特别说明

MSP430单片机

自学笔记

主　编　张福才
副主编　张　锐　汝洪芳

北京航空航天大学出版社
BEIHANG UNIVERSITY PRESS

CD-ROM INCLUDED

MSP430 单片机自学笔记

主　编　张福才
副主编　张　锐　汝洪芳

北京航空航天大学出版社

内 容 简 介

本书以 TI 公司的 MSP430 系列 16 位超低功耗单片机为核心，介绍了 MSP430 系列单片机的特点和选型，详细讲述了 MSP430 系列单片机的结构和指令系统，对 MSP430 全系列单片机（包括最新的 F15x、F16x）所涉及的片内外围模块的功能、原理、应用作了详尽的描述，并配有完整的经过测试的实用子程序；还包括 MSP430 单片机的开发环境安装方法、汇编语言、C 语言程序设计方法，以及单片机常用外设的接口电路，如显示（数码管、1602、12864）、键盘、DS18B20 温度传感器、24C02、DDS 芯片 AD9850/AD9851 等外设和软件编程的方法。

本书着重讲述 MSP430 单片机的基本原理，更侧重于应用，配有完整的例子和详细的源程序，非常适合个人自学使用；亦可作为高等院校计算机、电子、自动化类专业 MSP430 单片机课程的教材，也适合广大从事单片机应用系统开发的工程技术人员作为学习、参考用书。

图书在版编目(CIP)数据

MSP430 单片机自学笔记 / 张福才主编. -- 北京 : 北京航空航天大学出版社，2011.2

ISBN 978-7-5124-0303-1

Ⅰ. ①M… Ⅱ. ①张… Ⅲ. ①单片微型计算机 Ⅳ. ①TP368.1

中国版本图书馆 CIP 数据核字(2010)第 261426 号

MSP430 单片机自学笔记

主　编　张福才

副主编　张　锐　汝洪芳

责任编辑　张冀青

*

北京航空航天大学出版社出版发行

北京市海淀区学院路 37 号（邮编 100191）　http://www.buaapress.com.cn

发行部电话：(010)82317024　传真：(010)82328026

读者信箱：bhpress@263.net　　邮购电话：(010)82316936

涿州市新华印刷有限公司印装　各地书店经销

*

开本：787×960　1/16　印张：18　字数：403 千字

2011 年 2 月第 1 版　2011 年 2 月第 1 次印刷　印数：4 000 册

ISBN 978-7-5124-0303-1　　定价：35.00 元（含光盘 1 张）

前　言

MSP430 系列单片机是 TI 公司推出的一种 16 位的单片机。由于它具有较高的集成度、丰富的片内外设、超低功耗等优点，因此在许多领域都得到了广泛的应用。与其他单片机相比，MSP430 系列单片机具有以下几个方面的特点。

① 在超低功耗方面，其处理器功耗(1.8～3.6 V，0.1 μA/Powerdown，0.8 μA/Standby，250 μA/MIPS)和口线输入漏电流(最大 50 nA)在业界都是最低的，远低于其他系列产品。

② 在运算性能上，其 16 位 RISC 结构，使 MSP430 系列单片机工作在 16 MHz 晶振时，指令速度可达 16 MIPS(注意：同样 16 MIPS 的指令速度，16 位处理器比 8 位处理器在运算性能上高远不止 2 倍)；不久还将推出 25～30 MIPS 的产品。同时，MSP430 系列单片机中采用了一般只有 DSP 中才有的 16 位多功能硬件乘法器、硬件乘加(积之和)功能、DMA 等一系列先进的体系结构，大大增强了它的数据处理和运算能力，可以有效地实现一些数字信号处理的算法(如 FFT、DTMF 等)。

③ 在开发工具上，MSP430 系列单片机支持先进的 JTAG 调试，简单的硬件仿真工具(仿真器)只是一个并口转接器，一般人都可以自己制作，而且适用于所有 MSP430 系列单片机，既便于推广，又大大降低了用户的开发投入。其软件集成开发环境 EW430 由著名的 IAR 公司提供，其最新版本已比较完善。

④ 在系统整合方面，MSP430 系列单片机结合 TI 公司独到的高性能模拟技术，根据其不同产品，集成了多种功能模块，包括定时器、模拟比较器、多功能串行接口(SPI/IIC/UART)、LCD 驱动器、硬件乘法器、10/12/14/16 位模/数转换器(ADC)、12 位数/模转换器(DAC)、看门狗定时器(WDT)、I/O 端口(P0～P6)、DMA 控制器、多达 120 KB 的 Flash、10 KB 的 RAM 和 SVS，以及丰富的中断功能。用户可以根据应用需求，选择最合适的 MSP430 系列产品。另外，大部分 MSP430 系列单片机采用 Flash 技术，支持在线编程，并有保密熔丝。其 BOOTSTRAP 技术为系统软件的升级提供了又一种方便的手段，BOOTSTRAP 有 32 字节的口令字，具有很高的保密性。MSP430 系列单片机均为工业级产品，性能稳定，可靠性高，可用于各种民用、工业产品。

MSP430 系列单片机可采用汇编语言和 C 语言进行开发。由于采用 C 语言开发可以大大提高开发效率，缩短开发周期，并且采用 C 语言开发的程序具有非常好的可读性和移植性，

因此，C 语言成为开发 MSP430 系列单片机的主流语言。由于适用于 MSP430 系列单片机的 C 语言与标准 C 语言兼容度高，并且 IAR 公司提供的 Embedded Workbench 集成开发环境人机界面友好，能很好地支持 C 语言开发、在线下载与仿真，因此本书的程序都是采用 C 语言进行编写的。

本书融合了作者最近几年对 MSP430 系列单片机学习与开发应用的经验和体会，内容以硬件为基础，配有完整的应用源程序，源程序在光盘中。书中的所有源程序代码都经过实际验证和测试，应用举例和综合设计多取材于实际应用项目。本书附带的光盘上有实验板原理图、源程序、430 JTAG 原理图及 PCB 图等。最新的资料、代码等可以发 E-mail 至 msp430mcu@163.com，免费索取。

本书中所讲述的例子是在 F169 单片机基础上编写的程序。如果需 MSP430F169 中文资料，可以发 E-mail 至 qingtengzfc@yeah.net 索取。

在本书的编写过程中，张锐、汝洪芳、赵晓彦等同志做了很多代码验证、资料整理工作。在此向他们表示衷心的感谢。

由于时间仓促和水平所限，错误之处在所难免，恳请读者批评指正。读者在学习过程中，若发现什么疏漏之处，可以发邮件给我，我的邮箱是 qingtengzfc@yeah.net。

作　者

2011 年 1 月

于黑龙江科技学院

目　录

第 1 章

概　述

1.1　MSP430 系列单片机概述

单片机的应用范围日益广泛，生产批量大，成本低廉（目前最低的单片机价格为 3～5 元）；系统结构简单而可靠性高，采用 CMOS 工艺大大降低了功耗。因此，单片机已成为微型计算机的一个重要的分支，发展速度极快；从 4 位、8 位、16 位再到 32 位，单片机的种类有几百种，世界年销售量几亿片。

1.1.1　单片机的特点

单片机具有以下几个方面的特点：

① 封装形式上外形较小，成本低，易于产品化，能够方便地组装成各种智能的控制设备以及各类的智能仪器仪表。

② 面向控制，能够完成相对比较复杂的控制任务，具有较好的性价比。

③ 具有较广的温度适应范围，在各种恶劣的环境下仍可以正常工作，这是其他机型无法比拟的。

④ 可以很方便地实现多机和分布式控制。

⑤ 下载与仿真接口比较简单，便于程序的开发。

本书所介绍的 MSP430 系列单片机属于通用型的单片机，除了具有上述几个方面的特点外，还具有以下几方面的特点：

1. 超低功耗

MSP430 系列单片机的电源电压采用 1.8～3.6 V，待机电流小于 1 μA，在 RAM 数据保持方式时耗电电流小于 0.1 μA，在活动模式时耗电 250 μA/MIPS（MIPS 表示每秒百万条指令），I/O端口的漏电流最大为 50 nA。

MSP430 系列单片机有独特的时钟系统设计，包含两套不同的时钟系统：基本时钟系统和锁频环（FLL 和 FLL＋）时钟系统或 DCO 数字振荡器时钟系统。由时钟系统产生 CPU 和

各功能模块所需的时钟，这些时钟可以在指令的控制下打开或者关闭，从而实现对总体功耗的控制。由于系统运行时使用的功能模块不同，即采用的工作方式不同，芯片的功耗就明显不一样了。在系统中，有 1 种活动模式(AM)和 5 种低功耗模式(LPM0～LPM4)。

2. 强大的处理能力

MSP430 系列单片机是 16 位单片机，采用了目前流行的、颇受业界好评的精简指令集(RISC)结构，一个指令周期可以执行一条指令(在 51 单片机里需要 12 个指令周期才能执行一条指令)，MSP430 系列单片机使用 8 MHz 工作时，指令速度可达 8 MIPS(因为 MSP430 系列单片机是 16 位单片机，所以相对于同样工作在 8 MHz 的 51 单片机来说，MSP430 系列单片机的速度比 51 单片机的速度快得多)。TI 公司还在致力于推出 25～30 MIPS 的产品。同时，MSP430 系列单片机的某些型号，采用了一般只有 DSP 中才有的 16 位多功能硬件乘法器、硬件乘加(积之和)功能、DMA 等一系列先进的体系结构，大大增强了其数据处理和传输的能力，可以有效地实现一些数字信号处理的算法(如 FFT、DTMF 等)。这种结构在其他系列单片机中比较少见。

3. 高性能的模拟技术及丰富的片上模块

MSP430 系列单片机结合了 TI 公司的高性能模拟技术，各成员都集成了较为丰富的片内外设。根据型号分别组合了以下功能模块：10/12/16 位 ADC、12 位 DAC、比较器、LCD 驱动器、电源电压(SVS)、串行通信(UART、IIC、SPI)、红外线遥控器(IrDA)、硬件乘法器(MPY)、DMA 控制器(DMAC)、温度传感器、看门狗定时器(WDT)、定时器 A(Timer_A)、定时器 B(Timer_B)、端口 1～8(P1～P8，有些没有那么多端口，具体有多少根据型号而定)、基本定时器(Basic Timer)、实时时钟模块(RTC)、运算放大器(OA)以及扫描接口(Scan IF)等。其中，看门狗可以使程序在失控或跑飞(执行乱套)时迅速复位，模拟比较器进行模拟电压的比较，配合定时器可以设计出高精度(10～12 位)的 ADC；16 位定时器(Timer_A 和 Timer_B)具有捕获/比较功能，大量的捕获/比较寄存器，可用于事件的计数、时序发生、PWM 等；多功能串口(USART)可以实现异步、同步(SPI)、IIC 串行通信，可以方便地实现多机通信等应用；USCI 模块还包含红外控制部分，具有较多的 I/O 端口，最多达 10×8 条 I/O 线，每个 I/O 线不管是灌电流还是拉电流，每个晶体管都能限制输出电流(最大输入/输出电流25 mA)，保证芯片安全；P1、P2 端口接收外部上升沿或下降沿的中断输入；12 位 ADC 有较高的转换速率，最高可达 200 KSPS，能够满足大多数数据采集的场合。MSP430F4xx 单片机还带液晶驱动功能，最多可以驱动液晶的段数可达 160 段。F15x 和 F16x 系列有两路 12 位高速 DAC，可以实现直接数字波形合成等功能；硬件 IIC 串行总线接口可以扩展 IIC 接口器件；DMA 功能可以提高数据传输速度，减轻单片机的负荷；运算放大器(OA)可在模拟信号转换成数字信号前，支持前端模拟信号状态；扫描接口(Scan IF)能检测传感器，测量直线运动或者转动。

MSP430 系列单片机丰富的片内外设，在目前所有的单片机系列产品中是非常突出的，为

系统的单片解决方案提供了极大的可能性与方便性。

4. 系统工作稳定

上电复位后，首先由 DCO_CLK 启动单片机(数字控制振荡器 DCO 是 MSP430 系列单片机内置的)，以保证程序从正确的位置开始执行，保证外接的晶体振荡器有足够的起振和稳定时间。然后通过软件设置适当的寄存器和控制位来确定最后的系统时钟频率。如果晶体振荡器在用做 CPU 时钟 MCLK 时发生故障，则 DCO 会自动启动，以保证系统正常工作。这种结构和运行机制，在目前各系列单片机中是绝无仅有的。另外，MSP430 系列单片机均为工业级器件，运行环境温度为－40～＋85 ℃，运行稳定，可靠性高，所设计的产品适用于各类民用和工业环境。

5. 方便高效的开发环境

目前的 MSP430 系列单片机有 OPT 型、Flash 型、ROM 型和 EPROM 型 4 种，国内大量使用的是 Flash 型的器件。这些器件开发手段不同，对于 OPT 和 ROM 型的器件，使用专门的仿真器开发成功后再烧写到芯片中去，适于大批量的成型生产。

对于 Flash 型器件有十分方便的开发环境，因为这类器件带有 JTAG 调试接口，还有可电擦写的 Flash 存储器，因此采用先通过 JTAG 接口下载程序到 Flash 存储器内，再由 JTAG 接口控制程序的运行，可以读取单片机内部状态和相关存储器的内容回传到 PC 机上，整个开发(编译、调试)都可在同一个软件集成环境中运行。这种方式只需要一台 PC 机和一个 JTAG 调试器(完全可以自制)，不需要专门仿真器和编程器。开发语言有汇编语言和 C 语言。目前较好的开发环境是 IAR 的 IAR Embedded Workbench，本书介绍的程序就是在此软件的基础上编写的，具体版本是 Electronic Workbench for MSP 430 V3.42A。

这种以 Flash 技术、JTAG 调试、集成开发环境相结合的开发方法，具有方便、廉价、实用等优点，在单片机开发中还较为少见。其他系列单片机的开发一般均需要专门的仿真器或编程器。

2001 年 TI 公司又公布了 BSL(Boot Strap Loader)技术，利用它可在保密熔丝烧断以后，只要几根硬件连线，通过软件口令字(密码)，就可更改并运行内部程序，为系统固件的升级提供了又一方便的手段。BSL 具有很高的保密性，口令字长度可达 256 位。

1.1.2 单片机的应用及产品概况

MSP430 系列单片机种类繁多，在介绍产品之前，需要了解 MSP430 系列单片机命名规则。命名中包括片内存储器的类型、系列号、子系列号、温度范围、封装类型等信息。其中，片内存储器类型有 ROM(C)、EPROM(E)、Flash(F)、OTP(P)、USER(U)等，各类型存储器特性如表 1－1 所列。

表 1-1　各类型存储器特性

类　型	名　称	特　性
C	ROM	只读存储器，适合大批量生产
P	OTP	单次可编程存储器，适合小批量生产
E	EPROM	可擦除只读存储器，适合开发样机
F	Flash	闪存，具有 ROM 型的非易失性和 EPROM 的可擦除性

- 系列号码为存储器类型后面的一位数字，如 1、2、4 分别代表 MSP430 的 1、2、4 系列单片机。
- 子序列号码为包括系列号码在内的若干位数字，用来表示更加具体的芯片内部外围模块的配置和存储容量等信息。
- 封装类型表示方面有 DW（SOIC20，1. 27 mm 脚距）、RGE（QFN24）、DGV（TVSOP20）、PW（TSSOP20，0. 65 mm 脚距）、PM（QFP80，0. 5 mm 脚距）、PN（QFP80，0.5 mm 脚距）、PZ（QFP100，0.5 mm 脚距）。
- 温度范围包括 I（工业级）和 A（汽车级）。

以 MSP430F449 为例，F 表示 Flash 型；44 表示 MSP430 单片机 4 系列中的 44 子系列，片内具有 ADC12、LCD 和硬件乘法器等外围器件；9 表示存储容量为 60 KB。

TI 公司的 MSP430 系列的单片机种类齐全，用户可以根据应用需求选择合适的芯片。关于命名规则这部分可以参考附图 1。

1.2　MSP430 系列单片机的发展和应用

TI 公司从 1996 年推出 MSP430 系列单片机开始到 2000 年初，相继推出了 MSP33x、MSP32x、MSP31x 等几个系列，目前还有 16x 系列 MSP4xx 等，总之，型号比较多。MSP4xx 带有液晶驱动接口，有利于提高系统的集成度。每一系列有 ROM 型、OTP 型和 EPROM 型等芯片。EPROM 型的价格昂贵，运行环境温度范围窄，主要用于样机的开发。这也表明了这几个系列的开发模式。用户可以用 EPROM 型开发样机，用 OTP 型进行小批量生产，而 ROM 型适合大批量生产。

另外，随着 Flash 技术的迅猛发展，TI 公司也将这一技术引入到了 MSP430 系列单片机中。2000 年推出了 F11x/F11x1 系列，这个系列采用的是 20 引脚的封装形式，其内存容量、片上功能和 I/O 引脚都比较少，但是价格相对较低，适合于简单的产品开发。在 2000 年 7 月推出了带 ADC 或硬件乘法器的 F13x/F14x 系列。2001 年以后又推出了带 LCD 控制器的 F41x、F43x、F44x。

TI 公司在 2003—2004 年期间推出了 F15x 和 F16x 系列产品。在这一系列的产品中，有了两个方面的发展，一是增加了 RAM 的容量，如 F1611 的 RAM 容量增加到了 10 KB，这样就可以引入实时操作系统(RTOS)或简单文件系统等；二是从外围模块来说，增加了 IIC、DMA、DAC12 和 SVS 等模块。

MSP430F169 单片机引脚定义如图 1-1 所示。

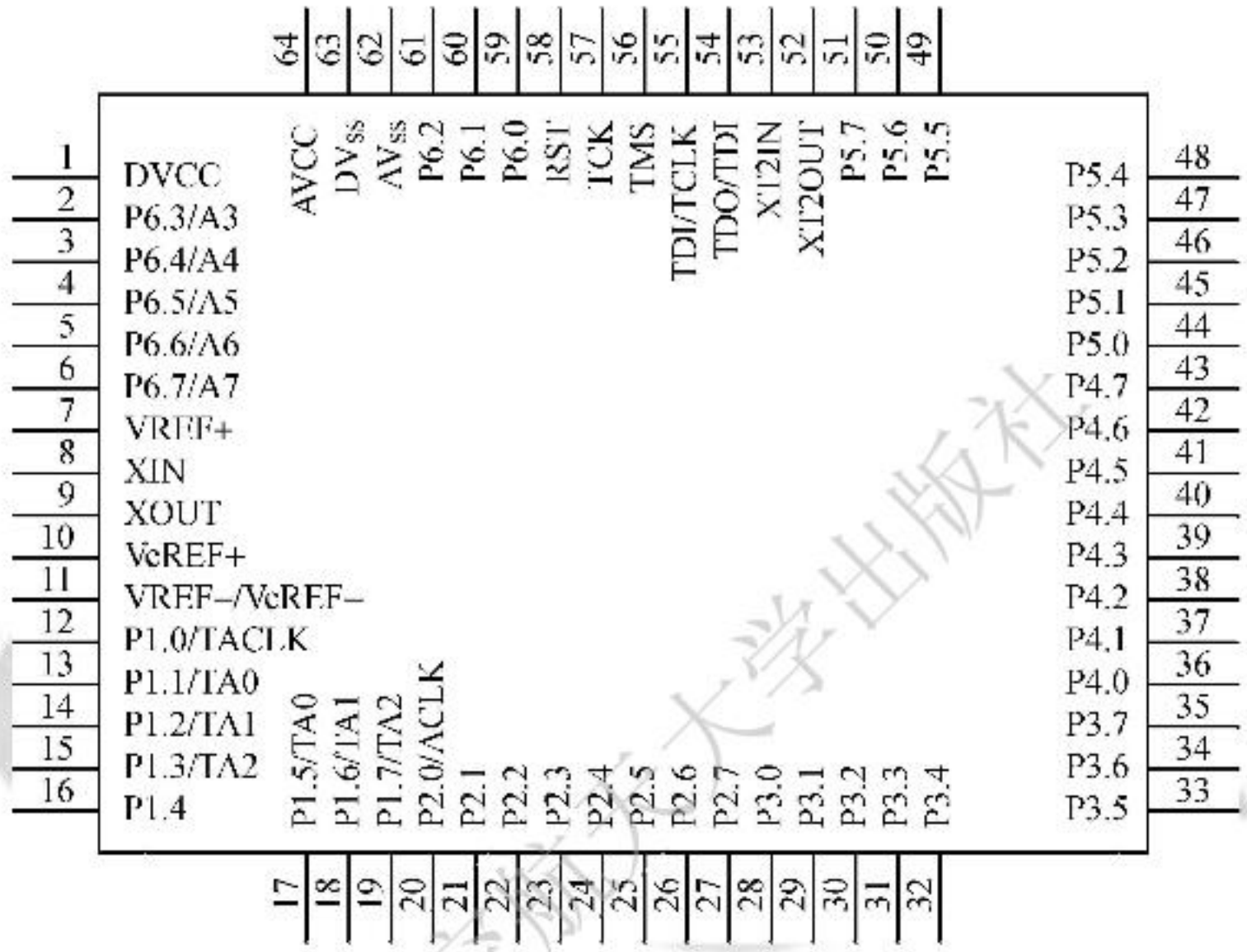


图 1-1　MSP430F169 单片机引脚定义

1.3　MSP430 系列单片机的选型

目前，MSP430 系列单片机的主流 Flash 产品如表 1-2 所列。

表 1-2　MSP430 系列产品选型表

型　号	Flash/KB	RAM/字节	A/D	D/A	DMA	LCD	USART	比较器	乘法器	定时器数量	封　装	端口数量
MSP430F1101A	1	128	Slope				软件	√		2	20SOIC TSSOP	14
MSP430F1111A	2	128	Slope				软件	√		2	20SOIC TSSOP	14

续表 1-2

型　号	Flash/KB	RAM/字节	A/D	D/A	DMA	LCD	USART	比较器	乘法器	定时器数量	封　装	端口数量
MSP430F1121A	4	256	Slope				软件	√		2	20SOIC TSSOP	14
MSP430F1122	4	256	10 位				软件			2	20SOIC TSSOP	14
MSP430F1132	8	256	10 位				软件			2	20SOIC TSSOP	14
MSP430F1222	4	256	10 位				硬件			2	20SOIC TSSOP	22
MSP430F123	8		Slope				硬件	√		2	20SOIC TSSOP	22
MSP430F1232	8		10 位				硬件 1			2	20SOIC TSSOP	22
MSP430F133	8		12				硬件 1	√		3	64LQFP	48
MSP430F135	16		12 位				硬件 1	√		3	64LQFP	48
MSP430F147	32		Slope				硬件 1	√	√	3	64LQFP	48
MSP430F1471	32		12 位				硬件 2	√	√	3	64LQFP	48
MSP430F148	48		Slope				硬件 2	√	√	3	64LQFP	48
MSP430F1481	48		12 位				硬件 2	√	√	3	64LQFP	48
MSP430F149	60		Slope				硬件 2	√	√	3	64LQFP	48
MSP430F1491	60		12 位	12 位	√		硬件 2	√	√	3	64LQFP	48
MSP430F155	16		12 位	12 位	√		硬件 1	√		3	64LQFP	48
MSP430F156	16		12 位	12 位	√		硬件 1	√		3	64LQFP	48
MSP430F157	32		12 位	12 位	√		硬件 1	√		3	64LQFP	48
MSP430F167	32		12 位	12 位	√		硬件 2	√	√	3	64LQFP	48
MSP430F169	60		12 位	12 位	√		硬件 2	√	√	3	64LQFP	48

说明：√表示该型号的芯片存在对应的功能。

根据 TI 公司 430 单片机的发展规划，今后将陆续推出性能更高、功能更强的 F5xx 系列单片机，这一系列的单片机运行速度可达 25～30 MIPS 以上，并具有更大的 Flash(128 KB)，以及更丰富的外设接口(CAN、USB 等)。

MSP430 系列单片机不仅可以应用于许多传统的单片机应用领域(如仪器仪表、自动控

制、消费品域),更适用于一些电池供电的低功耗的产品,如能量表(水表、电表、气表等)、手持式设备、智能传感器等,以及需要较高运算性能的智能仪器设备。

1.4 MSP430F1xx 系列单片机时钟模块

1.4.1 时钟模块结构

MSP430F1xx 系列单片机时钟模块相对于其他单片机来说比较复杂,而且在以后的编程中也有用到,所以这里要介绍一下。图 1-2 所示为 MSP430F1xx 系列单片机的时钟模块结构图。这里要说明一点,其中,MSP430F11x 和 MSP430F12x 内部没有 XT2 高频振荡器。

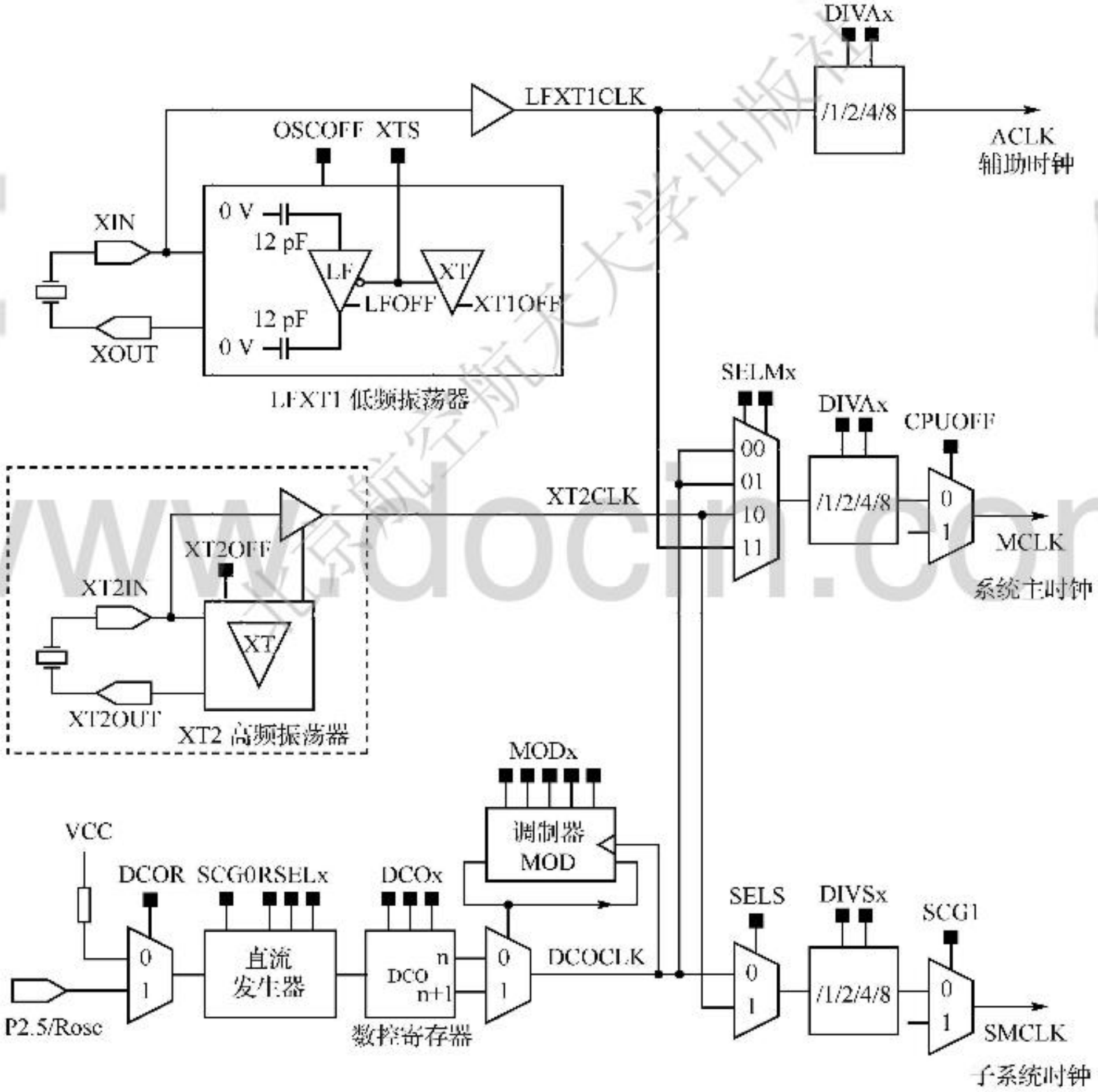


图 1-2 MSP430F1xx 系列单片机的时钟模块结构

从图 1-2 可以看出，MSP430F1xx 基础时钟模块有 3 个时钟输入源，分别是 LFXT1CLK（低频时钟源）、XT2CLK（高频时钟源）和 DCOCLK（数字控制 RC 振荡器）。

基础时钟模块可提供 3 种时钟信号，分别是 ACLK、MCLK 和 SMCLK。

- ACLK（辅助时钟）是 LFXT1CLK 信号经 1、2、4、8 分频得到的。ACLK 可由软件选作各个模块的时钟信号，一般用于低速外设。
- MCLK（系统时钟）可由软件选择，来自 LFXT1CLK、XT2CLK、DCOCLK 三者之一，然后经 1、2、4、8 分频得到。MCLK 主要用于 CPU 和系统。
- SMCLK（子系统时钟）可由软件选择，来自 LFXT1CLK 和 DCOCLK，或者 XT2CLK 和 DCOCLK（由具体器件决定），然后经 1、2、4、8 分频得到。SMCLK 主要用于高速外围模块。

1.4.2 低速晶体振荡器

MSP430 的每一种器件都包含有低速晶体振荡器（LFXT1CLK），低速晶体振荡器满足了低功耗及 32 768 Hz 晶体运行的要求。晶体只需通过 XIN 和 XOUT 两个引脚连接再加两个 30 pF 的电容即可，两个因素决定了选择人们熟悉和广泛使用的手表晶振，即低功耗的振荡器和时基，并且廉价。

在发生有效的 PUC 信号后 LFXT1CLK 振荡器开始工作，一次有效的 PUC 信号可以将 SR 寄存器（状态寄存器）中的 OSCOFF 位复位，即只允许 LFXT1CLK 工作。

LFXT1CLK 振荡器的默认状态工作在低频模式，或 32 768 Hz 的状态上，也可以通过外接 0.45～8 MHz 的高速晶体振荡器（或陶瓷谐振器）工作在高频模式，这时两个引脚还要外接电容，电容大小根据晶体或谐振器特性来选择。

如果 LFXT1CLK 信号没有用做 SMCLK 或 MCLK 信号，可以用软件将 OSCOFF 位置位以禁止 LFXT1CLK 工作。

LFXT1CLK 振荡器的逻辑控制如图 1-3 所示。

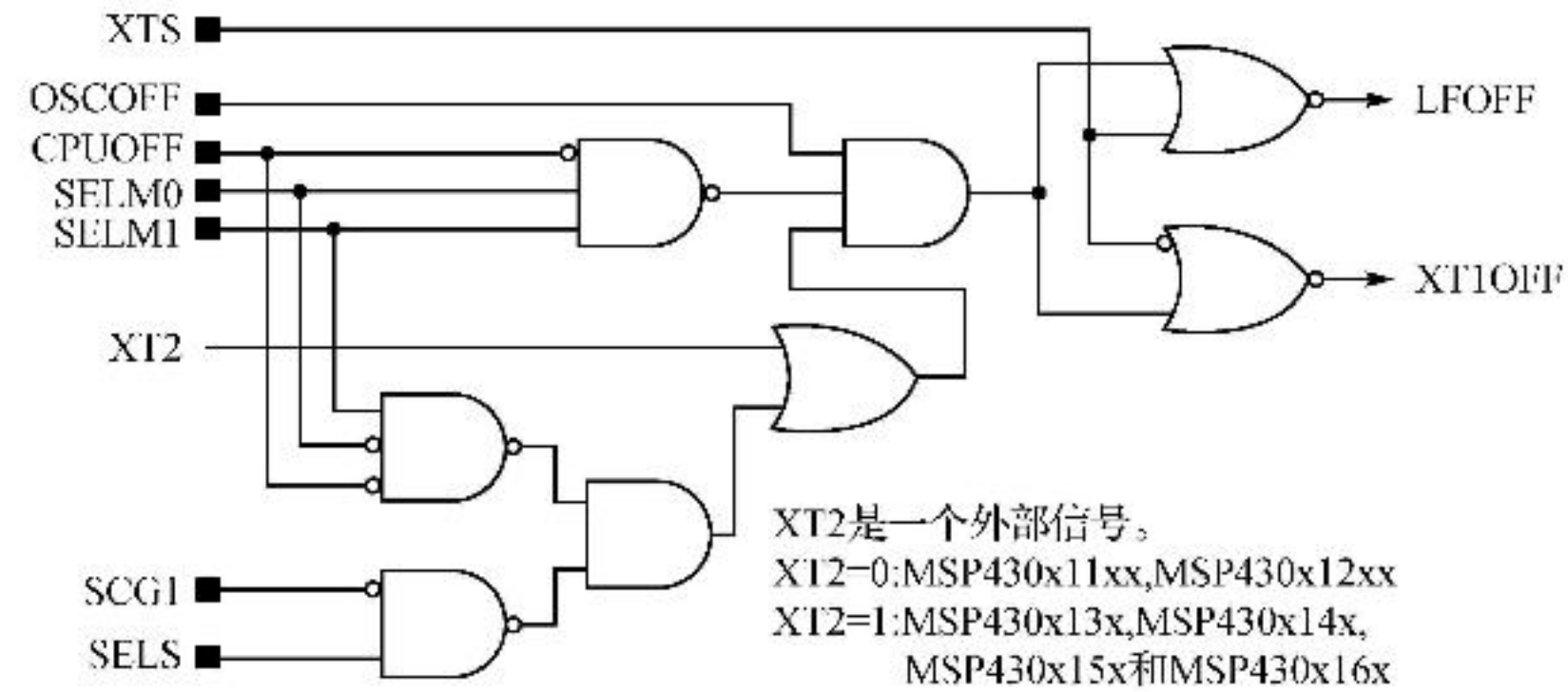


图 1-3 LFXT1CLK 振荡器逻辑结构图

1.4.3 高速晶体振荡器

高速振荡器主要存在于 x13x、x14x、x15x、x16x、x43x、x44x 等器件中，一般称为第二振荡器 XT2。它产生时钟信号 XT2CLK，它的工作特性与 LFXT1CLK 振荡器工作在高频模式时类似。如果 XT2CLK 信号没有用做 MCLK 和 SMCLK 时钟信号，可用控制位 XT2OFF 关闭 XT2。XT2 的控制逻辑结构如图 1-4 所示。

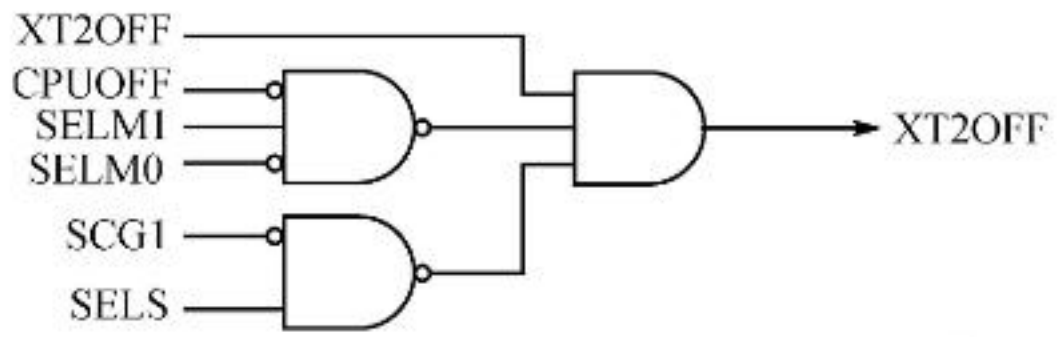


图 1-4　XT2 的控制逻辑结构图

MSP430 外接高频振荡器 XT2 的频率可以为 450 kHz～8 MHz。系统频率和系统的工作电压密切相关，某些应用需要较高的工作电压，所以也需要系统提供相应较高的频率。

1.4.4 DCO 振荡器

MSP430 单片机的两个外部振荡器产生的时钟信号经 1、2、4、8 分频后作为系统主时钟 MCLK。当振荡器失效后，DCO 振荡器会自动被选为 MCLK 的时钟源，因此由振荡器失效引起的 NMI 中断请求可以得到响应，甚至在单片机关闭的情况下也能得到处理。MSP430 可以让任意被允许的中断请求在低功耗模式下得到服务，甚至在 LPM4 模式下(所有振荡器停止工作，CPU、MCLK、SMCLK、ACLK 处于禁止状态)。MCLK 在中断服务时自动有效。

DCO 振荡器是一个可数字控制的 RC 振荡器，它的频率随供电电压、环境温度变化而具有一定的不稳定性。MSP430 可以通过操作寄存器软件调节来增强振荡器频率的稳定性。当 DCO 信号没有用做 SMCLK 和 MCLK 时钟信号时，可以用控制位 SCG0 关闭直流发生器，直流发生器消耗的电流定义了 DCOCLK 的基本频率。DCO 的控制逻辑如图 1-5 所示。

这里以 MSP430F1xx 为例，详细介绍如何控制 MSP430 的 DCOCLK 频率。在 MSP430F1xx 系列中，时钟模块的控制由 DCOCTL、BCSCTL1 及 BCSCTL2 这 3 个寄存器来完成，如表 1-3 所列。

表 1-3　时钟模块的 3 个寄存器

寄存器	缩写形式	类　型	地　址	初始状态
DCO 控制寄存器	DCOCTL	读/写	56H	60H
基本时钟系统控制寄存器 1	BCSCTL1	读/写	57H	84H
基本时钟系统控制寄存器 2	BCSCTL2	读/写	58H	复位

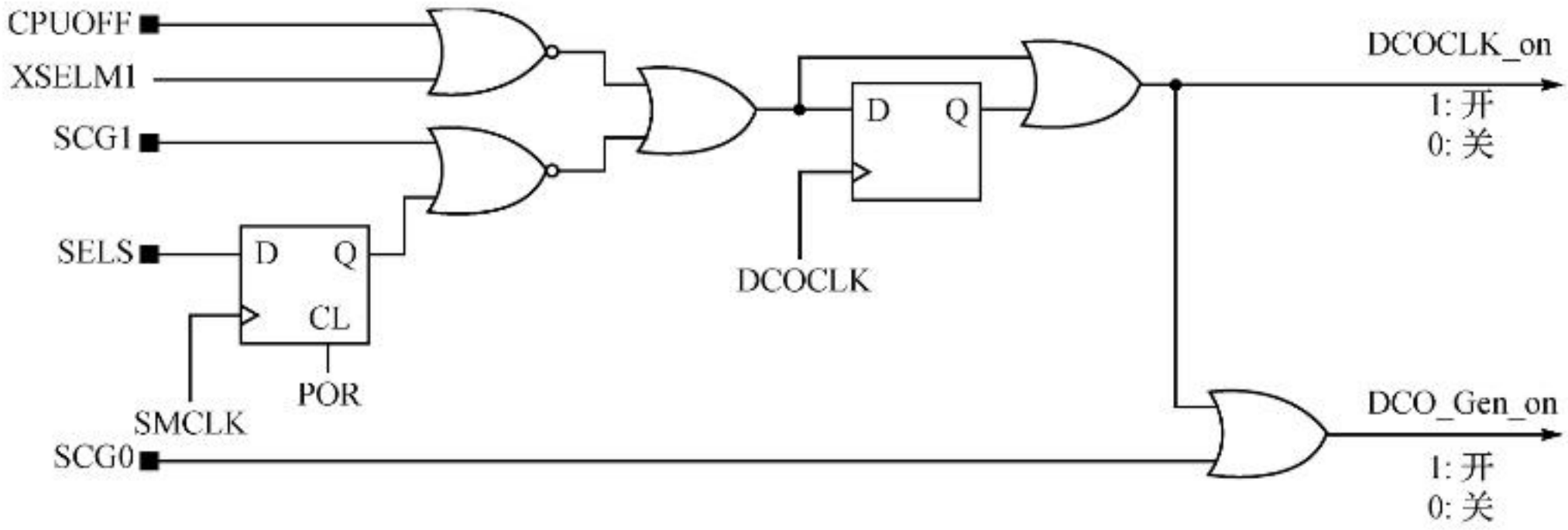


图 1-5　DCO 控制逻辑结构

1. DCO 控制寄存器

DCO 控制寄存器 DCOCTL 各位定义如表 1-4 所列。

表 1-4　DCO 控制寄存器 DCOCTL

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
DCO. 2	DCO. 1	DCO. 0	MOD. 4	MOD. 3	MOD. 2	MOD. 1	MOD. 0

DCO. 0～DCO. 2　定义 8 种频率之一，可分段调节 DCOCLK 频率，相邻两种频差为 10%。而频率由注入直流发生器的电流定义。

MOD. 0～MOD. 4　定义在 32 个 DCO 周期中插入的 $f_{DCO}+1$ 周期个数，而在余下的 DCO 周期中位 f_{DCO}周期，控制切换 DCO 和 DCO+1 选择的两种频率。如果 DCO 的常数为 7，表示已经选择了最高频率，此时 MOD. 0～MOD. 4 不能进行频率的调整。

2. 基本时钟系统控制寄存器 1

基本时钟系统控制寄存器 BCSCTL1 各位定义如表 1-5 所列。

表 1-5　基本时钟系统控制寄存器 BCSCTL1

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
XT2OFF	XTS	DIVA. 1	DIVA. 0	XT5V	RSEL. 2	RSEL. 1	RSEL. 0

XT2OFF　控制 XT2 振荡器的开启与关闭。

- XT2OFF=0，XT2 振荡器开启；
- XT2OFF=1，XT2 振荡器关闭。

XTS　控制 LFXT1CLK 的工作模式，选择需结合实际晶体的连接情况。

- XTS=0，LFXT1CLK 工作在低频模式(默认低频模式)；

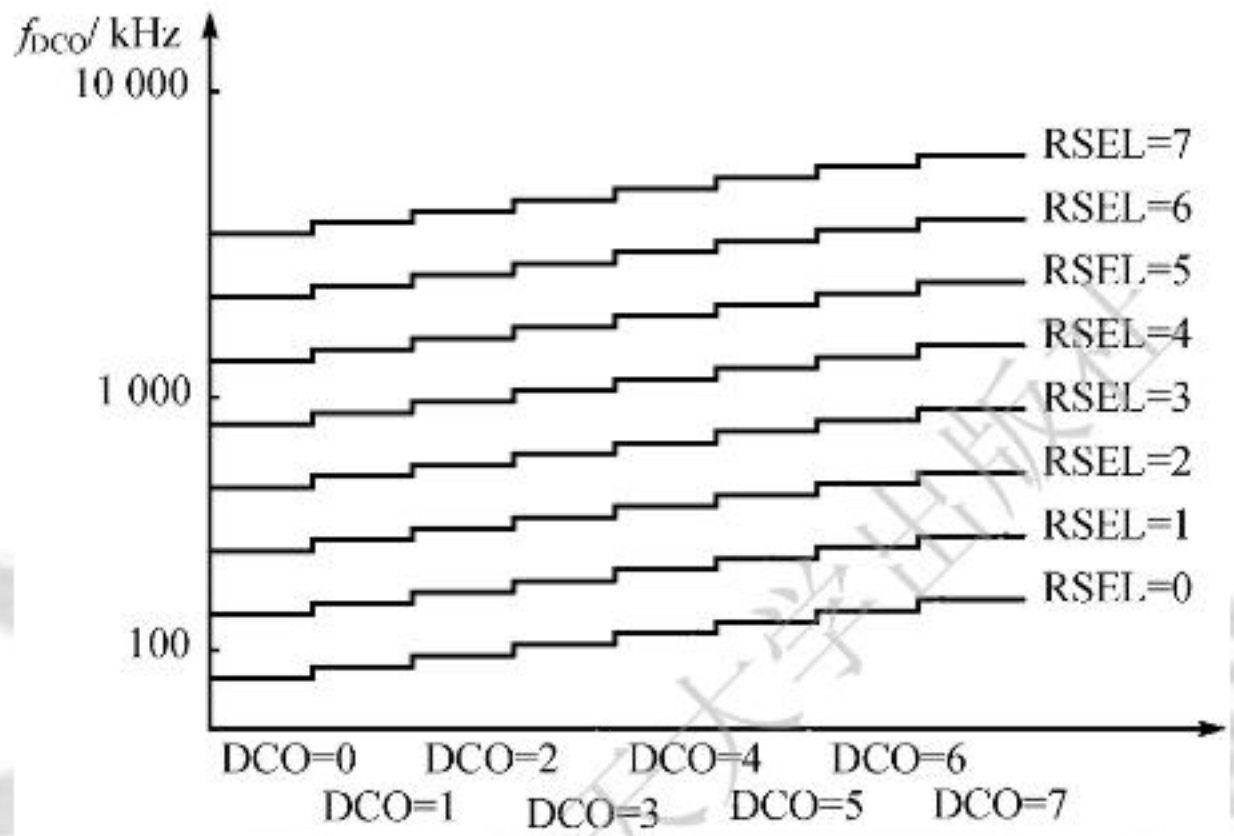
• XTS=1,LFXT1CLK 工作在高频模式(必须连接相应的高频晶振才行)。

DIVA.1 和 DIVA.0　控制 ACLK 分频关系。00 表示不分频;01 表示 2 分频;10 表示 4 分频;11 表示 8 分频。

XT5V　此位设置为 0。

RSEL.2、RSEL.1 和 RSEL.0　三个控制某个内部电阻以决定标称频率。000 表示选择最低的标称频率;111 表示选择最高的标称频率。

通过控制位 RSEL.2～RSEL.0 和 DCO.2～DCO.0 调节 DCO 的频率,如图 1-6 所示。



注:RSEL 代表 RSEL.2、RSEL.1、RSEL.0 的组合;DCO 代表 DCO.2、DCO.1、DCO.0 的组合。

图 1-6　DCO 频率的调节

3. 基本时钟系统控制寄存器 2

基本时钟系统控制寄存器 BCSCTL2 各位定义如表 1-6 所列。

表 1-6　基本时钟系统控制寄存器 BCSCTL2

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
SELM.1	SELM.0	DIVM.1	DIVM.0	SELS	DIVS.1	DIVS.0	DCOR

SELM.1 和 SELM.0　选择 MCLK 时钟源。

- 00:时钟源为 DCOCLK(默认时钟源);
- 01:时钟源为 DCOCLK;
- 10:对于 MSP430F11/12x,时钟源为 LFXT1CLK;对于 MSP430F13/14/15/16x,时钟源为 XT2CLK;
- 11:时钟源为 LFXTICLK。

DIVM.1 和 DIVM.0　选择 MCLK 分频。

- 00：1 分频(默认 MCLK＝DCOCLK)；
- 01：2 分频；
- 10：4 分频；
- 11：8 分频。

SELS　选择 SMCLK 时钟源。默认时钟源为 DCOCLK；对于 MSP430F11/12x，时钟源为 LFXT1CLK；对于 MSP430F13/14/15/16x，时钟源为 XT2CLK。

DIVS.1 和 DIVS.0　选择 SMCLK 分频。

- 01：分频(默认 SMCLK＝MCLK)；
- 01：2 分频；
- 10：4 分频；
- 11：8 分频。

DCOR　选择 DCO 电阻，内部电阻或外部电阻。

PUC 信号之后，DCOCLK 被自动选作 MCLK 时钟信号，根据需要，MCLK 的时钟源可以另外设置为 LFXT1 或者 XT2，设置顺序如下：

① 复位 OSCOFF；

② 清除 OFIFG；

③ 延时等待至少 50 μs；

④ 再次检查 OFIFG，如果仍然复位，则重复③、④步骤，直到 OFIFG＝0 为止。

4. 时钟模块应用举例

(1) 基础时钟模块工作方式和相关寄存器设置(程序见光盘：源程序\第 1 章)

例 1-1　设 MCLK＝XT2，SMCLK＝DCOCLK，将 MCLK 由 P5.4 输出。(注：MSP430x14x 中引脚 P5.4 和 MCLK 复用)

实现上述功能的程序如下：

```
#include <msp430x16x.h>
void main(void)
{
  unsigned int i;
  WDTCTL = WDTPW + WDTHOLD;                    //停止看门狗
  P5DIR |= 0x10;                               //P5.4 输出
  P5SEL |= 0x10;                               //P5.4 用做 MCLK 输出
  BCSCTL1 &= ~XT2OFF;                          //XT2 有效
  do
  {
IFG1 &= ~OFIFG;                                //清除振荡器失效标志
```

```
for (i = 0xFF; i > 0; i--);                    //稳定时间
  }
while ((IFG1 & OFIFG) != 0);                   //如果振荡器失效标志存在
BCSCTL2 |= SELM1;                              //MCLK = XT2
    for (;;);
   }
```

例 1-2　设 ACLK=MCLK=LFXT1=HF,将 MCLK 由 P5.4 输出。

实现上述功能的程序如下：

```
#include <msp430x16x.h>
void main(void)
{
  unsigned int i;
  WDTCTL = WDTPW + WDTHOLD;                    //停止看门狗
  P5DIR |= 0x10;                               //P5.4 输出
  P5SEL |= 0x10;                               //P5.4 用做 MCLK 输出
  BCSCTL1 |= XTS;                              //ACLK = LFXT1 = HF 模式
  do
  {
IFG1 &= ~OFIFG;                                //清除振荡器失效标志
for (i = 0xFF; i > 0; i--);                    //稳定时间
  }
while ((IFG1 & OFIFG) != 0);                   //如果振荡器失效标志存在
BCSCTL2 |= SELM1 + SELM0;                      //MCLK = LFXT1
   for (;;);
}
```

(2) 根据实际连接情况确定 ACLK、SMCLK 和 MCLK 时钟源

MSP430x1xx 系列单片机具有不同组合的时钟模块,如表 1-7 所列。

表 1-7　MSP430x1xx 系列单片机不同组合的时钟模块

MSP430	低速晶体	DCO	高速晶体	FLL+
MSP430x11x1	√	√		
MSP430F12x	√	√		
MSP430F13x/14x/15x/16x	√	√	√	
MSP430F4xx	√	√	√	√

1.5 MSP430 - DEMO16x 开发实验板

MSP430 系列单片机是美国德州仪器(TI)公司 1966 年开始推向市场的一种 16 位超低功耗的混合信号处理器(mixed signal processor)。称之为混合信号处理器,主要是其针对实际应用需要,把许多模拟电路、数字电路和一片微处理器集成在一个芯片上,以提供“单片”解决方案。虽然 MSP430 系列单片机推出时间不是很长,但由于其卓越的性能,在短短几年时间里发展极为迅速,应用也日趋广泛。MSP430 系列单片机针对各种不同应用,包括一系列不同型号的器件。

为了更好地开展大学中 MSP430 系列单片机的教学、实验及毕业设计,提高 MSP430 系列单片机的开发速度,TI 公司在中国的技术合作伙伴和 MSP430 单片机首家增值经销商——利尔达单片机技术有限公司,设计了功能齐全的 MSP430 - DEMO16x 开发实验板。另外,单片机的所有引脚都已经引出,便于学生进行扩展试验,并对实验的原理、实验环境配置和源程序都进行了详细的说明。这些实验都经过了作者的实际验证,都是正确无误的,大家可以放心使用。

1.5.1 开发实验板主要实验内容概要

下面介绍 MSP430 - DEMO16x 开发试验板,本实验板以 MSP430F169 为核心,其外观如图 1 - 7 所示。

MSP430 - DEMO16x 开发实验板集成了丰富的接口电路与模块,具体如下:

- 字符型 1602 LCD 一块,点阵型 12864 LCD(可显示汉字)一块。
- MSP430F169 一片。
- MAX7219 控制的 LED 数码管 8 个,蜂鸣器一个。
- 发光二极管 8 个。
- EEPROM 24C02 一片。
- A/D 转换接口(12 位)6 个。
- 4×4 行列式键盘、独立按键式键盘一个。
- DDS 芯片 AD9850 及其外围元件一套。
- 实时时钟芯片 DS1302 一个。
- DS18B20 温度传感器一个。
- AC 220 V 继电器一个。
- D/A 转换通道两个。
- MSP430 JTAG 接口一个。

MSP430 - DEMO16x 开发实验板还把 MSP430F169 全部引脚用插座引出,并可通过 DIP

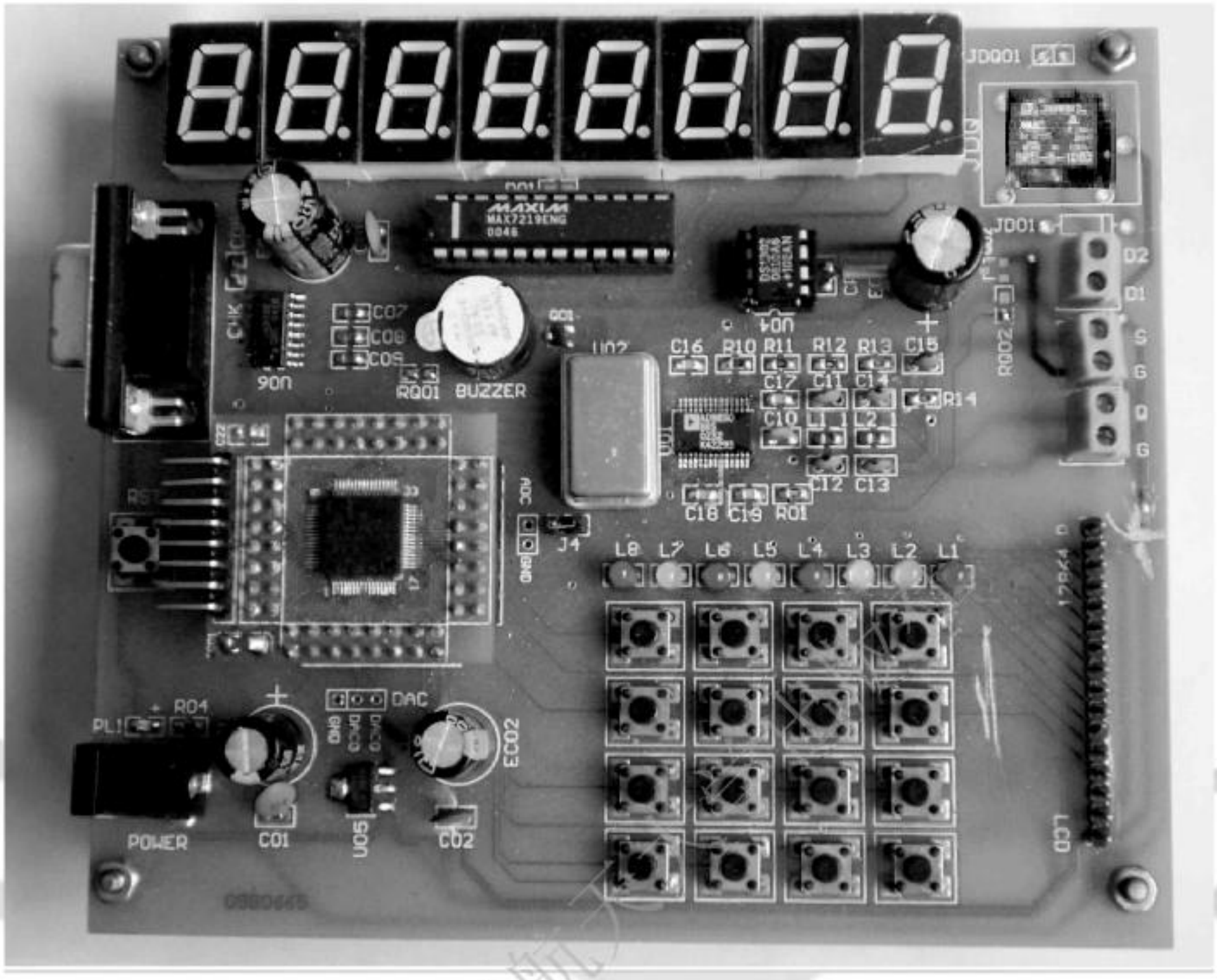


图 1-7　MSP430-DEMO16x 开发试验板照片

开关选择是否与外部电路相连，方便使用者自行扩展模块与单片机相连。开关闭合，单片机与实验板上电路连接；开关断开，单片机与试验班上外围电路断开。使用者可以方便地利用板上引出的引脚与自己的外围电路实现二次开发。

1.5.2　MSP430F169 特点与结构

MSP430-DEMO16x 开发实验板以 MSP430F169 为核心，MSP430F169 芯片是 MSP430 芯片中功能较多的一款 MCU。

1. 特　点

MSP430F169 的特点如下：

① 低工作电压：1.8～3.6 V。

② 超低功耗：活动模式 280 mA(1 MHz,2.2 V)，待机模式 1.1 mA，掉电模式(RAM 数据保持)0.1 mA。

③ 5 种节电模式。

④ 从待机到唤醒不到 6 μs。

⑤ 12 位 ADC 带有内部参考源、采样保持和自动扫描特性。

⑥ 16 位精简指令结构(RISC),150 ns 指令周期。

⑦ 带有 3 个捕获/比较器的 16 位定时器,即定时器 A 和定时器 B。

⑧ 2 个串行通信模块 USART0/1,可软件选择 UART/SPI 模式 。

⑨ 片内比较器配合其他器件可构成单斜边 ADC。

⑩ 可编程电压监测器。

⑪ 线串行编程,不需要外部编程电压。

⑫ 驱动液晶能力可达 160 段。

⑬ 可编程的保密熔丝可保护设计者代码。

⑭ Flash 存储器多达 60 KB,RAM 多达 2 KB。

2. 引脚定义

MSP430F16x 系列器件的封装形式为 64 引脚的 PLASTIC 64 - PIN QFP。其引脚说明如表 1 - 8 所列。

表 1 - 8　MSP430F16x 引脚说明

引脚名称	引脚号	端　口	说　明
AVCC	64		模拟供电电源正端,只为 ADC 和 DAC 的模拟部分供电
AVss	62		模拟供电电源负端,只为 ADC 和 DAC 的模拟部分供电
DVCC	1		数字供电电源正端,为所有数字部分供电
DVss	63		数字供电电源负端,为所有数字部分供电
P1.0/TACLK	12	I/O	通用数字 I/O 引脚/定时器 A;时钟信号 TACLK 输入
P1.1/TA0	13	I/O	通用数字 I/O 引脚/定时器 A;捕获：CCI0A 输入;比较：OUT0 输出
P1.2/TA1	14	I/O	通用数字 I/O 引脚/定时器 A;捕获：CCI1A 输入;比较：OUT1 输出
P1.3/TA2	15	I/O	通用数字 I/O 引脚/定时器 A;捕获：CCI2A 输入;比较：OUT2 输出
P1.4/SMCLK	16	I/O	通用数字 I/O 引脚/SMCLK 信号输出
P1.5/TA0	17	I/O	通用数字 I/O 引脚/定时器 A;比较：OUT0 输出
P1.6/TA1	18	I/O	通用数字 I/O 引脚/定时器 A;比较：OUT1 输出
P1.7/TA2	19	I/O	通用数字 I/O 引脚/定时器 A;比较：OUT2 输出
P2.0/ACLK	20	I/O	通用数字 I/O 引脚/ACLK 输出
P2.1/TAINCLK	21	I/O	通用数字 I/O 引脚/定时器 A;INCLK 上的时钟信号
P2.2/CAOUT/TA0	22	I/O	通用数字 I/O 引脚/定时器 A;捕获：CCI0B 输入/比较器输出
P2.3/CA0/TA1	23	I/O	通用数字 I/O 引脚/定时器 A;比较：OUT1 输出/比较器 A 输入

续表 1-8

引脚名称	引脚号	端　口	说　明
P2.4/CA1/TA2	24	I/O	通用数字 I/O 引脚/定时器 A;比较：OUT2 输出/比较器 A 输入
P2.5/Rosc	25	I/O	通用数字 I/O 引脚,定义 DCO 标称频率的外部电阻输入
P2.6/ADC12CLK	26	I/O	通用数字 I/O 引脚,转换时钟－12 位 ADC,DMA 通道 0 外部触发器
P2.7/TA0	27	I/O	通用数字 I/O 引脚/定时器 A,比较：OUT0 输出
P3.0/STE0	28	I/O	通用数字 I/O 引脚,USART0/SPI 模式从设备传输使能端
P3.1/SIMO0/SDA	29	I/O	通用数字 I/O 引脚,USART0/SPI 模式的从入/主出,IIC 数据
P3.2/SIMI0	30	I/O	通用数字 I/O 引脚,USART0/SPI 模式的从出/主入
P3.3/UCLK0/SCL	31	I/O	通用数字 I/O 引脚,USART0/SPI 模式的外部时钟输入,IIC 时钟输出
P3.4/UTXD0	32	I/O	通用数字 I/O 引脚,USART0/SPI 模式的传输数据输出
P3.5/URXD0	33	I/O	通用数字 I/O 引脚,USART0/SPI 模式的接收数据输入
P3.6/UTXD1	34	I/O	通用数字 I/O 引脚,USl1/UART 模式的发送数据输出
P3.7/URXD1	35	I/O	通用数字 I/O 引脚,USl1/UART 模式的接收数据输入
P4.0/TB0	36	I/O	通用数字 I/O 引脚,捕获 I/P 或者 PWM 输出端口_定时器 B7 CCR0
P4.1/TB1	37	I/O	通用数字 I/O 引脚,捕获 I/P 或者 PWM 输出端口_定时器 B7 CCR1
P4.2/TB2	38	I/O	通用数字 I/O 引脚,捕获 I/P 或者 PWM 输出端口_定时器 B7 CCR2
P4.3/TB3	39	I/O	通用数字 I/O 引脚,捕获 I/P 或者 PWM 输出端口_定时器 B7 CCR3
P4.4/TB4	40	I/O	通用数字 I/O 引脚,捕获 I/P 或者 PWM 输出端口_定时器 B7 CCR4
P4.5/TB5	41	I/O	通用数字 I/O 引脚,捕获 I/P 或者 PWM 输出端口_定时器 B7 CCR5
P4.6/TB6	42	I/O	通用数字 I/O 引脚,捕获 I/P 或者 PWM 输出端口_定时器 B7 CCR6
P4.7/TBCLK	43	I/O	通用数字 I/O 引脚,输入时钟 TBCLK_定时器 B7
P5.0/STE1	44	I/O	通用数字 I/O 引脚,USART1/SPI 模式从设备传输使能端
P5.1/SIMO1	45	I/O	通用数字 I/O 引脚,USART1/SPI 模式的从入/主出
P5.2/SOMI1	46	I/O	通用数字 I/O 引脚,USART1/SPI 模式的从出/主入
P5.3/UCLK1	47	I/O	通用数字 I/O 引脚,USART1/SPI 模式的外部时钟输入,USART0/SPI 模式的时钟输出
P5.4/MCLK	48	I/O	通用数字 I/O 引脚,主系统时钟 MCLK 输出
P5.5/SMCLK	49	I/O	通用数字 I/O 引脚,子系统时钟 SMCLK 输出
P5.6/ACLK	50	I/O	通用数字 I/O 引脚,辅助时钟 ACLK 输出
P5.7/TboutH/SVSOUT	51	I/O	通用数字 I/O 引脚,将所有 PWM 数字输出端口为高阻态－定时器 B7,SVS 比较输出
P6.0/A0	59	I/O	通用数字 I/O 引脚,模拟量输入 A0_12 位 ADC

续表 1-8

引脚名称	引脚号	端　口	说　明
P6.1/A1	60	I/O	通用数字 I/O 引脚,模拟量输入 A1_12 位 ADC
P6.0/A2	61	I/O	通用数字 I/O 引脚,模拟量输入 A2_12 位 ADC
P6.0/A3	2	I/O	通用数字 I/O 引脚,模拟量输入 A3_12 位 ADC
P6.0/A4	3	I/O	通用数字 I/O 引脚,模拟量输入 A4_12 位 ADC
P6.0/A5	4	I/O	通用数字 I/O 引脚,模拟量输入 A5_12 位 ADC
P6.0/A6/DAC0	5	I/O	通用数字 I/O 引脚,模拟量输入 A6_12 位 ADC,DAC.0 输出
P6.0/A7/DAC1	6	I/O	通用数字 I/O 引脚,模拟量输入 A7_12 位 ADC,DAC.1 输出,SVS 输入
RST/NMI	58	I	复位输入,非屏蔽中断输入
TCK	57	I	测试时钟,TCK 是芯片编程测试和 Bootstrap loader 启动的时钟输入端口
TDI	55	I	测试数据输入,TDI 用做数据输入端口,芯片保护熔丝连接到 TDI
TDO/TDI	54	I/O	测试数据输出端口,TDO/TDI 数据输出或者编程数据输出引脚
TMS	56	I	测试模块选择,TMS 用做芯片编程和测试的输入端口
VeREF+	10	I/P	外部参考电压的输入
VREF+	7	O	内部参考电压的正输出引脚
VREF-/VeREF-	11	O	内部参考电压或者外加参考电压的引脚
XIN	8	I	晶体振荡器 XT1 的输入端口,可连接标准晶振或者钟表晶振
XOUT/TCLK	9	I/O	晶体振荡器 XT1 的输出引脚或测试时钟输入
XT2IN	53	I	晶体振荡器 XT2 的输入端口,只能连接标准晶振
XT2OUT	52	O	晶体振荡器 XT2 的输出引脚

1.6　自学思考题

1. MSP430 系列单片机具备哪些独特的优点?
2. MSP430 系列单片机有几种时钟源? 各有何特点?
3. MSP430 系列单片机的低速晶体、高速晶体的频率是多少? 是怎样连接的?
4. MSP430F169 单片机片内外设有哪些主要的模块?

第2章

MSP430系列单片机C语言基础知识

本章主要介绍C语言，在对C语言有一定了解的基础上才能完成MSP430系列单片机程序的编写。MSP430系列单片机是一种16位的单片机，相对于8位的51单片机来说，它具有功能丰富、较大的内部RAM和程序存储空间，适合开发比较复杂的系统。除此之外，C语言功能全面，可以非常方便地完成较为复杂的控制与计算功能，是学习、开发项目的首选程序设计语言。采用C语言开发主要有以下优点：相对于汇编语言来讲，C语言没有汇编那么繁琐；当程序较大时，相对于汇编语言来说，C语言更容易编写；C语言具有较好的可读性，从而可以大大提高软件开发的工作效率。本章用相对较短的篇幅在本人自学的基础上介绍C语言程序设计的基本概念，同时也介绍一下MSP430系列单片机C语言的扩展特性，为后面的程序设计打下基础。

MSP430系列单片机支持标准的C语言，在标准的C语言基础上进行了扩展，因此掌握标准C语言对开发MSP430系列单片机有着非常重要的作用。下面针对MSP430系列单片机开发介绍一些C语言的开发基础。

2.1 标志符与关键字

1. 标志符

C语言中的标志符可以作为变量名、函数名、数组名、类型名及文件名。它可以是一个字符，也可以是多个字符。标志符必须以字母或者下划线开始，后面可以跟字母、数字或者下划线。例如，_Data、nIndex是正确的形式，而2Index、n%则是错误的形式(不能以数字或者特殊的字符开头)。在C语言中，标志符要求区分大小写，也就是说，大写和小写的标志符被当做不同的标志符(汇编语言一般不区分大小写，C语言区别大小写，这点大家要特别注意)。例如，Index和index在C语言里是两个不同的标志符，所以在这一点上需要特别引起注意，这就要求大家在写程序时要养成良好的习惯。

2. 关键字

关键字是一种含有特殊意义的标志符。关键字又称保留字。关键字在编译器中已经有了

定义，所以不能再重新定义，需要加以保留。用户在定义自己的变量或者函数的时候千万不要使用关键字，否则就会出现一些错误。在 C 语言的编译系统中主要有以下几种类型的关键字。

(1) 数据类型关键字

数据类型关键字包括 auto、char、const、double、enum、extern、float、int、long、register、sizeof、short、static、struct、typedef、union、unsigned、void、volitile 等。该类型关键字主要用于定义一些变量或者函数。以后编程中最常用的如 char、int 等，例如，“int nlndex;”语句就是定义一个 int 类型的数据。这里不同的关键字有不同的含义，需要在使用的时候加以区分，总结起来就是一句话：关键字不能当做普通的变量来使用。

(2) 程序控制关键字

程序控制关键字包括 break、case、continue、default、do、else、for、goto、if、return、switch、while 等。该类型的关键字主要用于程序的控制。例如：

```
int n = 9;                    //定义一个整型变量 n,并将其赋值为 9
int m;                        //定义一个整型变量 m,如果没有赋值,一般其值默认为 0
if(n<10)                      //上面已经将 n 赋值为 9 了,所以条件(n<10)成立
{
    m = 10;
}
else                          //n 大于或等于 10 时执行
{
    m = 11;
}
```

以上代码运行后的结果是 m=10。

(3) 预处理功能关键字

预处理功能关键字包括 define、end、if、include 等。该类型的关键字主要用于进行预处理，这些实际上用得不多。例如，“#include <msp430x16x.h>”表示包括 msp430x16x.h 头文件，这个头文件在以后编程中还会遇到。说明一点，不要看这么多的关键字，实际上有些关键字以后根本就用不到，真正用到的就那么十几个而已。

2.2 数据基本类型

标准 C 语言中数据主要有整型、实型和字符型。下面就这几种类型进行具体介绍。

1. 整型数据

这是一种比较常用的数据类型，整型数据中主要包括 int、short、long(三者之间的区别见

表 2-1)等。不同整数类型的变量具有不同的整数范围。MSP430 的 C 语言除了支持标准的 C 语言整数类型外,还支持其他几种整数类型。表 2-1 给出了 MSP430 系列单片机的 C 语言支持的几种整数类型,这张表格中每一种数据的类型所表示的数据范围是值得注意的,如果在进行计算的时候超出了数制所能表示的范围而自己还没有意识到,就要出大问题了!

表 2-1　MSP430 系列单片机的 C 语言支持的整数类型

整数类型	字　节	数值范围	说　明
sfrb	1		定义特殊功能寄存器
sfrw	2		定义特殊功能寄存器
unsigned char	1	0～255	无符号字符
char(默认)	1	0～255	等效于 unsigned char
signed char	1	－128～127	有符号字符
char (－c 选项)－	1	－128～127	等效于 signed char
short	2	－32768～32767	短整数
int	2	－32768～32767	整数
unsigned short	2	0～65535	无符号短整数
unsigned int	2	0～65535	无符号整数
long	4	－2147483648～2147483647	长整数
unsigned long	4	0～4294967295	无符号长整数
float	4	±1.18E－38～±3.39E＋38	浮点数
double	4	±1.18E－38～±3.39E＋38	双精度浮点数
long double	4	±1.18E－38～±3.39E＋38	长双精度浮点数
pointer	2		指针
enum	1～4		枚举

整型变量的定义如下:

```
main()                          //主程序开始
{
    int sum,n,m;
//定义三个整型变量 sum、n、m,这里要注意,每个整型变量占用两个存储空间,这两个存储空间都在内部
//RAM 单元中,由编译软件分配
    n = 12;                     //将整型变量赋值为 12
    m = 0x15;                   //将整型变量赋值为 0x15;0x 是十六进制表示形式,0x15 化成十进制是 21
    sum = n + m;                //将 n、m 进行求和
```

```
}
```

上面的例子给出了整型变量的定义方法。其他两种变量定义也可以参考这种方法。另外,在上面的例子中,整数有常用的两种形式:十进制和十六进制。在数值的前面加上 0x 就表示十六进制数了。

2. 实型数据

实型数据就是 C 语言中的浮点数据。它可以含有小数点,但是它表示的数是有精度的。实型数据有两种具体的表现形式:十进制小数点形式和指数形式。小数点形式如 10.1234。指数形式包括整数部分、尾数部分和指数部分,具体形式如 1.21E+1(十进制的值 1.21×10^1)。实型变量主要有 float 型、double 型和 long double 型。实型变量的定义方法很简单。例如:

```
float a;                    //定义一个 float 型的实型数据 a
double b;                   //定义一个 double 型的实型数据 b
```

3. 字符型数据

字符型数据主要处理与字符相关的内容,比如英文字母或者汉语句子。一般来说,会将多个字符型变量组成一个字符串来使用。在 C 语言中,字符是与 ASCII 码的值对应的,一个字符占一个字节,例如,英文字母 A 的 ASCII 码值是 41H。字符的 ASCII 码值可以在码表中查找到。在这里需要强调一下,整数和字符常量在表现形式上是有区别的,比如加单引号的'5'表示的是字符,而 5 表示的是整数;所以,字符是以单引号表示的,而单引号的输入要在英文状态下才有效,不能在汉字的状态下输入。

字符变量主要包括 char 型的字符变量,字符变量的定义方法非常简单。例如:

```
char chrTemp;
chrTemp = 'A';
```

上面的例子定义了字符变量 chrTemp,并给它赋值为'A'。在使用字符变量时,需要了解转义字符。转义字符是一种特殊的字符,通常使用转义符表示 ASCII 码字符中不可打印的控制字符和特殊功能字符。转义字符是使用反斜杠(\)后面跟一个字符来表示。例如,"\n"表示回车,"\'"表示单引号。当然这些所谓的转义字符一般用到的较少,用到时再查也来得及。在某些应用场合需要不同数据类型的转换,如把字符类型的变量转换成整数类型,把 int 类型的数据转换成 long 类型的数据等。在这里,进行数据类型转换的方法是采用强制转换类型。下面例子给出的是数据之间的转换。

```
main()
{
    char chrTemp;
    int n;
```

```
    chrTemp = 'A';
    n = (int)(chrTemp);      //强制转换类型,将字符型数据强制转换成整型数据
while(1);                    //等待或执行其他程序
}
```

上面的例子是从字符类型转换成 int 类型,n 的最终值为 65。需要注意的是:并不是所有类型之间都可以进行类型转换;当占字节多的类型转换成占字节少的类型时,有可能会造成数据的丢失。基于以上原因,使用类型转换的时候需要特别小心。

2.3 C 语言的运算符

C 语言的内部运算比较丰富,比如可以直接进行加、减、乘、除等运算而不再像汇编语言那么繁琐。C 语言的运算主要包括算术运算、关系运算、逻辑运算、赋值运算和位运算。下面简要介绍各种具体的运算。

1. 算术运算

算术运算主要是指加、减、乘、除等运算。表 2-2 给出了算术运算符。

表 2-2 算术运算符

运算符	含 义	说 明
++	单目加(加 1 操作)	只有一个操作数
—	单目减(减 1 操作)	只有一个操作数
+	加	需要两个操作数
-	减	需要两个操作数
*	乘	需要两个操作数
/	除	当两个操作数是整数的时候,结果为整数
%	模运算(求余)	操作数必须是整数

下面给出算术运算的例子。

```
main()
{
    int n,m;
    int y;
    n = 5;
    m = 2;
    y = m + n;                  //y 的值为 7
```

```
    y = m * n;                    //y 的值为 10
    y = m / n;                    //y 的值为 2
    y = m % n;                    //y 的值为 1
    n++ ;                         //执行这句代码后,n 的值为 6
}
```

上面的例子比较简单,复杂的运算也是同样如此。

2. 关系运算

关系运算符主要是对操作数进行某种逻辑上的判断,结果只有 true(结果为真)和 false(结果为假)两种结果。表 2-3 给出了关系运算符。

表 2-3 关系运算符

关系运算符	含 义	举例(设 a=3,b=4)
>	大于	a>b 结果：false
>=	大于或等于	a>b 结果：false
==	等于	a==b 结果：false
<	小于	a<b 结果：true
<=	小于或等于	a<=b 结果：true
!=	不等于	a!=b 结果：true

由表 2-3 可知,关系运算主要就是处理操作数之间的结果,这里有一点要说明,a==b 是关系运算,其结果有两种情况,即 true 或 false;a=b 是赋值运算,就是 b 的值赋给 a。这二者区别很大,大家一定要注意。

3. 逻辑运算

逻辑运算和关系运算比较相似,也是处理操作数之间的关系,结果只有 true 和 false 两种结果。表 2-4 给出了逻辑运算符。

表 2-4 逻辑运算符

逻辑运算符	含 义	举例(设 a=true,b=false)
&&	“与”运算	a&&b 结果：false
||	“或”运算	a||b 结果：true
!	“非”运算	! a 结果：false

4. 赋值运算

通常把“=”称为赋值运算符。该运算符是一个二元运算符,需要两个操作数,左边的操作

数是变量或者数组，右边的是表达式。例如：

```
int n,m;
m = 5;                    //赋值运算
n = m * m                 //赋值运算
```

另外，“=”还可以和其他的运算符结合起来使用。比如+=、-=、/=、%=等，它们的意义分别是：“x+=a;”等价于“x=x+a;”、“x-=a;”等价于“x=x-a;”、“x*=a;”等价于“x=x*a;”、“x/=a;”等价于“x=x/a;”、“x%=a;”等价于“x=x%a;”。

上面几种运算形式在以后的编程中会经常用到，大家要熟记。当然“=”还可以和“>>”等运算符结合起来使用，使用的含义和上面的含义相同，所以在此不再赘述。

5. 位运算

位运算在单片机的开发中非常重要，比如设置某个引脚的输出电平为高电平的操作就是通过位运算来实现的。位运算主要包括“与”(&)、“或”(|)、“反”(~)、“左移”(<<)和“右移”(>>)等运算。例如：

```
main()
{
  int m,n,k,result;
  m = 10;
  n = 13;
  k = 0x0a;
  result = m & n;              //result 的值为 8
  result = m | n;              //result 的值为 15
  result = ~(k);               //result 的值为 0xFA
  result = m << 2;             //result 的值为 40
  result = m >> 2;             //result 的值为 2
}
```

通过以上的例子就很容易理解位运算，说到这里，大家可能就要有疑问了，“&&”和“&”、“||”和“|”有什么区别呢？其实区别很大，首先，前者“&&”和“||”叫做关系运算，其结果只有true 或者 false 两种情况，true 就是 1，而 false 就是 0；“&” 和“|”等运算是位运算，其结果千变万化、五花八门，可以为 1、为 0，也可以为其他的数值。为了便于理解可以简单地认为关系运算是两个“事件”之间的运算，那么其结果就是 1 或 0 了，位运算是两个数据内部按照比特(位)的关系进行的运算，当然其结果就有可能出现多种情况了。

6. 运算的优先级

通过前面的介绍，读者应该对 C 语言的几种运算有了基本的了解。在实际应用中，一个

计算可能是上面的几种运算的组合，这样在进行运算的时候，执行的顺序就非常重要，这时就需要了解运算的优先级。

表 2-5 给出了运算的优先级，如果在有括号运算的时候，应该先运算括号里面的，再运算括号外面的表达式，而括号里面的表达式的运算优先级顺序应该参照表 2-5 所列。

表 2-5　运算的优先级

优先级	符　号	操作数个数
1	!,～,++,-	单操作数
2	*,/,%	双操作数
3	+,-	双操作数
4	<<,>>	双操作数
5	<,<=,>,>=,==,!=	双操作数
6	&	双操作数
7	|	双操作数
8	&&	双操作数
9	||	双操作数
10	=,+=,-=,*=,/=等	双操作数

2.4　函　数

在C语言中，函数是程序的基本组成单位。函数不仅可以实现程序的模块化，使程序设计得简洁和直观，提高程序的易读性和可维护性，而且还可以把程序中经常用到的一些计算或者操作做成通用的函数，以便随时调用。

函数定义的语法如下：

```
函数类型　函数名
{
语句 1;
......
语句 n;
返回;
}
```

函数类型确定了函数返回值的类型。函数的类型可以是任何一种有效的类型，比如整数类型，也可以是用户自定义的类型。

函数的参数表确定了函数的输入参数和输出参数，多个参数之间用逗号分开。

函数体主要由一系列的语句组成，语句的组成结构可以是顺序结构，也可以是分支结构或者循环结构。在函数体的最后，需要返回函数的值，如果函数定义成 void，可以不用返回值，其他类型必须返回函数值，并且返回函数值的类型必须和函数定义时函数的类型一致。下面举例来说明函数的定义。

```
memset(int a[10],n)
{
    int i;
    for(i = 0;i < 10;i ++ )
    {
        a[i] = n;
    }
    return;
}
```

在上面的例子中，return 可以省略，因为这个函数的返回类型是 void。这里的 void 没有出现在函数名 memset 前，所以可以不写，默认是 void 的。

```
int sum(int n,int m)
{
    int sum;
    sum = n + m;
    return sum;
}
```

在上面的例子中，return 不可以省略，需要将计算的结果返回。

1. 局部变量与全局变量

在引入了函数定义后，需要知道变量的作用域，就是变量的使用范围。根据变量的作用域可以将变量分为全局变量和局部变量。局部变量就是在函数内部定义的变量，局部变量只被函数内部访问。全局变量与局部变量不同，它一般定义在程序的顶端，能贯穿整个程序，能被任何一个模块使用。在程序的设计中，如果全局变量和某一个局部变量的名字相同，那么在局部变量使用的函数内部当使用同一名字的两个变量时，实际使用的是局部变量，这一点需要引起注意。正因为如此，所以在定义变量的时候绝对不能重名，并且在使用变量的时候应该合理使用局部变量和全局变量。

结构化的程序需要程序代码和数据分离，C 语言是通过局部变量和全局变量来实现这一分离的。如果大量使用全局变量就破坏了结构化程序设计的要求。下面举例说明局部变量和

全局变量的使用。

```
int a = 3;                          //全局变量
int c = 3;                          //全局变量
int Mult(int x,y);
main()
{
  int sum,a;                        //全局变量
  sum = Mult(a,c);
}
int Mult(int x,int y);
{
  int res;
  res = x * y;
  return res;
}
```

上面的例子具体演示了局部变量和全局变量的使用。上面程序的运行结果应该是48。通过例子也能明确变量的作用域,直观来讲,局部变量就是在函数或子程序里再重新定义的变量,出了这个函数或子程序变量的数值就不再保存了;而全局变量在整个程序运行过程中数值都会保留,直到程序再对其进行修改为止。全局变量一般在整个程序的起始位置来定义,后面的编程中可以直接使用。

2. 形式参数与实际参数

函数定义时的参数称为形式参数,简称形参。它们同函数内部的局部变量作用相同。形参的定义在函数名后的括号内。在进行函数调用时,传入的参数成为实际参数,简称实参。实参和形参的顺序必须一致,否则就会出现错误,这一点需要引起注意。下面举例说明。

```
int GetMax(int x,int y);
main()
{
  int m,n,k;
  m = 9;
  n = 10;
  k = GetMax(m,n);                  //m,n 为实参
}
int GetMax(int x,int y);            //x,y 为形参
{
if(x > = y) return x;
```

```
else return y;
    }
```

程序中 k 的值为 10。

3. 函数调用方式

在函数体实现完成后，需要具体调用函数才能执行函数，也才能利用函数实现的功能。在 C 语言中，函数有标准的库函数，也有用户自定义的函数。对于库函数而言，需要包括具体的头文件，比如 #include<stdio.h>。对于系统预定义的函数，如果函数不在调用它的函数的那个 C 文件里面，则需要包括头文件，比如 #include<stdio.h>。对于用户自定义的函数，如果函数不在调用它的函数的那个 C 文件里面，则需要包括头文件，比如 #include“user.h”。这里需要注意的是头文件不能用“<>”括起来，只能用双引号括起来。如果函数和调用它的函数在同一个文件里，则只需要在函数前面声明一下就可以了。

关于函数的调用主要有以下形式。

① 函数作为执行语句。

```
memset (int a[10],int n);
main()
{
int a[10];
int n;
n = 10;
memset (a,n);                    //函数作为执行语句
}
memset (int a[10],int n);
{
int i;
for(i = 0;i < 10;i ++ )
{
    a[i] = n;
}
return;
}
```

最终结果是 a[0],…, a[9]的值均为 0。

② 函数作为表达式，这种形式就是将函数作为表达式里面的一部分。

```
int GetMax(int x,int y);
main()
```

```
{
int m,n,k;
m = 9;
n = 10;
k = 20;
k = k + GetMax(m,n);                    //函数作为表达式
}
int GetMax(int x,int y)
{
if(x > = y) return x;
else return y;
    }
```

最终结果是 30。

③ 函数作为参数，这种形式就是将函数作为一个函数的实参进行传递。

```
int GetMax(int x,int y);
main()
{
int m,n,k;
m = 9;
n = 10;
k = 20;
k = GetMax(m,GetMax(n,k));          //函数作为参数
}
int GetMax(int x,int y)
{
if(x > = y) return x;
else return y;
    }
```

最终结果是 20。

4. 函数嵌套调用

在 C 语言中，所有的函数都可以互相调用，如果函数调用自己的函数，就是所说的递归调用。函数也可以调用其他函数，函数之间可以实现多次调用，下面举例说明。

```
int max2(int x,int y);
int max3(int x,int y,int z);
main()
```

```
{
int m,n,k,res;
n = 10;
m = 20;
k = 30;
res = max3(n,m,k);
}
int max2(int x,int y)
{
    if(x >= y) return x;
    else return y;
}
int max3(int x,int y,int z);
{
    int res;
    res = max2(x,y);
    return = max2(res,z);
}
```

上面的例子给出了函数的嵌套调用，主要强调的是当实现递归的时候，一定要注意，因为很容易引起死循环。

2.5 数 组

数组是一个由同种类型变量组成的集合，引用这些变量时可以使用同一名字。数组由连续的存储区域组成，最低地址对应数组的第一个元素，最高地址对应最后一个元素。数组可以是一维的，也可以是多维的。

1. 一维数组

一维数组的定义如下：

数组类型　数组名[数组元素个数]；

例如：

```
int a[10];                    //定义一个整数类型的数组 a,数组元素的个数为 10
```

一维数组可以在定义的时候初始化值。例如：

```
int a[10]={0,1,2,3,4,5,6,7,8,9};
```

数组的访问通过下标来实现，数组元素的下标从 0 开始，而不是从 1 开始。比如上面数组中的第 3 个元素就是 a[2]。在赋值的时候可以全部赋值，也可以部分赋值。如果是全部赋

值，那么可以写成下面的形式：

```
int a[]={1,2,3,4,5,6,7,8,9};
```

如果是部分赋值，那么没有被赋值的元素的值默认为 0。例如：

```
int a[10]={1,2,3,4,5};
```

在上面的数组中，只有前 5 个元素被赋了值，而后面几个元素没有被赋值，所以后面几个元素的值为 0。

数组元素的访问和一般变量的使用差不多。下面举例说明。

```
main()
{
int a[10];
int n;
for(n=0;n<10;n++)
{
    a[n]=n;
}
}
```

上面的例子说明数组的元素 a[n]与一般变量的使用基本相同。

2. 多维数组

C 语言中可以定义多维数组，最简单的多维数组就是二维数组。

二维数组的定义如下：

```
数组类型  数组名[行数][列数];
```

二维数组元素的个数等于行数乘以列数。例如：

```
int a[3][5];          //定义一个 3 行 5 列的数组，数组元素的个数为 15
```

二维数组可以在初始化的时候赋值。例如：

```
int a[3][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}}
```

通过以上例子可以看出：二维数组可以看成多个一维数组的集合。

另外，二维数组也可以部分赋值。例如：

```
int a[3][4]={{1,2,3},{5,6,7,8},{9,10}}
```

对于二维数组的每个元素而言，也可以看成一般的变量。下面举例说明。

```
main()
{
int a[10][5];
int n,m;
for(n=0;n<10;n++)
```

```
{
    for(m = 0;m < 5;m ++ )
    {
a[n][m] = m + n;
    }
}}
```

2.6 自学思考题

1. MSP430 系列单片机 C 语言有哪些基本的数据类型?
2. 函数和过程有何区别? 如何定义一个函数?
3. 如何定义一个二维数组?
4. 如何使用 ASCII 码表? A 的 ASCII 值是多少?

第3章

MSP430系列单片机并口JTAG仿真器及其使用

3.1 并口JTAG仿真器功能简介

对于初学者来说，仿真器是必备工具，借助于编程器可以完成程序的下载及在线调试功能，这里介绍一款价格低廉而且实用的并口JTAG仿真器。

图3-1是并口JTAG的实物照片，JTAG仿真器的核心是一个244芯片，购买244芯片时一定要注意型号的问题，比如TTL类型的244，如74LS244，就不能买，因为这种TTL型号

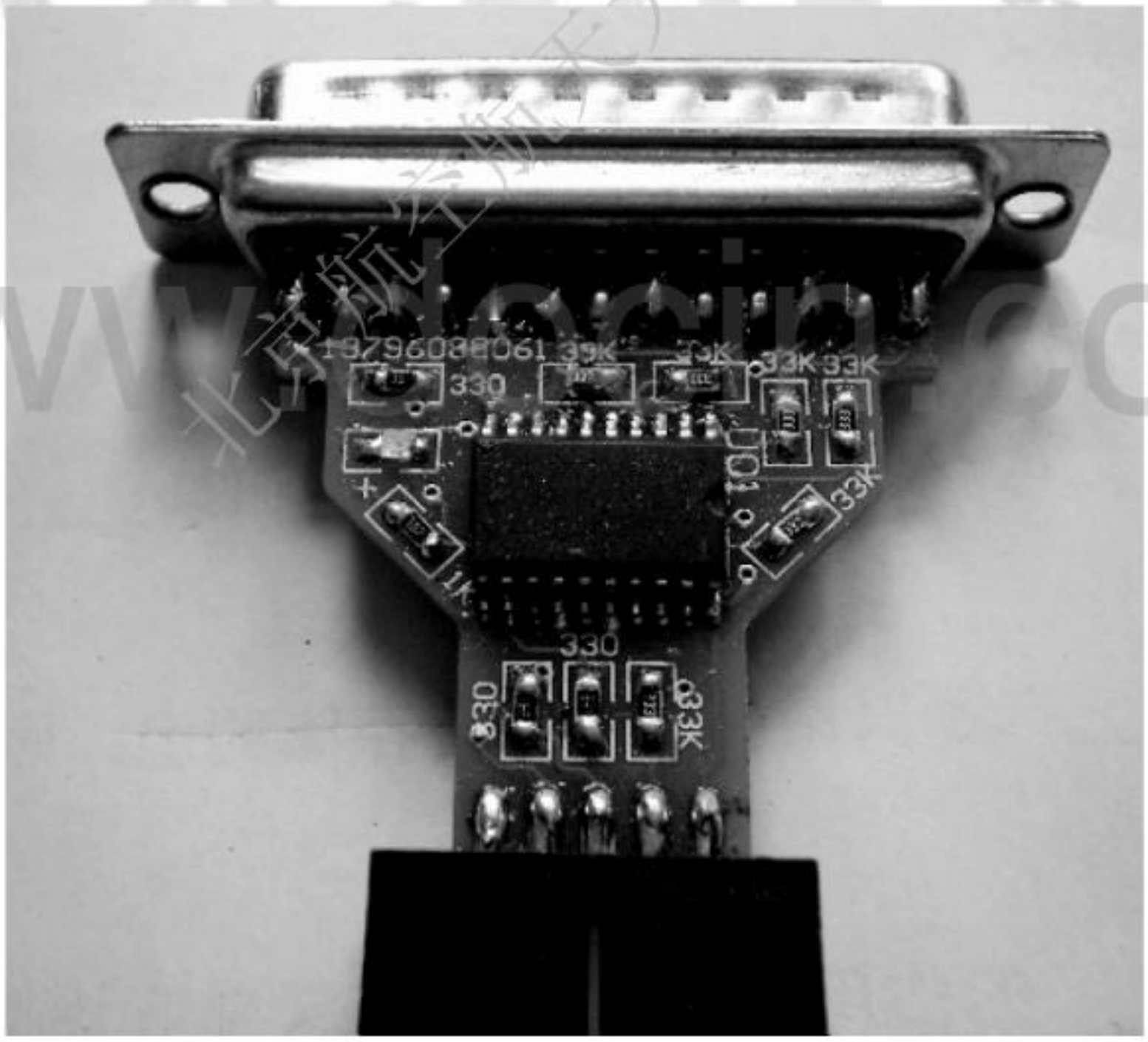


图3-1 并口JTAG的实物照片

的 244 在 3.3 V 电压下可能无法正常工作。这里选用的型号是 74HC244，该芯片可以工作在 3.3 V 电压下。这种芯片目前至少有三种封装形式，一种是直插的，另外两种是贴片的。对于直插封装的 74HC244，由于其引脚间距较大(2.54 mm)，非常适合自己动手时用，可以选择该芯片在实验板上搭建自己的 JTAG 电路，省去了画板、制板的时间与繁琐；缺点是对焊工要求高一些，如果焊工一般，就千万不要去尝试了，因为那是自找麻烦。至于贴片的 74HC244，有两种封装形式，一种是宽体的，一种是窄体的，功能上没有区别，引脚也兼容，就是大小不一样。具体使用时，应先买来芯片再画 PCB 板，在本 JTAG 中采用的是宽体封装的 244 芯片。这种芯片的封装形式可以在光盘中找到。使用时通过上端的 DB25 针形(公头)接口和计算机的打印机的输出接口相连，当然二者之间可能有一定的电阻，然后经过 244 芯片输出，输出端采用一个 IDC10 针形插座输出(实际上只用了 8 根，其中 6 根信号线，2 根电源线)，与目标板的 MSP430 系列单片机相连，实现仿真、下载功能。

图 3-1 上的 DB25 插头是一个针形的插头(因为计算机主机箱后边的输出插口是一个孔型的插座，这样二者才能配合起来)，这个插头和电路板的连接位置属于非标准连接，在做 PCB 板时，要先测量它的外观尺寸，再画它的封装。当然输出接口 IDC10 的插座和电路板连接时也是非标准连接关系，这个封装也要自己来画。至于 74HC244 芯片和电阻，在 PCB 库中都可以找到，74HC244 芯片的封装是 SOL20；电阻的封装是 0805。

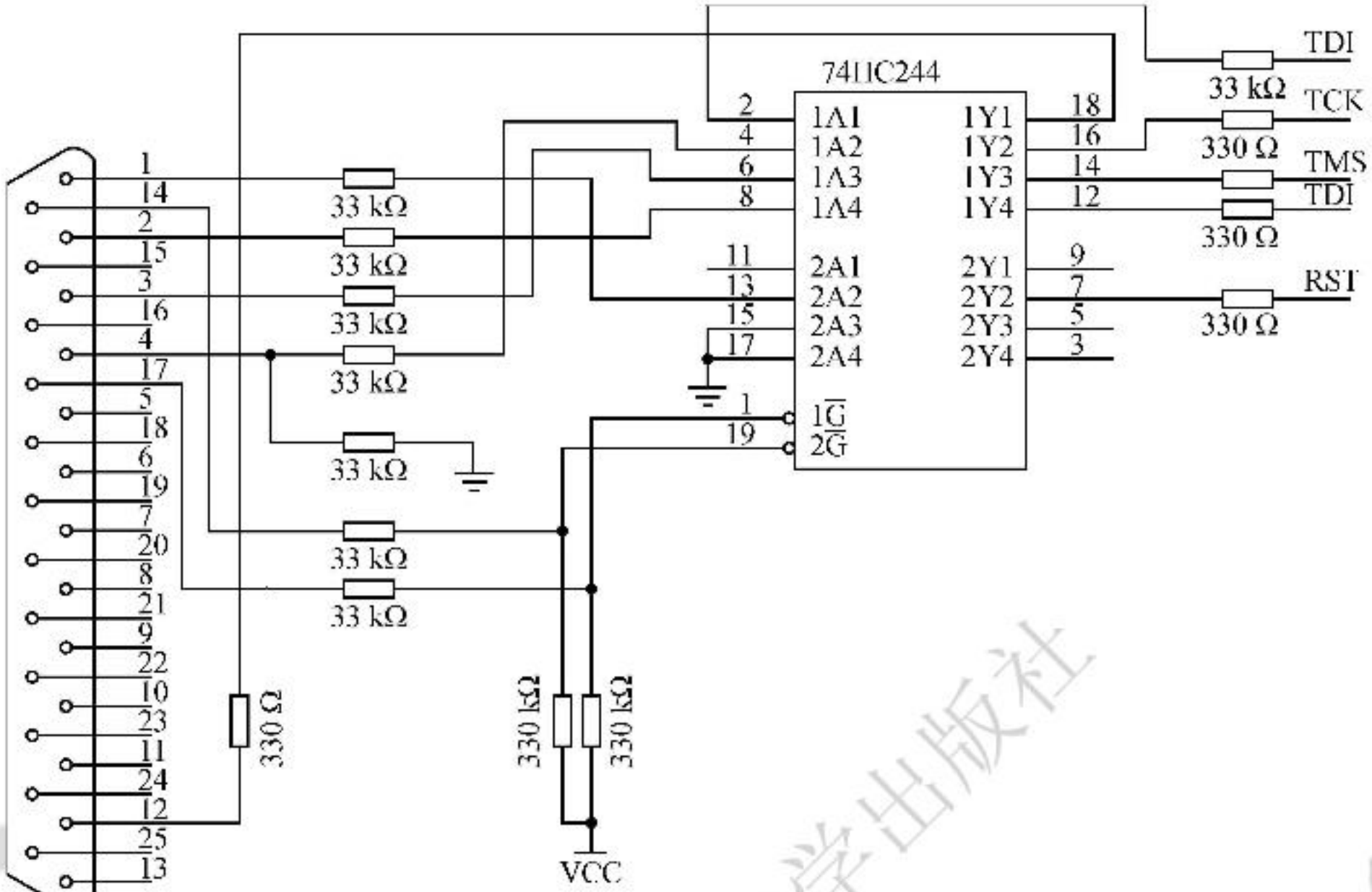
3.2 JTAG 仿真器原理图与自制过程

JTAG 仿真器是 MSP430 系列单片机学习中必不可少的工具，可以购买但价格不便宜，其实 JTAG 仿真器并不复杂，有一定经验的同学完全可以自制。下面介绍 JTAG 的原理图与自制的方法，方便大家自己动手制作。

3.2.1 JTAG 仿真器原理图

图 3-2 是并口的 JTAG 的原理图，从图中可以看出，除使用了一片 74HC244 外，就是一些电阻了。对于电阻的精度，没有什么特殊要求，常用的即可。阻值有三种，330 kΩ、33 kΩ 和 330 Ω。笔者做的编程器中，所有电阻都采用的是 0805 的封装。该编程器结构简单，价格低廉，使用方便，可以实现程序的下载、单步、全速、断点等功能，可以按照原理图自制，也可以购买。如果需要原理图及 PCB，可以发邮件给笔者，邮箱地址是 qingtengzfc@yeah.net。

该编程器中使用的 74HC244 芯片是 8 缓冲器及线驱动器，用来改善三态存储地址驱动器，其引脚如图 3-3 所示。74HC244 芯片可以工作在 3.3 V 的电源电压下，芯片是八同相三态缓冲器/线驱动器，芯片内部共有两个 4 位三态缓冲器，使用时可分别以 $1\overline{G}$和 $2\overline{G}$作为它们的选通工作信号。A 侧为输入端，Y 侧为输出端，当 $1\overline{G}$和 $2\overline{G}$都为低电平时，输入端 A 和输出端 Y 状态相同；当 $1\overline{G}$和 $2\overline{G}$都为高电平时，输出呈高阻态。74HC244 真值表如表 3-1 所



注：74HC244的20引脚接VCC，10引脚接GND。

图 3-2　并口 JTAG 的原理图

列。图 3-2 的原理图的左边与计算机的打印机接口相连，右边与目标板相连。

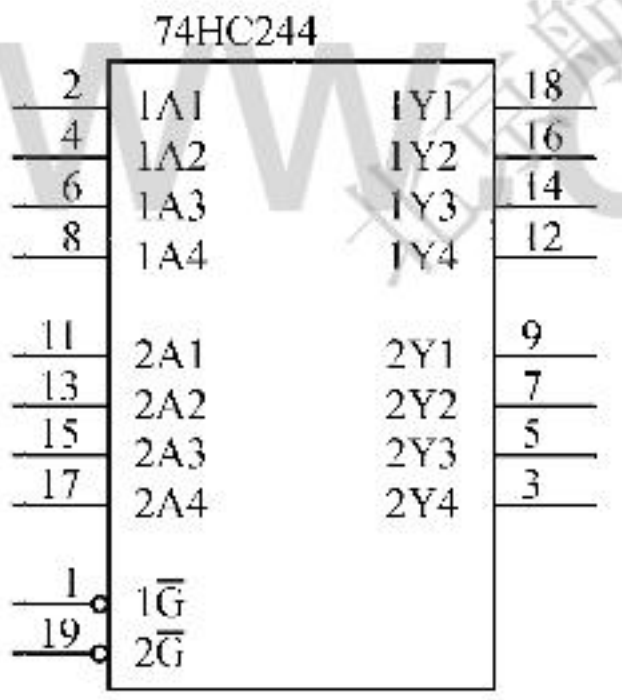


图 3-3　74HC244 芯片引脚图

表 3-1　74HC244 真值表

输　入		输　出
使　能	1Ax/2Ax	1Yx/2Yx
L	L	L
L	H	H
H	×	Z

注：L 表示低电平；H 表示高电平；Z 表示高阻抗；
×表示任意状态；x 表示 1～4。

3.2.2 JTAG 仿真器的自制过程

图 3-4 是 JTAG 仿真器的 PCB 照片，该板是双面板，为节省空间，所有元件均采用贴片封装；在市场上可以买到一种并口的小盒子，大家可以把做好的编程器放到小盒子里面。这里要说明一点，大家自己动手制作的时候，可以购买一个实验板，买一个直插的 74HC244 就行了。如果没有制作 PCB 板就不要买贴片的，否则，特别不容易焊接。如果需要 PCB 图，可以联系笔者。

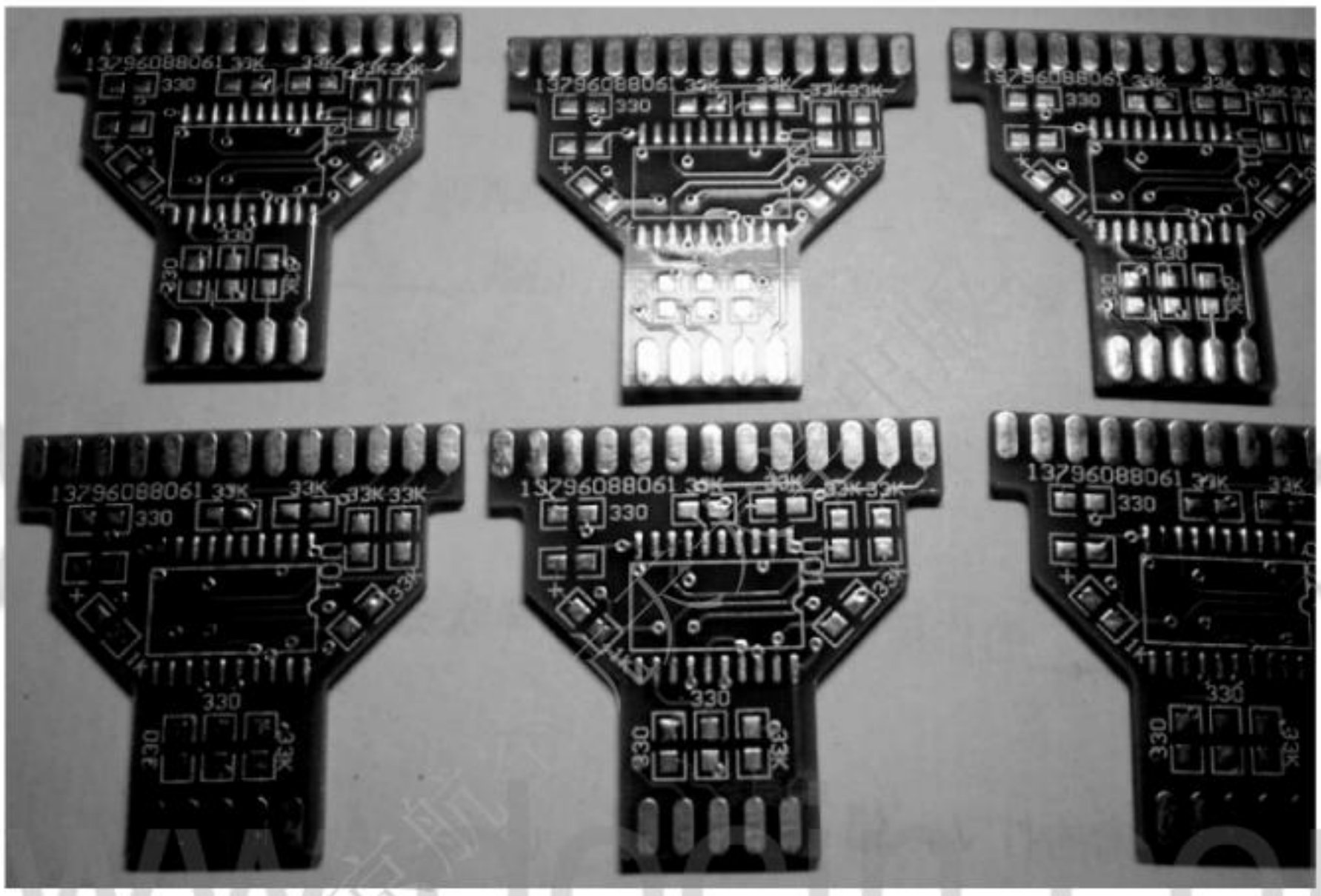


图 3-4 并口的 JTAG 的 PCB 照片

该编程器只采用了一片 74HC244 芯片及少量的贴片元件，特别适合初学者自己动手制作。贴片元件焊接时首先在 PCB 的一个焊盘上点上少许的焊锡，然后用镊子夹住元件，将其焊在刚才点过焊锡的位置，然后再将贴片元件的另一端也焊好就 OK 了。焊接 0805 的贴片大家实验几次就没有问题了。贴片的 74HC244 焊接起来有点难度，基本上有两种方法。一种是一个一个引脚慢慢焊接，这种方法比较容易理解，这里就不必细说了。下面介绍第二种方法，这种方法在焊接贴片时比较常用，通过图片的形式简单介绍一下。

如图 3-5 所示，首先，在元件的一侧挂上焊锡，起固定作用，可以随便焊几个引脚，也可以用堆锡的方式，一堆锡也没关系，不要害怕更不用担心，后面自有处理的方式。

在元件的另一侧，只采用堆锡的方式，如图 3-6 所示。看到图中的铅“疙瘩”害怕了没有？是不是有“废”了的感觉啊？没关系，下面给出解决的方法。

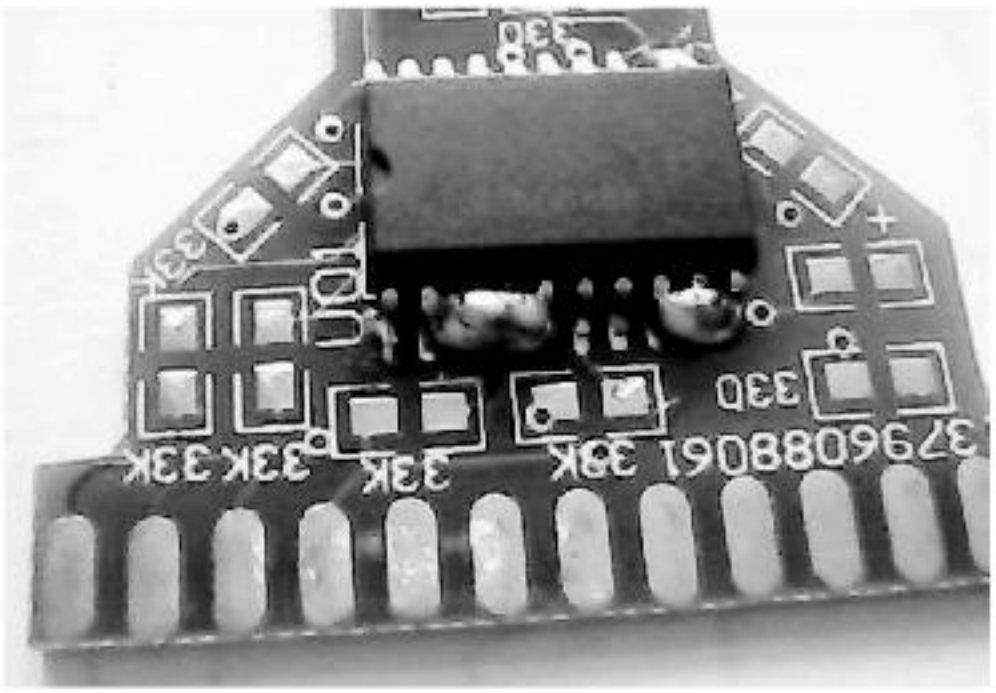


图 3-5　在元件的一侧挂锡,起固定作用

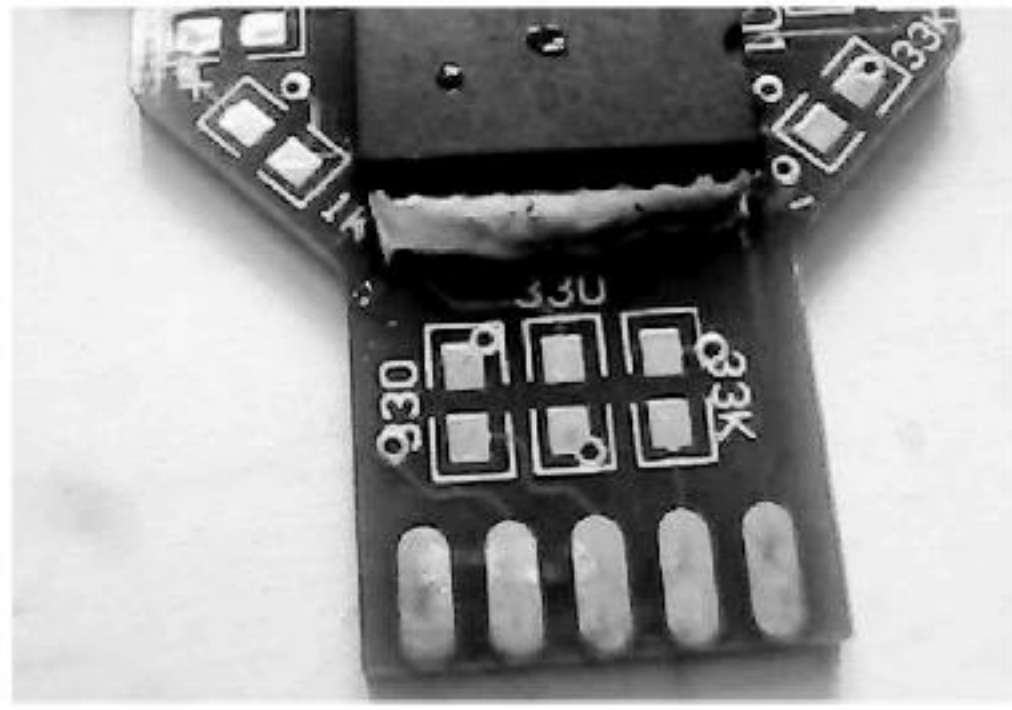


图 3-6　在另一侧大面积堆锡

在图 3-7 中,使用烙铁在堆锡的一侧不断加热,在加热过程中,不断左右移动烙铁,只要烙铁足够热,焊锡就完全变为融化状态了,就像水一样,这时就到了最关键的时候了! 用手拿起电路板用力向桌面“撞”去,这时焊锡就会从元件的一侧脱落下来,引脚间只会残留很少的焊锡,但这一点焊锡就足够保证元件和电路板连接用了。如果引脚还有部分粘连在一起的话,可以重复上述方法,用同样的方法向桌面“撞”几下即可。至于元件的另一侧,也就是起固定作用的那一侧,也可以采用相同的方法,细节在这里就不再赘述了,请大家自己尝试。

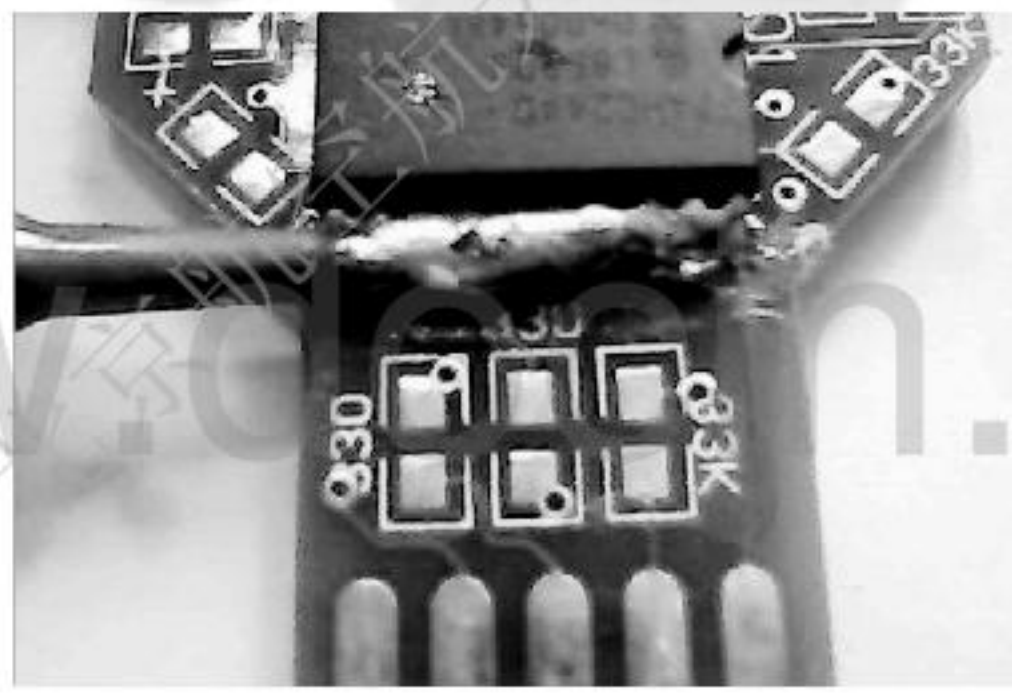


图 3-7　烙铁在一侧加热示意图

3.3　自学思考题

1. JTAG 仿真器具有哪些功能?
2. 自制 MSP430 系列单片机 JTAG 仿真器时都需要哪些元件?
3. 如何焊接贴片元件?

第4章

MSP430 系列单片机集成开发调试环境

目前，MSP430 系列单片机的开发调试环境版本比较多，本章介绍的是 IAR 公司提供的开发调试环境，版本是 IAR Embedded Workbench V3.42A。该编程软件可以在网上下载，选择非限制版本就行了。

4.1 IAR Embedded Workbench V3.42A 概述

IAR Embedded Workbench V3.42A 支持全系列 MSP430 单片机，它具备以下特点。

① 支持 Windows 98/Windows NT/Windows XP 操作系统；

② Windows 风格的可视化开发环境；

③ 集成所有开发、调试工具(编译、连接、仿真)，方便使用；

④ 可以采用 Make 进行重新编译、连接和调试。

IAR Embedded Workbench V3.42A 采用创建项目(Project)的方式来进行软件的开发和管理。IAR Embedded Workbench V3.42A 包含的实用工具有：编辑器、汇编器、连接器、函数管理器、Make 工具和调试工具。

用户使用集成开发环境的文本编辑器编写源程序代码，该编译器特点如下：

① 根据 C 语言的语法来区别字体的颜色。

② 具有查找和替换功能，能够非常方便地对程序进行编辑。

③ 可以从出错的列表中直接跳到文本中相应的出错位置。

④ 能够检查括号是否匹配，这一点在程序编写、纠错时非常有用。

⑤ 可以多个窗口进行编辑。

程序编写完成后，用户可以对程序代码进行编译连接。编译连接完成后，可以运行程序，可以对程序进行调试。

4.2 Electronic Workbench for MSP430 V3.42A 安装过程简介

光盘中的编译软件是非限制版本，所以安装有点复杂，需要破解才能正常使用，现将

MSP430 系列单片机的开发环境介绍如下，可以按照下列步骤完成安装。

1）双击名为 keygen 的图标，出现注册机的画面，在 Product 选项的下拉菜单里选择 Electronic Workbench for MSP430 V3.42A。

2）查看 Hardware ID 文本框中的字符，将 0x 以后的字符中的小写字母全部改为大写。具体的操作方法是：直接选中某个小写字母，然后从键盘输入大写字母即可；更改完毕以后，选中文本栏内的所有字符就能看到原来的小写字母是否已经被改为大写了。例如，笔者计算机中的操作流程如图 4-1～图 4-4 所示。

① 打开软件以后看到的 Hardware ID，见图 4-1。

图 4-1　Hardware ID

② 选中小写字母 d，见图 4-2。

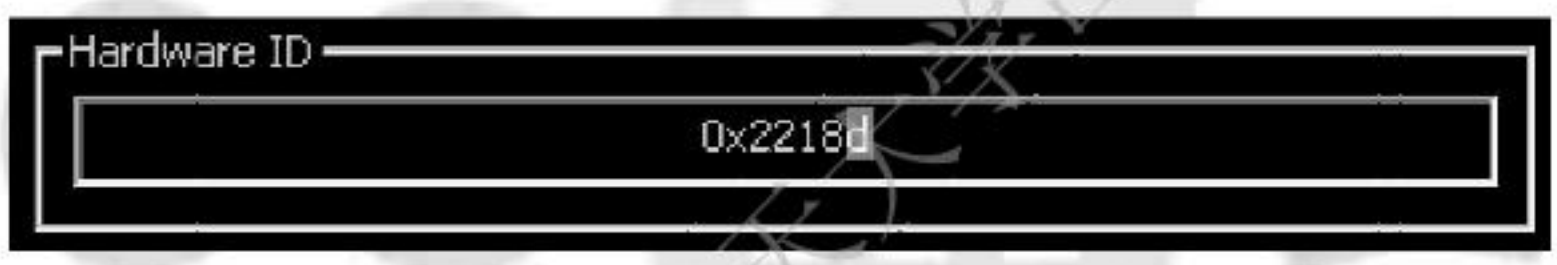


图 4-2　输入小写字母

③ 从键盘输入大写字母 D，见图 4-3，此时整个文本框内的字符会变得模糊不清，这是正常现象，不用担心；如果要看清楚，将工具条最小化后再恢复一下即可。

图 4-3　输入大写字母 D

④ 再次选择整个文本框中的内容，见图 4-4，可以看到模糊的字符又变清晰了，注意原来的 d 已经被更改为 D。

图 4-4　选择整个文本框

这时，单击注册机软件的左下角的 Generate 图标就可以得到需要的序列号了。

3）双击名为 EW430-ev-web-342A 的图标，等待解压缩完毕后出现安装界面，单击 Next 按钮后看到关于 License 的说明，再单击 Accept 按钮就可以看到如图 4－5 所示的对话框。

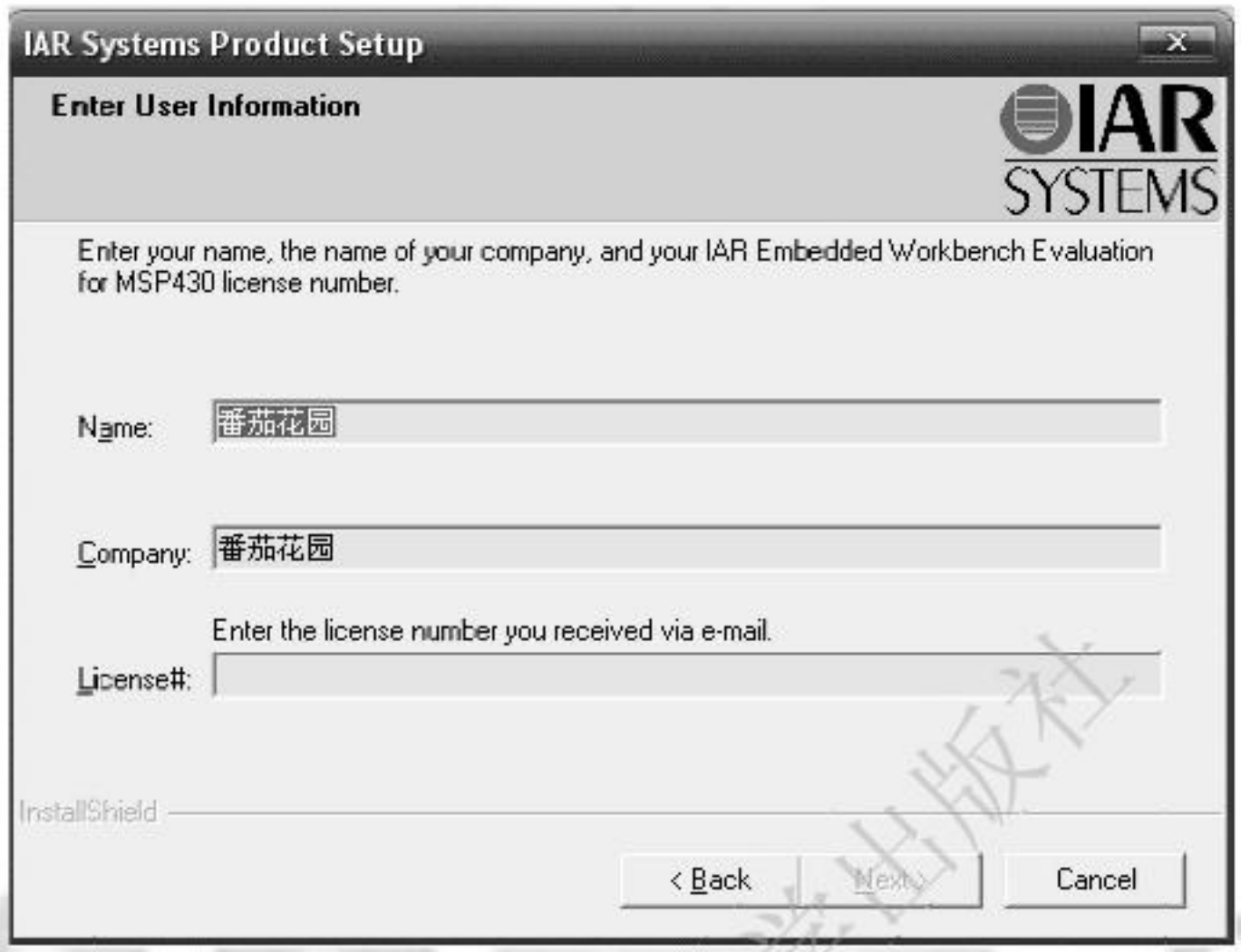


图 4－5　安装对话框 1

4）用户可以随意更改 Name 和 Company 文本框中的内容；然后选择已经打开的注册机，将 License number＋key 文本框中的数字复制到图 4－5 的 License 的文本框中，可以看到原来灰色的 Next 按钮变成了黑色，单击 Next 按钮进入下一对话框，如图 4－6 所示。

图 4－6　安装对话框 2

5）再次回到已经打开的注册机，将最后一栏文本框中的全部数字复制到 License Key 下面的文本框中，单击 Next 按钮，看到图 4－7。

图 4－7　安装对话框 3

6）单击 Browse 按钮可以更改安装目录，然后单击 Next 按钮继续。等待安装完成，单击 Finish 按钮就大功告成了。

7）安装以后，您的桌面上会出现图标，双击即可打开这个软件了。

4.3　Electronic Workbench for MSP430 V3.42A 应用方法简介

4.3.1　Electronic Workbench 仿真设置

1）创建一个新的项目。选择 Project→Create New Project 命令，显示如图 4－8 所示对话框。选择项目模板 Empty project 后，会出现“另存为”的对话框，如图 4－9 所示，选择文件的保存位置并在文件名文本框中输入项目名字，单击“保存”按钮就可以了。这样将建立一个空的不包含任何文件的项目。

2）也可以单击图 4－8 中 C 旁边的“＋”号，选中下面新出现的 main，单击 OK 按钮以后新建一个带有 main.c 源文件的项目。单击 OK 按钮后同样会出现图 4－9 的画面，进行类似操作保存即可。

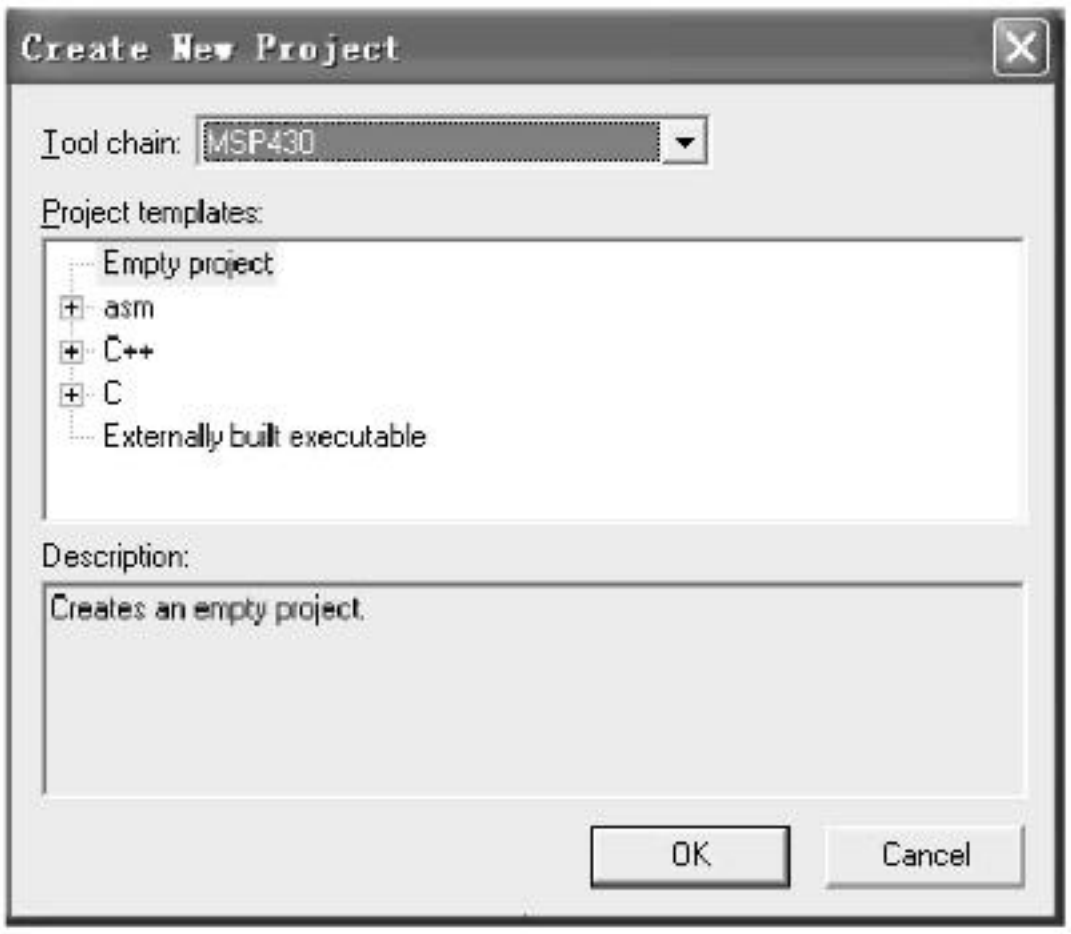


图 4-8　创建一个新的项目

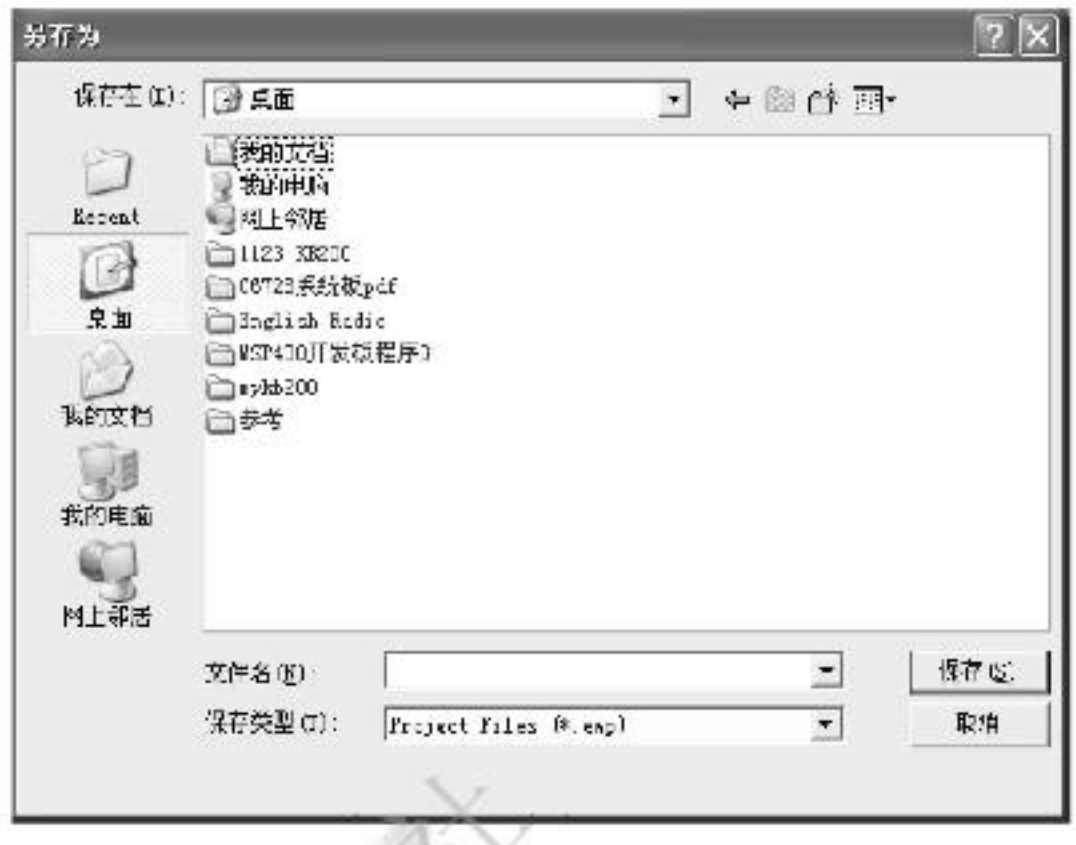


图 4-9　“另存为”对话框

3) 之后可以看到当前项目出现在 Workspace 窗口内，如图 4-10 所示。在 Workspace 中有一个下拉列表框，这里有系统的创建配置(build configurations)，默认时系统有两种创建(build)配置：Debug 和 Release。缺省配置是 Debug，在这种模式下，用户可以进行仿真和调试；在 Release 模式下，是不能进入调试状态的，所以建议在产品研发阶段一定不要修改这个创建配置，否则就不能进行调试了。

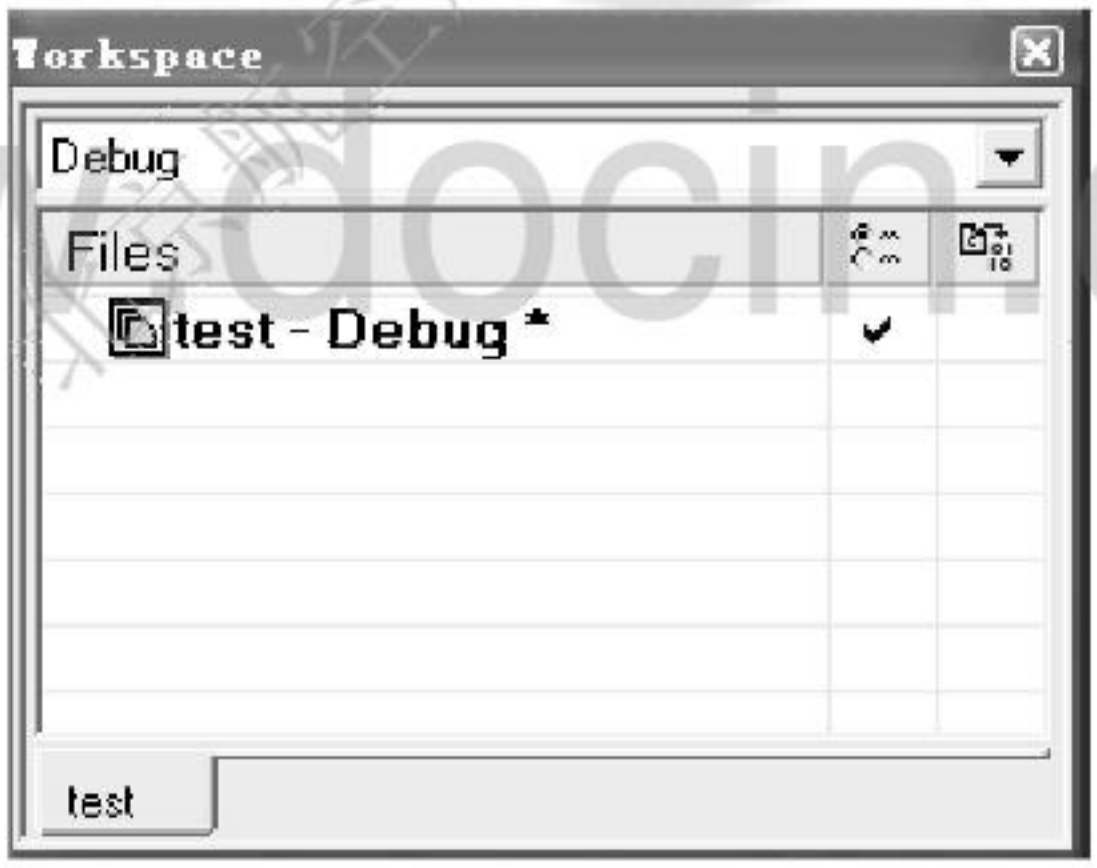


图 4-10　Workspace 窗口

4) 选择 File→Save Workspace 命令，保存当前的工作空间(Workspace)，以后直接打开工作空间就可以了。系统会为每个 Workspace 单独保存一套配置信息，所以不同项目的设置可以保留而不会相互冲突，因此建议用户每次建立一个项目都单独存储一个 Workspace 文

件，这样日后使用起来相当方便。

5）如果用户已经编辑好了源文件，则选择 Project→Add Files 命令就可以打开一个添加文件的对话框，见图 4-11。在这里可以向项目中添加源文件。在 Files of type 下拉列表框中可以选择要添加的文件类型。用鼠标同时选择多个文件或者按住 Ctrl 键单击多个文件名可以一次性地向项目中添加多个文件。当然，对于一般的初学者来讲，一般只有源文件是要添加的，另外，引用的那些是不需要添加的，那些一般都可以自动由编译软件自动加载进来。

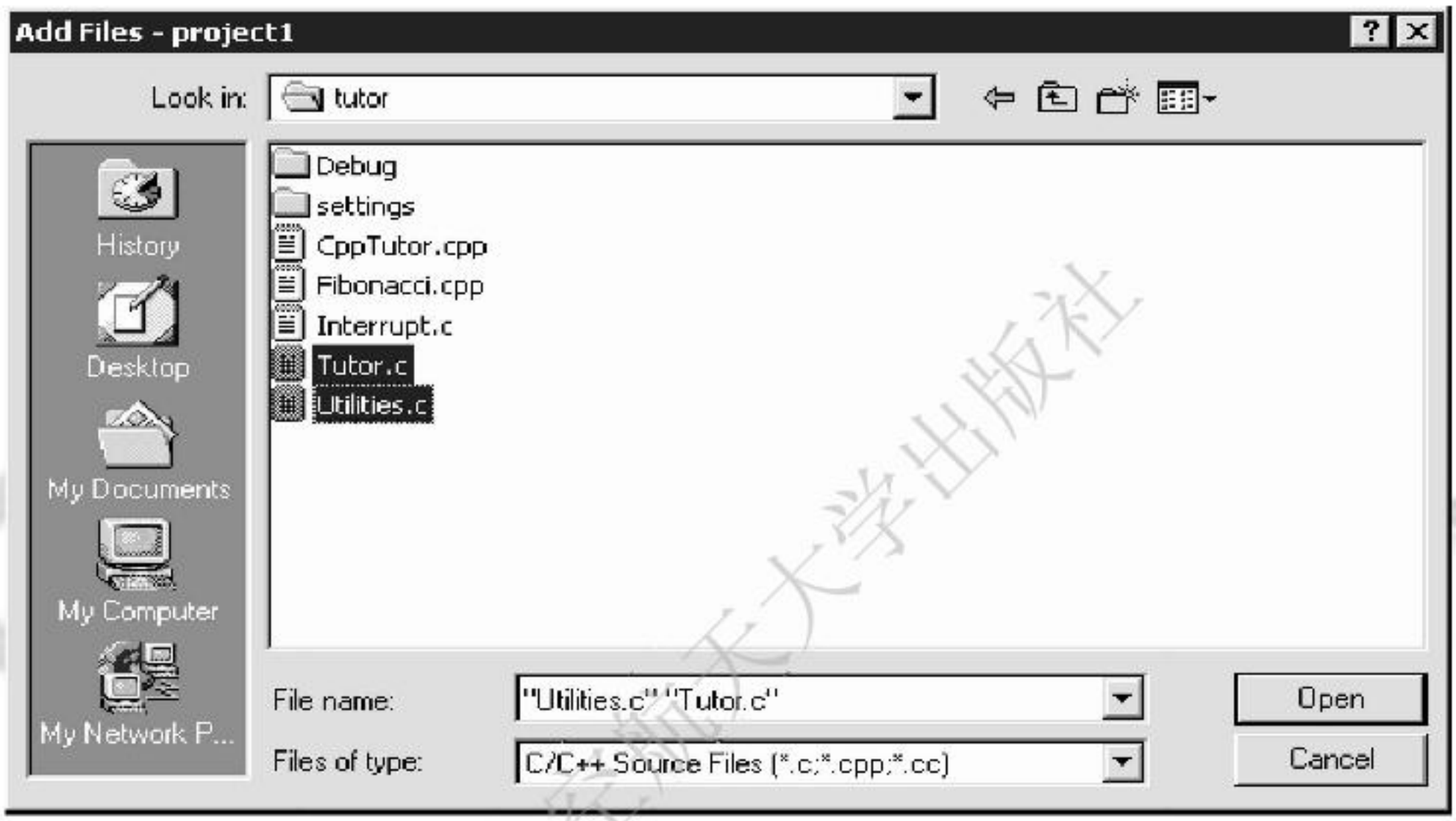


图 4-11　添加文件的对话框

6）如果用户需要手动输入源文件，则选择 File→New File 命令，也可使用工具栏左侧的图标按钮新建一个文本文件，然后用户可在其中输入自己的源程序，选择 File→Save 命令用于保存输入的文件。

7）所有的源文件都输入完毕以后，需要设置项目选项(Project Options)。选择 Project→Options 命令，或者将光标放在窗口左边的 Workspace 窗口的项目名字上单击右键(简称右击)选择 Options 命令，可以看到如图 4-12 所示对话框。

8）图 4-12 中是有关对本项目进行编译(compile)和创建(make)时的各种控制选项，系统的默认配置已经能够满足大多数应用的需求，所以通常情况下用户只需要更改两个地方。

① 单击 Category 下面的 General Options(见图 4-13)，再单击 Device 下方文本框右侧的图标按钮，显示图 4-14 所示的 MSP430 单片机型号，用户可以选择要使用的单片机。若使用本书介绍的开发板，请选择 MSP430F169，此时可以看到如图 4-15 所示的画面。

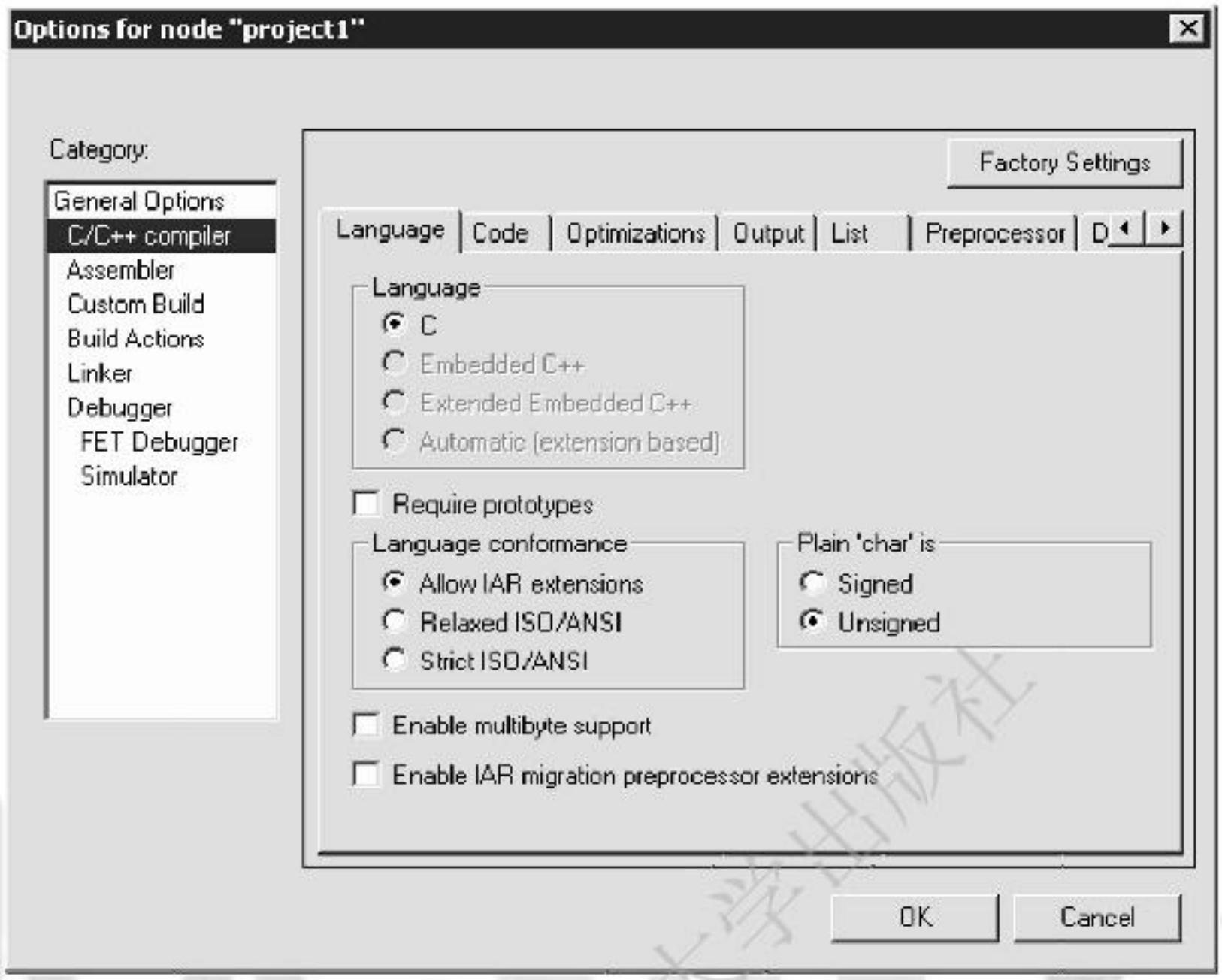


图 4-12　设置项目选项对话框

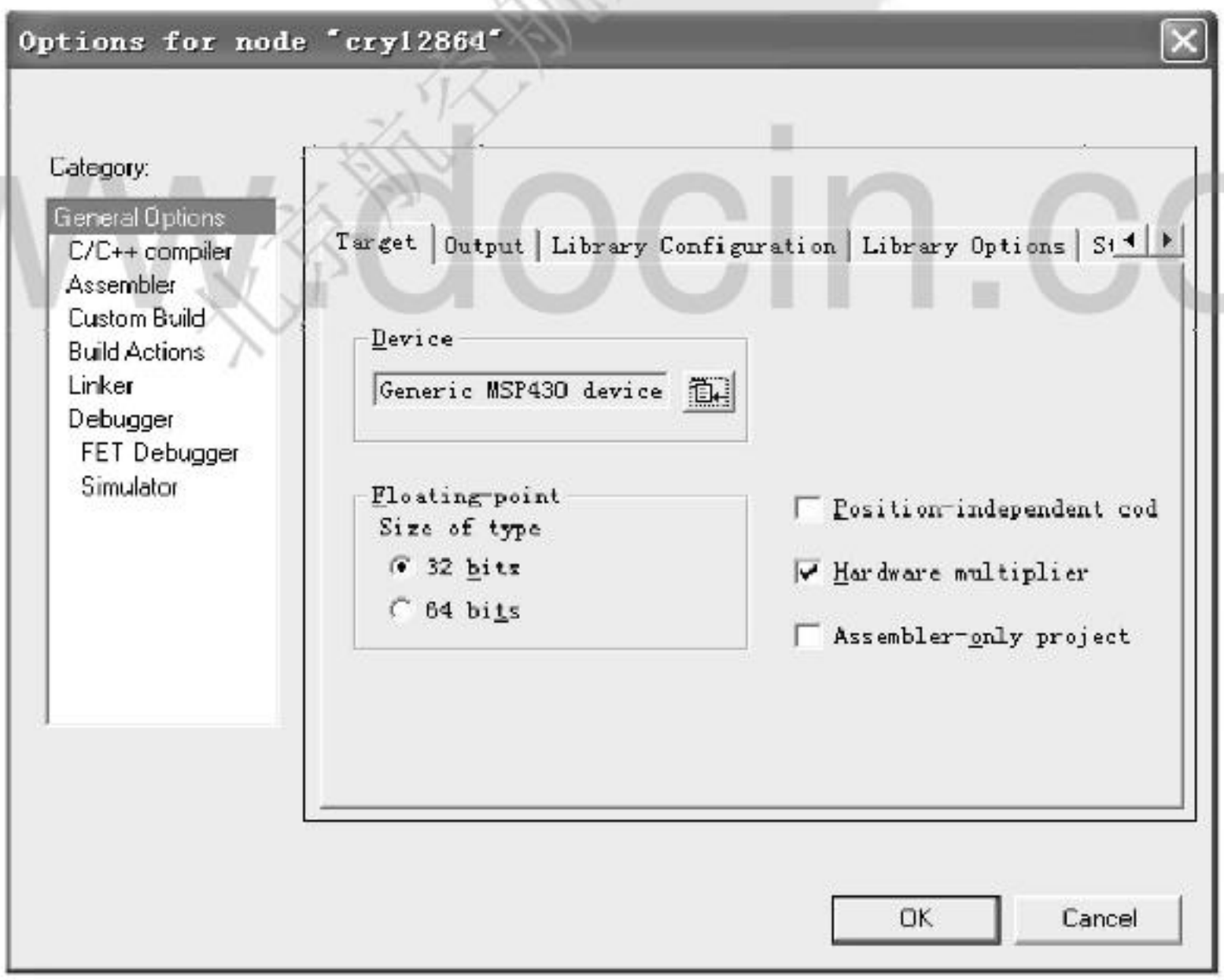


图 4-13　系统配置对话框

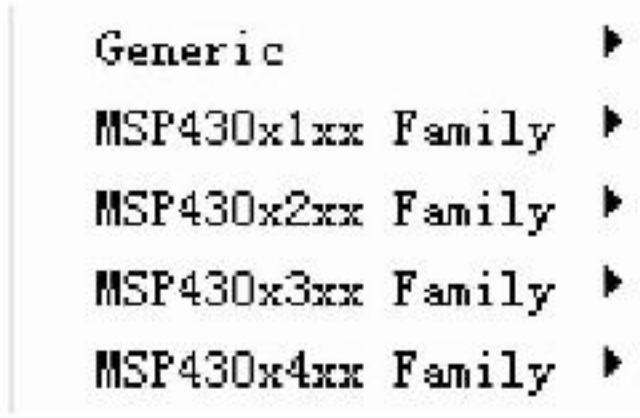


图 4-14　单片机型号选择

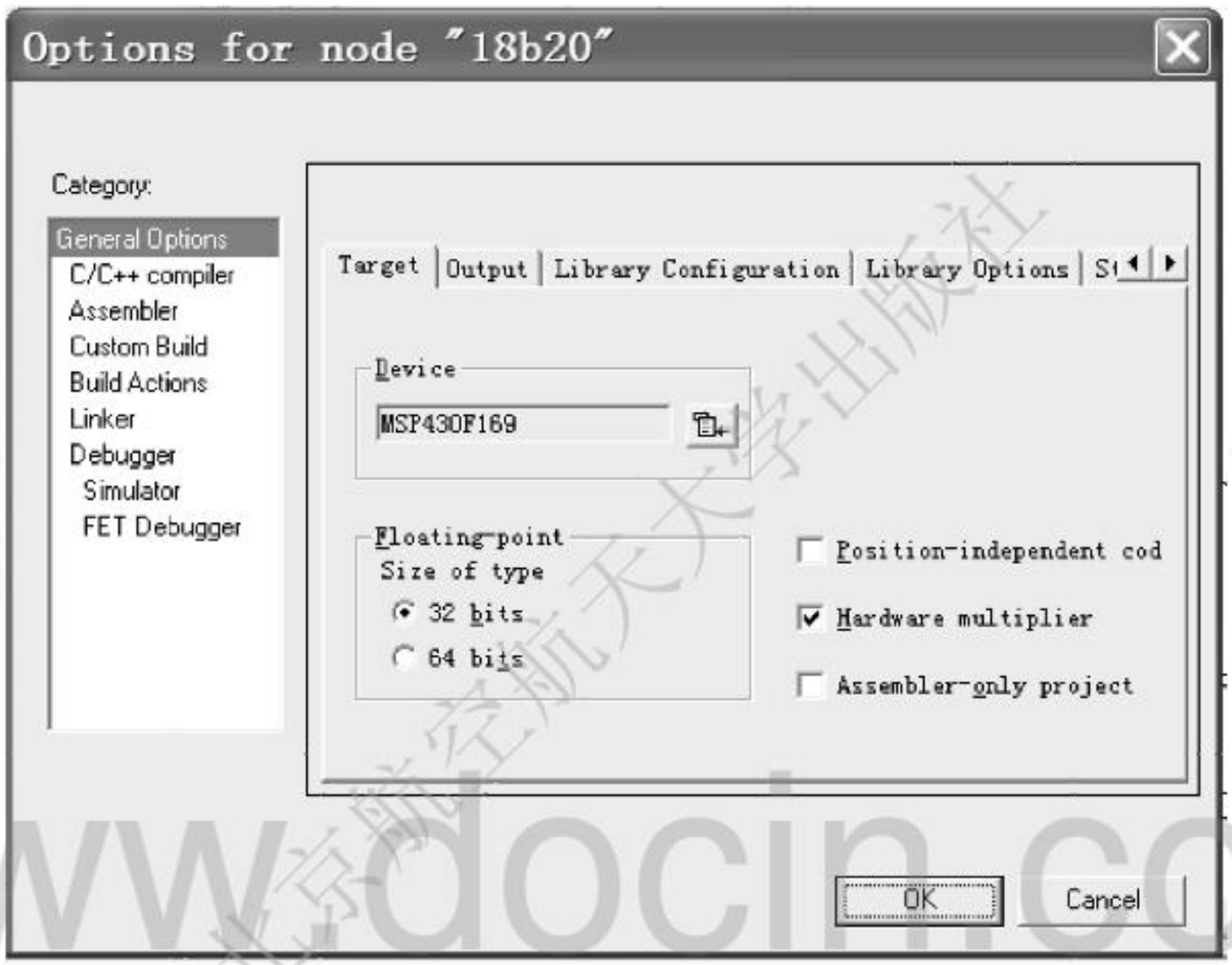


图 4-15　选择 MSP430F169

② 单击 Category 下面的 Debugger(见图 4-16),在 Driver 下拉列表框中,可以看到 Simulator 和 FET Debugger 两个选项。若选择 Simulator,则可以用软件模拟硬件时序,实现对程序运行的仿真观察;若选择 FET Debugger,则需要通过仿真器将 PC 上的软件与开发板上的 MCU 进行连接,然后就可以进行硬件仿真了。

如果用户只想进行软件仿真,那么选择 Simulator 以后单击 OK 按钮,就可以完成设置。如果用户需要进行硬件仿真,那么选择 FET Debugger 后,可以看到如图 4-17 所示的画面。在 Connection 下拉列表框中选择要使用的仿真器类型,就可以了。如果使用精简版仿真器,请选择 Texas Instrument LPT-IF。

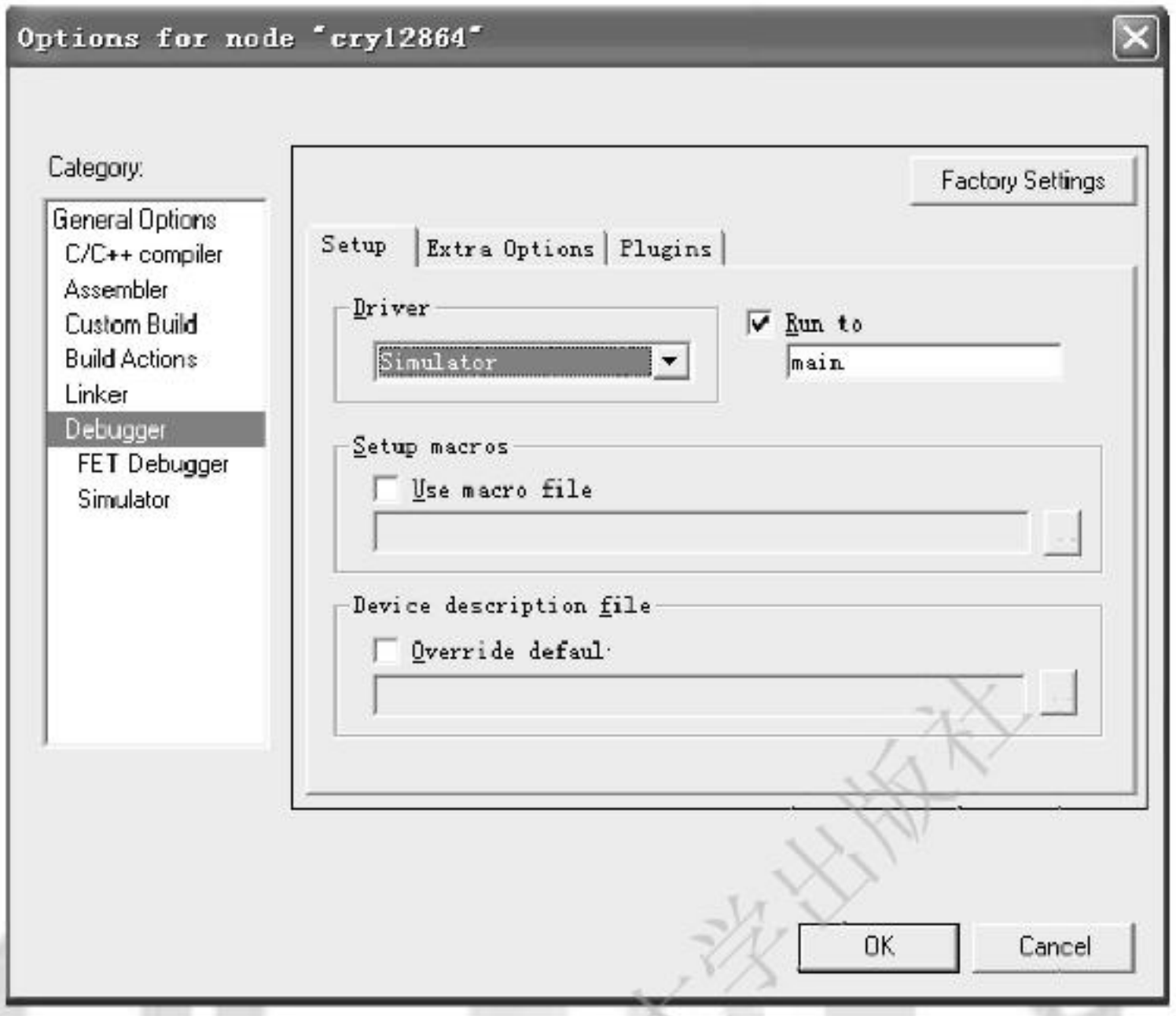


图 4-16　仿真器选择

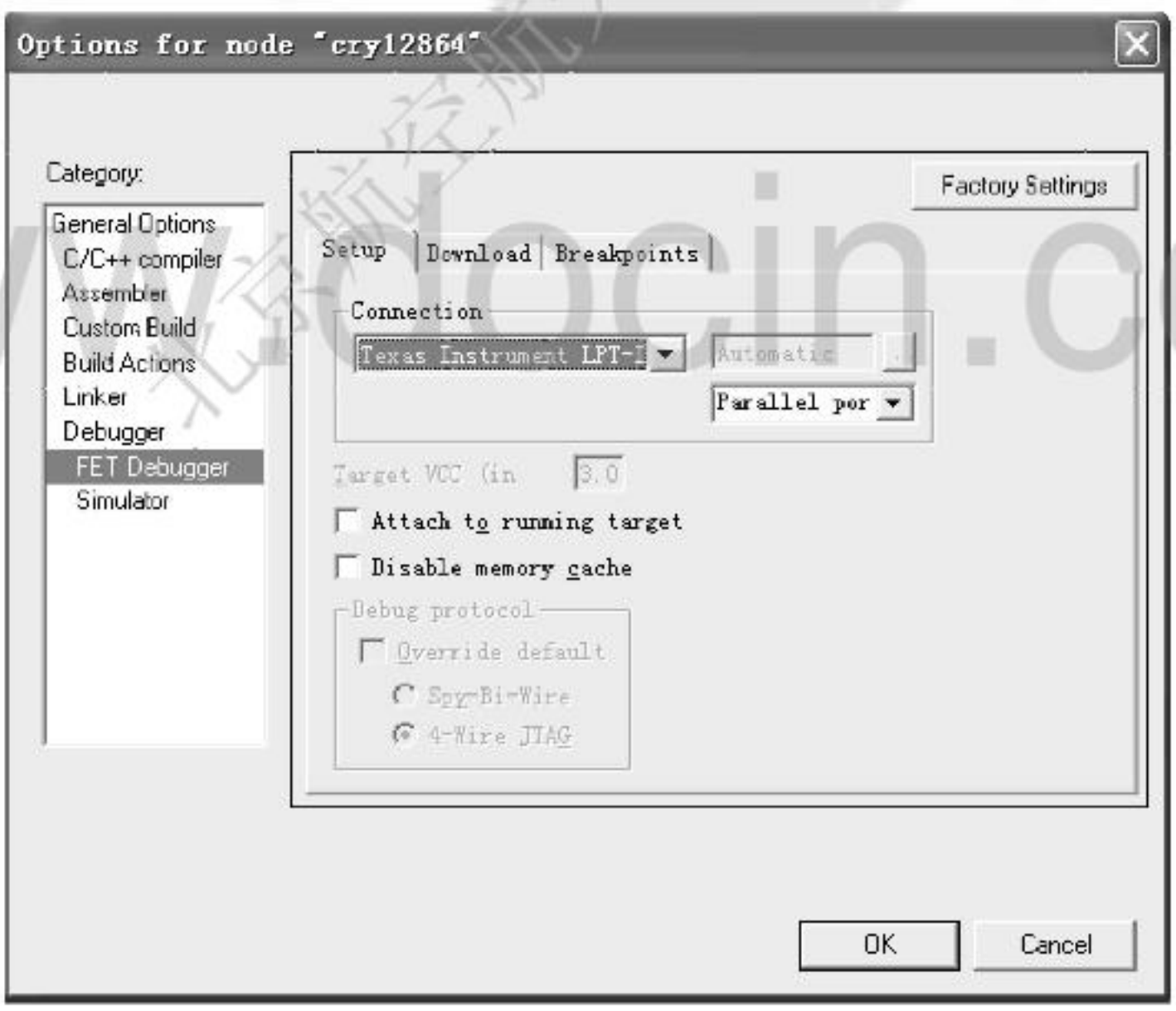


图 4-17　选择 JTAG 仿真器

4.3.2 Electronic Workbench 仿真过程举例

① 完成前面的设置以后，选中项目中的一个源文件(如. c、. cpp、. cc、. s、. asm、msa)，选择 Project→Compile 命令，或者单击工具栏中的图标按钮，对源文件进行编译。如果有错误，请根据提示的出错信息，将错误修正以后重新编译。

② 保证所有的源文件都编译通过以后，选择 Project→Make 命令，或者单击工具栏中的图标按钮，对源文件进行创建连接。如果有错误，请根据提示的出错信息，将错误修正以后重新创建连接。

③ 创建通过以后，就可以进入调试阶段了。选择 Project→Debuger 命令，或者单击工具栏中的图标按钮 就可以进入调试界面了。图 4-18 是编辑界面整个项目创建连接成功以后的截图；图 4-19 是进入调试界面以后的截图。

图 4-18 IAR for MSP430 界面

④ 在调试界面用户可以看到有一行被箭头选中，这表示程序计数器指向了此行的程序。将光标放在程序中的某一行，单击图标按钮 可以在这里设置一个断点，当程序运行到此处时会自动停止，用户可以观察某些变量；另外，也可以单击图标按钮 ，程序将自动运行到当