C2023 DSP 试验箱

试验指导用书

1　概述

1.1　TMS320LF2407A　简介

TMS320LF2407A　芯片作为是　TI 公司 TMS320C2023　系列中的一种　16 位定点 DSP 芯片，是目前应用最为广泛的芯片。它为　　C2xxCPU　功能强大的　TMS320　DSP　构造设计供给了低本钱、低功耗、高性能的处理力量，对电机的数字化掌握格外有用。同时，几种先进的外设被集成到该芯片内，形成了真正意义上的数字掌握器。

TMS320LF2407A　的主要特点为：

- 承受了高性能静态 CMOS 技术，使得供电电压降为 3.3V，削减了掌握器的功耗。最高 40MIPS 的执行速度使得指令周期缩短为　25ns，从而提高了掌握 器的实时掌握力量
- 片内有高达 32K 字的 FLASH 程序存储器，数据存储器包含 2K 字的 SARAM 和 544 个字的 DARAM
- 可扩展的外部存储器总共有 192K 字：64K 字程序存储器；64K 字数据 存储器；64K 字 I/O 寻址空间
- 片内集成有两个大事治理器模块　　EVA 和 EVB，每个大事治理器包括：两个　　16 位通用定时器〔　GP〕、三个比较单元、三个捕获单元以及一个正交编码脉冲电路
- 片内集成有模数转换模块〔　ADC〕，该模块是内置采样和保持〔　S/H〕的 10 位精度的模式转换器，共有　16 个模拟输入通道〔　ADCIN0~ADCIN15　〕，最小 A/D 转换时间是 375ns
- 掌握器局域网〔　CAN〕2.0B 模块，该模块是一个完全的　　CAN 掌握器，完全支持 CAN2.0B 协议，有六个邮箱可用于发送接收数据
- 片内集成有正交编码脉冲〔　　QEP〕电路可用于检测电机的角位移和转向
- 40 个独立可编程的双向通用 I/O 口
- 两个串行通讯口：　SPI 和 SCI
- 看门狗定时器模块〔　　WDT〕和电源驱动保护电路，以提高系统的安全牢靠性

1.2　DSP 应用软件的开发流程

DSP 软件的开发流程如图　　1 所示，涉及 C 编译器、汇编器、链接器等软件开发工具　〔图中的灰色局部〕　。假设只是开发一个汇编程序，则不需要用到　　C 编译器。

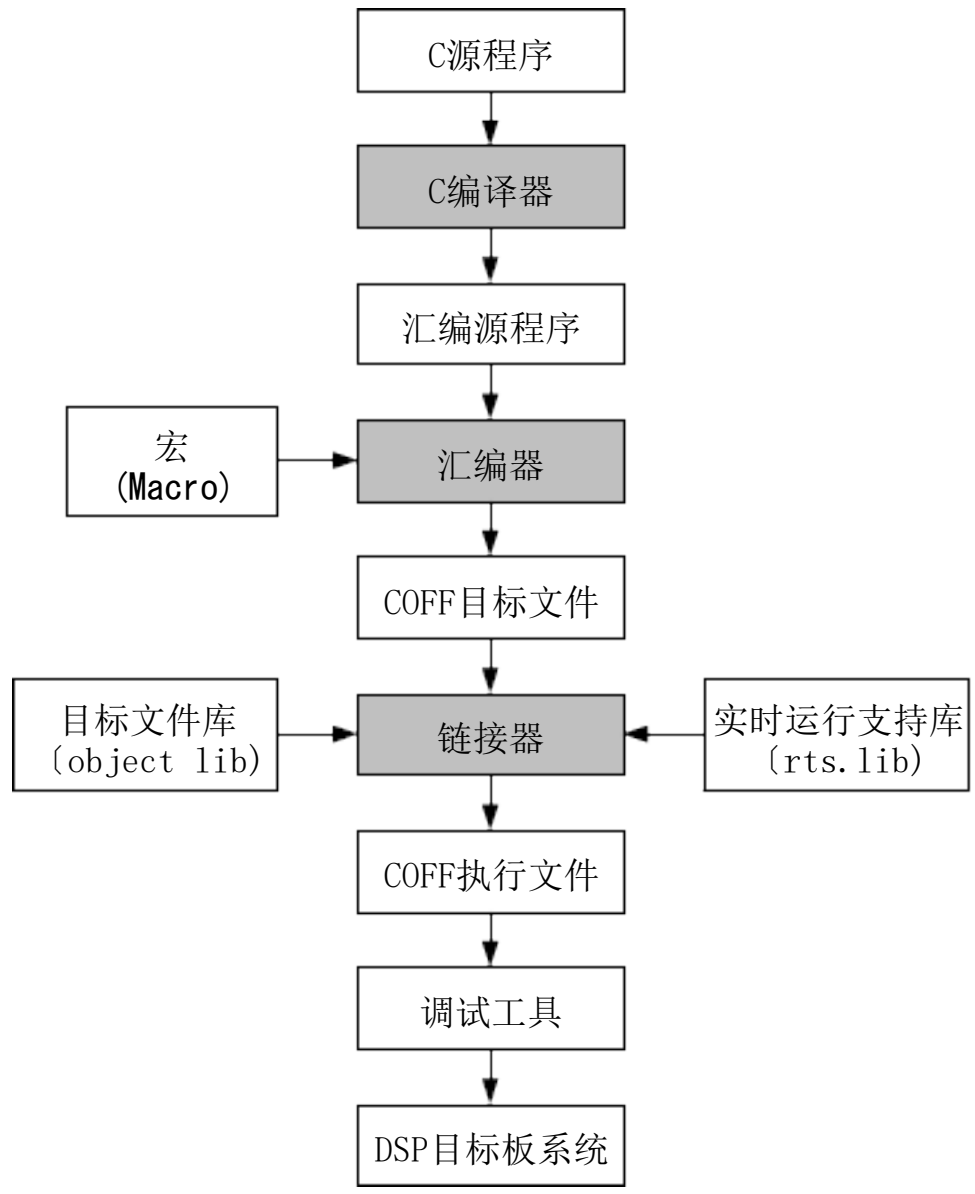


图 1　C 语言应用软件的开发流程图

1.1　如何编写源程序和 cmd 文件

TMS320　汇编语言程序　是分段编写的，这就是按所谓的　　COFF 文件格 式组织程序；在程序中除了有硬指令语句外，还有很多汇编指令〔伪指令〕语句，它们是汇编源程序的重要组成局部。在每条语句后面可以用分号隔开，写上注释，注释不参与汇编连接和最终的操作，只是为了便于阅读和修改而作的程序说明。

一个完整的汇编程序至少有三种根本的文件：汇编语言文件、头文件和命令文件。汇编语言文件名的后缀为　.ASM 。书写该文件所用指令为　LF2407 支持的 汇编语言指令。通常在该文件的最开头会写上　.include　“F2407REGS.H 〔或者 2407regs.h 〕”，说明该程序包含了　F2407REGS.H　头文件里面的一些存放器定义。

头文件中定义　DSP 系统用到的一些存放器映射地址，　用户用到的常量和用户自定义的存放器。头文件的后缀为　.H。

命令文件名的后缀为　D ，该文件实现对程序存储器空间和数据存储器空间的安排。该文件中常用到的伪指令有　MEMORY 和 SECTIONS 。

2　DSP 集成开发环境

TI 公司 DSP 的集成开发环境　CCS〔Code Composer Studio　〕供给了环　境配置、源文件编辑、程序调试、跟踪和分析等工具，可以帮助用户在一个软件环境下完成编辑、编译、链接、调试和数据分析等工作。

CCS 一般工作在两种模式下：软件仿真和与硬　　件开发板相结合的在线仿真。软件仿真〔Simulator 〕可以脱离 DSP 芯片，在 PC 机上模拟 DSP 的指令集与工作机制，主要用于前期算法实现和调试。与　硬件开发系统相结合的仿真〔　　Emulator〕是程 序实时运行在 DSP 芯片上，可以在线编制和调试应用程序。不同的　　DSP 芯片系列要承受不同型号的　CCS，对于 TMS320C2023　系列的 DSP 可承受 CCS(‘C2023) 来仿真调试。

2.1　CCS 的安装与设置

CCS 的安装过程包括三个阶段：

1）安装 CCS 到系统中。将 CCS 安装光盘放入到光盘驱动器中，运行安 装程序setup.exe 进展安装。安装完成后，在桌面上会有“　　CCS (‘C2023) ”和“ Setup CCS (‘C2023) ”两个快捷方式图标。分别对应 CCS 应用程序和 CCS 配置程序。

2）安装开发系统的软件 驱动程序，假设用北京 瑞泰公司的仿真器 ICETEK-5100 PP 来开发 2023 系列的 DSP 目标系统，则安装驱动程序 Itk2xxpp.dvr 。

3）运行“ Setup CCS(‘C2023) ”来配置程序设置 驱动程序。系统将显示如下界面：

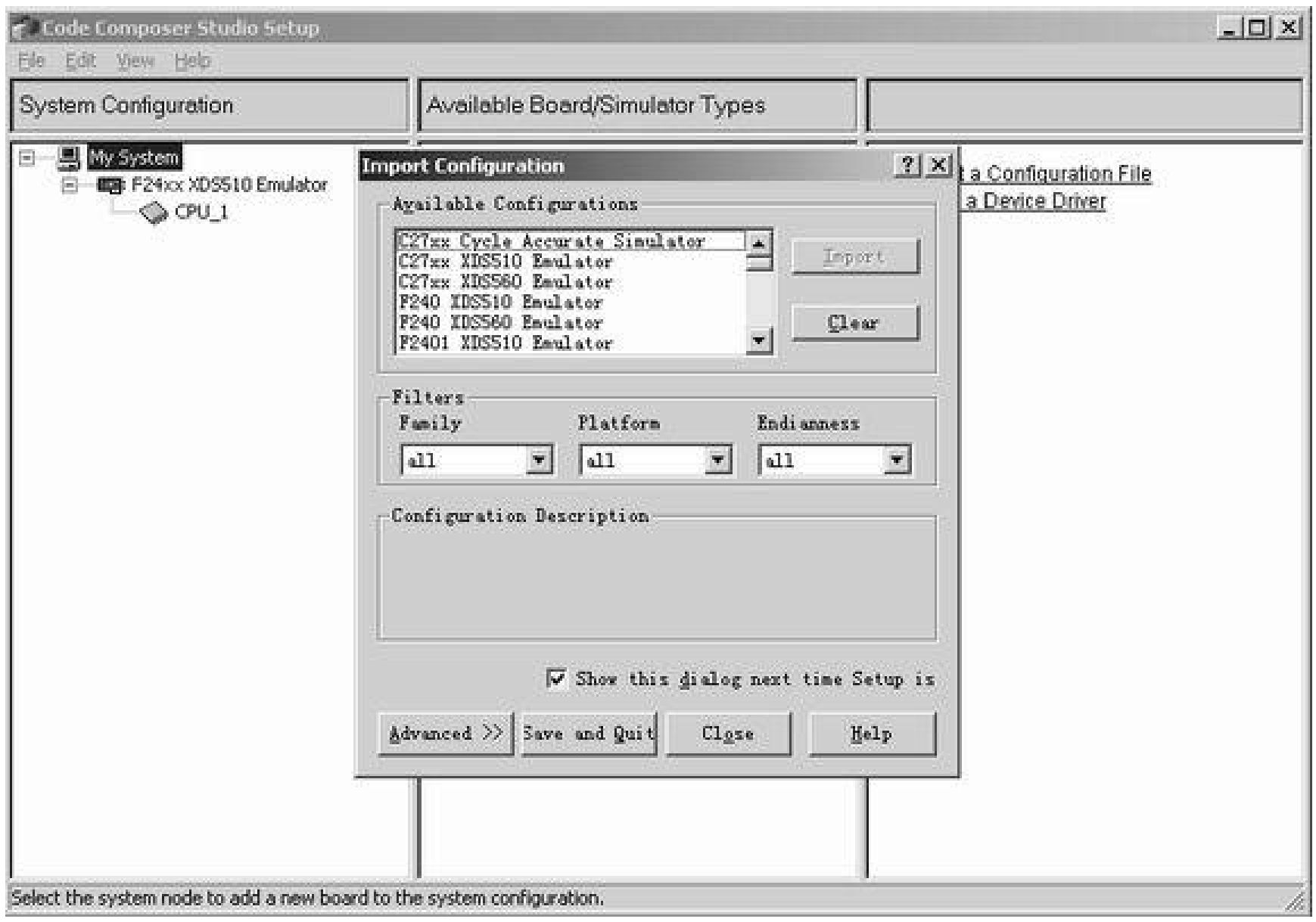


图 2 驱动程序配置界面

4）〕关闭弹出的“ Import Configuration ”对话 框〔单击 Close 按钮〕。

② 〕点击 Edit → Install Driver 选择相应 DSP 和仿真器的驱动程序 Itk2xxpp.dvr 。

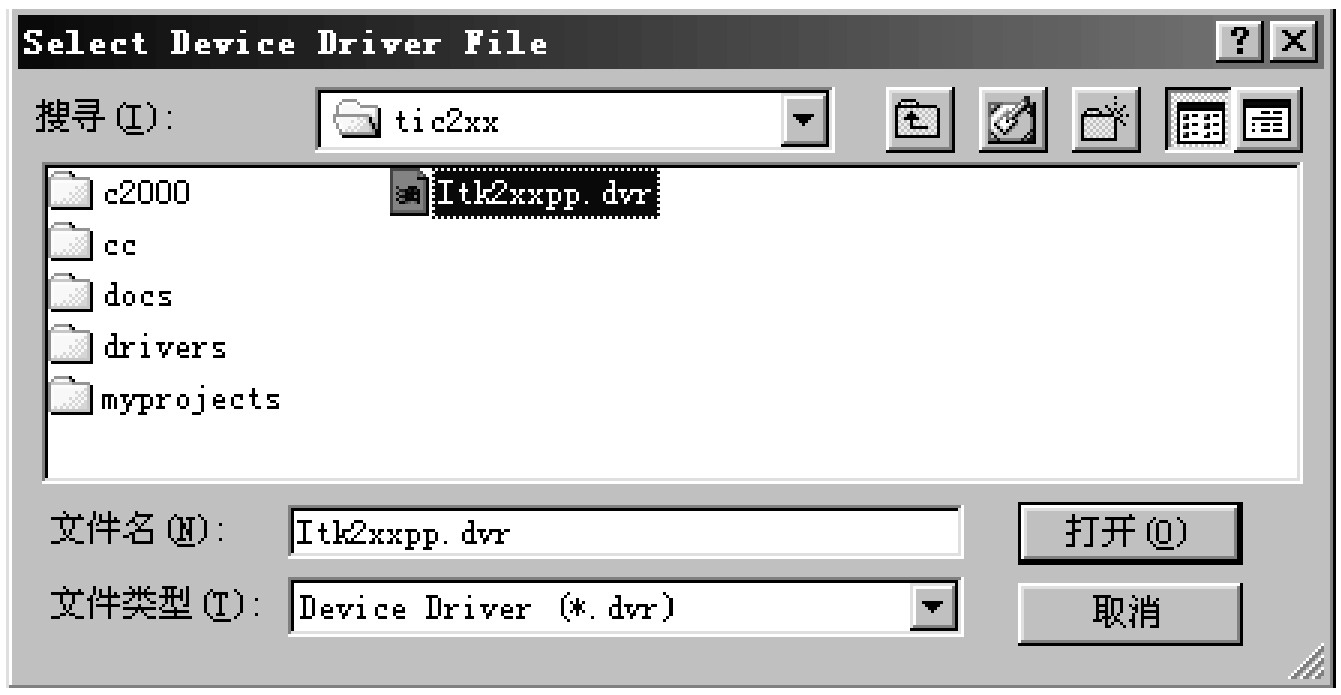


图 3 选择驱动程序界面

③ 〕点击图 3 中的“翻开”，就消灭了如下界面：

图 4 设备驱动程序的属性页

④ 〕点击图 4 所示的 OK，驱动程序就加到了 可用目标板列表中了。

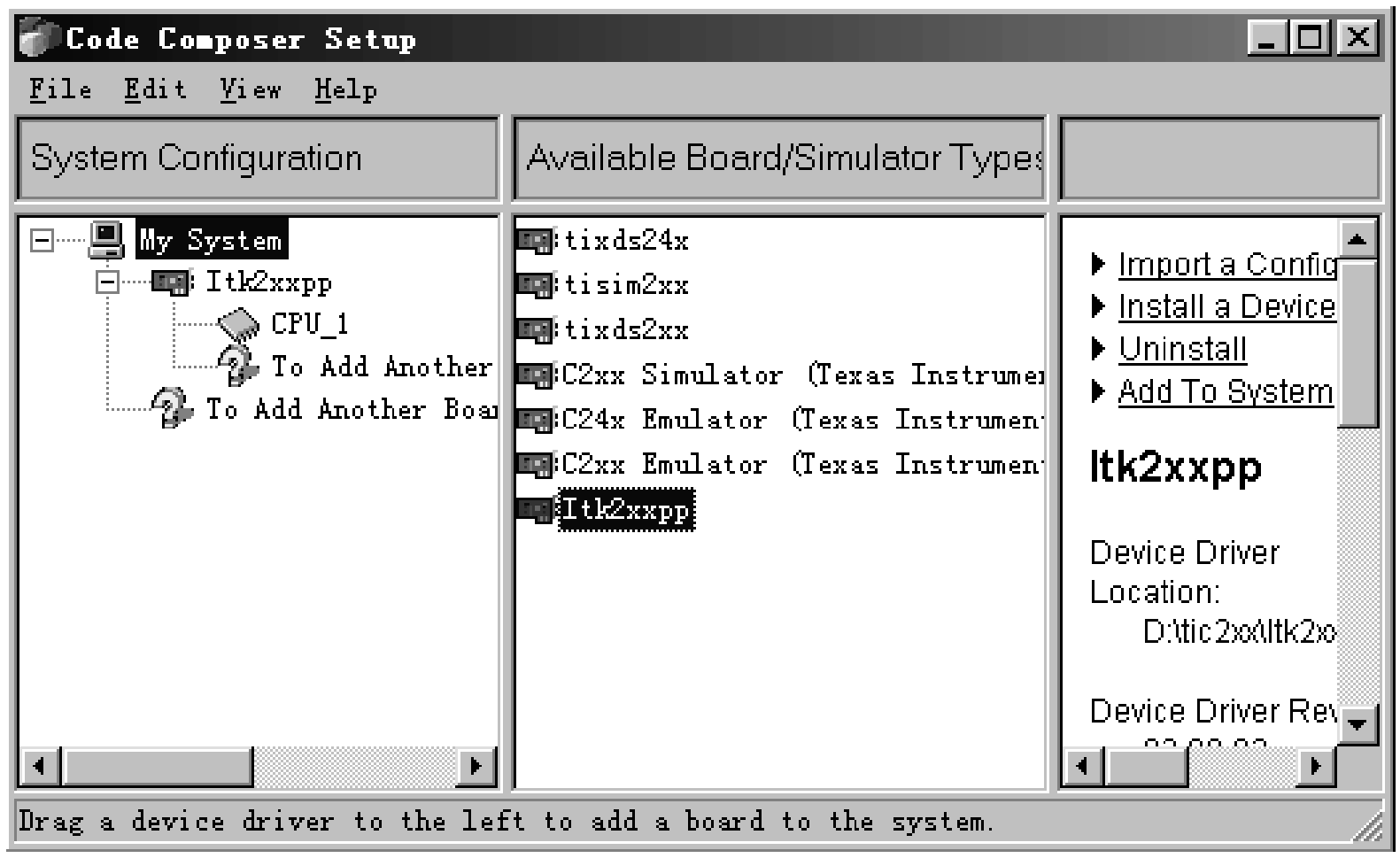


图 5 程序设置界面

① 〕选中图 5 中的 Itk2xxpp ，点击右键选中 Add to System …， 弹出 界面：

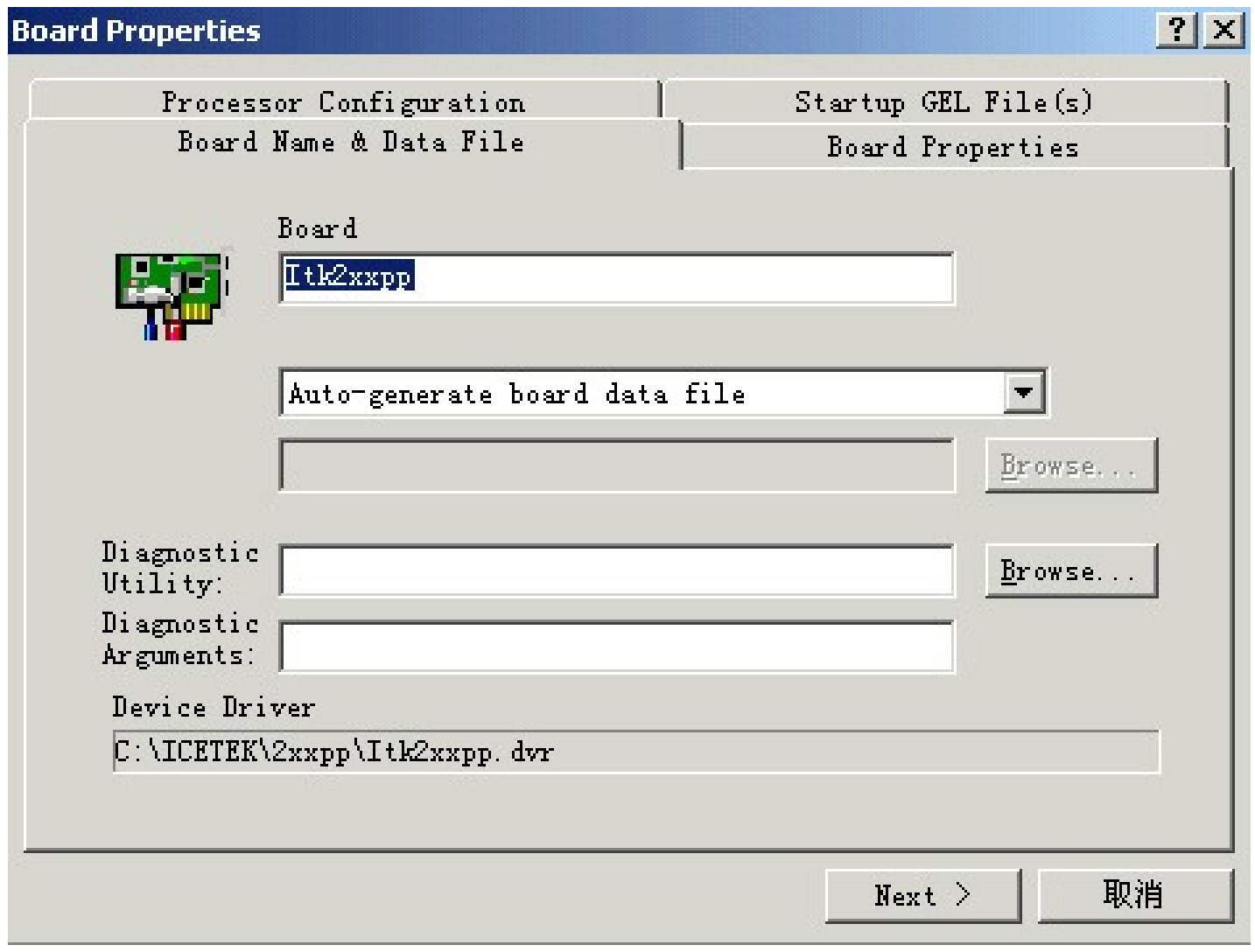


图 6 开发系统的属性选项

② 〕按图 6 中的“ Next > ”按钮连续，在弹出的 如图 7 的界面中修改相应的参数设置，把 0x240 改为0x378 ，再连续

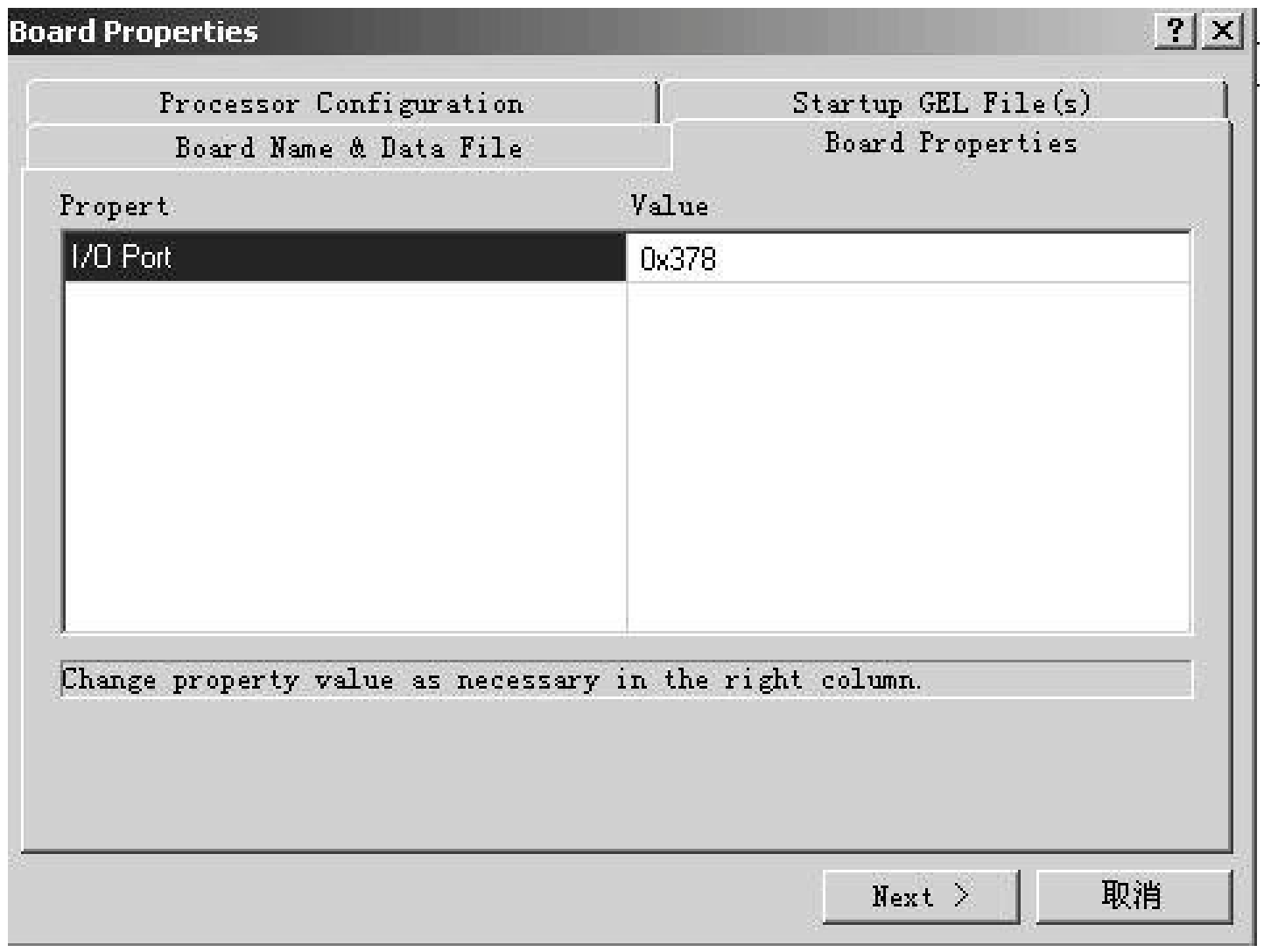


图 7 开发系统的属性配置界面

③ 〕在弹出的如图 8 的界面中点击“ Add Single ”按钮，把 cpu_1 参加到配置中，再连续。

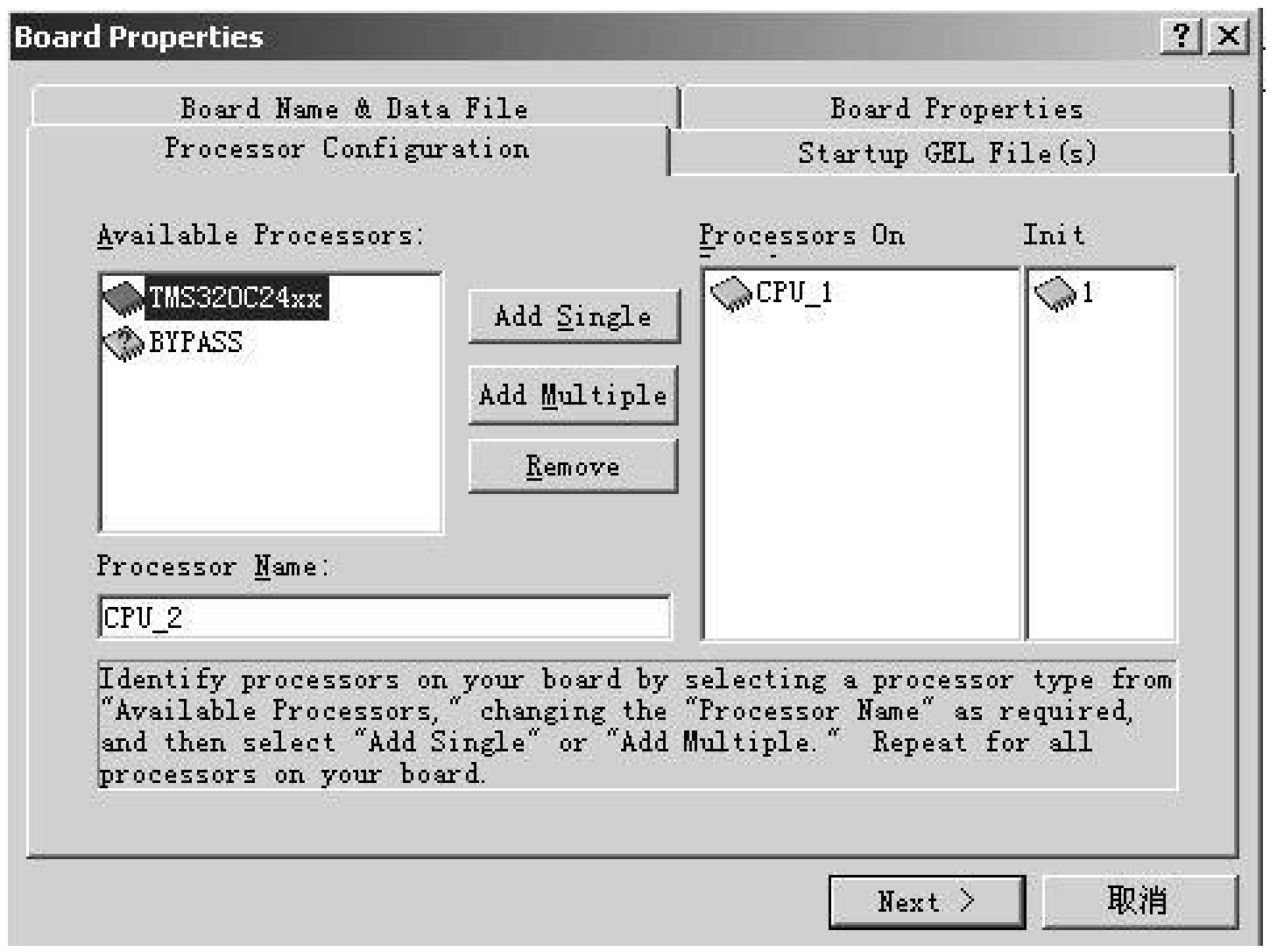


图 8 开发系统中处理器的配置选项

⑪ 〕按图 22 中的“ Finish ”按钮完毕配置。

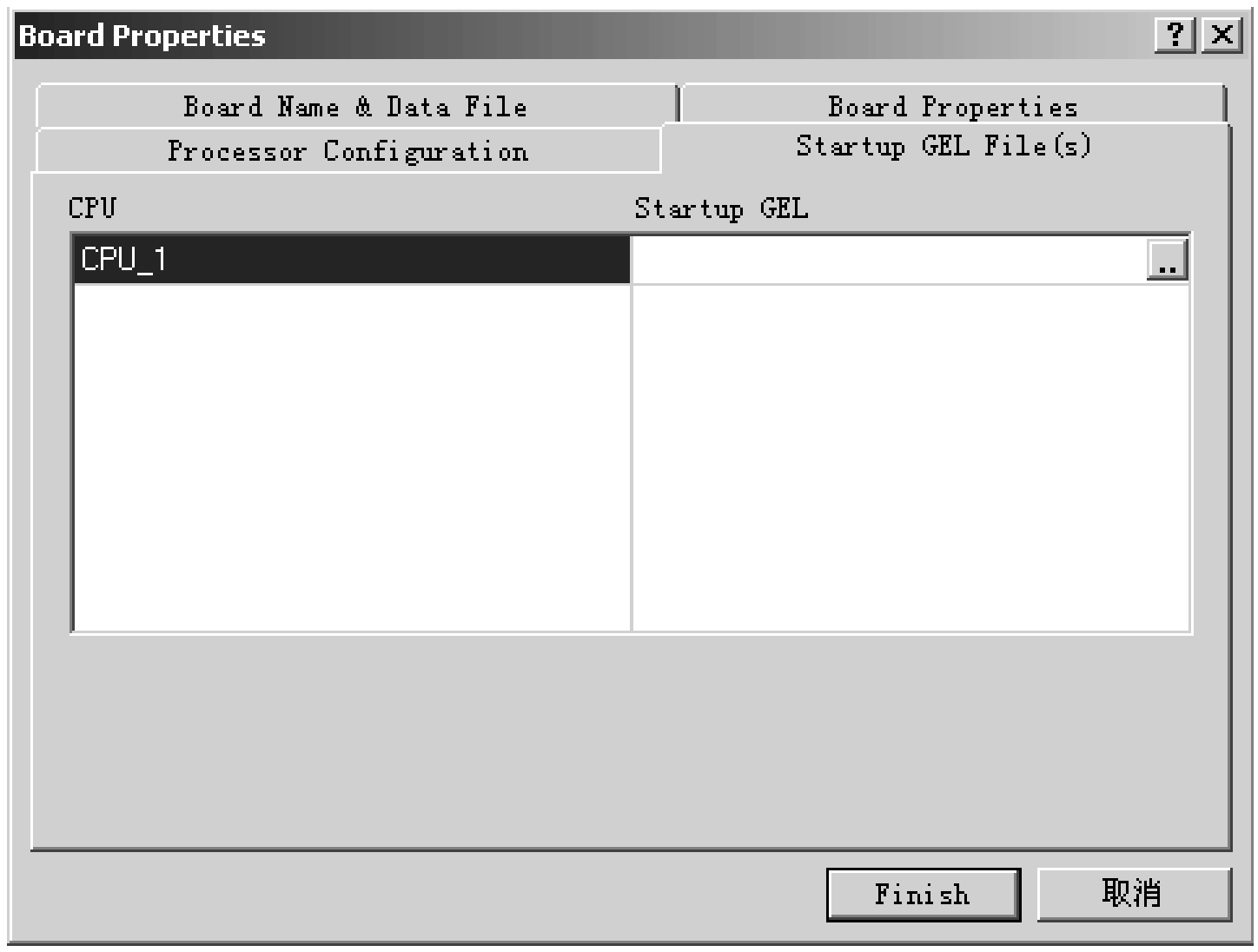


图 9 系统的初始化 GEL 文件配置

⑫ 〕选择配置主界面“ Code Composer Setup ”中的 File →Save 保存设置并退出。

2.2 CCS 的应用

2.2.1 用 CCS2023 开发 应用 程序的 一般步 骤为

⑪ 〕创立一个工程。 运行 CCS 程序，进入 CCS 集成调试环境， 点击下拉菜单 Project ，

选择 New …，就可以创立工程，如图 10 所示。

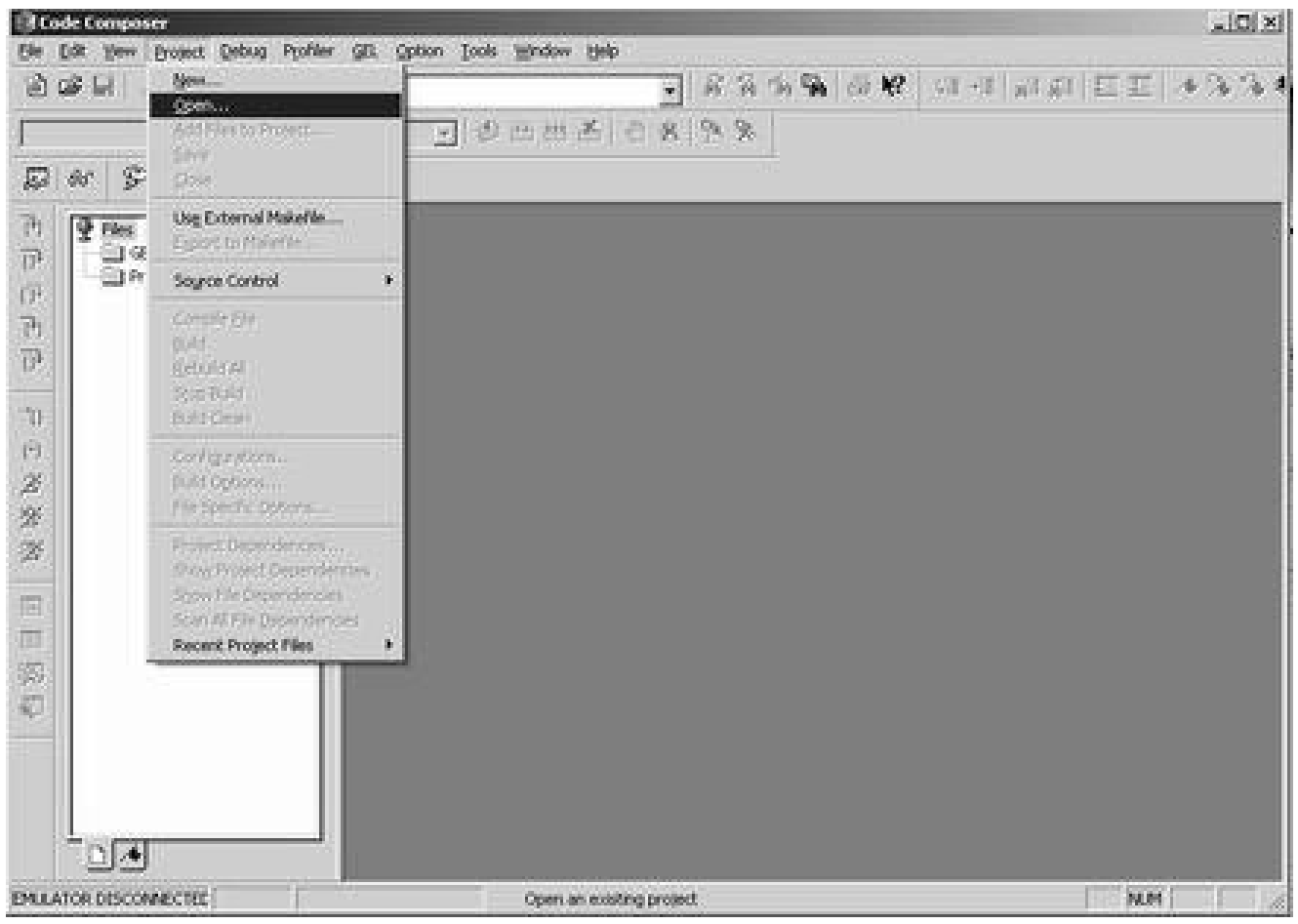


图 10 CCS 的开发界面

然后，选择好路径，输入所要建立的工程名〔以 lfdac.pjt 为例〕，点击“保存”，如图 11 所示。

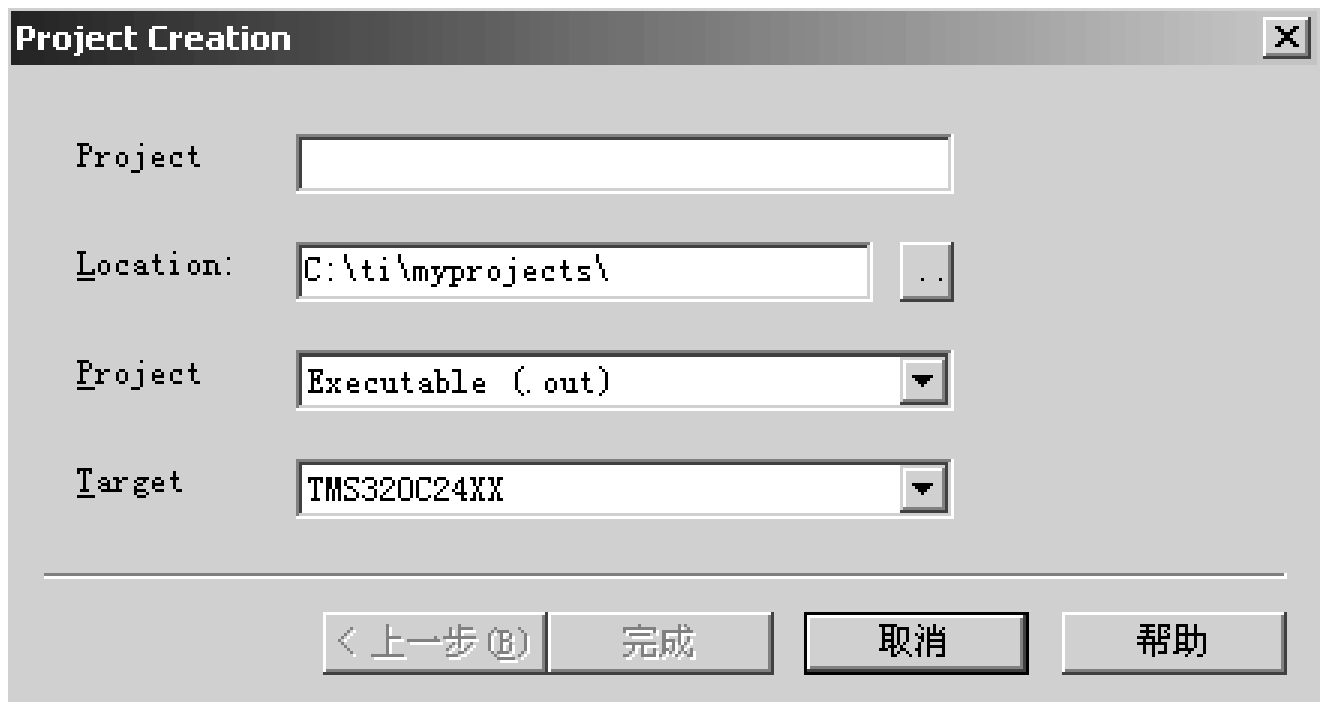


图 11 建立工程弹出窗口

这样工程 lfdac.pjt 就建立了，下一次就可以直接翻开这个工程了。现在这个工程中就只有空文件夹，是一个空的工程。接下来就是在此工程中参加用户所需要的程序。

⑵ 〕编写各类文件。包括源文件〔C 或汇编〕、命令文件〔 d 文件〕、头文件〔 .h 文件〕等。

⑶ 〕添加各类文件到工程中去，然后对工程进行编译。从下拉菜单 Project 选择“ Add Files to Project …”或者右击工作窗口中工程 lfdac.pjt，选择“ Add Files …”，如图 12 所示：

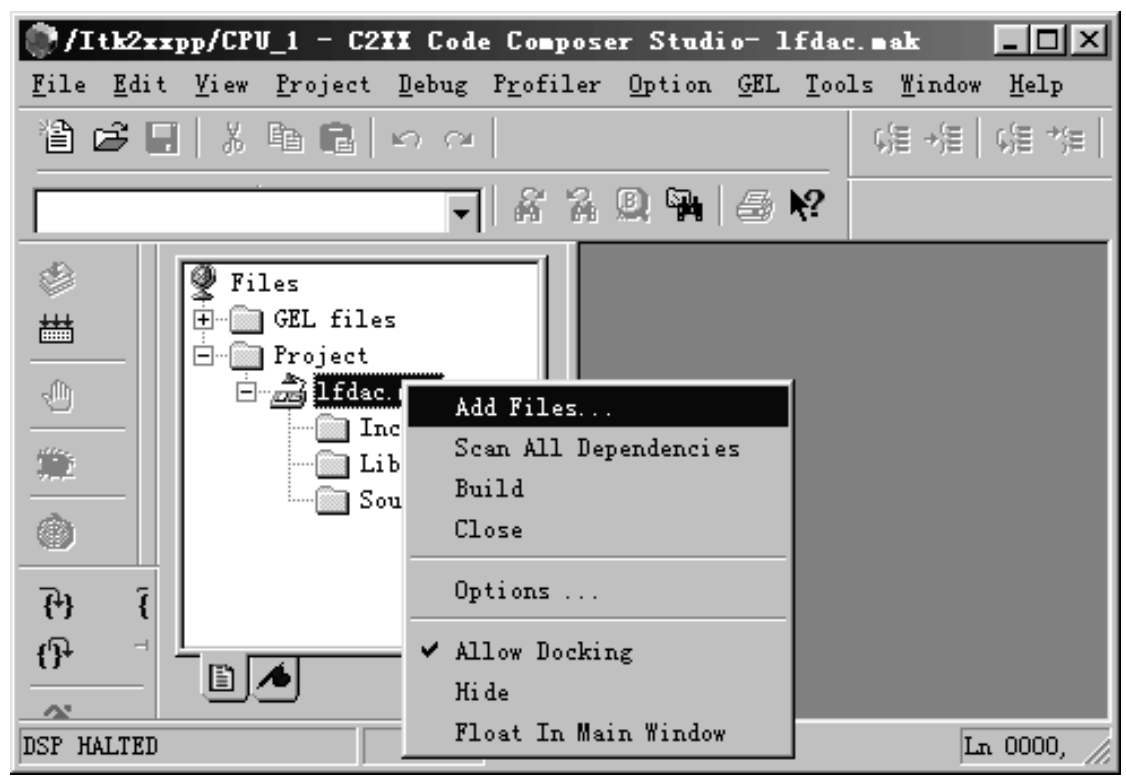


图 12 工程显示窗口

这时，要依据需要，添加 *.C，*.asm，* d 等需要的文件到工程中去，添加时 CCS 会自动把文件放到相应的名目中，添加后文件会在 CCS 下显示出来。

假设在汇编或 C 语言中包含了某些头文件， 则在工程中会将这些头文件自动包含 进去，在工程名目下的 inlcude 文件夹中可以看到。

留意：对于 C 语言程序，除了要添加*.C,* d 文件外，还必需添加实时运行库 rts2xx.lib ，这个工程才 能编译通过。

对于有中断效劳的程序，必需添加中断向量表进去，例如 verctors.asm ，中断程序才能正常执行。

等文件添加完了后，就可以进展汇编、链接了，只要点击工具条中的  图标或者下拉菜单 Project 下的“ Rebuild All ”，这样汇编和链 接就一步完成了。 在 CCS 最下面的 Message 窗口中会消灭如图 13 所示的窗口。

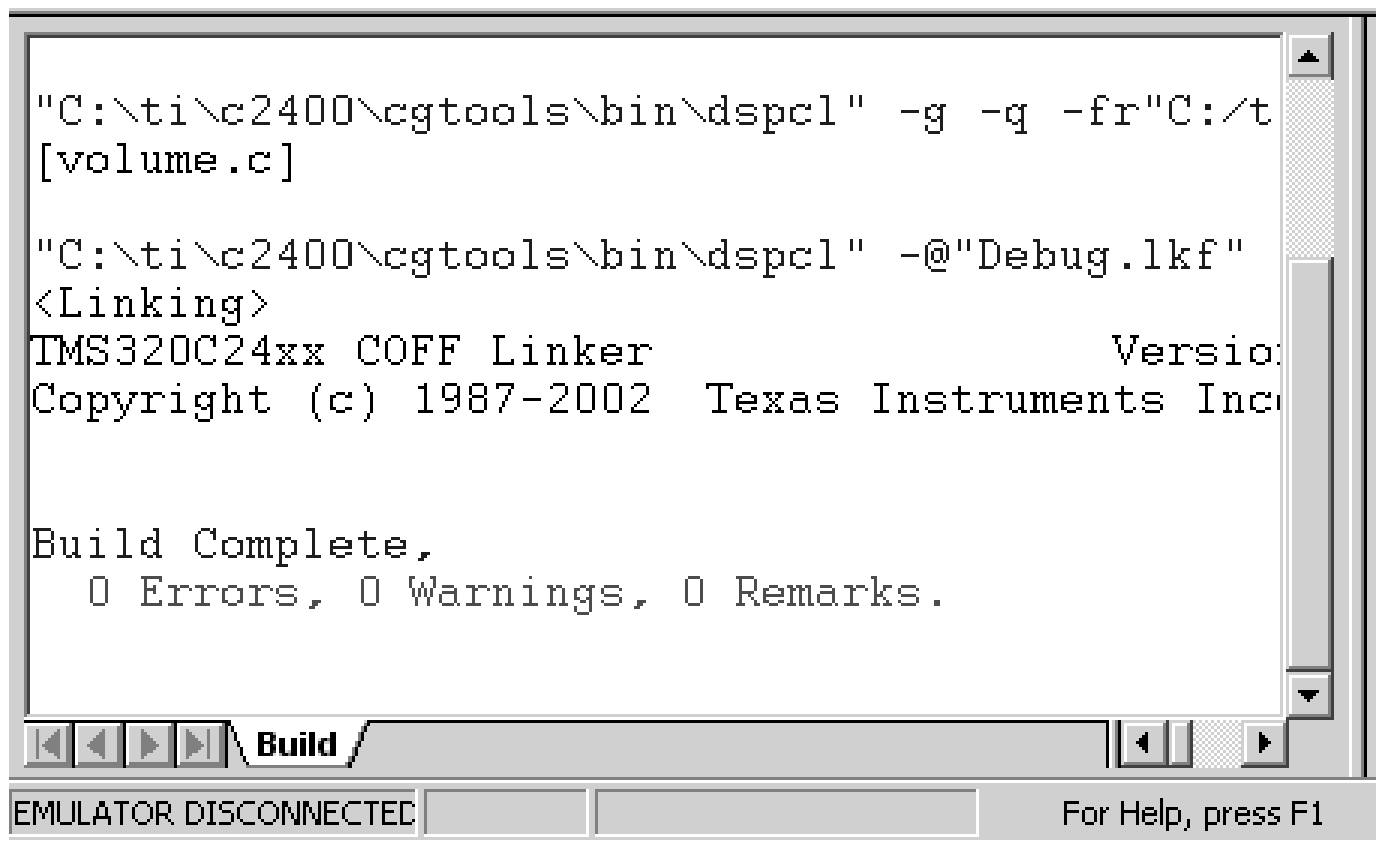


图 13 输出窗口

图中说明全部文件没有任何语法错误。假设有语法错误，将在 Message 窗口中显示出来。编程人员可以依据显示的信息，直接点击该信息，鼠标就定位到了错误位置，然后可

依据提示更改错误，每次修改完了后都要重编译。

①　〕加载程序。选择下拉窗口　File　下的　Load　Program，在弹出的　窗口中加载刚刚生成的 *.out　文件，　点击“　翻开”，可执行文件就自动加载到　　DSP　中去了。

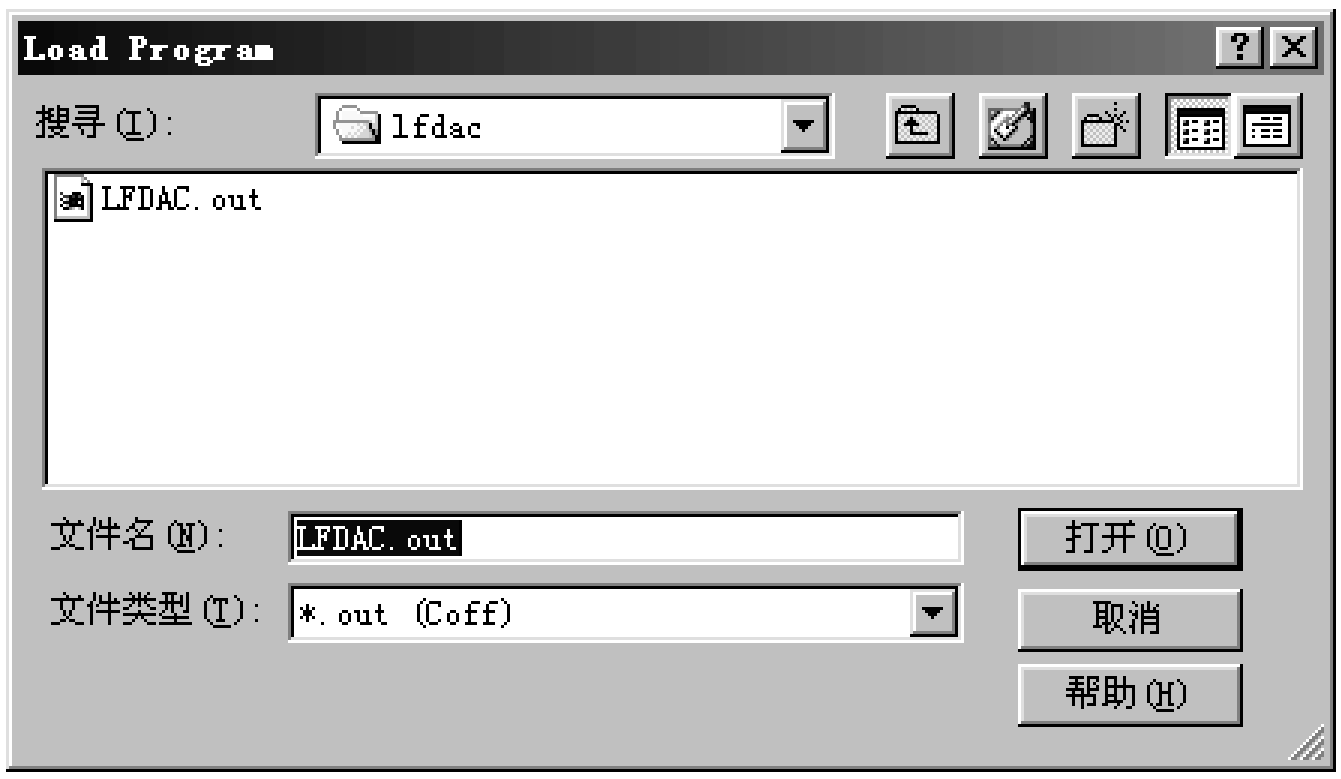


图　14　加载程序窗口

2.2.2　工程调试

1）　调试工具

可以使用断点、观看窗　口、探针等调试工具对　错误进展调试；还可以　对计算结果　/输出数据进展分析，评估算法性能。

①　〕断点调试

设置断点是调试程序的必要手段。双击程序代码段中的某一行，或单击工具栏断点开关按钮  ，还有可以单击　快捷键　F9。设置了断　点后，该行就变成了　　粉红色高亮。断点设置成功后，　点击菜单“ Debug/Run　”，或者点击调试工具条按钮  ，　程序就会运行到断点处停住，　PC 指针指到断点位置，黄色和粉红色同时在该行消灭。前半段位黄色，后半段为粉红色。双击呈现粉红色的断点行，就把断点消退了。

②　〕 Watch 窗口调试

选择下拉菜单　View　下的选项　Watch Window　，　就会弹出一个空白的　Watch　窗口，在这个空白　Watch　窗口中点　击右键，选择　Insert　New　Expression　就会弹出　如图　15　所示的对话框。

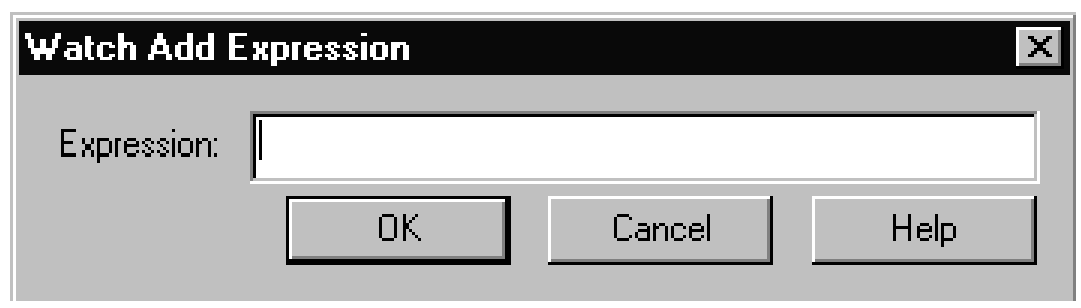


图　15　设置表达式观看窗口对话框

在上图窗口中的空白处填入　“　*〔 int* 〕变量名”〔例如变量为　DAC0VAL 〕，消灭如图　16　所示的窗口。

图 16 观看窗口 1

程序运行时 Watch 窗口将显示要查看的变量的值，如图 17 所示。

图 17 观看窗口 2

③ 〕探针 Probe Point

探针点调试可以让用户查看程序执行到某一位置时，各存储器窗口值、文件 I/O 等。当设置了探测点与 C I/O 、CPU 存放器等连接起来后，实际程序是运行到断点处，但各窗口的更值并不是断点处，而是探测点处的。由于 CCS 增加了文件 I/O 功能，利用探测点调试，可以在某一探测点位置看到外部文件数据流与 DSP 算法程序代码交换的状况。

按调试工具条按钮“ Toggle Probe Point ”将光 标当前位置设置为探测点，可以看到如图 18 所示的对话框。探测点设置成功后，当前位置会消灭蓝色的线条。假设光标仍停留在探测点位置，再次点击工具条按钮“ Toggle Probe Point ”，设置的探测点将被删除。

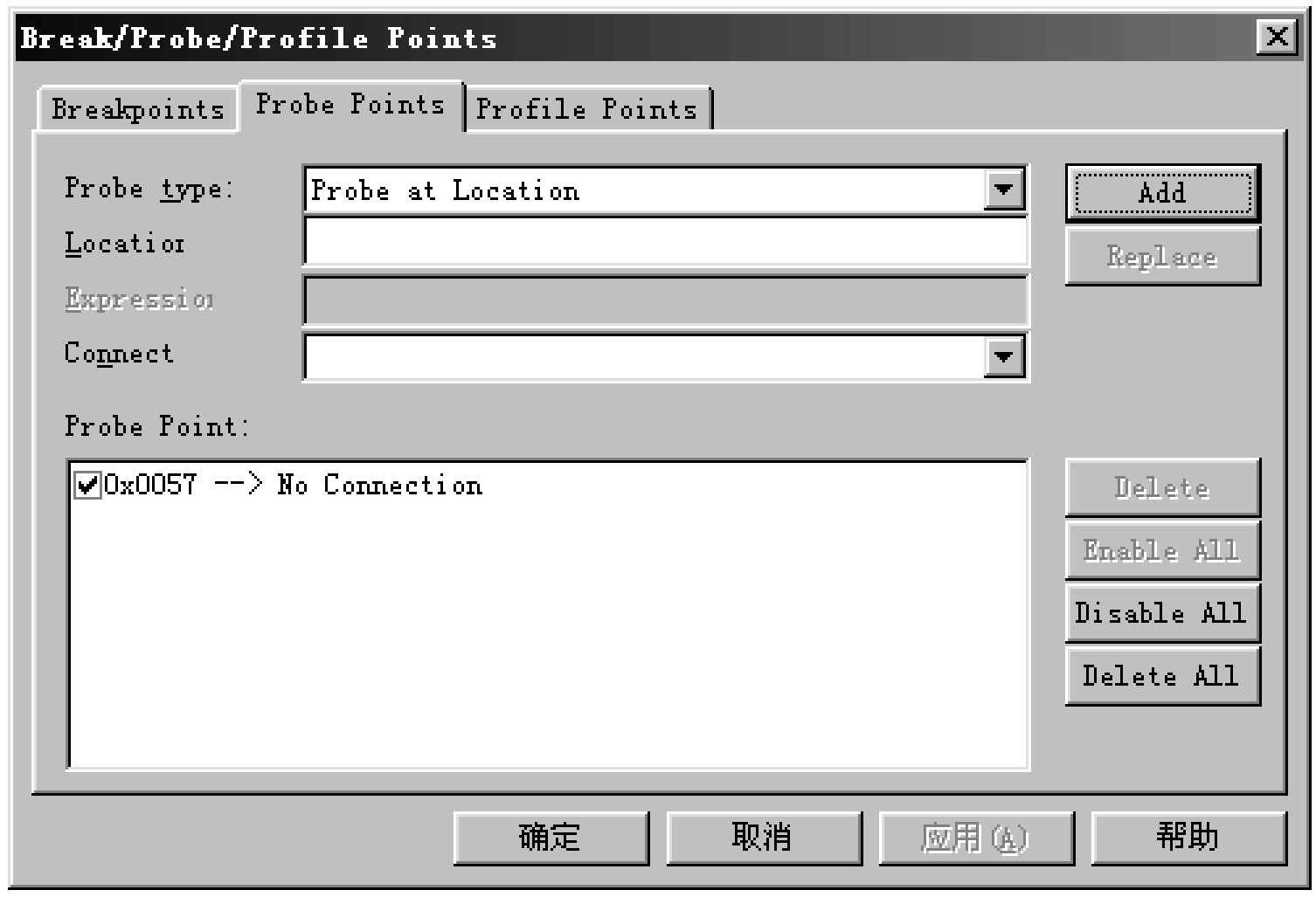


图 18 探针工具设置窗口

另外， CCS 还有剖析点 Profile Point 等工具。

2） 查看调试中的信息

在程序调试工程中，需要 CCS 显示出程序运行的结果，以和预期的结果进展比较，从而顺当地调试成功程序。可以查看的信息有：查看存放器，查看数据，查看反汇编程序，查看多个窗口等。

① 〕查看存放器

点击菜单“ View/Registers/CPU Register ，可以查 看到 CPU 存放器的值。 如图 19 所示。

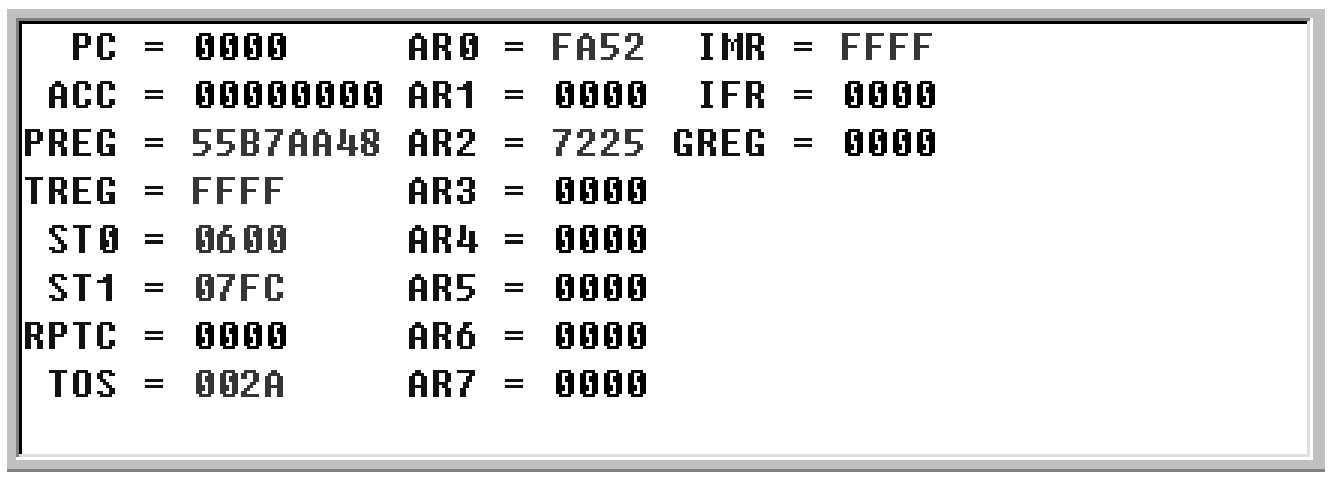


图 19 CPU 存放器观看窗口

窗口中存放器的内容以黑色或红色表示，未更的值为黑色，更的值为红色。在这个窗口中，用户也可以直接修改其中的存放器的值，以便利调试程序。

② 〕查看数据

查看数据也是程序调试中常用的手段之一， CCS 中可以以多种形式查看数据单元中的值，可以满足各种状况的需要。

查看数据单元，选择 View 菜单中的 Memory 弹出窗口如图 20 所示。

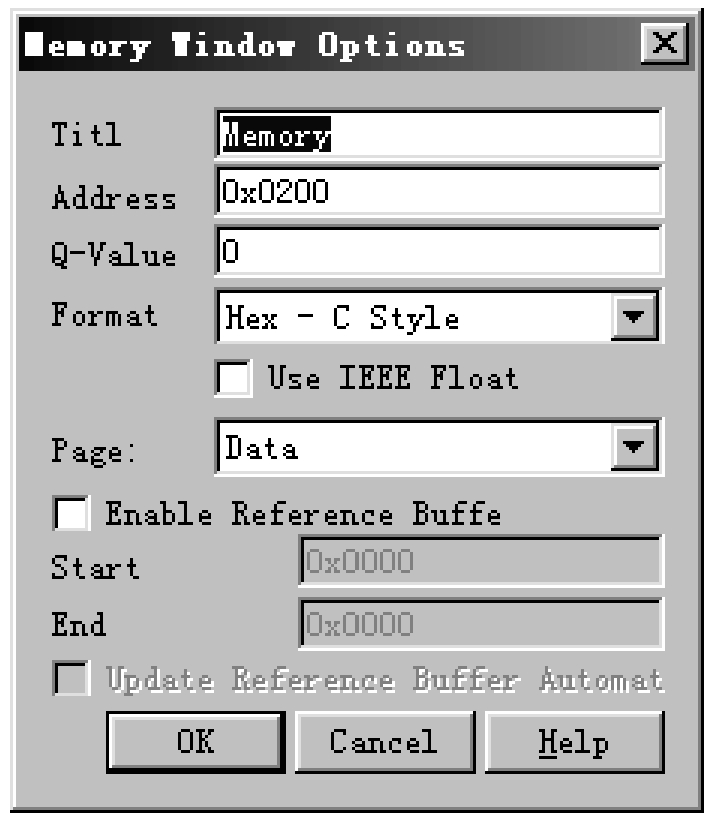


图 20 存储器观看窗口的设置选项

在 Address 一栏中写上 要查看的数据的起始地址， Format 一栏中可以 选择查看数据的格式，点击 OK 就可以在 Memory 窗口中看到 要查看的数据了。对于数据单元也可以直接修改其内容以便利调试，修改方法同修改存放器方法一样。

3 DSP 仿真器的连接与使用

3.1 并口仿真器

1 仿真器的特点

- 兼容 TI 全系列 DSP 产品： TMS320C2023 /C5000 /C6000 /C3X,4X /VC33 ；
- 对于全部系列 DSP 仿真器完全通用， 只需更换软件就可以实现全部 DSP 器件的开发，同时每种软件均支持 C 语言和汇编源代码调试；
- 支持 Code Composer Studio 集成调试环境；
- 仿真不占用任何 DSP 资源；
- 支持多 DSP 同时调试仿真。对于多 CPU 系统，只需购置一套开发系统，配上多 CPU 调试软件，就可以对它们进展并行调试。

2 仿真器系统工作 环境

仿真器使用并口与计算机连接，在使用仿真器之前需要正确配置并行端口和仿真器的相应设置。对于台式机 并行端口 有三种工作模式： SPP 、EPP 、ECP 。ICETEK-5100PP 可以在 SPP 和 EPP 模式下工作，所以要把计算机的并行口配置成 SPP 或 EPP 模式。假设主机支持 EPP 工作模式，推举使用 EPP 模式。

3 安装步骤

第一步 ： 预备安装

在进展安装前请确认已经具备了以下部件 :

〔1〕并口仿真头；

〔2〕开发系统供电电源〔注：假设目标板是 +5V 的文件 ,可以不用〕；

③ 〕并口电缆；

④ 〕目标板。

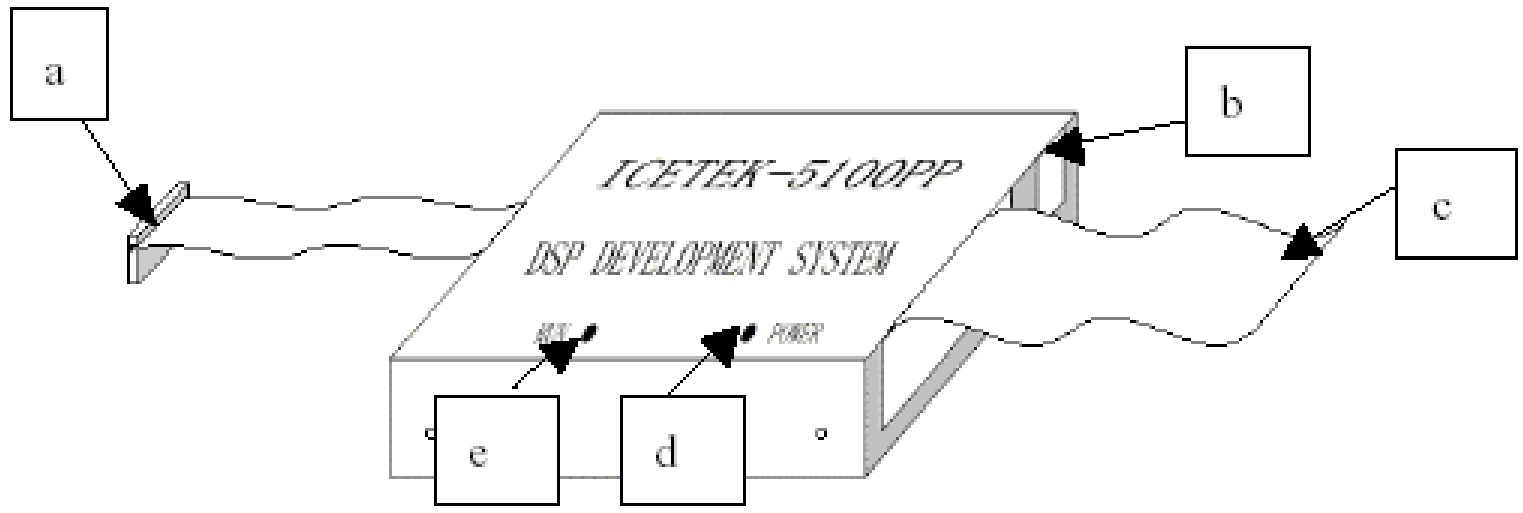


图 21 ICETEK-5100PP 开发系统外观图

如图 21 所示为仿真器的外形图，它的各个部件的说明如下：

a) 仿真头： JTAG 接头；

b) 开发系统电源接口；

c) 并口电缆；

d) 开发系统供电指示灯 ；

e〕正确运行指示灯。

其次步：在预备工作做完之后，下面可以进展安装。

①　〕关计算机

②　〕把主机的并口电缆与仿真器的接口相接

③　〕+5V 电源与仿真器的电源接口相接

④　〕仿真头与目标板相接

⑤　〕翻开目标板电源，这时仿真头上的　Power　灯会亮

⑥　〕开 PC 机

注：假设是开发 +5V 供电的 DSP 器件，仿真器可以不用外接电源。

4 软件安装使用 说明：

1）　并行端口配置：

启动计算 机，进 入 BIOS 界面， 进入“ INTEGERATED PERIPHERALS ”子选 项，将“ Parallel Port Mode ”项改为“ EPP ”并保存设置，同时记录“ Onboard Parallel ”的值。设置后的典型值如下：

Parallel Port Mod	： EPP
Onboard Parallel Port	： 378/IRQ7

2）　ICETEK-5100PP　的配置：

在确定了计算机的并行端口工作模式之后　　，还要把 ICETEK-5100PP　的参数设置与计算机对应起来，此时　，配置方法如下：正确安装　ICETEK-5100PP　的软件，编辑安装后软件路径下的 XDS510PP.INI　文件。 XDS510PP.INI　文件有三个选项，缺省值如下：

Port	＝378
Mode	＝EPP
Speed	＝0

用记录的“Onboard Parallel Port“ 的值替换“port “参数的缺省值。例如:假设记录值为：278/IRQ7, 则把 XDS510PP.INI 的“port“ 参数改 为 278 。用设置的 “ Parallel Port Mode“ 的值替换“mode“ 参数的缺省值， 例如： 假设设置值为： SPP ， 则把 XDS510PP.INI 的“mode“ 参数值改为“SPP“ ，最终，保存修改后的文件。

在设置并行端口和调试器之后，　假设连接不稳定， 则需要修改　XDS510PP.INI　中“speed“参数的值。修改方法如下：

将“speed“ 参数的值改为　20，保存并运行调试软件。假设连接稳定，可以渐渐减小“speed“的参数值，直到保持连接稳定的最小值为止。假设把参数值改为　20　后仍旧连接不稳定，请检查计算机的并行端口。

4 基于 TMS320LF2407A 试验箱

本试验装置主要可以进展以下几方面试验：

- □ 片内外设试验
- ■ 系统总线试验、 A/D 输入试验、 PWM 信号的产生、脉冲捕获试验、旋转码盘脉冲解码试验、定时器相关试验和 SCI 、SPI 通信试验。
- □ 片外设备掌握试验
- ■ 键盘试验、拨码开关试验、数码管显示试验、 D/A 输出试验、 PWM 模拟 D/A 输出试验、 I2C 总线试验、四线串行接口〔 microwire 〕试验。
- □ 数字信号处理试验
- ■ FFT 试验、 FIR 和 IIR 试验、相关分析和自适应滤波器试验。
- □ 系统综合试验
- ■ BLDC 掌握系统：包含 根本的闭环掌握、过流 过压保护、人机界面和 掌握信号输入输出等功能；
- ■ 可扩展相应外设进展三相沟通电机变频掌握系统，包含根本的 V/F 掌握、空间矢量掌握、过流过压保护、人机界面和掌握信号输入输出等功能。

4.1 系统组成

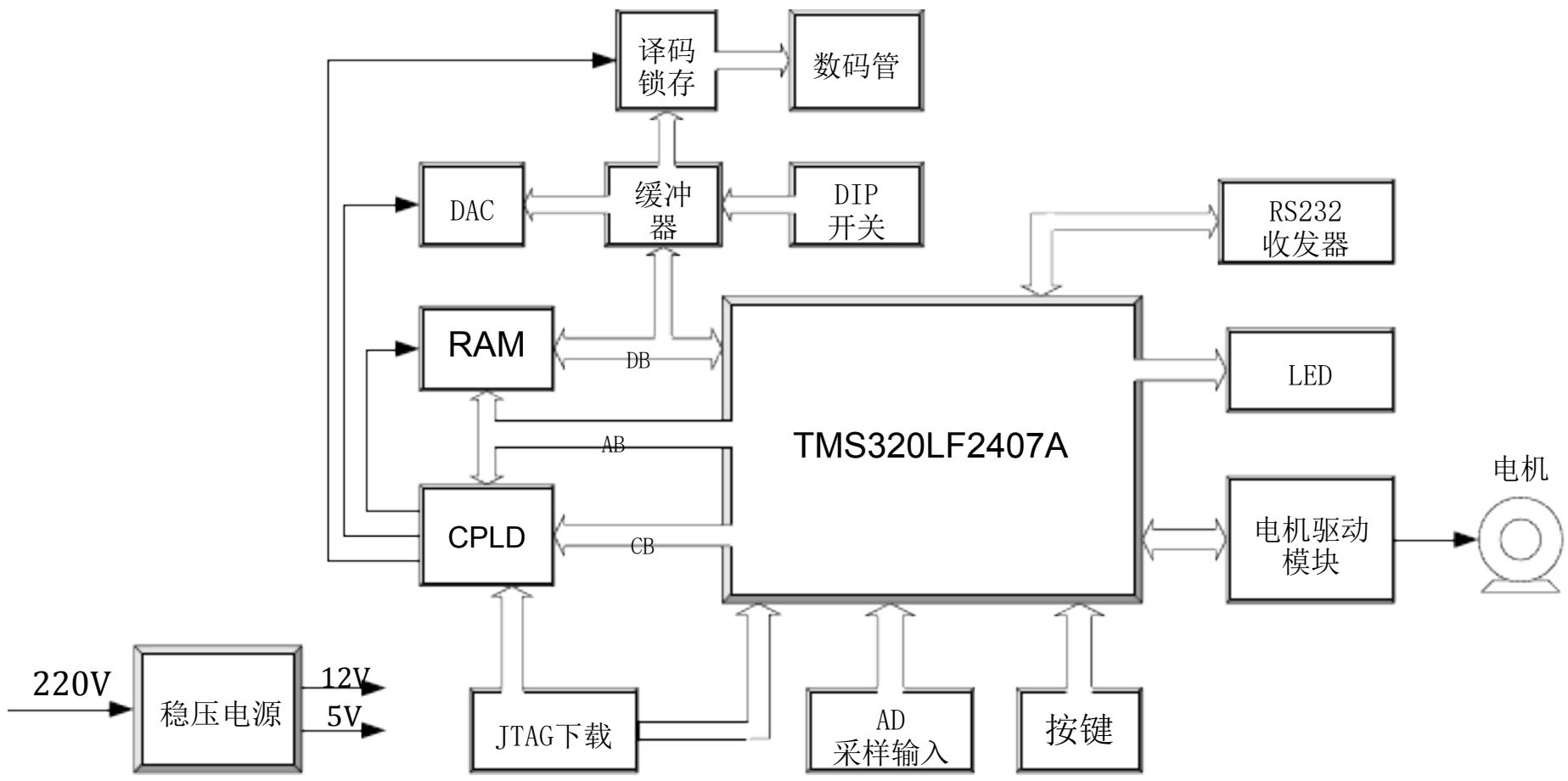


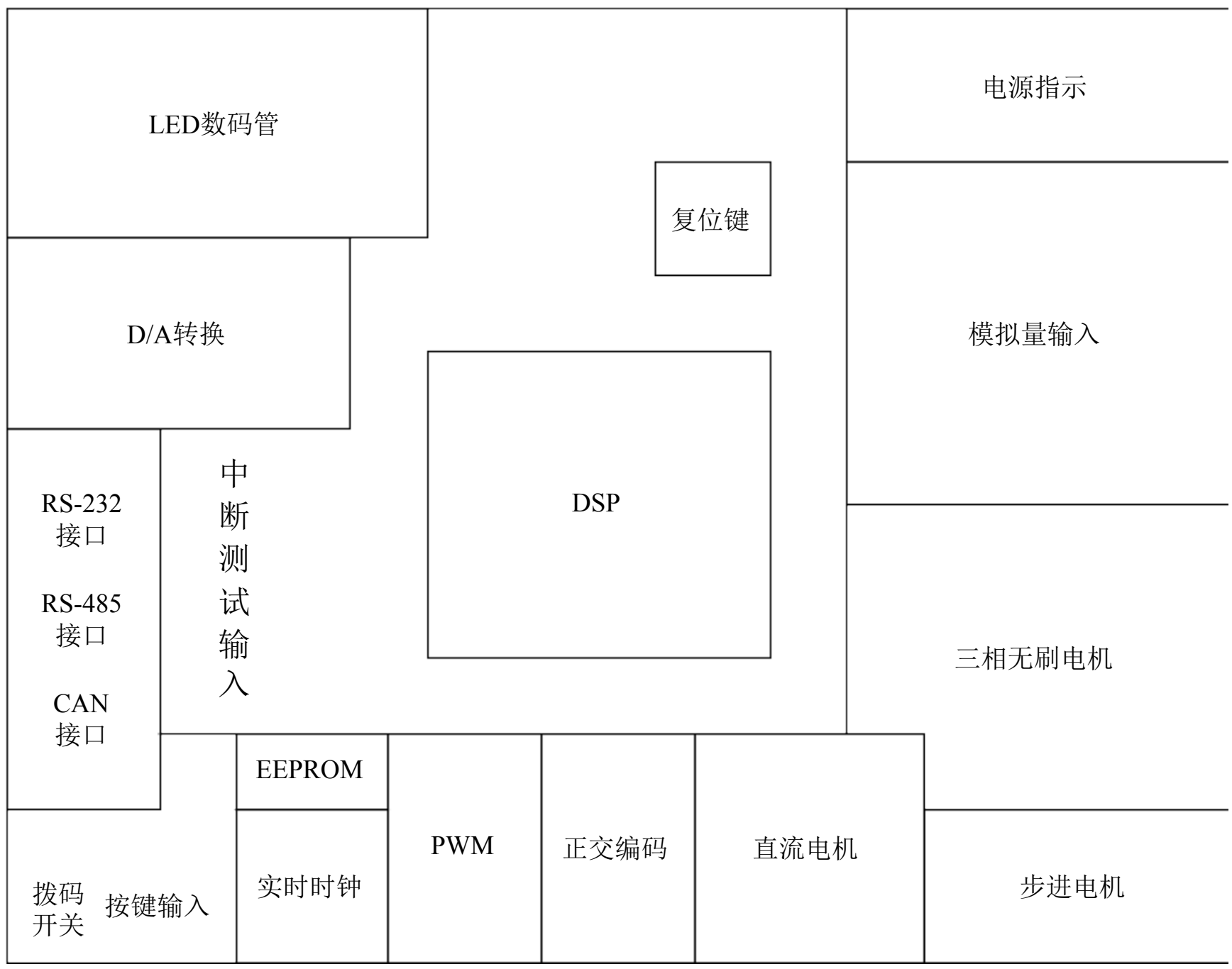
图 22 硬件系统框图

硬件系统框图如图 22 所示，系统通过 CPLD 来产生各外设的片选信号，大大简化片选规律电路的设计。板上外扩 64K 的 SRAM ，在微处理器模式下作为程序和数据存储器，而在微掌握器模式下，其中 32K 作为数据存储 器使用，而其片内的 32K FLASH 作程序存储器用。除此之外，试验装置硬件主要包括输入输出模块和电机驱动模块等，今后的试验中将分别进展具体介绍。

4.2 性能指标

- TMS320LF2407A 运行速度为 40MIPS ，128K 存储器空间
- 16 路 10Bit 集成片上 A/D 接口
- 双大事治理器 EVA 和 EVB
- 2 路的 TLC7528 D/A 转换
- 符合 RS232 标准的 UART 串行接口
- 32K 片上 FLASH
- CAN 总线标准接口
- 用户开关、测试指示灯
- 数据地址 I/O 掌握 4 处扩展连接器
- 具有与 IEEE1149.1 相兼容的规律扫描电路，用于测试和仿真

4.3 试验箱布局



5　试验例程

5.1　试验一　Code Composer　使用教程

这个教程可以帮助你了解　Code　Composer　Studio　的根本功能。它无意　供给一个关于 Code Composer Studio　的具体描述，但可以作为开发者使用　　Code Composer Studio　的起点。

教程使用一个例子程序从输入缓冲区中读取数据，然后乘以某个系数后放到输出缓冲区中。教程内容包括：

①　〕创立工程环境

②　〕根本调试功能

③　〕使用观看窗口

④　〕图形功能简介

1.　创立工程环 境

这局部为使用者供给了生疏工程环境的时机，　　CCS　供给工程文件来治理应用程序。全部有关的应用程序的信息保存在工程文件中。　　工程文件记录生成一个目标　　DSP 程序和库程序需要的全部文件和运行库。它也包含了为编译、汇编、链接目标代码而配置的开关参数。

在这局部，你将学习到如何创立工程文件并添加源程序文件和库文件到工程中，以及编辑源程序文件和编译连接生成目标代码。

⑴　创立工程：　执行“　Project　→New　”命令，显示文件选择对话框，　转变名目到　tutorial 名目下〔缺省位置在　c:\ti2xx\c2023　〕，输入“　volume　”作为工程文件名　并保存；

⑵　添加源程序文件：执行“　　Project　→Add　File　to　Projects　”命令，打　开添加文件对话框.反复使用这个命令，　添加以下文件到工程中：

volume.c　〔在　C:\TI\MyProjects\volume　　名目中〕

rts2xx.lib　〔在　C:\TI\C2400\cgtools\lib　　名目中〕

⑶　转变工程设置：执　行“　Project　→Build Options　”命令，显示对话框　。这个对话框用于配置编译，汇编和链接的开关；

①　〕点击　linker 属性页

②　〕在　Autoinit Model栏选择　Run-time Autoinitialization

③　〕选择“确定”对话框保存修改并关闭这个对话框　　4

〕编译工程：执行“　Project →Build　”命令完成　对工程的编译；

2.　根本调试功 能

在这一节，我们将生疏　　CCS　的一些根本调试功能。

⑴　执行　File →Load Program　，在随后翻开的对话　框中选择刚刚建立的　　volume.out　文件；

⑵　在工程扫瞄窗口中，双击　　volume.c　激活这　个文件，移动光标到　　main　〔〕行上，右击鼠标选择　Toggle Breakpoint　或按　F9 设置断点　；

⑶　选择　Debug →Run　或按　F5 运行程序，程序　会自动停在　main　〔〕函　数上。

①　〕按　F8，执行到　write_buffer　　〔〕函数上

⑵ 〕持续按 F8，直到 程序转到 write_buffer 函数中运行

留意：在执行 C 语言的程序时，为了快速的运行到主函数调试自己的代码，可以使用 Debug →Go main 命令。

3. 使用观看窗 口

这一节介绍如何在 CCS 中观看和修改程序变量。

1）执行 View →Watch window 翻开观看窗口；

2）在 volume.c 中，选 中任意一个变量，右击 鼠标，选择“ Quickwatch ”，CCS 将翻开 quickwatch 窗口并显示 选中的变量；

3）或者在 volume.c 中，选中任意一个变量，右击鼠标，选择“ Add to watch window ”，CC 将把变量添加到观看窗口并显示选中的变量值；

4）在观看窗口中选中变量， 单击对应栏中“ value ” 项， 可以在该栏中转变程序变量的值；

5）依据上述方法将 str 变量加到观看窗口中，点击变量左边的“ +”，观看窗口可以开放构造变量，并且显示构造变量的每个元素的值；

6）把 num 变量加到观 察窗口中，执行程序进入 write_buffer 函数， 此时 num 变量超出了作用范围，可以利用 stack call 窗口观察在 不同作用范围的变量。

⑴ 〕执行 View →Call Stack 翻开堆栈窗口

⑵ 〕双击堆栈窗口的 main 〔〕选项，此时可 以观察 num 变量的值

4. 文件输入 /输出

这一节介绍如何从 PC 机上加载数据到目标机上。这是使用的数据流测试算法的正确性的好方法。

在完成下面的操作以前，先介绍 Code Composer Studio 的 Probe 断点。这种断点允许用户在指定位置提取/ 注入数据。Probe 断点可以设置在程序的任何位置。当程序运行到 Probe 断点时，与 probe 断点相关的大事将会被触发。当大事完毕后，程序会连续执行，在这一节里， Probe 断点触发的大事是： 从 PC 机的数据文件加载数据到目标系统的缓冲器中。

⑴ 在真实的系统中， read_signals 函数用于读取 A/D 模块的数据并放到 DSP 缓冲区中。在这里，代替 A/D 模块完成这个工作的是 Probe 断点。当执行到函数 read_signals 时，Probe断点完成这个工作；

⑴ 〕先在程序行 read_signals 〔input 〕上设置 断点；

⑵ 〕然后右击鼠标，选择 Toggle Probe Point, 设置 Probe 断点。

⑵ 执行 File →File I/O ，翻开对话框；

⑶ 点击 Add File 把 sine2.dat 文件加到对话框 中；

⑷ 完成设置；

⑴ 〕在 Address 中,输入 inp_buffer,

⑵ 〕在 Length 中,输入 0x64

⑶ 〕保证 warp around 被选中

⑸ 关联大事和 Probe 断点；

① 〕点击 Add Probe Point 按钮，翻开对话框

② 〕点击 Probe Point 中的内容，使之被选中

③ 〕在 Connect 中选择 sine2.dat 文件

④ 〕点击 Replace 按钮确认设置

⑤ 〕点击“确定”关闭对话框

⑥ 点击“确定”关闭对 话框，此时，已经配置 好了 Probe 断点和与之关联的大事。进一步的结果在下一节显示。

5. 图形功能简 介

这一节使用 CCS 的图形功能检验上一节的结果。

1）执行 View →Graph →Time/Frequency 翻开 Graph Property Dialog 窗口；

2）修改属性为如下值并确定；

Graph Title	：	Input
Satrt Address	：	inp_buffer
Acqusition Buffer Size	：	100
Display Data Size	：	100
DSP Data Type	：	16-bit signed integer

3）按 F12 运行程序，观看 input 窗口的内容 。

5.2 试验二　编制链接器掌握文件

链接的目的是生成可执行的目标代码，在这一节，我们将学习如何正确　　的链接目标代码，并且学会编制链接工程文件和使用一些有用的链接开关，最终，为后继试验编写一个链接批处理命令。

1. 建立 map 文件

为了把最终的程序编程到程序的　　ROM 中，我们需要知道程序的大小和位置，下面通过建立目标程序的 map 文件了解 DSP 代码的精准信息。

1〕翻开上一个试验建立　的 volume 工程，执行 Project →Build Options　命令，翻开 Build Options 对话框；

2〕在 Build Options 对话框中，点击 linker 属性页，在 Map Filename 中输入“Volume.map ”；

3）按“确定”保存对设置的修改；

4）执行“ Project →Rebuild All ”重编译和链 接整个工程文件；

5）执行 File →Open 命令，翻开 volume.map 文件，观看 Memory Configuration 和 SECTION ALLOCNTION MAP 了解链接器如何缺省定位程序代码和安排数据变量。在 map 文件中，我们可　以了解很多有关于输出代码的信息，比方　　代码的长度，数据空间的长度等等。在 map 文件中 ，有一个术语　，它代表了 DSP 的存储空间。其中，　　0 表示程序存储空间，　1 表示数据存储空间，而　　2 表示 I/O 空间。它们分别与芯片管脚 PS#， DS#和 IS#片选的存储空间对应。

2. 建立链接文 件

在实际的应用中，我们总是需要定位自己的存储器，同时，也需要掌握文件的输出到需要的地方去。下面的内容是创立链接文件，并用这个文件掌握程序的地址映射。

1）执行 File →New →Source File 创立一个的 文件；

2）执行 File →Save 保存创立的文件，并命名为　　volume d ；

3）依据以下的内容输入

```
MEMORY
{
    0:
   VECT  :   origin = 0    ,length  =40h
   PROG   : origin= 2023h ,length =7000h
   1:
   SARAM :origin = 800h  ,length = 1000h
}
SECTIONS
{
   .text         : { } > PROG   0
   .cinit        : { } > PROG   0
```

```
    .switch       : { } > PROG   0
    .data         : { } > PROG   0
    .const        : { } > SARAM        1
    .bss          : { } > SARAM        1
    .stack        : { } > SARAM        1
    .sysmem       : { } > SARAM        1
}
```

4）执行 File →Save 保存程序；

5）执行 Project →Add File to Project 把 volume d 加到工程中；

6）执行 Project →Rebuild All 重编译链接工程 ；

7）执行 File →Open 翻开 volume.map ，比照与 刚刚有哪些不同 ；

8）执行 Project →Build Option 翻开对话框；

9）单击 linker 属性页， 在 stack size 中输入 0x100 ；

10）〕按“确定”保存对配置的修改；

11）执行“ Project →Rebuild All ”指令重编译 整个工程；

12）〕执行 File →Open 翻开 volume.map ，比照与 刚刚有哪些不同；

13）〕修改 volume d 文件，如下： FLASH : origin = 0x8000 , length =0x100 ；

14）〕重编译链接程序，观看 map 文件有哪些 变化。

5.3 试验三 汇编语言程序设计

由内置的硬件模块支持， 数字信号处理器可以 高速的完成加法和乘法运算。 但 TMS320 系列不供给除法指令，为实现除法运算，需要编写除法子程序来实现。二进制除法是乘法的逆运算。乘法包括一系列的移位和加法，而除法可分解为一系列的减法和移位。本节要求编写一个 16 位的定点除法子程序。

1. 除法运算的 过程

设累加器为 8 位，且除法运算为 10 除以 3，除的过程包括与除数有关的除数逐步移位，然后进展减法运算，假设所得商为正，则在商中置 1，否则该位商为 0.例如 4 位除法例如：

1） 除数的最低有效位对齐被除数的最高有效位

```
  00001010
- 00011000
  11110010
```

2） 由于减法结果为负，丢弃减法结果，将被除数左移一位再减

```
  00010100
- 00011000
  11111000
```

3） 结果仍为负，丢弃减法结果，将被除数左移一位再减

```
  00101000
- 00011000
  00010000
```

4） 结果为正，将减法结果左移一位后把商置 1，做最终一次减

```
  00100001
- 00011000
  00001001
```

5） 结果为正，将减法结果左移一位加 1 得最终结果， 高 4 位是余数，低 4 位商：00010011

2. 除法运算的 实现

为了尽量提高除法运算的效率， 2xx 系列供给 了条件减指令 SUBC 来完成除法操作。SUBC 指令的功能如下：

假设〔 ACC 〕≥ 0 且〔数据存储器地址〕≥ 0

PC+1 然后

〔 ACC 〕－ [〔数据存储器地址〕× 2^{15}] ⇨ALU 输出

假设 ALU 输出≥ 0

则：〔ALU 输出〕× 2+1 ⇨ ACC

否则：〔 ACC ×2〕 ⇨ ACC

实际上， SUBC 指令完成的是除法中的减除数求商的过程，即余数末位补 0，减去除数，假设结果为正，该位商为 1，否则商为 0。

SUBC 指令实现条件减，可以用如下除法：把 16 位的正被除数放在累加器的低 16 位，

累加器的高 16 位清 0，16 位的正除数放在数据存储单元中。执行 SUBC 指令 16 次，最终一次 SUBC 指令完成后，累加器的低 16 位是除法的商，高 16 位是余数。假设累加器和 /或数据存储单元的内容为负，则不能用 SUBC 指令实现除法。为了完成屡次的 SUBC 指令，还需要用到循环指令 RPT，它可以使 RPT 后的一条指令重复 1—256 次。SUBC 指令仅能对正数除法进展运算，因此，要扩展到全部数值的除法，还需要做如下工作：

在程序开头对被除数和除数做乘法， 并保存到 临时变量， 除数和被除 数分别取确定值，在除法运算完成后，依据临时变量的值修改商的符号。

3. 除法运算程 序流程

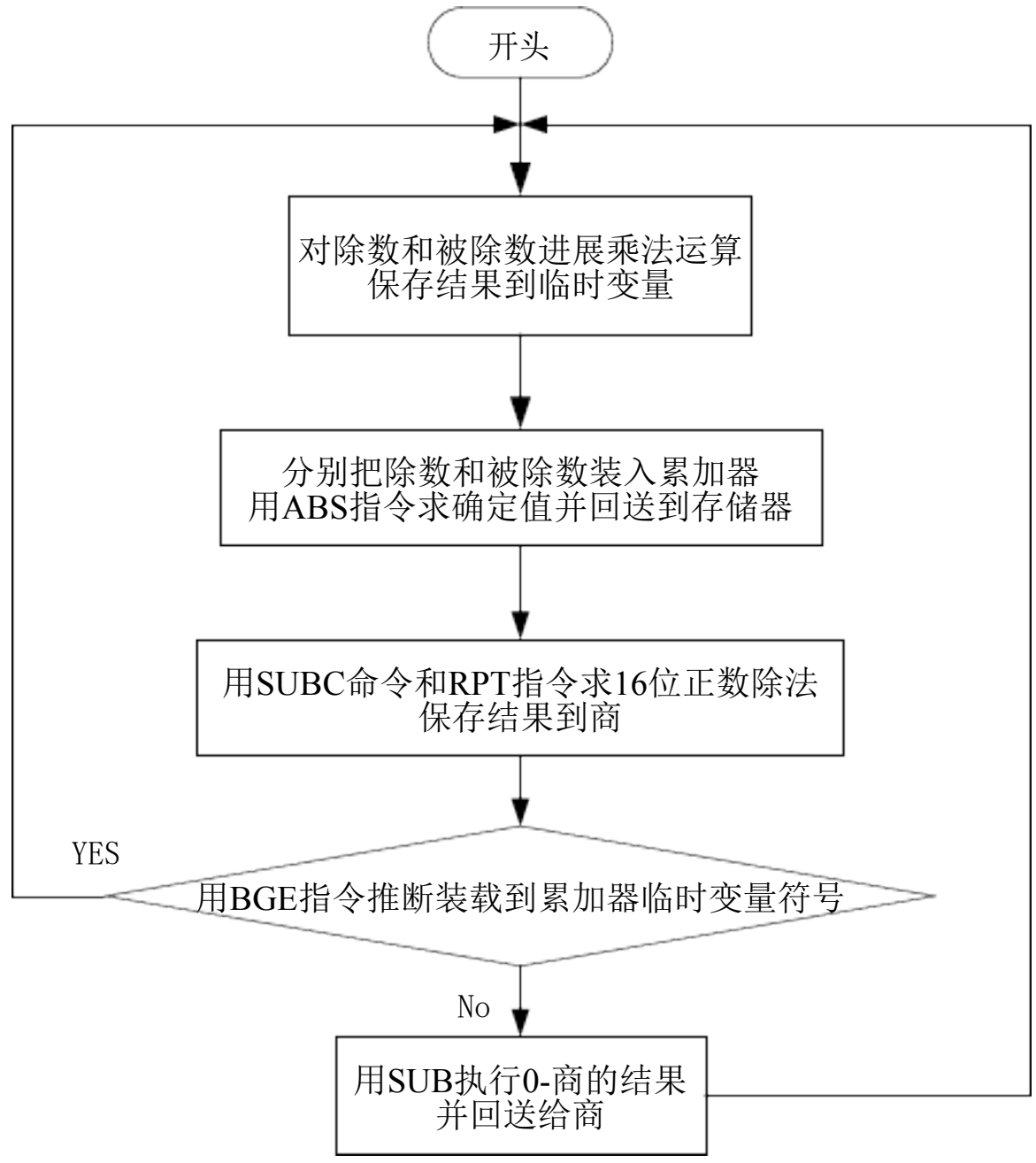


图 23 除法运算软件流程图

4. 试验步骤

下面是完成除法运算的试验步骤： 1

〕重启动 Code Composer Studio ；

2〕执行 Project →New 建立的工程；3〕

输入 lab2 作为工程的名称；

4） 执行 File →New →Source File 建立的程序 文件；

5） 为创立的程序文件命名为 lab2.asm 并保 存；

6） 执行 Project →Add New File to Project ，把 lab2.asm 参加工程中；

7〕执行 File →New →Source File 建立的文件 并保存为 lab2 d；8〕执行 Project →Add New File to Project ，把 lab2 d 参加工程中；

9）编辑 lab2.asm 参加如下内容：

```
   .bss    NUMERA,1
   .bss    DENOM ,1
   .bss    QUOT  ,1
   .bss    ARIT  ,1
   .bss    TEMSGN,1
   .text
   nop
start:ldp     #NUMERA
       lacc    NUMERA
       call    DIV
       b       start
DIV:  lt      NUMERA
       mpy     DENOM
       pac
       sach    TEMSGN
       lac     DENOM
       abs
       sacl    DENOM
       lacc    NUMERA
       abs
       rpt     #15
       subc    DENOM
       sacl    QUOT
       sach    ARIT
       lac     TEMSGN
       bgez    done
       zac
       sub     QUOT
       sacl    QUOT
       zac
       sacl    ARIT
done: lac     QUOT
       ret
```

10）〕编辑 lab2 d 参加如下内容

```
MEMORY
{
    0:
```

```
    VECT: o = 0        , l = 40h
    PROG: o = 02023h,    l = 7000h

     1:
    SARAM: o = 0800h, l = 01000h
}
SECTIONS
{
    .text     : { } > PROG        0
    .bss      : { } > SARAM       1
}
```

11） 执行 Project →Rebuild All 编译工程；

12）〕编译错误如下：

warning: entry point symbol _c_int0 undefined

缺省时 Code Composer 设置工程程序为 C 语言编译。因此，当我们编译汇编程序时，要对工程做配置；

13）〕执行 Project →Build Options翻开编译选项 ；

14）〕在 linker 上单击， 把 Autoinit Model栏改为 No Autoinitialization ，在 Compiler 属性页上单击 Assembly 名目选中“ Keep Labels as Symbols ”,然后执行 Project →Rebuild All ；

15）〕按“确定”保存对配置的修改；

16）〕执行“ File →Load Program ”装载程序 ，装 载完程序后， Code Composer 把指针指向 0000 处。为了执行我们的程序代码，需要修改 DSP 的 PC 值；

17）〕执行 View → Registers →CPU Register 翻开存放器窗口；

18）〕双击窗口中的 PC 标号， CC 弹出修改对话框供修改存放器；

19）〕在对话框中输入“ start ”，程序将处于我们 的程序入口点上

； 20〕执行 View →Watch Window 翻开观看窗口 ；

21〕在观看窗口中添加如下变量：

①〕 在观看窗口右击鼠标，在 Watch 窗口“ name ”栏下，输入 “ *(int*)NUMERA ”，观看该变量的值

②〕重复 a 的操作，参加 *(int*)DENOM 、*(int*)QUOT 和 *(int*)ARIT 四个变量〔留意变量名要与程序中的全都〕这样，我们就添加了为完成除法操作而需要的输入和输出。

其中 NUMERA 是被除数， DENOM 是除数， QUOT 是商，而 ARIT 是余数

22〕〕双击观看窗口上的 NUMERA 变量，输入数据“ 10 ”；

23〕〕双击观看窗口上的 DENOM 变量，输入数据“ 3 ”；

24〕〕按 F10 执行程序到“ b start ”，观看程序的 QUOT 和 ARIT 两个变量是否是正确的结果。此外，还可以多运行几次程序，分别用不同的

数据测试除法程序；

25〕 〕为汇编程序添参加口地址。在刚刚的 C 语言程序中 ,当我们装载完 程序后， PC 指针指向程序的入口地址，现在，我们要为自己的汇编语言添参加口地址。双击工程扫瞄窗

口中的 Lab2.asm 激活源 程序窗口；

26 〕在程序的开头参加下述代码： **.global start**

27〕执行 Project →Build Option 翻开对话框；

28〕在对话框中单击 Linker 属性页，在 Code Entry Point 中输入“ start ”；

29〕按“确定”保存对设置的修改；

30) 〕执行“ Project →Rebuild All ”选项，重编 译链接整个工程；

31) 〕执行“ File →Reload Program ”，Code Composer 将会自动把上次选 中的文件装载到目标系统中。观看这次装载与上次是否不同；

32) 〕执行“ View →Registers →CPU Register ”翻 开 存放器观看窗口，观察 PC 指针的值是否有些不同；

33) 〕构造完备的应用程序， C2023 系列的芯片在上电复位后将 PC 机的值置为 0x0000 ，这里实际上是复位向量的地址。一般的，要在这里添加一个无条件跳转指令，跳到程序真正的入口地址去。双击工程扫瞄窗口中的 Lab2.asm 激活 源程序窗口；

34) 〕在程序中 .bss NUMER, 1指令前参加如下指令：

```
.sect “.vectors ”
    b start
```

35) 〕保存对 lab2.asm的 修改；

36) 〕双击工程扫瞄窗口中的 lab2 d, 激活该窗 口；

37) 〕在 .text :{}> PROG 0 的下一行参加如下行：

```
.vectors :{}>VECT 0 38〕
```

执行 Project →Build 重编译源程序； 39〕执

行 File →Exit 退 出 Code Composer ；

40) 〕在桌面上双击 CCS (C2023) 图标重启动 Code Composer ；

41) 〕执行 File →Load Program 装载 lab2.out ；

42) 〕按 F5 运行程序；

43) 〕按 Shift-F5 完毕程 序运行；

44) 〕执行 Debug →Reset CPU 命令，观看程序的运行过程。

5.4 试验四　数模转换试验

1. 试验目的

- 理解 TMS320LF2407A 存储器空间的概念：程序空间、数据空间和 I/O 空间
- 把握 DA 转换器的原理和使用方法
- 把握 CCS 开发环境平台的使用

2. 试验内容

- DSP 的初始化配置
- 存储器 I/O 空间的使用
- DA 转换的写入、转换结果的观看
- 可编程运算放大器试验

3. 硬件原理图

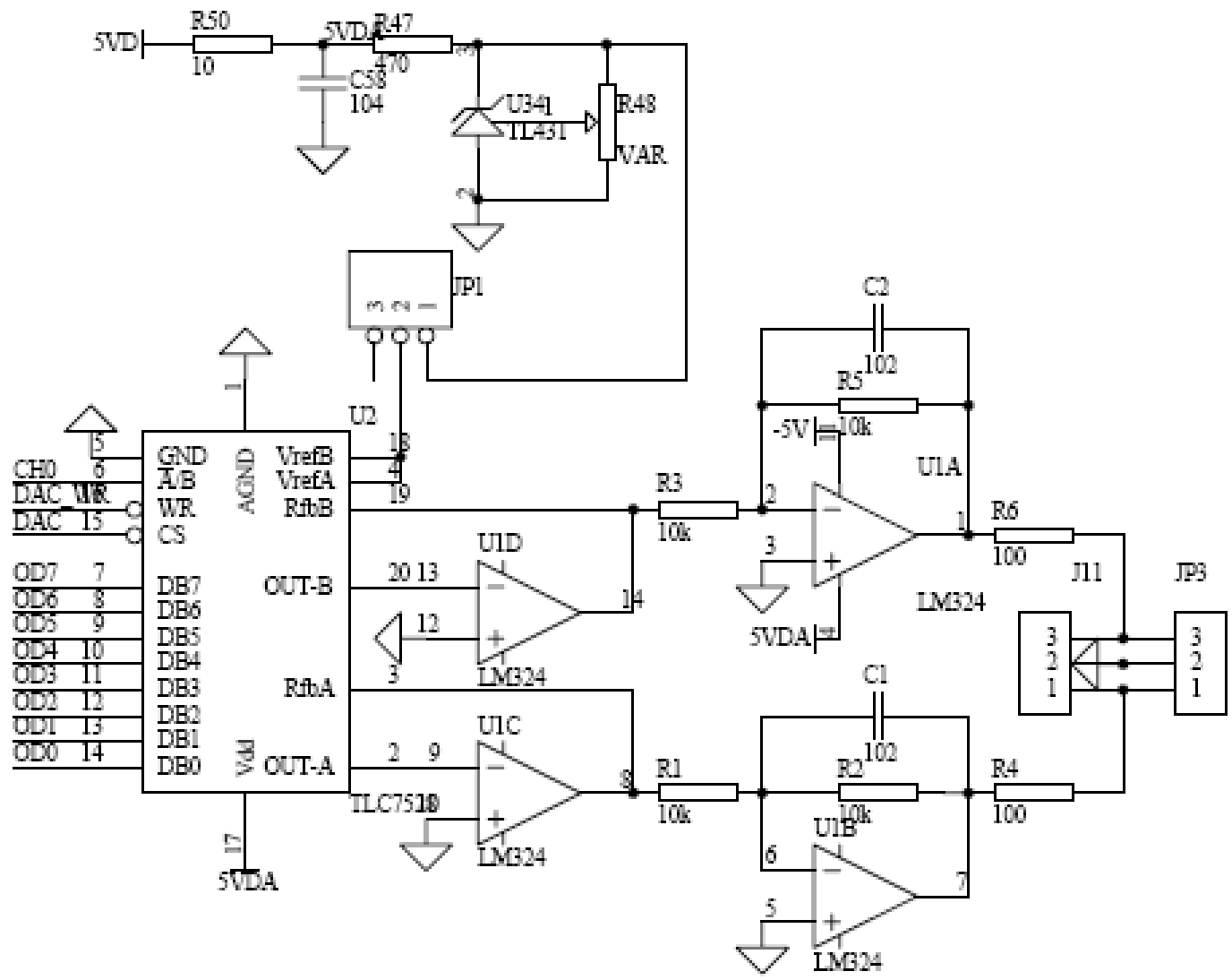


图 24 数模转换电路硬件原理图

试验箱上配置的是一个 2 通道 8 位并行输入 DA 转换器，电流型输出，单电源供给。由于输出的电流与输入的电压极性正好相反，因此电路上配置了一个电流/电压转换和一个电压反向器，使得输出信号是与参考源极性全都的电压信号。DAC 转换器输出电压公式为：

$$Vo = +Vref\left(\frac{D}{256}\right)$$，D＝输入的 8 位二进制转换为 10 进制的值

DA 转换器有 2 个输出通道，每一个通道的数字量输入通过内部的锁存器进展锁存，DSP 通过 DA 转换器的 WR、A0 和 A1 三个引脚，经过 CPLD 译码后，进展 DA 数字量的写入，图 26 为操作时序图。

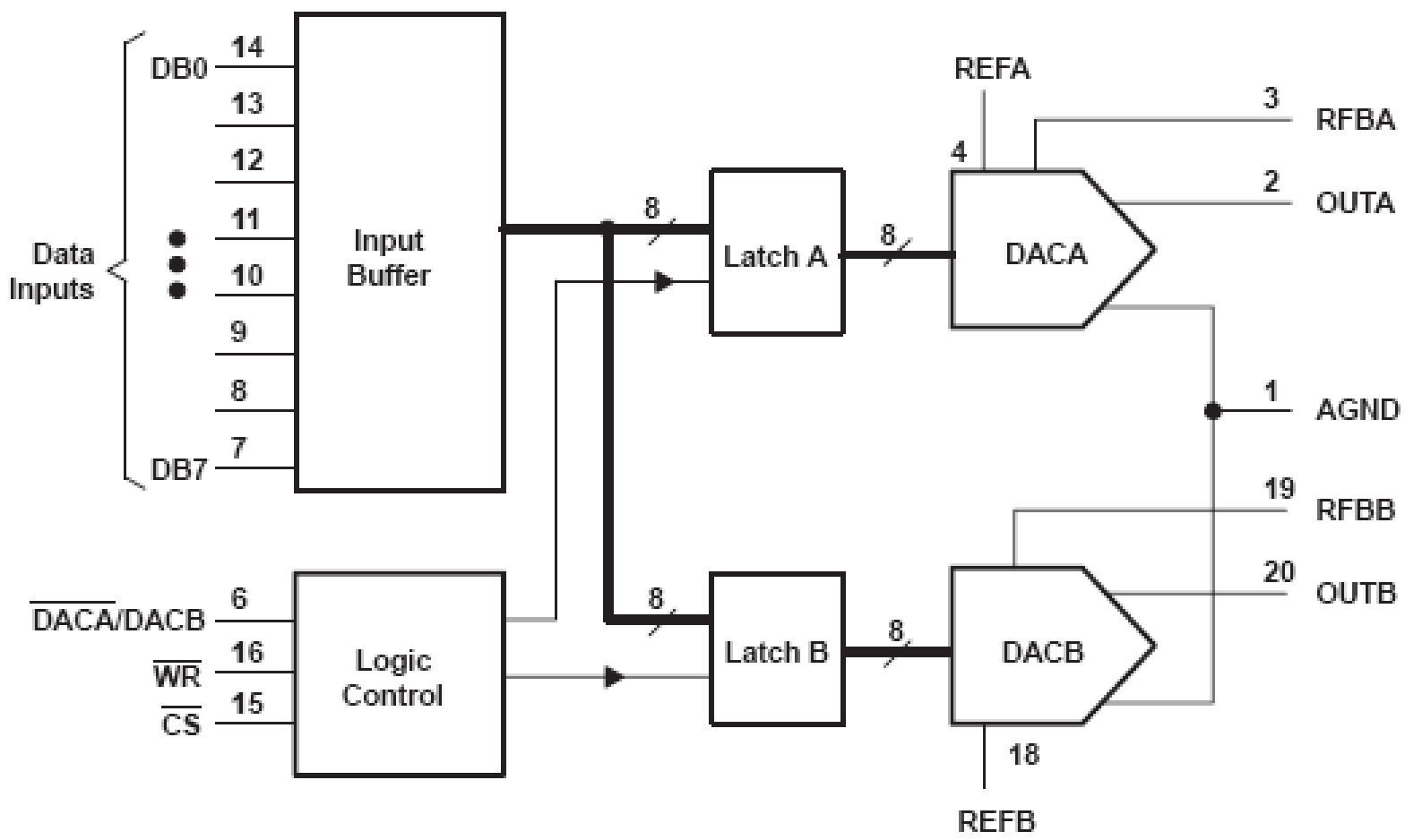


图 25 数模转换芯片的内部原理构造

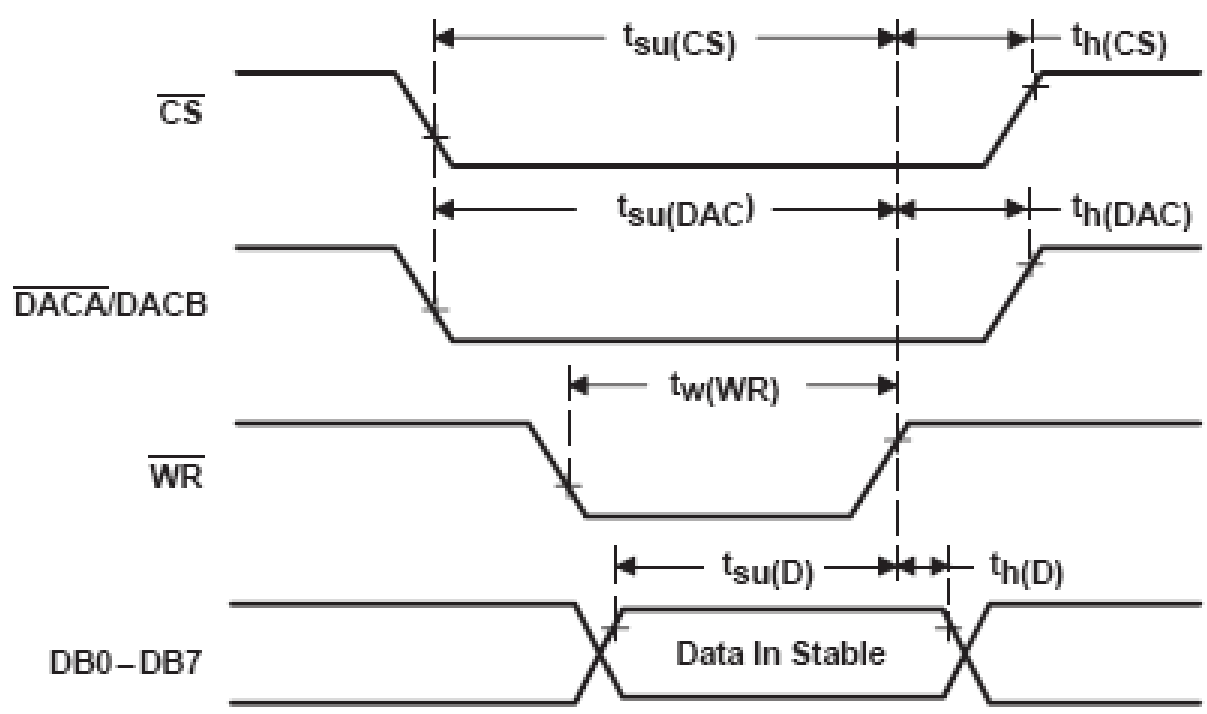


图 26 数模转换芯片的操作时序图

4. 试验步骤

数模转换试验是利用 DSP 将内存中正弦表〔或者余弦表〕所设定数字量，由 DSP 定时依次将这些数字量送入到 DAC 的 2 路通道内，通过 DAC 转换为的模拟正弦波，我们可以通过示波器显示出波形。图 27 为产生正弦波信号的软件流程图。

1〕如果你的软件安装在 C 盘的根目录下，那么实验程序目录应该是 C:\ti\myprojects\DA\ 〔X\ti\myprojects\DA\ ，其中 X 代表 CCS 的安装名目〕；

2） Code Composer Studio 主界面中翻开 Project 选项选择 New 选项；

3）在 Save New Project 对话框中，选择试验 程序所在的工作名目。在文件名编辑框中输入 DA 做为工程名称，建立工程工程文件 DA.pjt ；

4）翻开 Project 选项选择 Add file to project ，在随后翻开的窗口中转变文件类型为〔*.*〕，选择全部后 缀名为 asm、c 和 cmd 的文件 并按翻开按钮；

5）屏幕左侧的窗口是 Project List 单击列表项旁 的+ 开放 Project DA.pjt 和 Source 项，观看上述文件是否都被包含到工程中；

留意：工程扫瞄器，假设在屏幕上看不到工程扫瞄器，请翻开 View 选项选择 Projects 工程。假设工程扫瞄器已经翻开但看不到工程文件 ，在工程扫瞄器窗口下的 File 标签上单击。

6）在工程扫瞄器中，双击 main.c ， 激活 main.c 文件，扫瞄该文件的内容；

7） Code Composer Studio 可以自动的保存工程工程的状态。你可以使用 Project 下的 Open 翻开一个工程工程文件的同时，恢复上 次退出 CodeComposer Studio 时工作环境的设置值；

8）翻开 Project 选项，选择 Rebuild all 选项， Code Composer Studio 重编译和链接这个工程工程，整个的处理过程在屏幕下方的 Message 窗口中返回信 息，当转变了设置后必需从编译全部的文件；

9）翻开 File 选项，选 择 Load Program 选项，在 Load Program 对话框中，选中建名目下的 DA.out 文件，此时 Code ComposerStudio 将把这个目标文件 装载到 LF2407 试验箱中，同时 Code Composer Studio 翻开反汇编 窗口，显示被加载程序的汇编指令码；

10）〕翻开 Debug 选项，选择 Run 选项或按 F5 运行程序，将示波器一端接在从板上引出的地线上〔 JP3 的 2 脚〕，一端分别接在DA 芯片的输出引脚 JP3 的 1、3 脚上，这样我们就可以通过示波器看到这 2 路通道中输出的正弦波；

11）翻开 Debug 选项，选择 Halt 选项或按 Shift-F5 终止试验结果。

12）〕转变在 main.c 文件 中，将 periodDA 添加 到变量观看窗口中，再次运行，在变量观看窗口转变 periodDA 的值，同时观看输出正弦波形的变化。

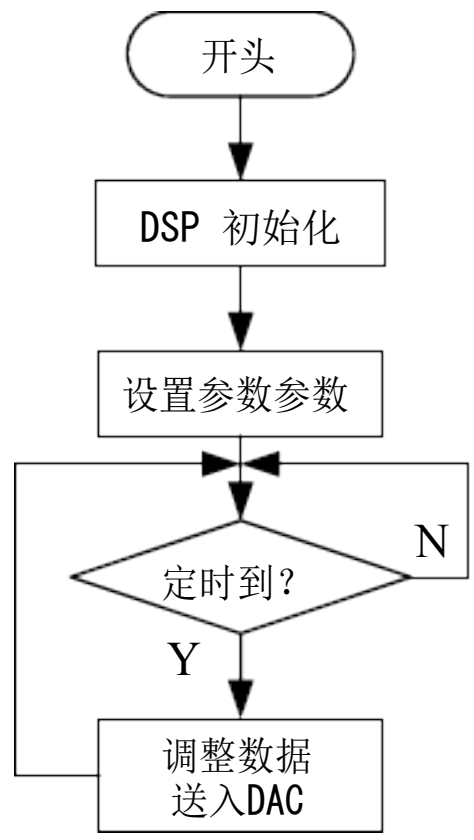


图 27 用 D/A 发生正弦波信号的软件流程

5. 试验结果

试验的最终现象：通过示波器可以看到 1 路标准的正弦波和 1 路标准的锯齿波 。

扩展：请修改程序，把输出的锯齿波变为三角波。

5.5 试验五　用户使用开关试验

1. 试验目的

- 理解 TMS320LF2407A 存储器空间的概念：程序空间、数据空间和 I/O 空间
- 把握 CCS 开发环境平台的使用

2. 试验内容

- DSP 的初始化配置
- 存储器 I/O 空间的使用

3. TMS320LF2407A 存储器空间简介

LF240xA 的存储器包括 RAM〔单口 RAM 和双口 RAM〕、ROM 和 Flash。对于以 LF 为前缀的 DSP 芯片具有 Flash，而以 LC 为前缀的 DSP 芯片具有 ROM。240xA 具有 16 位地址总线，可以独立访问如下三种空间〔共 192K〕：64K 字的程序空间、64K 字的数据空间和 64K 字的 I/O 空间。

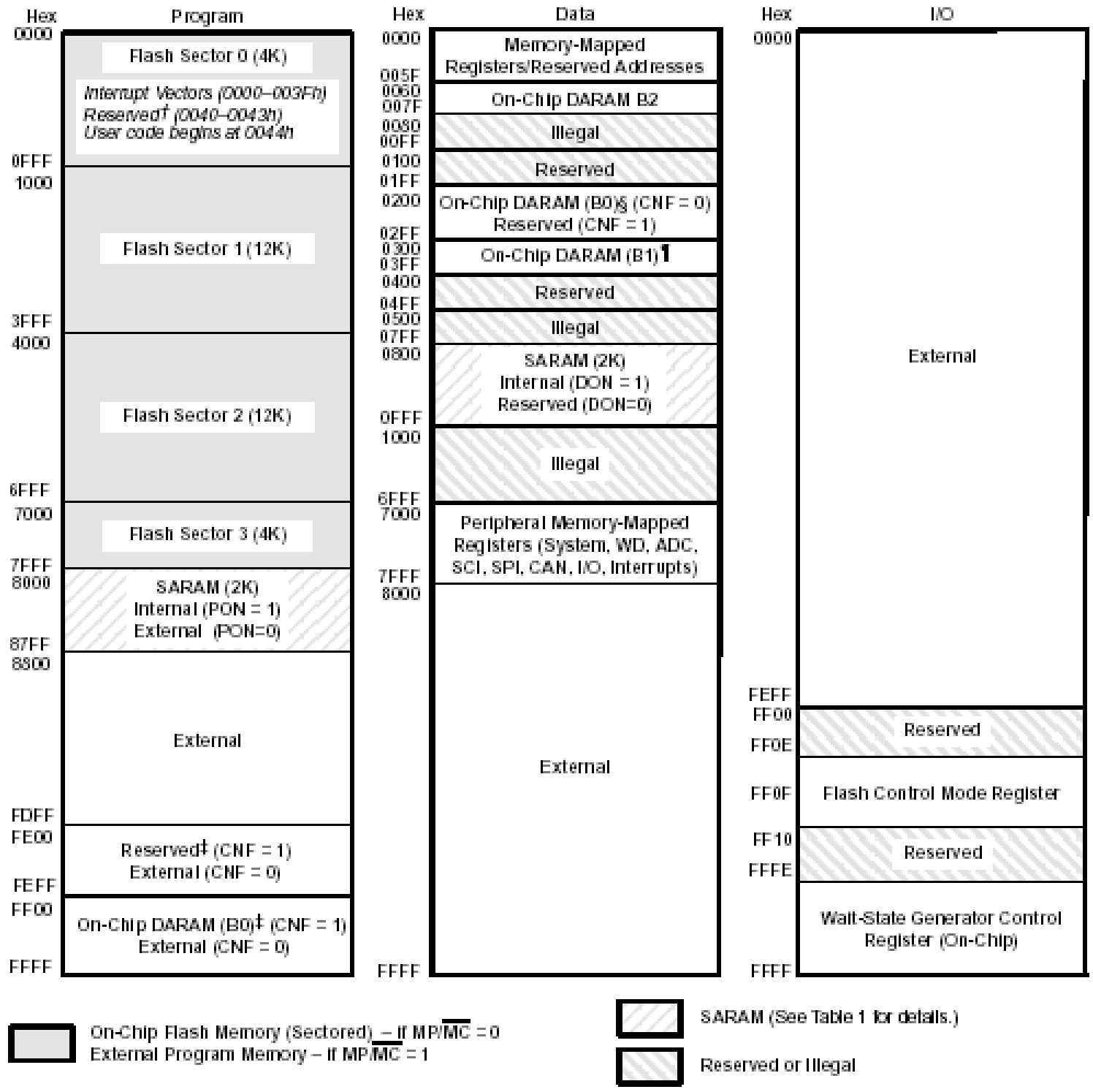


图 28 2407A 的存储器映射图

本章还介绍程序、数据和 I/O 空间的地址映像以及 240xA 的可访问的存储器配置选项。图 28 是 LF2407A 的存储器映像。

2407A 芯片的设计基于增加型的哈佛构造，它具有多块存储空间，通过程序地址总线〔PAB〕、数据读 地址 总 线〔 DRAB 〕和数据 写地址总 线〔 DWAB〕这 三条并 行总 线可 以被访问。每条总线在操作的不同阶段访问不同的存储空间。由于 CPU 的总线操作是相互独立的，所以 CPU 能够同时对程序存储空间和数据存储空间进展访问。在一个给定的机器周期内，中心算术规律单元〔 CALU 〕能够执行三 条并行的存储器操作。

240xA 器件的地址映像可以分为 3 个独立的选择空间，总计 192K 字的地址范围：

- 程序存储器〔 64K 字〕：存放指令机器码以及在执行程序时要使用的数据；
- 数据存储器〔 64K 字〕：保存指令使用的数据；
- 输入 /输出 I/O 空间〔 64K 字〕：用于和外设 接口，并包含片内外围设备的存放器。

2407A 芯片内还含有大量的片内存储器。这些存储器集成度高，运行速度快，功耗和本钱较低且使用便利，有助于提高系统的整体性能，较片外存储器操作有更多的优点。但是对于片外存储器操作来说，可以访问更大的地址空间。在 240xA 系列的器件中，只有 2407A 具有外部存储器接口，其他的器件只 有片内存储器。

4. 试验硬件原 理图

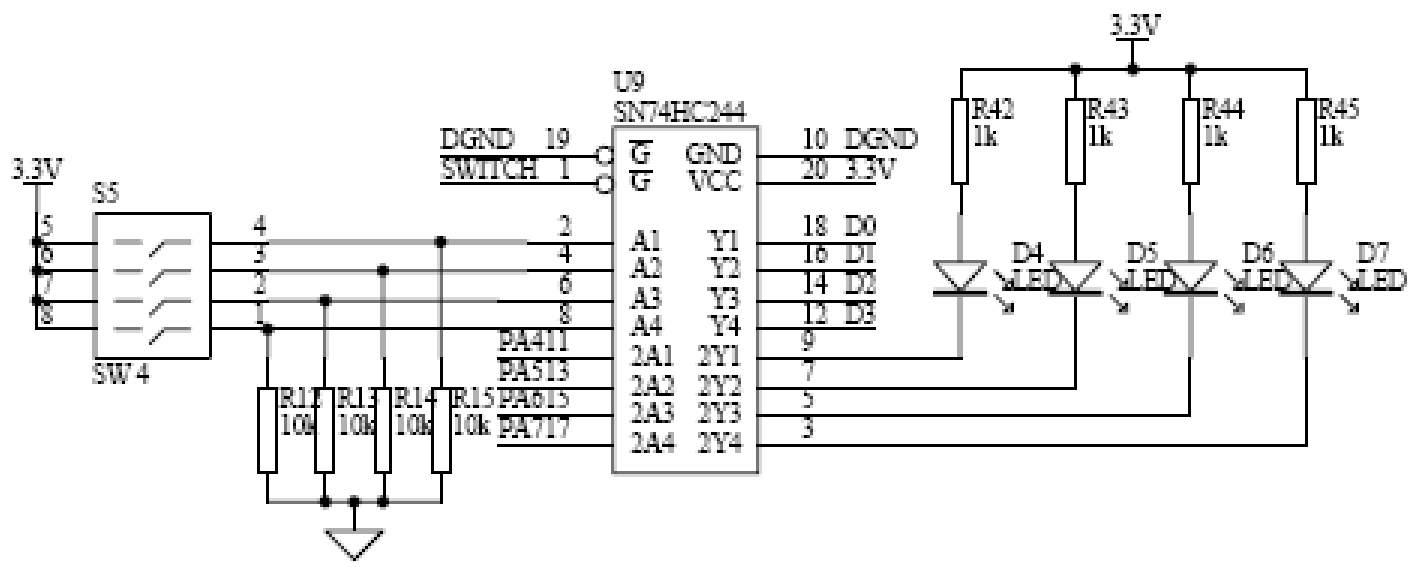


图 29 用户拨码开关输入及 LED 显示硬件电路原理

如图 29 所示，拨码开关通过缓冲器连接到 DSP 的数据总线上，缓冲器的规律使能是经过 CPLD 的内部译码 产生的。 从 DIP 开关取数据到 I/O 空间，再送到 LED 显示； LED 的状态反映了对应的 DIP 的状态。用户使用开关的使用试验主要是将用户使用开关的状态与指示灯的状态相统一，当开关设置在 ON 时，指示灯为亮；当设置 为 OFF 时，指示灯为灭。

2〕软件流程图

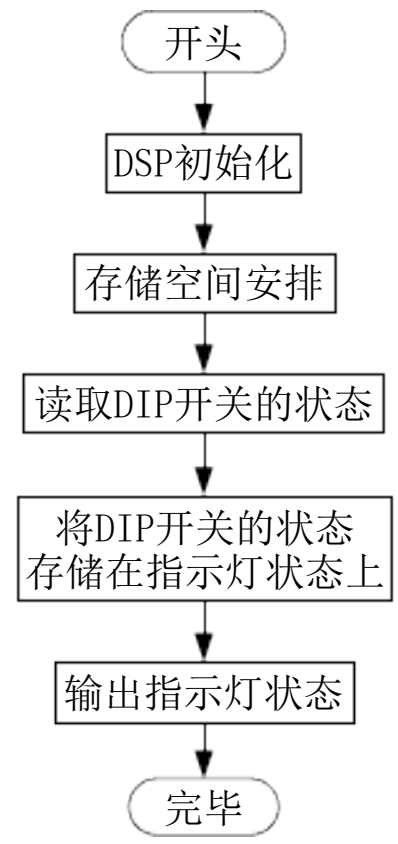


图 30 用户使用开关软件流程图

5. 试验步骤

1〕如果你的软件安装在C盘的根目录下，那么实验程序目录应该是C:\ti\myprojects\dip\〔或者X\ti\myprojects\dip\，其中X代表CCS的安装名目〕；

2）Code Composer Studio主界面中翻开Project选项选择New选项；

3）在Save New Project对话框中，选择试验程序所在的工作名目。在文件名编辑框中输入dip做为工程名称，建立工程工程文件dip.pjt；

4）翻开Project选项，选择Add file to project，在随后翻开的窗口中选择main.c并按“翻开”按钮；同样添加initsystem.c,vectors.asm到工程中。

5）翻开Project选项，选择Add file to project，在随后翻开的窗口中转变文件类型为〔* d〕选择test2 d并按“翻开”按钮；

6）屏幕左侧的窗口是Project List，单击列表项旁的“+”开放Project、dip.pjt和Source项，观看上述文件是否都被包含到工程中；

留意：工程扫瞄器，假设在屏幕上看不到工程扫瞄器，请翻开View选项，选择Projects工程。假设工程扫瞄器已经翻开，但看不到项目文件，在工程扫瞄器窗口下的File标签上单击；

7）在工程扫瞄器中，双击main.c，激活main.c文件，扫瞄该文件的内容；

8）翻开Project选项，选择Rebuild all选项，Code Composer Studio重编译和链接这个工程工程，整个的处理过程在屏幕下方的Message窗口中返回信息，当转变了设置后必需重编译全部的文件；

9）翻开File选项，选择Load Program选项，在Load Program对话框中选中，建名目下的dip.out文件，此时Code Composer Studio将把这个目标文件装载到LF2407试验箱系统中，同时，Code Composer Studio翻开反汇编窗口显示被加载程序的汇编指令码；

10）〕翻开Debug选项，选择Run选项或按F5运行程序，我们对用户使用开关S5进展设置就可以定制指示灯是否为亮；

11）翻开Debug选项，选择Halt选项或按Shift-F5终止试验结果。

试验的最终现象：可以看到哪一个用户开关设置在　　ON　状态上哪一盏指示灯就为亮。