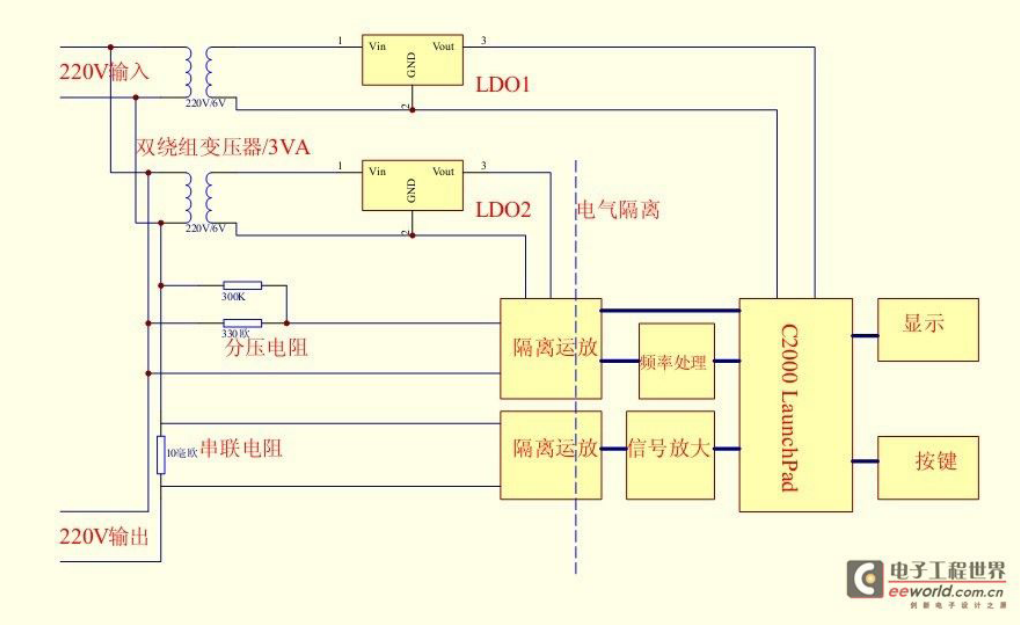
更多详情：http://bbs.eeworld.com.cn/thread-369567-1-1.html

3.3.5 家用智能功率监视器

[功能描述]:随着家用电器的越来越普及,用电量大成了家庭的一个重大问题,其实用电量大除了家用电气增多这一客观因素外,更多的是不合理的使用家用电器。比如让一些耗电量大的电器长期处于待机状态。还有一些就是家用电器的实际耗电量和标称的不符,这一点集中表现在冰箱和空调上。如果能找出家中耗电量大的电器并合理利用,不仅可以节约一笔电费,也有助于节能减排,造福子孙(帽子有点儿大)。本装置的目的是监视某一个家用电器的瞬时功率以及累计电能,同时显示实时电压电流,外加实时时钟。暂定这么多功能吧。这个东西 TI 已经有了,具体的名称忘了,改天找一下,国内一家做电能采集的厂家也做了类似的东西。

[实施方案描述]:CPU 当然使用 C2000 LaunchPad 上的 TMS320F28027 了,利用 TMS320F28027 的内部 AD 采集电压电流值,计算实时电压电流和功能,并计算电能量。显示部分初步确定采用12864或者再小一些的OLED屏(因为没玩过,想试一下)。三个按键,操作简单。比较麻烦的是电压电流采集部分,暂时有两个方案,一个是隔离方案,一个是非隔离的。隔离方案使用 TI 的 AMC1200 隔离运放作为隔离器件,电压经过分压,电流经过串联电阻后再通过隔离运放给 28027 采集。非隔离的方案就是省去了隔离运放直接采集分压后的电压值,电流采用互感器隔离一下。考虑到调试的时候的安全,先做隔离方案,后续再做非隔离的。电源部分也是比较头疼的,由于使用交流 220V 供电,而实际使用的是 3.3V 的直流电,使用现成的 220V 转 5V 的开关电源模块是比较简单,但体积比较大,自己做吧体积可能会小点儿,但没做过。而且如果采用隔离方案的话就需要两个电源,就更麻烦了。所以暂定采用线性电源的方案,用一个 3-5VA 双路输出的的变压器加两片 LDO 解决,体积可能会小一些。通讯留一个串口,当然就是 LaunchPad 上那个 USB 转的串口啦,和计算机通讯,当然上位机软件我没时间做啦,这个是后续的工作。功能很简单,关键是电压电流功率及电能的精度。初步的目标是实现电压电流 0.5% 的精度。功率电能 1% 的精度。电压范围当然就是 220V 了,当然往低应该可以测到 100V 或者更低,但是由于采用线性电源供电,100V 的有可能无法工作,往高的话到 264V,220V 的 120%,再高了也就没什么用了。电流的范围应该会比较大,一般几百 W 的电器比较多,电烤箱的话一般得在两千 W,这样算的话电流至少得达到 10A,考虑到更大功率的电器,暂定最大电流测到 15A。而笔记本,电灯等小电器的功率一般在 100W 以内,电流也就 0.5A 左右,这个范围有点儿大了,有点儿难度。暂定 0.5A-15A 吧。如果不能在这个范围内保证精度,就降低一下,在某个范围内达到上述的精度。

[软硬件框图] : 简单的示意图。



更多详情：http://bbs.eeworld.com.cn/thread-369831-1-1.html

3.3.6 C2000 外围电路设计 - 光伏逆网逆变器

一、项目名称：光伏离网逆变器

二、功能描述：

光伏离网逆变器主要功能是白天给蓄电池充电，晚上蓄电池放电供负载使用，如太阳能路灯。这次设计的光伏离网逆变器是功率在 500W，专门用于牧区的供电。白天太阳能电池板要给蓄电池充电，控制器要控制充电过程，防止畜电池过充，同样充电的同时也可以放电使用，晚上控制器给负载供电。

三、实施方案与描述：

本次设计方案是一款成熟的产品，主要部分包括太阳能控制器、逆变控制回路、主回路、检测电路、保护电路。

1) 控制器方案：

小型的光伏逆变器一般选用 8 位单片机作为控制器，但是单片机处理速度有限，只能起到简单的控制和监控作用，逆变器的脉冲产生一般要靠模拟电路设计，这样机器的整体性能就会下降。

选择 C2000 的 TMS320F28027 芯片，可以提高控制器的性能。C2000 是一款低端 C2000 家庭系列。最高频率达到 60MHZ，三个 32 位定时器，增强型 PWM 输出，灵活配置的 ADC 模块，能够满足机器的控制要求。TMS320F28027 主要负载太阳能充电控制、逆变器控制、电流电压检测、过压、过流保护的功能，取代传统的模拟电路设计，大大提高产品性能。

2) 充电器结构方案：

光伏充电器采用太阳能电池板与蓄电池串联使用，用 MOS 管控制两者的导通。这样要合理配置电池板电压和蓄电池电压。。

3) 逆变控制器：

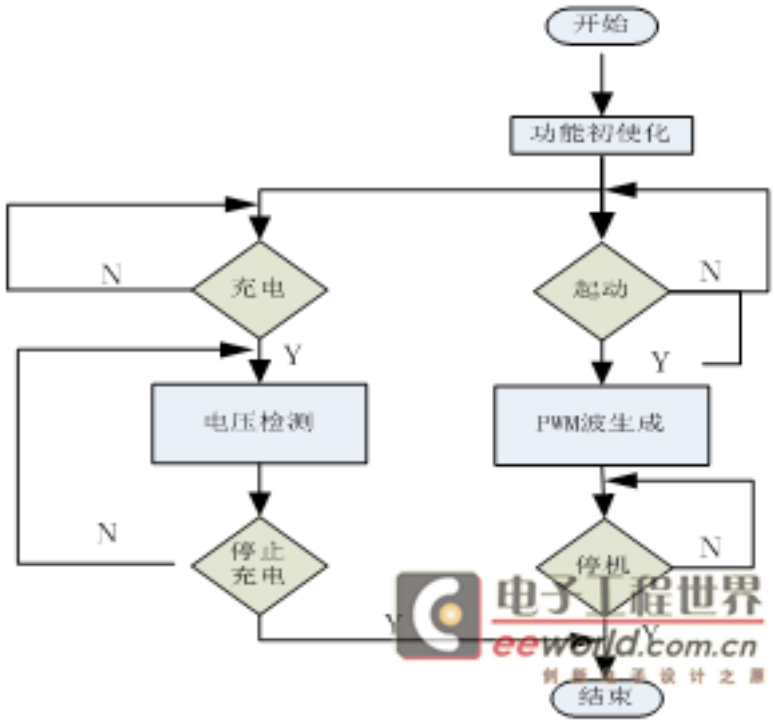
主要负责将交流电变为直流电，控制器产生 SPWM 脉冲，控制 H 桥 MOSFET 的导通，产生高频脉冲方法，经过滤波电路，然后经过变压器升压到 220V，输出。

4) 主回路：H 桥使用 MOSFET 管，为了扩大容量，上下桥臂的 MOSFET 都使用三颗并联。

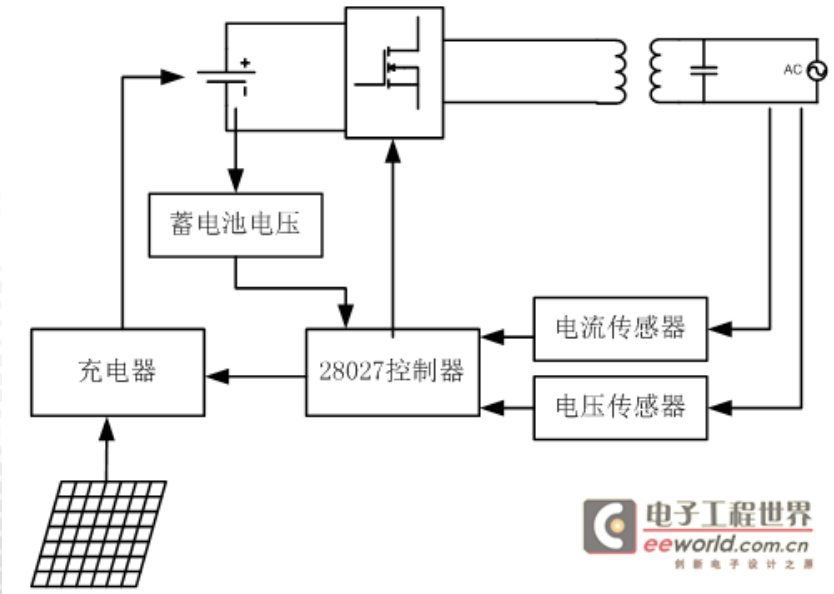
5) 检测电路：包括电压检测、电流检测。接合保护电路，起到过压、过流保护的功能。

四、软硬件框图

软件框图：



硬件电路图



更多详情：http://bbs.eeworld.com.cn/thread-371292-1-1.html

3.3.7 基于 C2000 的卫星导航软件接收机的设计

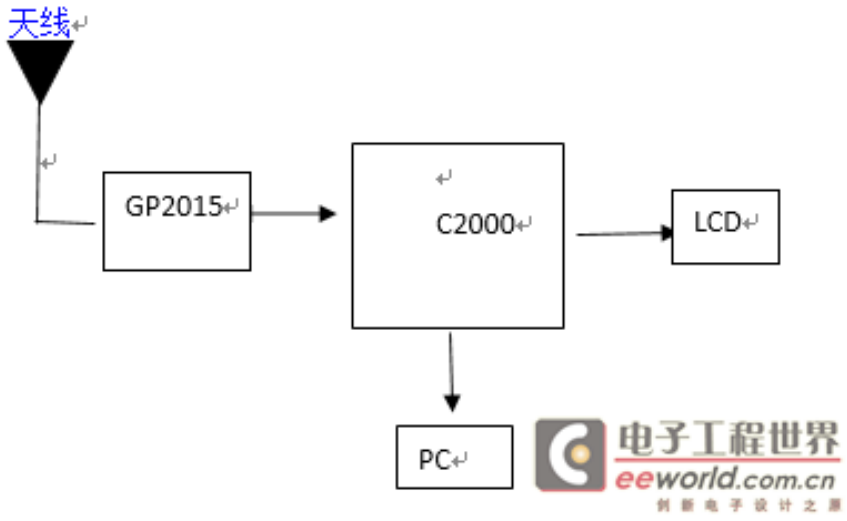
【项目名称】：基于 C2000 的卫星导航软件接收机的设计

【功能描述】：通过变频芯片将卫星信号降到较低的中频，然后通过 ADC 模块对信号实现数字化，接收机的捕获、跟踪、定位等功能则由 DSP 实现。

【实施方案描述】：由射频前端 GP2015 对天线接收的信号进行变频，输出模拟中频信号，ADC 对中频信号进行采样和量化，然后传输到 TMS320F28027 DSP 进行相关的运算处理，完成卫星信号的捕获、跟踪和定位等功能，最后将结果通过输出模块传送到显示终端显示定位等信息。

【整机性能分析】：Piccolo 处理器支持高达 80 MHz 的速度和高达 256kB 的集成型闪存、专用的高分辨率 PWM、功能强大的 ADC、模拟比较器及成本敏感型混合信号器件中的通信接口。LCD 选用 TFT 彩屏（2.6 寸），处理器选用 TMS320F28027. 理论上是可行的。

其他 :Piccolo 处理器支持高达 80 MHz 的速度和高达 256kB 的集成型闪存、专用的高分辨率 PWM、功能强大的 ADC、模拟比较器及成本敏感型混合信号器件中的通信接口。



附录一：
C2000 Launchpad 入门资料集锦

- 太阳能微逆变器方案介绍
  http://www.eeworld.com.cn/training/2013/TI_C2000_0424/15.html
- C2000 系列之 concerto 讲座
  http://www.eeworld.com.cn/training/2012/TI_C2000_1106/72.html
- TI PLC 开发套件简介
  http://www.eeworld.com.cn/training/2013/TI_C2000_G_0428/62.html
- 采用 C2000 TMS320F28027 的 LaunchPad 启动开发工作
  http://www.eeworld.com.cn/whitepaper/show.php?itemid=546
- 探索 C2000 ™ Launchpad ™评估套件
  http://bbs.eeworld.com.cn/thread-346840-1-5.html
- 非库方式新建 C2000 工程入门
  http://bbs.eeworld.com.cn/thread-362200-1-1.html
- 【C2000 LaunchPad】让 LED 闪起来
  http://bbs.eeworld.com.cn/thread-360676-1-2.html
- 【C2000 LaunchPad】PWM 模块原理及使用
  http://bbs.eeworld.com.cn/thread-361436-1-1.html
- 【C2000 LaunchPad】UART 作为调试监视
  http://bbs.eeworld.com.cn/thread-360852-1-1.html
- C2000 Piccolo LaunchPad 之 CCS5 安装
  http://bbs.eeworld.com.cn/thread-361212-1-2.html
- C2000 Piccolo LaunchPad 之 controlSUITE
  http://bbs.eeworld.com.cn/thread-361564-1-2.html
- C2000 LAUNCHPAD DEMO 程序学习
  http://bbs.eeworld.com.cn/thread-363097-1-1.html

- C2000 Piccolo LaunchPad 资料
  http://bbs.eeworld.com.cn/thread-359694-1-2.html
- DSP C2000 程序员高手进阶
  http://bbs.eeworld.com.cn/thread-285376-1-2.html
- CCS5 使用说明文档
  http://bbs.eeworld.com.cn/thread-347800-1-1.html
- 非库方式新建 C2000 launchpad 工程入门
  http://bbs.eeworld.com.cn/thread-362200-1-1.html
- C2000 LaunchPad 原理图 PDF 文件格式
  http://bbs.eeworld.com.cn/thread-371438-1-1.html
- 【玩转 C2000 LaunchPad】ADC 使用技巧
  http://bbs.eeworld.com.cn/thread-363491-1-1.html

附录二：
C2000 Launchpad 实用问答

- F28027 的 AD 采样结果寄存器什么时候清零？
  http://bbs.eeworld.com.cn/thread-327372-1-1.html
- C2000 launchpad 无法连接目标板
  http://bbs.eeworld.com.cn/thread-361762-1-1.html
- C2000 LaunchPad 的 usb 转串口驱动我找不到
  http://bbs.eeworld.com.cn/thread-363856-1-1.html
- 求助：F28027 的 cmd 文件中的“>>”和“|”
  http://bbs.eeworld.com.cn/thread-366332-1-1.html
- 谁知道 C2000 Launchpad 几个拨键做用吗？
  http://bbs.eeworld.com.cn/thread-373689-1-1.html
- F28027 用 SPI 操作 SD 卡相关问题
  http://bbs.eeworld.com.cn/thread-372120-1-1.html

附录三：
C2000 LAUNCHPAD 应用实例

- 【C2000 LaunchPad】让 LED 闪起来
  http://bbs.eeworld.com.cn/thread-360676-1-1.html
- 【C2000 LaunchPad】加个 LCD
  http://bbs.eeworld.com.cn/thread-362930-1-1.html
- 教你制作如同 LaunchPad 上的 XDS100V2，自带串口
  http://bbs.eeworld.com.cn/thread-360974-1-1.html
- 【C2000 Launchpad】+EPWM 模块实现 SPWM 波形输出
  http://bbs.eeworld.com.cn/thread-363124-1-1.html
- 【C2000 Launchpad】SCI to ADC
  http://bbs.eeworld.com.cn/thread-363356-1-1.html
- 【C2000 Launchpad】节日呼吸灯
  http://bbs.eeworld.com.cn/thread-363380-1-1.html
- 【C2000 Launchpad】SCI+ADC+NIKIA5110
  http://bbs.eeworld.com.cn/thread-364086-1-1.html
- C2000launchpad 仿真器成功连接 28030 开发板
  http://bbs.eeworld.com.cn/thread-373089-1-1.html
- 【玩转 C2000 Launchpad】数字功放
  http://bbs.eeworld.com.cn/thread-365306-1-1.html
- 成功实现 C2000LAUNCH 和我的“家庭物联网”无线数据传输
  http://bbs.eeworld.com.cn/thread-364702-1-1.html

附录四：
C2000 程序员高手进阶节选

摘自《C2000 程序员高手进阶》，原名《一个 DSP 程序员应具备的素质》

1. 在定点 DSP 中，能熟练运用 Q 格式。

2. 能编写出大量符合 TI 公司规范的代码程序，虽然不可将规范看的太死。

很多小公司没有软件规范，即使是一些大公司，其制定的规范也失于简单读者可以在网上搜索看看国内某些大公司的规范,再与 TI 公司的作比较。（例如华为）

3. 能熟练运用 C 语言,从 C51 到 DSP、ARM 等,这是个嵌入式编程泛 C 的时代，程序员最好还能知晓一些面向对象语言，如 C++,JAVA 或 C#, 灵活吸收其适合小规模程序的思想和技巧。

4. 好的嵌入式 C 语言程序员，并不是明白 3 重指针怎么用之类的人。除了了解语言的细节，一个好的嵌入式 C 语言程序员应该能根据器件的特点，大约估算出 C 语言可能对应的汇编指令和时间性能，并能编写汇编函数来加快代码速度。

5. 专业知识。专家之所以称为专家，是因为他对研究的具体对象了然于胸。

6. 如果上述的条件还不具备，不要灰心，只要具备这点就够了---- 保持良好的学习心态，不断的学习能带来成长感！

附录五：编委信息与后记

《C2000 Launchpad 炼成记》通过对 TI 教室“采用 C2000 TMS320F28027 的 LaunchPad 启动开发工作”培训教程的整理，同时结合了 EEWORLD 社区资深工程师对 TI 官网资料的整合以及一手的评测报告。帮助更多工程师更快完成设计。

在此特别感谢：

EEWORLD 社区（http://bbs.eeworld.com.cn/forum-92-1.html）坛友对我们活动的支持与关注，产生了大量新鲜、一手的应用经验。

感谢 TI 公司的大力支持，希望《C2000 Launchpad 炼成记》，能够为电子工程师设计工作提速！

EEWOLRD 社区

2013.6.4

附录六：版权说明

1、《C2000 Launchpad 炼成记》著作权属 TI 和 EEWORLD 共同所拥有；

2、本着开源思想，我们授权任何对 C2000 Launchpad 有兴趣的工程师免费下载、复制、传播该书；

3、如用于商业用途须经 EEWORLD 书面同意。

重要声明

德州仪器(TI) 及其下属子公司有权根据 JESD46 最新标准, 对所提供的产品和服务进行更正、修改、增强、改进或其它更改， 并有权根据 JESD48 最新标准中止提供任何产品和服务。客户在下订单前应获取最新的相关信息, 并验证这些信息是否完整且是最新的。所有产品的销售都遵循在订单确认时所提供的TI 销售条款与条件。

TI 保证其所销售的组件的性能符合产品销售时 TI 半导体产品销售条件与条款的适用规范。仅在 TI 保证的范围内，且 TI 认为 有必要时才会使用测试或其它质量控制技术。除非适用法律做出了硬性规定，否则没有必要对每种组件的所有参数进行测试。

TI 对应用帮助或客户产品设计不承担任何义务。客户应对其使用 TI 组件的产品和应用自行负责。为尽量减小与客户产品和应 用相关的风险，客户应提供充分的设计与操作安全措施。

TI 不对任何 TI 专利权、版权、屏蔽作品权或其它与使用了 TI 组件或服务的组合设备、机器或流程相关的 TI 知识产权中授予 的直接或隐含权限作出任何保证或解释。TI 所发布的与第三方产品或服务有关的信息，不能构成从 TI 获得使用这些产品或服 务的许可、授权、或认可。使用此类信息可能需要获得第三方的专利权或其它知识产权方面的许可，或是 TI 的专利权或其它 知识产权方面的许可。

对于 TI 的产品手册或数据表中 TI 信息的重要部分，仅在没有对内容进行任何篡改且带有相关授权、条件、限制和声明的情况 下才允许进行复制。TI 对此类篡改过的文件不承担任何责任或义务。复制第三方的信息可能需要服从额外的限制条件。

在转售 TI 组件或服务时，如果对该组件或服务参数的陈述与 TI 标明的参数相比存在差异或虚假成分，则会失去相关 TI 组件 或服务的所有明示或暗示授权，且这是不正当的、欺诈性商业行为。TI 对任何此类虚假陈述均不承担任何责任或义务。

客户认可并同意，尽管任何应用相关信息或支持仍可能由 TI 提供，但他们将独力负责满足与其产品及在其应用中使用 TI 产品 相关的所有法律、法规和安全相关要求。客户声明并同意，他们具备制定与实施安全措施所需的全部专业技术和知识，可预见 故障的危险后果、监测故障及其后果、降低有可能造成人身伤害的故障的发生机率并采取适当的补救措施。客户将全额赔偿因 在此类安全关键应用中使用任何 TI 组件而对 TI 及其代理造成的任何损失。

在某些场合中，为了推进安全相关应用有可能对 TI 组件进行特别的促销。TI 的目标是利用此类组件帮助客户设计和创立其特 有的可满足适用的功能安全性标准和要求的终端产品解决方案。尽管如此，此类组件仍然服从这些条款。

TI 组件未获得用于 FDA Class III（或类似的生命攸关医疗设备）的授权许可，除非各方授权官员已经达成了专门管控此类使 用的特别协议。

只有那些 TI 特别注明属于军用等级或“增强型塑料”的 TI 组件才是设计或专门用于军事/航空应用或环境的。购买者认可并同 意，对并非指定面向军事或航空航天用途的 TI 组件进行军事或航空航天方面的应用，其风险由客户单独承担，并且由客户独 力负责满足与此类使用相关的所有法律和法规要求。

TI 已明确指定符合 ISO/TS16949 要求的产品，这些产品主要用于汽车。在任何情况下，因使用非指定产品而无法达到 ISO/TS16949 要求，TI不承担任何责任。

	产品		应用
数字音频	www.ti.com.cn/audio	通信与电信	www.ti.com.cn/telecom
放大器和线性器件	www.ti.com.cn/amplifiers	计算机及周边	www.ti.com.cn/computer
数据转换器	www.ti.com.cn/dataconverters	消费电子	www.ti.com/consumer-apps
DLP® 产品	www.dlp.com	能源	www.ti.com/energy
DSP - 数字信号处理器	www.ti.com.cn/dsp	工业应用	www.ti.com.cn/industrial
时钟和计时器	www.ti.com.cn/clockandtimers	医疗电子	www.ti.com.cn/medical
接口	www.ti.com.cn/interface	安防应用	www.ti.com.cn/security
逻辑	www.ti.com.cn/logic	汽车电子	www.ti.com.cn/automotive
电源管理	www.ti.com.cn/power	视频和影像	www.ti.com.cn/video
微控制器 (MCU)	www.ti.com.cn/microcontrollers		
RFID 系统	www.ti.com.cn/rfidsys		
OMAP应用处理器	www.ti.com/omap		
无线连通性	www.ti.com.cn/wirelessconnectivity	德州仪器在线技术支持社区	www.deyisupport.com

邮寄地址： 上海市浦东新区世纪大道 1568 号，中建大厦 32 楼 邮政编码： 200122