dsPIC30F 数字信号控制器入门用户指南

© 2007 Microchip Technology Inc.

请注意以下有关 Microchip 器件代码保护功能的要点：

- Microchip 的产品均达到 Microchip 数据手册中所述的技术指标。
- Microchip 确信：在正常使用的情况下， Microchip 系列产品是当今市场上同类产品中最安全的产品之一。
- 目前，仍存在着恶意、甚至是非法破坏代码保护功能的行为。就我们所知，所有这些行为都不是以 Microchip 数据手册中规定的操作规范来使用 Microchip 产品的。这样做的人极可能侵犯了知识产权。
- Microchip 愿与那些注重代码完整性的客户合作。
- Microchip 或任何其他半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是 “牢不可破”的。

代码保护功能处于持续发展中。 Microchip 承诺将不断改进产品的代码保护功能。任何试图破坏 Microchip 代码保护功能的行为均可视为违反了《数字器件千年版权法案 （Digital Millennium Copyright Act）》。如果这种行为导致他人在未经授权的情况下，能访问您的软件或其他受版权保护的成果，您有权依据该法案提起诉讼，从而制止这种行为。

提供本文档的中文版本仅为了便于理解。请勿忽视文档中包含的英文部分，因为其中提供了有关 Microchip 产品性能和使用情况的有用信息。Microchip Technology Inc. 及其分公司和相关公司、各级主管与员工及事务代理机构对译文中可能存在的任何差错不承担任何责任。建议参考 Microchip Technology Inc. 的英文原版文档。

本出版物中所述的器件应用信息及其他类似内容仅为您提供便利，它们可能由更新之信息所替代。确保应用符合技术规范，是您自身应负的责任。**Microchip 对这些信息不作任何明示或暗示、书面或口头、法定或其他形式的声明或担保，包括但不限于针对其使用情况、质量、性能、适销性或特定用途的适用性的声明或担保。** Microchip 对因这些信息及使用这些信息而引起的后果不承担任何责任。如果将 Microchip 器件用于生命维持和 / 或生命安全应用，一切风险由买方自负。买方同意在由此引发任何一切伤害、索赔、诉讼或费用时，会维护和保障 Microchip 免于承担法律责任，并加以赔偿。在 Microchip 知识产权保护下，不得暗中或以其他方式转让任何许可证。

商标

Microchip 的名称和徽标组合、 Microchip 徽标、 Accuron、 dsPIC、 KEELOQ、 KEELOQ 徽标、 microID、 MPLAB、 PIC、 PICmicro、 PICSTART、 PRO MATE、 rfPIC 和 SmartShunt 均为 Microchip Technology Inc. 在美国和其他国家或地区的注册商标。

AmpLab、 FilterLab、 Linear Active Thermistor、 Migratable Memory、 MXDEV、 MXLAB、 SEEVAL、 SmartSensor 和 The Embedded Control Solutions Company 均为 Microchip Technology Inc. 在美国的注册商标。

Analog-for-the-Digital Age、 Application Maestro、 CodeGuard、 dsPICDEM、 dsPICDEM.net、 dsPICworks、 dsSPEAK、 ECAN、 ECONOMONITOR、 FanSense、 FlexROM、 fuzzyLAB、 In-Circuit Serial Programming、 ICSP、 ICEPIC、 Mindi、 MiWi、 MPASM、 MPLAB Certified 徽标、 MPLIB、 MPLINK、 PICkit、 PICDEM、 PICDEM.net、 PICLAB、 PICtail、 PowerCal、 PowerInfo、 PowerMate、 PowerTool、 REAL ICE、 rfLAB、 Select Mode、 Smart Serial、 SmartTel、 Total Endurance、 UNI/O、 WiperLock 和 ZENA 均为 Microchip Technology Inc. 在美国和其他国家或地区的商标。

SQTP 是 Microchip Technology Inc. 在美国的服务标记。

在此提及的所有其他商标均为各持有公司所有。

© 2007, Microchip Technology Inc. 版权所有。

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2002

Microchip 位于美国亚利桑那州 Chandler 和 Tempe 与位于俄勒冈州 Gresham 的全球总部、设计和晶圆生产厂及位于美国加利福尼亚州和印度的设计中心均通过了 ISO/TS-16949:2002 认证。公司在 PIC® MCU 与 dsPIC® DSC、KEELOQ® 跳码器件、串行 EEPROM、单片机外设、非易失性存储器和模拟产品方面的质量体系流程均符合 ISO/TS-16949:2002。此外，Microchip 在开发系统的设计和生产方面的质量体系也已通过了 ISO 9001:2000 认证。

© 2007 Microchip Technology Inc.

目录

© 2007 Microchip Technology Inc.

注：

© 2007 Microchip Technology Inc.

dsPIC30F 数字信号控制器入门

前言

客户须知

所有文档均会过时，本文档也不例外。 Microchip 的工具和文档将不断演变以满足客户的需求，因此实际使用中某些对话框和 / 或工具说明可能与本文档所述之内容有所不同。请访问我们的网站 （www.microchip.com）获取最新文档。

文档均标记有 “DS”编号。该编号出现在每页底部的页码之前。 DS 编号的命名约定为 “DSXXXXXA”，其中 “XXXXX”为文档编号，“A”为文档版本。

欲了解开发工具的最新信息，请参考 MPLAB® IDE 在线帮助。从 Help （帮助）菜单选择 Topics （主题），打开现有在线帮助文件列表。

简介

欢迎使用此关于单片机综合解决方案的最佳文档。 Microchip Technology， 8 位单片机付运量居全球首位的单片机供应商，亦提供 dsPIC®16 位数字信号控制器系列。旨在满足多种应用的需求，dsPIC30F 器件将单片机的灵活性和控制能力与数字信号控制器的计算和数据吞吐能力融合在了一起。

dsPIC30F 得到了以业界领先的 MPLAB® 集成开发环境 （Integrated Development Environment， IDE）为中心的多种开发工具的支持。在本指南中，您将学会如何使用 MPLAB IDE 及相关的汇编器、编译器、链接器、软件模拟器、调试器以及仿真器工具。本指南涵盖了所有这些工具，所以即使您还没有相应的硬件，也会发现它十分有用。动手实验教程可让您在最短的时间内获得最佳的学习体验。这些教程通过 dsPICDEM™ 入门演示板或dsPICDEM™ 1.1通用开发板来使用MPLAB ICD 2在线调试器，对器件进行编程和调试。这些教程也同样适用于 dsPICDEM™ 28 引脚入门演示板和 dsPICDEM™ 2 开发板。

本前言中讨论的内容包括：

- 关于本指南
- 推荐读物
- Microchip 网站
- 开发系统变更通知客户服务
- 客户支持

关于本指南

本入门指南涵盖了 dsPIC 系列器件的架构和开发工具，并提供了许多技巧帮助您选择适合您设计的正确的 dsPIC 器件。

文档编排

本手册的内容编排如下：

- **第 1 章：dsPIC30F 数字信号控制器**——本章将帮助您选择适合您设计的正确的 dsPIC30F 器件，或只是帮助您了解更多的关于此数字信号控制器的性能。
- **第 2 章：Microchip 开发工具**——本章介绍了 MPLAB IDE，并使您熟知相关的汇编器、编译器、链接器、软件模拟器、调试器和仿真器工具。
- **第 3 章：MPLAB 集成开发环境**——Microchip 提供了一个功能强大的 MPLAB IDE 开发环境，该开发环境完全免费！本章使用教程模式，通过创建一个项目并汇编和链接一个程序来让您熟悉 MPLAB IDE。
- **第 4 章：MPLAB SIM 软件模拟器**——MPLAB SIM30 允许您在没有 dsPIC30F 硬件的情况下调试代码。软件模拟器完全集成在 MPLAB IDE 中，本章中您将学会如何使用该软件模拟器。
- **第 5 章：MPLAB ICD 2 在线调试器**——MPLAB ICD 2 在线调试器允许您灵活地直接调试自己电路板上的 dsPIC 芯片。 MPLAB ICD 2 是一个特例，在本章及本指南中的其他动手实验教程中您将学习如何使用它。
- **第 6 章：MPLAB ICE 4000 在线仿真器**——本章帮助您着手使用 MPLAB ICE 4000 在线仿真器。 MPLAB ICE 4000 是调试 dsPIC 器件的最成熟的工具。它在执行时对指令和数据路径提供全速仿真和可视性。
- **第 7 章：MPLAB ASM30 汇编器**——本章重点阐述代码的生成。它描述了代码的一般格式并给出指令和伪指令的一些例子，这些指令可通过 MPLAB ASM30 汇编器汇编到目标代码中。
- **第 8 章：MPLAB C30 C 编译器**——本章从生成 dsPIC30F 器件的 “C”代码开始着手。本教程演示了如何使用 MPLAB C30 C 编译器将应用程序源代码和库合并以生成目标文件。
- **第 9 章：MPLAB LINK30 链接器**——本章对链接描述文件进行逐步分析，以此来了解 MPLAB LINK30 链接器。
- **附录 A：dsPICDEM 1.1 通用开发板代码**——本附录包含 dsPICDEM 1.1 通用开发板的代码示例。
- **附录 B：dsPICDEM 入门演示板代码**——本附录包含 dsPICDEM 入门演示板的代码示例。
- **附录 C：dsPICDEM 28 引脚入门演示板代码**——本附录包含 dsPICDEM 28 引脚入门演示板的代码示例。
- **附录 D：dsPICDEM 2 开发板代码**——本附录包含 dsPICDEM 2 开发板的代码示例。

© 2007 Microchip Technology Inc.

本指南使用的约定

本文档采用以下文档约定：

文档约定

说明	涵义	示例
Arial 字体：		
斜体字	参考书目	*MPLAB® IDE User's Guide*
	需强调的文字	... *仅有*的编译器 ...
首字母大写	窗口	Output 窗口
	对话框	Settings 对话框
	菜单选项	选择 Enable Programmer
引用	窗口或对话框中的字段名	“Save project before build”
带右尖括号且带有下划线的斜体文字	菜单路径	*File>Save*
粗体字	对话框按钮	单击 **OK**
	选项卡	单击 **Power** 选项卡
尖括号 < > 括起的文字	键盘上的键	按 <Enter>，<F1>
Courier 字体：		
常规 Courier	源代码示例	`#define START`
	文件名	`autoexec.bat`
	文件路径	`c:\mcc18\h`
	关键字	`_asm, _endasm, static`
	命令行选项	`-Opa+, -Opa-`
	位值	`0, 1`
	常数	`0xFF, ‘A’`
斜体 Courier	可变参数	*`file`*`.o`，其中 *`file`* 可以是任一有效文件名
方括号 []	可选参数	`mcc18 [options]` *`file`* `[options]`
花括号和竖线：{|}	选择互斥参数； “或”选择	`errorlevel {0|1}`
省略号 ...	代替重复文字	`var_name [, var_name...]`
	表示由用户提供的代码	`void main (void) { ... }`

推荐读物

以下 Microchip 文档均已提供，并建议读者作为补充参考资料。

dsPIC30F 系列参考手册 （DS70046E_CN）

请参考本文档以获取 dsPIC30F 器件操作的详细信息。本手册介绍了 dsPIC30F DSC 系列器件的架构和外设模块的工作原理，但并不涉及每个器件工作的具体情况。欲知某个器件的具体信息，请参见下述相应器件数据手册。

dsPIC30F Data Sheet, Motor Control and Power Conversion Family （DS70082）

有关 dsPIC30F 电机控制和电源转换器件的信息，请参考本文档。本数据手册中的参考信息包括：

- 器件存储器映射
- 器件引脚排列和封装细节
- 器件电气规范
- 器件上包含的外设列表

dsPIC30F Data Sheet, General Purpose and Sensor Families （DS70083）

有关 dsPIC30F 传感器和通用器件的信息，请参考本文档。本数据手册中的参考信息包括：

- 器件存储器映射
- 器件引脚排列和封装细节
- 器件电气规范
- 器件上包含的外设列表

dsPIC30F 程序员参考手册 （DS70157B_CN）

本手册是 dsPIC30F 16 位 DSC 系列器件软件开发人员的参考手册。本手册详细介绍了指令集，并提供了通用信息以帮助用户进行 dsPIC30F DSC 系列器件的软件开发。

dsPIC30F 系列概述 , dsPIC 高性能 16 位数字信号控制器 （DS70043F_CN）

本文档概述了 dsPIC® 产品系列的特性和功能。它帮助您确定不同的 16 位 dsPIC 数字信号控制器系列所适合的具体产品应用。有关各个功能的详细信息，请参见《dsPIC30F 系列参考手册》（DS70046E_CN）。

MPLAB® ASM30、 MPLAB® LINK30 和实用程序用户指南 （DS51317F_CN）

本文档详细描述了 Microchip Technology 的基于 GNU 技术的 dsPIC 器件的语言工具。所讨论的语言工具包括：

- MPLAB ASM30 汇编器
- MPLAB LINK30 链接器
- MPLAB LIB30 归档器 / 库管理器
- 其他实用程序

MPLAB® C30 C 编译器用户指南 （DS51284F_CN）

本文档的目的是帮助您使用 Microchip 针对 dsPIC 器件的 MPLAB C30 C 编译器开发应用程序。 MPLAB C30 C 编译器是一款基于 GNU 的语言工具，它以自由软件基金会（Free Software Foundation， FSF）的源代码为基础。关于 FSF 的更多详细信息可登录网站 www.fsf.org 查看。

自述文件 （Readme）

关于使用其他工具的最新信息，请阅读与工具相关的 Readme 文件，该文件位于 MPLAB IDE 安装目录的 Readme 子目录下。此 Readme 文件包含了本用户指南中可能没有包括的最新信息和已发行版本。

© 2007 Microchip Technology Inc.

MICROCHIP 网站

Microchip 网站 （www.microchip.com）为客户提供在线支持。客户可通过该网站方便地获取文件和信息。只要使用常用的因特网浏览器即可访问。网站提供以下信息：

- **产品支持**——数据手册和勘误表、应用笔记和样本程序、设计资源、用户指南以及硬件支持文档、最新的软件版本以及存档软件
- **一般技术支持**——常见问题 （FAQ）、技术支持请求、在线讨论组以及 Microchip 顾问计划成员名单
- **Microchip 业务**——产品选型和订购指南、最新 Microchip 新闻稿、研讨会和活动安排表、 Microchip 销售办事处、代理商以及工厂代表列表

开发系统变更通知客户服务

Microchip 的客户通知服务有助于客户了解 Microchip 产品的最新信息。注册客户可在他们感兴趣的某个产品系列或开发工具发生变更、更新、发布新版本或勘误表时，收到电子邮件通知。

欲注册，请登录 Microchip 网站 www.microchip.com，点击 “变更通知客户（Customer Change Notification）”服务并按照注册说明完成注册。

开发系统产品的类别如下：

- **编译器**——Microchip C 编译器及其他语言工具的最新信息，包括 MPLAB C17、MPLAB C18 和 MPLAB C30 C 编译器、 MPASM™ 和 MPLAB ASM30 汇编器、MPLINK™和MPLAB LINK30目标链接器，以及MPLIB™和MPLAB LIB30目标库管理器。
- **仿真器**——Microchip 在线仿真器的最新信息，包括 MPLAB ICE 2000 和 MPLAB ICE 4000。
- **在线调试器**——Microchip 在线调试器 MPLAB ICD 2 的最新信息。
- **MPLAB IDE**——关于支持开发系统工具的 Windows® 集成开发环境 Microchip MPLAB IDE 的最新信息，主要针对 MPLAB IDE、 MPLAB SIM 和 MPLAB SIM30 模拟器、 MPLAB IDE 项目管理器以及一般编辑和调试功能。
- **编程器**——Microchip 编程器的最新信息，包括 MPLAB PM3、 PRO MATE® II 器件编程器以及 PICSTART® Plus 开发编程器。

客户支持

Microchip 产品的用户可以通过以下渠道获得帮助：

- 代理商或代表
- 当地销售办事处
- 应用工程师 （FAE）
- 技术支持
- 开发系统信息热线

客户应联系其代理商、代表或应用工程师 （FAE）寻求支持。当地销售办事处也可为客户提供帮助。本文档后附有销售办事处的联系方式。

也可通过 http://support.microchip.com 获得网上技术支持。

© 2007 Microchip Technology Inc.

第 1 章 dsPIC30F 数字信号控制器

1.1 简介

16 位 dsPIC30F 数字信号控制器（Digital Signal Controller，DSC）是 Microchip 最新最先进的处理器系列。在本章中，您将了解到此处理器的特性、各种可用的 dsPIC 器件以及如何为您的应用选择最合适的 dsPIC 器件。

dsPIC30F 是一款高性能的 16 位处理器，它通过单片机最根本的实时控制功能来提供真正的 DSP 功能。可区分优先级的中断、扩展的内置外设和电源管理功能与功能齐全的 DSP 引擎相结合。两个 40 位累加器、单周期 16x16 MAC、 40 位桶形移位器、同时取两个操作数的操作和零开销循环控制也是促使它成为功能强大的 DSC 的因素。

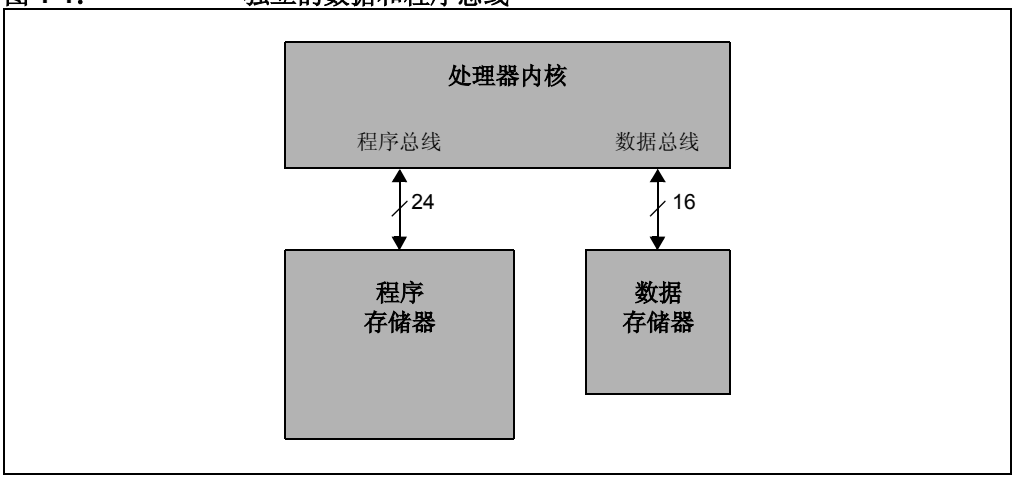
如果您无法理解这些术语，不必担心，本章中将更加详细地介绍它们。

1.2 架构

1.2.1 哈佛架构

dsPIC 处理器采用改进的哈佛架构，具有独立的程序和数据存储器总线，如图 1-1 所示。

图 1-1： 独立的数据和程序总线



哈佛架构允许使用不同大小的数据字 （16 位）和指令字 （24 位）。这种设计提高了指令集的效率。由于 dsPIC 处理器在执行访问数据 RAM 的当前指令的同时可以从程序存储器中预取下一条指令，这就加快了处理速度。

1.2.2 程序存储器和程序计数器

程序计数器 （Program Counter， PC）为 24 位宽，可寻址最大 4M × 24 位用户程序存储空间。对于每条 24 位指令，程序计数器以 2 为增量递增，这简化了对存储在程序存储器中的 16 位数据常量的寻址操作。程序存储空间包含了复位单元、中断向量表、用户程序存储空间、数据 EEPROM 以及配置存储空间 （见图 1-2 中的 “程序存储器映射”）。

处理器从复位单元 0x000000 处开始执行程序。用户可通过一条 GOTO 指令 （转移到代码开始处）对此单元进行编程。复位单元处的 GOTO 指令后跟中断向量表。代码的程序存储空间位于向量表后，起始地址为 0x100。

使用 DO 和 REPEAT 指令实现最低开销的程序循环结构；这两条指令在任何时间都可被中断。这些特性使重复 DSP 算法非常有效，并保持了处理实时事件的能力。

1.2.3 数据存储器

数据空间为 64 KB，大多数指令将其看作一个线性地址空间。当使用某些 DSP 指令，如我们熟知的 DSP 乘法指令时，该存储空间被分成两块，分别称为 X 和 Y 数据空间（见图 1-2 中的 “数据存储器映射”）。因此，这些 DSP 指令支持双操作数读操作，即，同一条指令可同时从 X 存储空间和 Y 存储空间中取数据。对于任何给定器件， X 和 Y 数据空间的边界是固定的。当未执行 DSP 指令时，所有存储空间都被看作是一个 X 存储空间。

数据存储器的前 2 KB 分配给特殊功能寄存器 （Special Function Registers， SFR）。SFR 是 dsPIC 器件内核和外设功能的控制和状态寄存器。

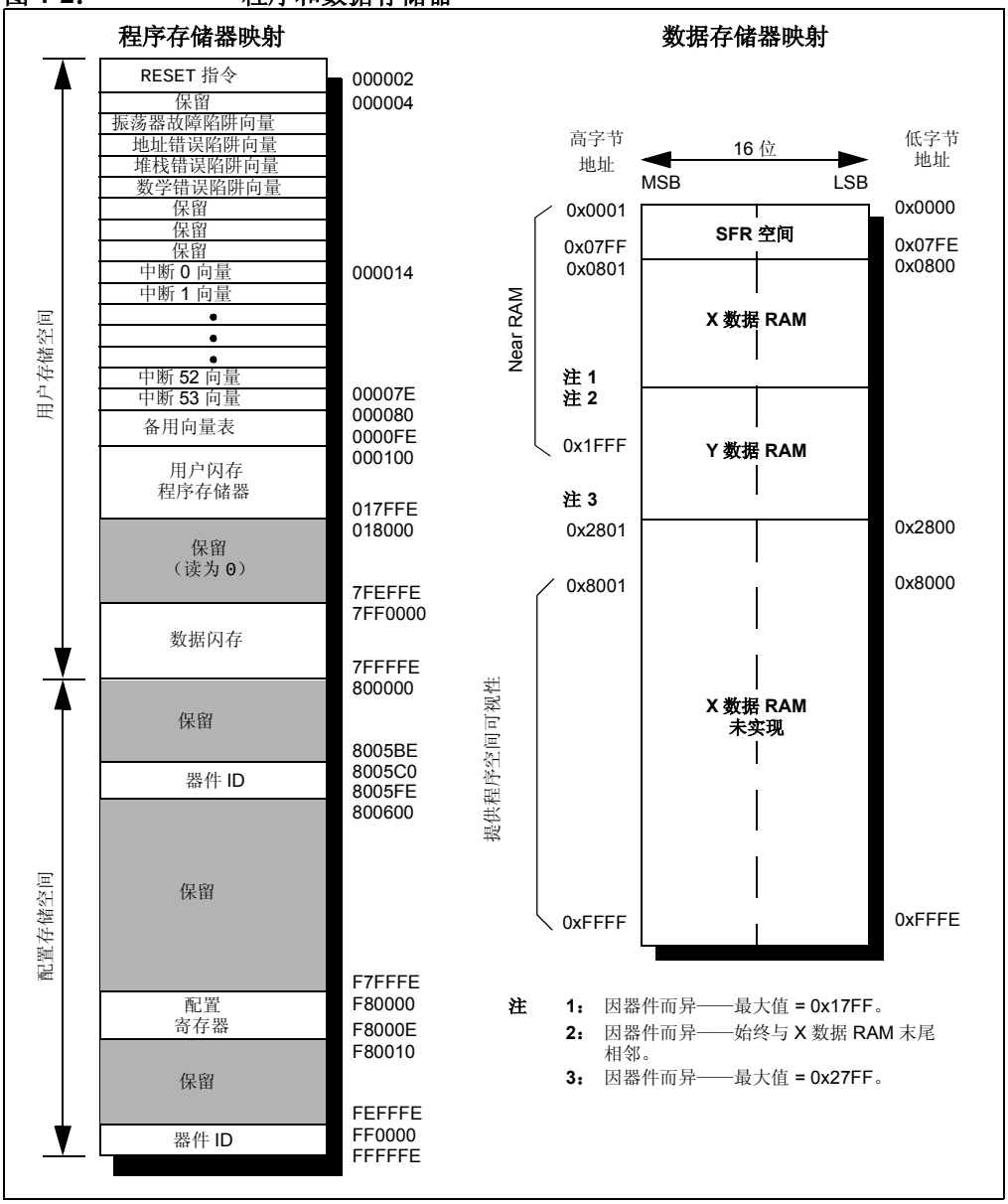
在 SFR 之后，最大 8 KB 的存储空间被用作数据 RAM。它可作为存储数据的通用存储器。对于 DSP 指令，它被分成 X 和 Y 存储空间。

数据空间的前 8 KB （即， SFR 的 2 KB 和数据 RAM 的前 6 KB）被称为 Near RAM。此 RAM 区域可直接通过文件寄存器指令访问。某些指令无法直接访问非 Near 的 RAM，必须使用间接寻址来访问。

数据 RAM 空间的最后 32 KB 未实现，但可被映射到程序空间上，以用于程序空间可视性 （Program Space Visibility， PSV）。 PSV 允许对程序存储器中的数据表进行读操作，就好像它们在数据 RAM 中一样。（此特性对于访问 DSP 滤波器系数相当有用。）

© 2007 Microchip Technology Inc.

图 1-2：　　程序和数据存储器



1.2.4　　工作寄存器阵列

dsPIC 器件具有 16 个 16 位工作寄存器。最后一个工作寄存器 （W15）始终用作软件堆栈指针。 W15 不能用于其他目的。其余工作寄存器可用作数据寄存器、数据地址指针或地址偏移量寄存器。软件堆栈用于存储中断和调用的返回地址，在执行 PUSH 和 POP 指令时，会对该堆栈进行操作。 C 编译器将软件堆栈广泛用于存储局部变量。

1.2.5　　数据寻址模式

CPU 支持固有 （无操作数）寻址、相对寻址、立即数寻址、存储器直接寻址、寄存器直接寻址和寄存器间接寻址模式。不必担心，它们并没有听起来那么复杂。每条对数据存储器进行寻址的指令都可以使用某种可用的寻址模式。每条指令最多支持 6 种寻址模式。工作寄存器则广泛地用作间接寻址模式地址指针。它们可被修改 （如递增）并用作同一指令中的指针。

1.2.6 模寻址和位反转寻址

模寻址允许实现循环缓冲区，这样省去了检查缓冲区边界所需的处理器开销。缓冲区的指针可设置为到达缓冲区末端后自动返回到缓冲区开始处，反之亦然。此操作可在 X 和 Y 存储空间中进行，从而极大地减少了 DSP 算法的开销。

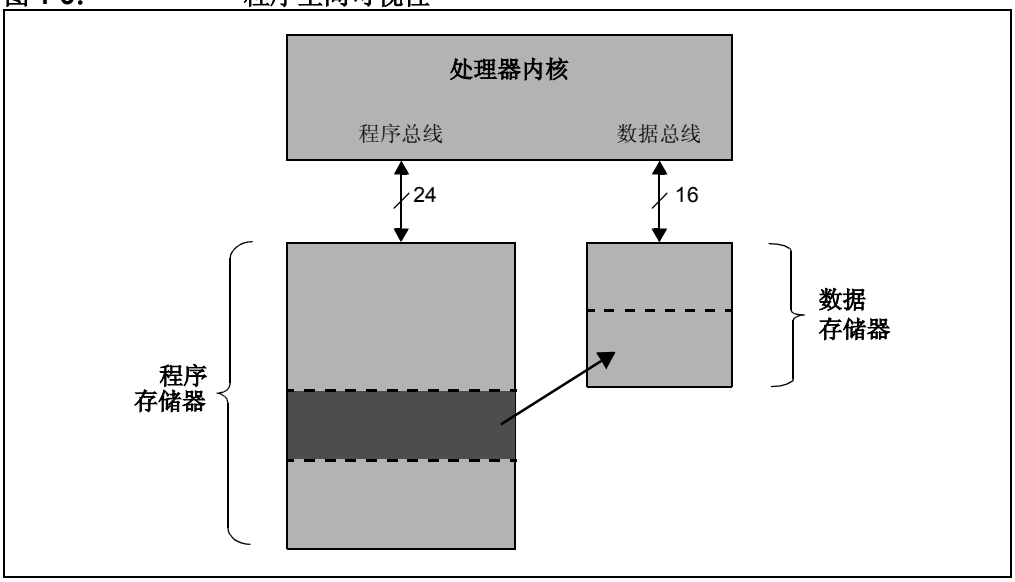
位反转寻址极大地简化了基 2 FFT 算法对输入或输出数据的重新排序。 X 存储空间支持位反转寻址。

1.2.7 程序空间可视性

可选择将数据存储空间映射的高 32 KB 映射到任何 16K 程序字 （32 KB）边界的程序空间内，该边界是由 8 位程序空间可视性页 （Program Space Visibility Page，PSVPAG）寄存器定义的。

程序空间到数据空间的映射功能使得任何指令都能象访问数据空间一样访问程序空间。此功能对于查找表，尤其是 DSP 算法中滤波器系数表非常有用。

图 1-3： 程序空间可视性



1.2.8 指令集

dsPIC30F 指令集有两类指令：MCU 指令和 DSP 指令。这两类指令可无缝地集成到该架构中并从单一执行单元处执行。指令集包括很多寻址模式，指令集的设计旨在优化 C 编译器的效率。

除了少数特例外，大多数指令的执行都在一个指令周期内完成。特例是：更改程序流的指令 （BRA 和 CALL 等）、双字传送指令 （MOV.D）和程序存储器读 / 写 （表）指令。

对于大多数指令而言，在每个指令周期内， dsPIC30F 都能执行一次数据存储器读操作、一次工作寄存器数据读操作、一次数据存储器写操作和一次程序存储器 （指令）读操作。因此，可以支持 3 个操作数的指令，使得 A + B = C 类操作能在单周期内完成。

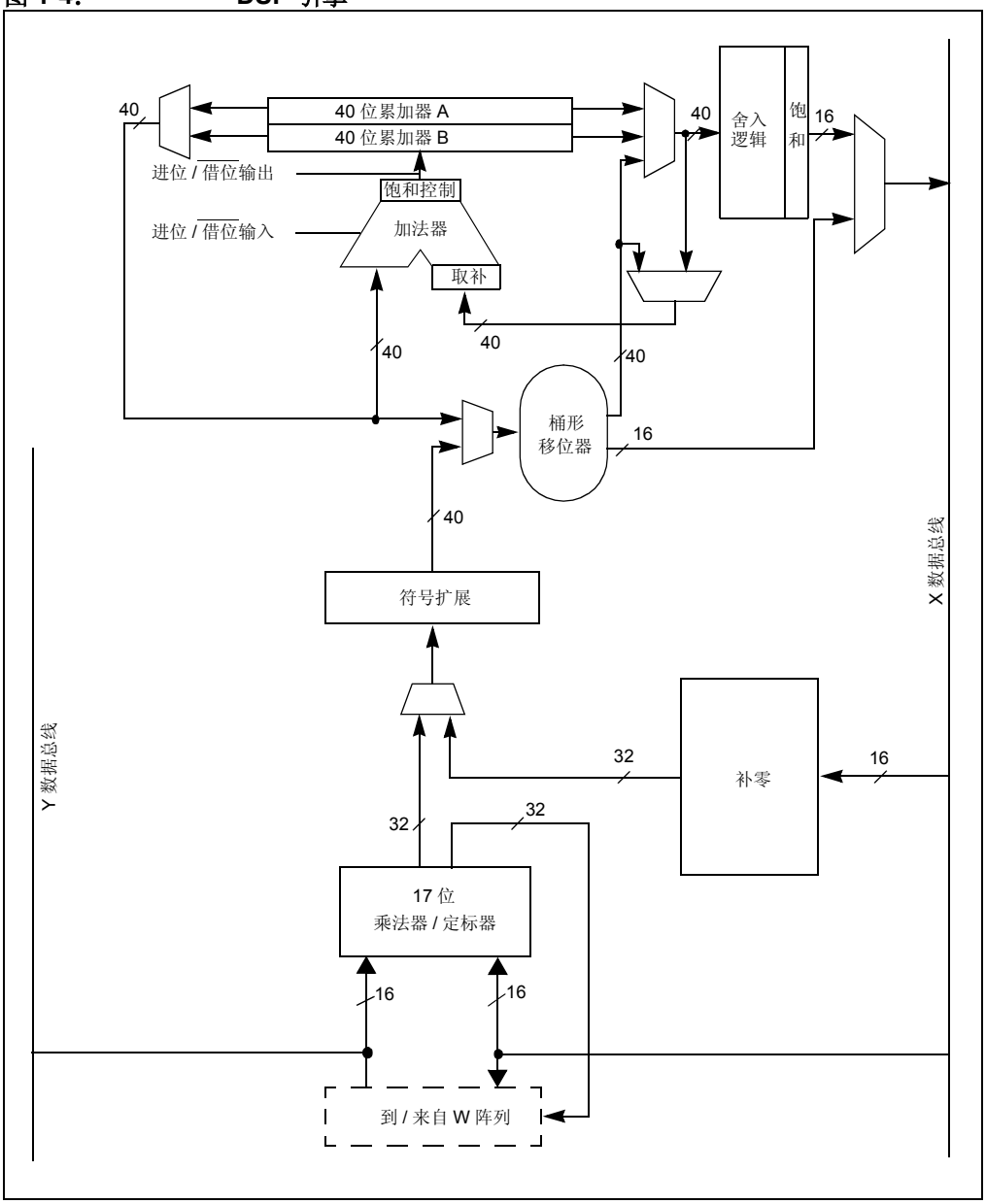
© 2007 Microchip Technology Inc.

1.2.9 DSP 引擎

DSP 引擎（图 1-4）具有一个高速 17 位 x 17 位定点数据乘法器、一个 40 位 ALU（算术逻辑单元）、两个 40 位饱和累加器和一个 40 位双向桶形移位寄存器。该桶形移位器能在一个周期内将一个 40 位的值右移最多 15 位或左移最多 16 位。

DSP 指令可以无缝地与所有其他指令一起操作，且设计为能获得最佳实时执行性能。MAC 指令和其他相关指令可以同时从存储器取出两个数据操作数，将两个 W 寄存器相乘。由于对于 DSP 指令，数据存储区被分成 X 和 Y 存储空间，因此这种操作是可行的。

图 1-4： DSP 引擎



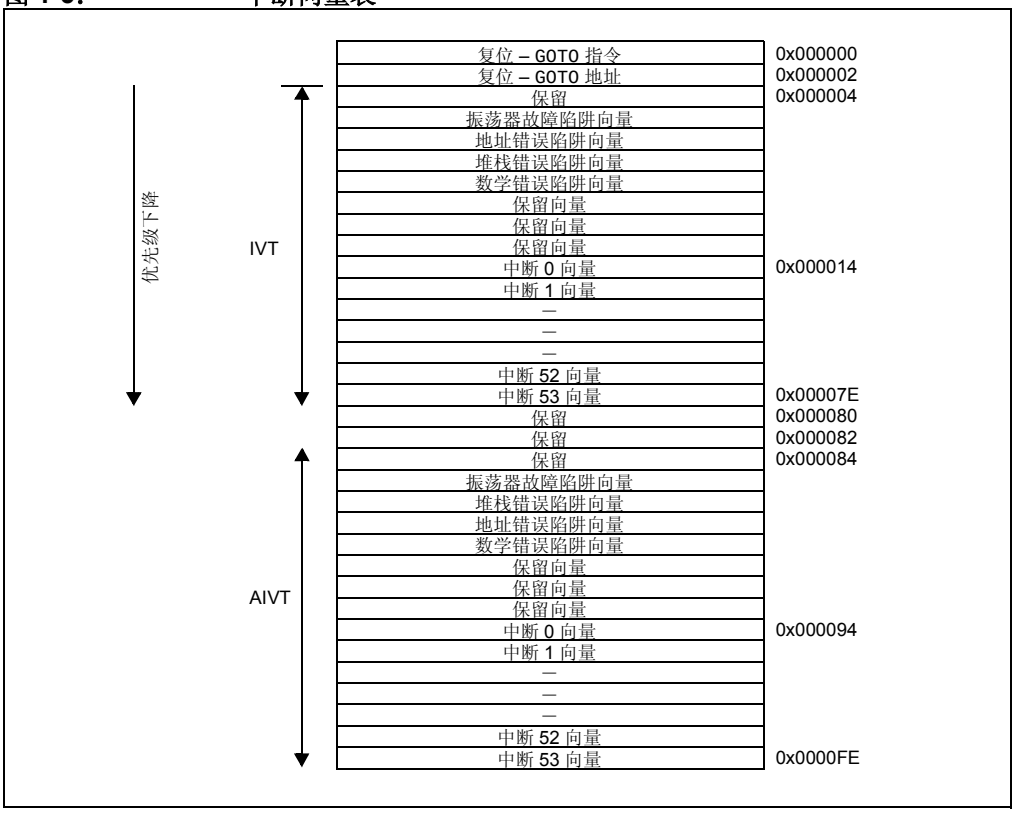
1.2.10 中断

dsPIC30F 具有一个向量中断机制。每个中断源都有自己的向量，且可被动态分配 7 个优先级中的一个。中断入口地址和返回延时是固定的，为实时应用提供确定的时序。

中断向量表 （Interrupt Vector Table， IVT）位于程序存储器中，且紧跟在复位单元处的指令之后，如图 1-5 所示。 IVT 包含 62 个向量，由最多 8 个不可屏蔽 （始终允许）错误陷阱向量和最多 54 个中断源组成。每个中断向量都包含相关中断服务程序（Interrupt Service Routine， ISR）的 24 位宽的起始地址。

备用中断向量表 （Alternate Interrupt Vector Table ， AIVT）位于程序存储器中的 IVT 之后。如果 ALTIVT 位置 1，则所有中断和错误陷阱将使用备用向量而不是默认向量。备用向量与默认向量的结构相同，并提供了一种无需再编程中断向量就可在应用和测试、设置或引导加载环境之间切换的方法。

图 1-5： 中断向量表



1.2.11 系统和功耗管理

现代应用通常需要灵活的工作模式，以降低电池供电器件的功耗、降低 EMI 和处理故障。 dsPIC 器件具有多种系统和功耗管理功能。例如，它具有若干种带有时钟切换和振荡器故障检测功能的振荡器模式。具有多种可选择性地关断和唤醒处理器和外设的节能模式。还具有其他安全功能，例如，低电压检测、欠压复位、看门狗定时器复位以及若干个错误陷阱。

© 2007 Microchip Technology Inc.

1.2.12 外设

dsPIC 器件可与多种外设一起使用以满足不同类型的应用的需要。这些主要外设包括：

- I/O 端口
- 定时器
- 输入捕捉
- 输出比较 /PWM
- 电机控制 /PWM
- 正交编码器
- 10 位或 12 位 A/D 转换器
- UART
- SPI™
- I^2C™
- 数据转换器 （CODEC）接口
- 控制器局域网 （Controller Area Network， CAN）

每个器件都或多或少的具有这些外设中的一个或几个。

1.3 器件分类

dsPIC 器件可分为三大系列，这种类型划分有助于您挑选出最适合您应用的器件：

- 通用
- 电机控制 / 电源转换
- 传感器

1.3.1 通用系列

通用器件是 40 至 80 引脚的器件，是各种 16 位嵌入式应用的理想选择。此器件系列包含：

- 12 位 100 ksps A/D 转换器
- 双 UART
- CODEC 接口 （多数器件）
- CAN 接口 （多数器件）
- 定时器、输入捕捉和输出比较
- UART、 SPI 和 I^2C 串行端口

具有 CODEC 接口的器件可支持多种音频应用。

1.3.2 电机控制和电源转换系列

电机控制器件是 28 至 80 引脚的器件，用于支持电机控制应用。它们也适用于不间断电源 （ Uninterruptible Power Supplies， UPS）、逆变器、开关电源及相关设备。此器件系列包含：

- 10 位 500 ksps A/D 转换器
- 电机控制 PWM
- 正交编码器
- 定时器、输入捕捉和输出比较
- UART、 SPI、 I^2C 以及 CAN 串行接口

1.3.3 传感器系列

传感器器件是小型 18 至 28 引脚器件，用于支持低成本的嵌入式控制应用。它们具有通用系列的大部分特性，但只具有少数外设。此器件系列包含：

- 12 位 100 ksps A/D 转换器
- 定时器、输入捕捉和输出比较
- UART、 SPI 和 I^2C 串行端口

1.4 应用

现在，您已对 dsPIC 架构有了基本的了解，可以考虑 dsPIC 器件是否适合您的特定应用。 dsPIC 器件的应用场合举不胜举，但这里只讨论 dsPIC 器件的最常见应用：

1.4.1 电机控制

对于无法由基本 8 位单片机实现的电机控制功能而言， dsPIC 器件是一种理想选择。无刷直流电机、交流感应电机以及开关磁阻电机都可用 dsPIC 器件来控制。该应用可能需要无传感器控制、转矩管理、变速、定位或伺服控制。 dsPIC 器件还可处理噪音消除和能效应用。

1.4.2 电源转换和监控

快速 A/D 转换器和多个 PWM 模块使得 dsPIC 器件非常适用于许多电源转换和电源管理应用。 dsPIC 器件还可控制复杂设备的不间断电源 （UPS）、逆变器和电源管理单元。

1.4.3 Internet 连接性

Microchip 现成的 TCP/IP、以太网驱动器以及软件调制解调器应用程序库支持用于 Internet 连接的以太网和调制解调器应用。

1.4.4 语音和音频

dsPIC 器件可支持多种音频应用，例如消除噪音和回声、语音识别以及语音回放。它还可用作高端音频应用中主 DSP 的附属芯片，以处理其他任务，例如数字调谐、均衡器等等。

1.4.5 传感器控制

较小的 dsPIC 器件是高级传感器控制的理想选择。 A/D 转换器和串行通信外设与电源管理功能结合使用，可创建智能传感器接口模块。

1.4.6 汽车

Microchip Technology Inc. 通过 QS-9000 和 ISO/TS-16949 认证，生产汽车温度级部件。通常，我们的产品具有很长的使用寿命以支持汽车应用要求的典型产品使用寿命。

© 2007 Microchip Technology Inc.

第 2 章 Microchip 开发工具

2.1 简介

现在您已选择了一款符合您应用的 dsPIC 器件，需要开发工具的支持。开发过程可分为以下三步：

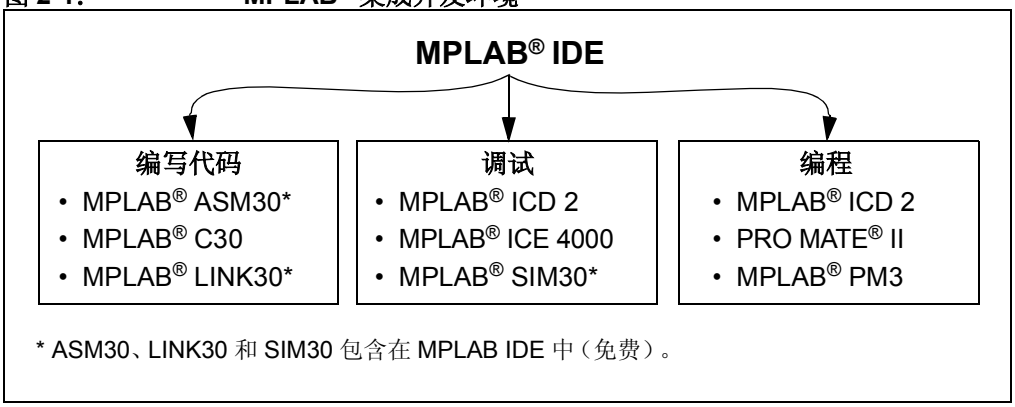
- 编写代码
- 调试代码
- 编程器件

此时，您需要一种工具实现上述功能，而 MPLAB® 集成开发环境 （IDE）就是支持开发工具的关键。首先，您会发现 MPLAB ICD 2 在线调试器是高性价比的调试和编程解决方案。 MPLAB ICD 2 可与任何 dsPIC 开发板配合使用，从而成为理想的学习平台。

2.2 MPLAB IDE

MPLAB IDE 允许您自始至终在同一个环境中开发项目。您无需使用单独的编辑器、汇编器 / 编译器和编程实用程序来创建、调试及编程应用程序。 MPLAB IDE 可以控制该过程的所有事宜，如图 2-1 所示。记住， MPLAB IDE 是**免费**的！

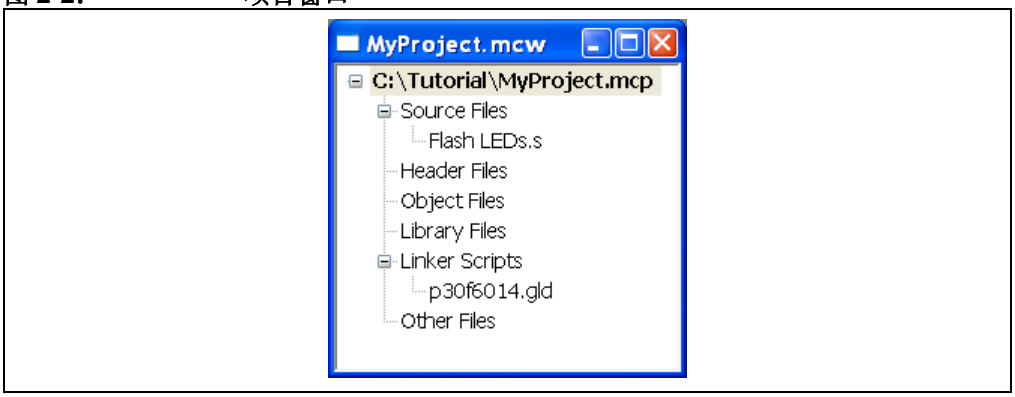
图 2-1： MPLAB® 集成开发环境



2.2.1 项目

MPLAB IDE 包含创建及使用项目和工作区的工具。工作区存储了项目的所有设置，所以您可以毫不费力地在项目间进行切换。通过 Project Wizard （项目向导）您只需点击几次鼠标即可轻松创建项目。您还可以使用项目窗口视图 （图 2-2）方便地添加文件到项目中以及删除项目中的文件。

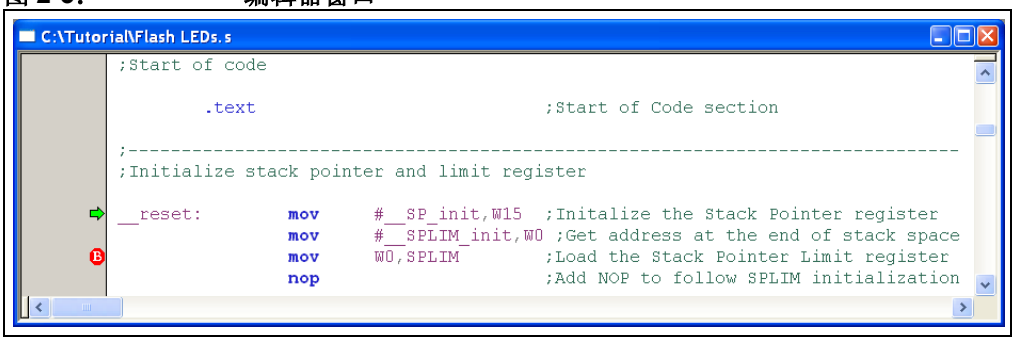
图 2-2： 项目窗口



2.2.2 编辑器

编辑器是 MPLAB IDE 的组成部分，它提供了多种功能使得代码编写更为轻松：语法突出显示、自动缩进、大括号匹配、块注释、书签以及许多其他功能。编辑器窗口（图 2-3）直接支持工具调试、当前执行位置和断点以及跟踪点显示，以及悬停鼠标来查看变量等等。

图 2-3： 编辑器窗口



2.3 语言工具

2.3.1 汇编器 / 链接器

MPLAB IDE 中包含基于行业标准 GNU 工具包的 MPLAB ASM30 汇编器和 MPLAB LINK30 链接器。您无需购买其他的软件来开发代码。MPLAB ASM30 汇编器将源文件汇编为目标文件，并由链接器将目标文件与项目中可能包含的库 （归档）文件一起转换为 hex 输出文件。

© 2007 Microchip Technology Inc.

2.3.2 编译器

Microchip 为需要 C 编译器的用户提供了 MPLAB C30 C 编译器，用户需单独购买它。该编译器使得代码更容易移植、阅读和维护。 MPLAB C30 C 编译器可以在 MPLAB IDE 内使用，以无缝地集成代码开发、调试和编程。

除了 MPLAB C30 C 编译器外，我们还可以使用由第三方生产商提供的 dsPIC 编译器。 HI-TECH Software （www.htsoft.com）、 CCS Inc.（www.ccsinfo.com）和 IAR Systems （www.iar.com）都生产支持 dsPIC 系列的 C 编译器。

对于熟悉这些生产商的 PIC® 编译器的用户来说，他们的 dsPIC 产品也是一种不错的选择。您不必费心学习一个全新的编译器，这些编译器本来就很容易移植到 dsPIC 系列。

2.3.3 模板、头文件和链接描述文件

想开始编写一段代码，却不知道如何开始？那么请查看 MPLAB ASM30 汇编器目录中的模板文件，这些模板文件是 MPLAB IDE 的一部分。可以复制这些模板以形成您自己的代码的基础。您还可以在该目录下找到处理器头文件；这些文件用来定义所有寄存器和位名称以及它们的位置，使之与数据手册中的定义保持一致。链接描述文件为链接器提供一个 dsPIC 器件的存储器映射，以便正确地自动放置代码和数据。

2.3.4 应用笔记

是否不确定该如何实现您的设计？只是想提高您的设计技能？想消磨时间？那么请登录我们的网站 （www.microchip.com）获取最新的应用笔记。我们会不断添加更多应用笔记，以便提供范例，使您在日益膨胀的应用中学会如何使用 dsPIC 器件。

2.4 调试工具

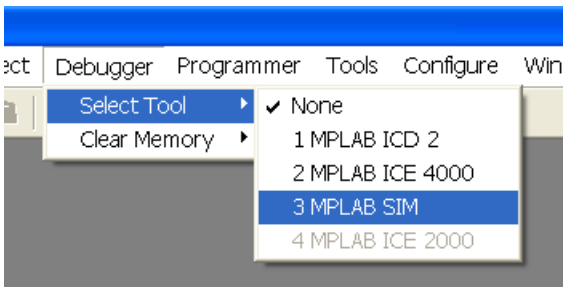
MPLAB IDE 可与三种不同的调试工具配合使用：软件模拟器 （MPLAB SIM30 软件模拟器）、在线调试器 （MPLAB ICD 2）和在线仿真器 （MPLAB ICE 4000）。所有这些调试器都允许您单步运行代码 （直到您选择暂停或遇到断点为止）、观察寄存器更新以及查看存储器内容。每个调试器都有自己独特的优缺点。

2.4.1 MPLAB SIM30 软件模拟器

MPLAB SIM30 软件模拟器是包含在 MPLAB IDE 中的一款功能强大的调试工具 （图 2-4）。该软件模拟器在 PC 机上运行，并模拟 dsPIC 器件上的代码执行。它不仅可以模拟代码执行，还可以响应模拟的外部输入和外设操作，以及测量代码执行时间。

MPLAB SIM30 软件模拟器为在没有外部硬件的情况下调试代码提供了一种快速易行的方法。尤其是当文件中出现重复数据时对于测试数学运算和 DSP 函数特别有用。通常，由于重现数据比较困难，故用实际硬件中的模拟信号测试代码是极具挑战性的。但如果有了采样数据或合成数据作为激励，测试就会变得简单得多。

图 2-4：　　MPLAB® IDE 中的软件模拟器选择菜单



MPLAB SIM30 软件模拟器除具有所有的基本调试功能之外，还具有一些高级功能：

- 跑表——用于测量代码执行时间。
- 激励——用于模拟外部输入和数据接收。
- 跟踪——用于查看执行记录。

图 2-5：　　MPLAB® ICE 4000

2.4.2　　MPLAB ICE 4000

MPLAB ICE 4000 在线仿真器（图 2-5）是一款功能齐全的调试工具，它可以全速仿真所有 dsPIC30F 器件，是我们提供的一款功能最强大的调试工具，并为处理器提供了极好的可视性。它可通过 USB 接口完全集成到 MPLAB IDE 中，该 USB 接口允许 MPLAB IDE 快速更新存储器和数据视图。

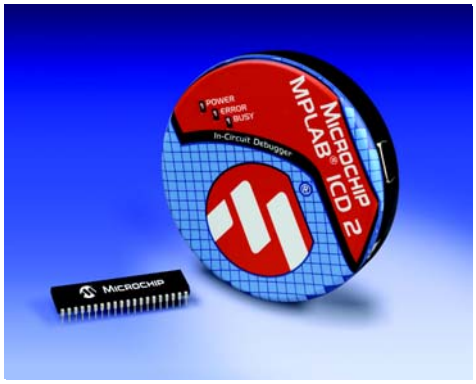
MPLAB ICE 4000 是一个模块化系统，它支持多种处理器和封装选项。您可以使用我们网站（www.microchip.com）上的 *“产品选型指南”* 选择正确的处理器模块、器件适配器和转换插座，以仿真您希望使用的特定器件。

MPLAB ICE 4000 除具有所有的基本调试功能之外，还具有一些高级功能：

- 复杂触发设置——检测事件顺序，如写入寄存器的顺序。
- 跑表——用于测量代码执行时间。
- 跟踪——用于查看执行记录。
- 逻辑探针——由外部信号触发，生成测试器件的触发信号。

© 2007 Microchip Technology Inc.

图 2-6：　　MPLAB® ICD 2

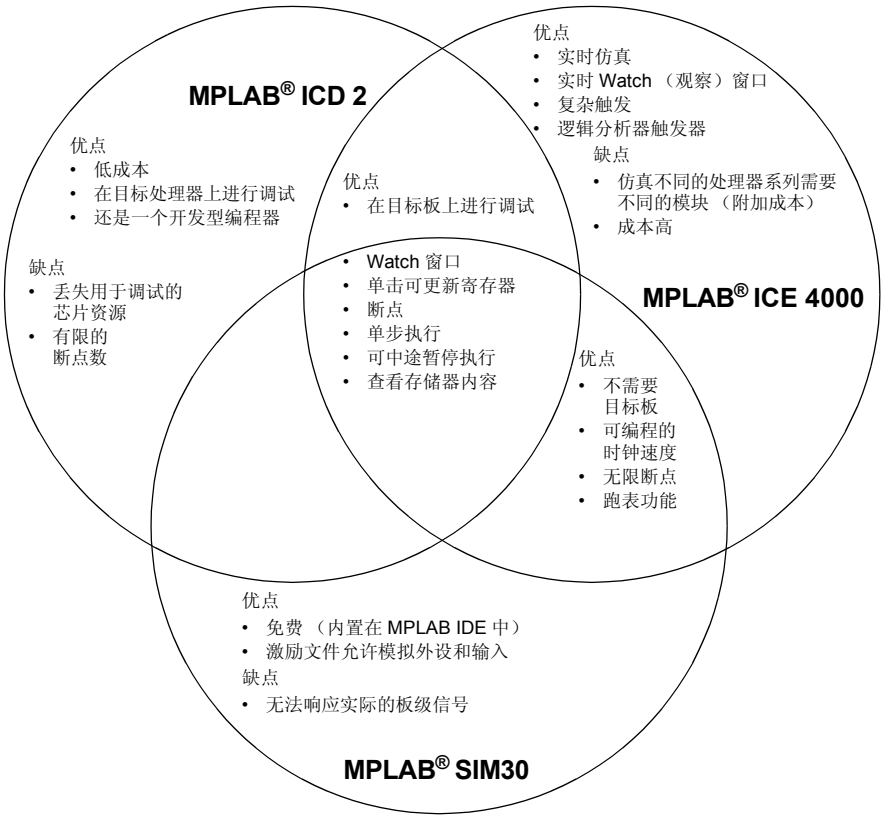


2.4.3　　MPLAB ICD 2

MPLAB ICD 2 在线调试器是允许在目标电路板上测试代码的一款高性价比的调试工具。如果用户不想增加与 MPLAB ICE 4000 相关的成本，并且不需要 MPLAB ICE 4000 的高级功能，那么 MPLAB ICD 2 是一个可行的备选工具。MPLAB ICD 2 允许您直接在目标板上调试 dsPIC 器件。您还可以使用它在线编程器件。

虽然 MPLAB ICD 2 具有基本的调试功能，但是它不具有 MPLAB ICE 4000 的一些功能如跟踪存储区和复杂触发功能。同样，免费的 MPLAB SIM30 软件模拟器允许您调试代码，但是缺少 MPLAB ICD 2 中包含的功能。图 2-7 是这三种调试工具的综合比较。

图 2-7：　　MPLAB® 调试工具比较



2.5 编程工具

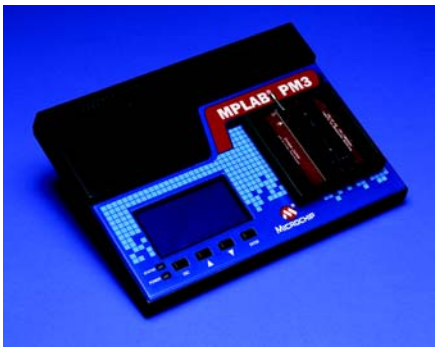
下列两个编程器可以和 MPLAB IDE 配合使用来编程 dsPIC 器件：MPLAB PM3 和 MPLAB ICD 2。每个编程器都有各自的优缺点。

MPLAB PM3 可以编程所有封装类型的器件，并且比 MPLAB ICD 2 具有更多的编程选项和存储器。但是两者都可以对器件进行在线编程。 MPLAB ICD 2 是一个可以对器件进行在线编程的在线调试器。

注： 通常， MPLAB PM3 是生产编程的最佳选择。而 MPLAB ICD 2 是在开发期间测试代码的最佳选择 （如果目标板支持在线编程的话）。

老版本的 PRO MATE II 编程器也支持 dsPIC 器件，但是已经被新型 MPLAB PM3 所取代。

图 2-8： MPLAB® PM3 通用器件编程器



2.5.1 MPLAB PM3 通用器件编程器

对于想要购买生产编程器的用户来说， MPLAB PM3 （图 2-8）是首选。它由一个基本的编程器单元和交互式插座模块组成，从而支持各种器件封装。它由 MPLAB IDE 或命令行实用程序控制，也可以独立工作。 MPLAB PM3 具有以下功能：

- 本身支持在线串行编程。
- 串行编程惟一的 ID 号。
- 用于代码安全性的安全模式。
- 通过 USB 进行高速编程和下载。
- 便于程序存储的安全数字和多媒体卡插槽。

2.5.2 MPLAB ICD 2

MPLAB ICD 2 不但是在线调试器，还是一个低成本的开发编程器。使用 MPLAB ICD 2 可以直接在目标板上对器件进行在线编程，还可以通过通用编程模块对电路板外的 DIP 封装器件进行编程。

© 2007 Microchip Technology Inc.

2.6 开发板

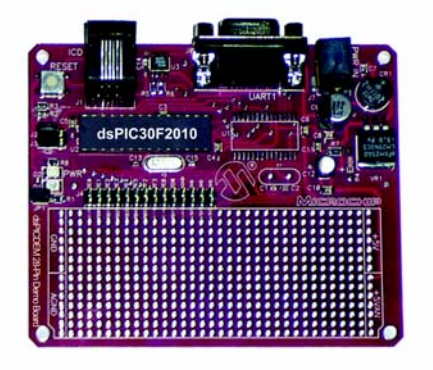
一些 dsPIC 开发板可用于简化代码开发和测试。这些开发板中包含示例代码和教程，对于新用户了解 dsPIC 处理器十分有用。

注： 本教程中提供了四个版本的代码示例，分别用于在以下四种开发板上运行：

- dsPICDEM 入门演示板
- dsPICDEM 28 引脚入门演示板
- dsPICDEM 1.1 通用开发板
- dsPICDEM 2 开发板

图 2-9： dsPICDEM™ 入门演示板

dsPICDEM™ 入门演示板



dsPICDEM™ 28 引脚入门演示板

2.6.1 dsPICDEM 入门演示板

dsPICDEM 入门演示板 （图 2-9 中的左图）是一个低成本演示板，该演示板采用 dsPIC30F6012 处理器，还包含一个连接器，以与 MPLAB ICD 2 连接以便进行编程和调试。它具有以下特性：

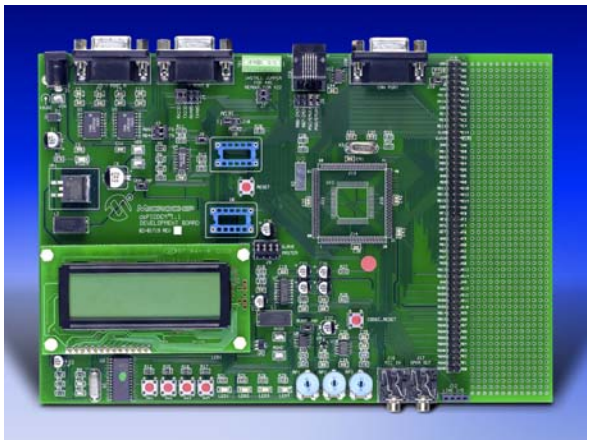
- RS-232 接口，以与 UART 配合使用
- 连接 I/O 口的开关和 LED
- 通过数字电位器控制的模拟输出
- 来自电位器的模拟输入
- 缓冲外部模拟输入
- 实验布线区

2.6.2 dsPICDEM 28 引脚入门演示板

dsPICDEM 28 引脚入门演示版 （图 2-9 中的右图）是 dsPIC 器件的又一个低成本演示板。该演示板采用 dsPIC30F2010 28 引脚处理器，还包含一个连接器，以与 MPLAB ICD 2 连接以便进行编程和调试。它具有以下特性：

- RS-232 接口
- 一个 LED
- 用于访问器件所有 I/O 引脚的连接头
- 28 引脚 DIP 插座和 28 引脚 SOIC 器件的布线焊盘
- 实验布线区

图 2-10：　　dsPICDEM™ 1.1 通用开发板



2.6.3　　dsPICDEM 1.1 通用开发板

dsPICDEM 1.1 通用开发板 （图 2-10）是一个相对成熟的通用开发板，它采用了 dsPIC30F6014 处理器。除了具有 dsPICDEM 入门演示板的大多数特性外，它还具有如下特性：

- 图形和文本 LCD 显示模块
- 用于 MPLAB ICE 4000 器件适配器的连接头引脚
- 控制器局域网 （CAN）接口
- RS-232、 RS-422 和 RS-485 UART 接口
- CODEC 模拟输入和输出，以与 DCI 接口配合使用

2.6.4　　dsPICDEM 2 开发板

dsPICDEM 2 开发板 （图 2-11）是一个多用途开发板，通过此开发板可使用 18、 28 和 40 引脚 PDIP 和 SPDIP 封装的 dsPIC30F 数字信号控制器来开发嵌入式应用。dsPICDEM 2 开发板的主要特性包括：

- dsPIC30F4011 40 引脚 PDIP 样片器件
- 用于 18、 28 和 40 引脚 DIP 器件的多个插座
- 连接 MPLAB ICD 2 在线调试器的连接器
- RS-232 接口
- CAN 接口
- 模拟 A/D 输入的温度传感器、模拟电位器和按钮开关
- LCD 屏幕和 LED 指示器

© 2007 Microchip Technology Inc.

图 2-11：　　dsPICDEM™ 2 开发板

2.6.5　　dsPICDEM MC1 电机控制开发系统

dsPICDEM MC1 电机控制开发系统为应用开发人员提供了三个主要组件，用于快速制作原型和验证无刷直流电机 （Brushless DC Motor， BLDC）、永磁交流电机（Permanent Magnet Alternating Current， PMAC）和交流感应电机 （AC Induction Motor， ACIM）应用。这三个主要组件是：

- dsPICDEM MC1 电机控制开发系统
- dsPICDEM MC1L 3 相低电压功率模块
- dsPICDEM MC1H 3 相高电压功率模块

dsPICDEM MC1 电机控制开发系统包含一片 dsPIC30F6010 器件，并支持定制的接口连接头，从而可将不同的电机功率模块连接到 PCB。该控制板还包括用于连接机械位置传感器 （诸如增量式旋转编码器和霍尔效应传感器等）的连接器，和一个用于定制电路的实验布线区。

图 2-12：　　dsPICDEM™ MC1 电机控制开发系统

dsPICDEM MC1L 三相低电压功率模块经过优化，适用于要求直流母线电压低于 50V 而输出功率高达 400W 的三相电机应用。该低电压模块旨在为 BLDC 和 PMAC 电机供电。

dsPICDEM MC1H 三相高电压功率模块经过优化，适用于要求直流母线电压低于 400V 而输出功率高达 1 kW 的三相电机应用。此高电压模块具有一个由 dsPIC30F 器件控制的有源功率因数校正电路。该功率模块旨在用于直接使用交流线电压的交流感应电机和电源逆变器应用。

图 2-13：　　dsPICDEM.net™ 网络连接开发板

2.6.6　　dsPICDEM.net 1 和 dsPICDEM.net 2 网络连接开发板

dsPICDEM.net 1 和 dsPICDEM.net 2 网络连接开发板提供了一个基础平台，用于开发和评估各种网络连接解决方案，以及实现基于 PSTN 或以太网通信通道的 TCP/IP 协议层和 V.22bis/V.22 ITU 规范。与该板一起提供的还有符合 ITU-T 标准的 V.22bis/V.22 调制解调器演示程序，该程序已经加载到安装在开发板上的 dsPIC30F6014 器件中。它用于 dsPIC 软件调制解调器和符合 ITU-T 标准的参考调制解调器之间的连接和数据传输。 dsPIC 软件调制解调器可以使用 AT 命令进行控制，它们之间的通信是通过片上 UART 通道进行的。与此开发板一起提供的还包括 CMX-MicroNet™ 网络和 FTP 服务器演示文件，把它们下载到 dsPIC30F6014 器件上，就可演示两个基于以太网数据链路层的 TCP/IP 协议栈应用。

dsPICDEM.net 1 和 dsPICDEM.net 2 都支持 dsPIC30F5013 和 dsPIC30F6014 器件，而且都有相应的以太网接口和 PSTN 接口。 dsPICDEM.net 1 支持 FCC/JATE PSTN，dsPICDEM.net 2 支持 CTR-21 PSTN。

2.6.7　　下一步——学习如何使用 MPLAB IDE

现在您已了解了用于 dsPIC 器件的开发工具，就可以开始使用 MPLAB 集成开发环境（IDE）了。

但是在进行编译、编程或调试之前您必须学会使用 MPLAB IDE，所以请继续**第 3 章“MPLAB 集成开发环境”**。

© 2007 Microchip Technology Inc.

第 3 章 MPLAB 集成开发环境

3.1 MPLAB IDE 概述

在了解 dsPIC30F 及其开发工具后，您可能很想编写一些代码。如**第 2 章 “Microchip 开发工具”**所述，在整个代码开发过程中都要使用 MPLAB IDE 软件：编写、编译、调试和编程。它具有以下主要功能：

- 项目管理器——用于组织代码文件
- 编辑器——用于输入代码
- 汇编器和链接器——用于汇编和生成代码
- 编译器接口——用于使用不同的编译器编译代码
- 软件模拟器——用于测试代码运行
- 调试器 / 仿真器接口——用于使用不同调试器或仿真器测试代码
- 编程器接口——用于使用不同编程器编程器件

我们就不花时间讨论这些枯燥、乏味的功能了，快点进入教程吧。通过实践来学习始终是有效的。

首先，安装并运行最新的 MPLAB IDE 软件。注意——这一步很重要。 MPLAB IDE 软件的更新相当频繁，以不断添加新功能和最新器件支持。您可从 Microchip Technology 网站 （www.microchip.com）下载最新版的 MPLAB IDE。

本教程的源代码文件及其文档可从 Microchip Technology 网站或光盘上获取。本教程将使用 `Flash LEDs with dsPIC30F6014.s` 文件。您不需要任何硬件，但是稍后，您也能使用 dsPICDEM 1.1 通用开发板在 dsPIC30F6014 上运行代码。

MPLAB IDE 中的 Project Wizard 是创建新项目的一个极好的方法，有了它新项目的创建变得非常简单。首先，创建一个 `C:\Tutorial` 文件夹，并将 `Flash LEDs with dsPIC30F6014.s` 文件复制到该文件夹。如果该文件是从光盘复制来的，则文件属性为只读。如果需要编辑该文件，切记更改其属性。

> **注：** 如果您拥有 dsPICDEM 入门演示板，则可使用 `Flash LEDs with dsPIC30F6012.s` 文件。如果是 dsPICDEM 28 引脚入门演示板，可使用 `Flash LED with dsPIC30F2010.s` 文件；如果是dsPICDEM 2开发板，可使用 `Flash LED with dsPIC30F4011.s` 文件。这些文件包含的代码和功能非常相似。

3.2 项目和工作区

通常， MPLAB IDE 中的所有操作都是在项目内完成的。

项目包含了编译应用程序所需的文件 （源代码和链接描述文件等），还包含这些文件与各种编译工具 （语言工具和链接器）和编译选项 （这些工具的设置）之间的对应关系。

工作区包含一个或多个项目，及选定器件、调试工具和 / 或编程器、打开窗口及其位置的信息，以及其他 IDE 配置设置。通常，一个工作区只有一个项目。（可通过菜单 *Configure>Settings* 进行更改）。

3.3 创建项目

3.3.1 项目和工作区

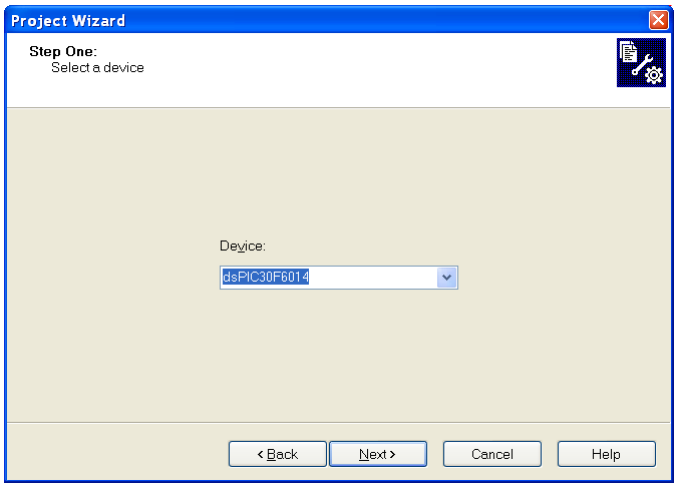
启动 MPLAB IDE 并使用 *File>Close Workspace* 菜单关闭任何已打开的工作区。然后使用 *Project>Project Wizard* 菜单启动 Project Wizard。出现 Welcome （欢迎）屏幕时，单击 **Next** （下一步）继续。

步骤 1——选择器件

下一屏幕 （图 3-1）允许您选择器件。从下拉菜单中选择 “dsPIC30F6014”。

注：	如果使用其他 dsPIC 开发板，需要为该开发板选择合适的器件。如果是 dsPICDEM 入门演示板，请选择 dsPIC30F6012。如果是 dsPICDEM 28 引脚入门演示板，请选择 dsPIC30F2010。如果是 dsPICDEM 2 开发板，请选择 dsPIC30F4011。

图 3-1： PROJECT WIZARD——步骤 1



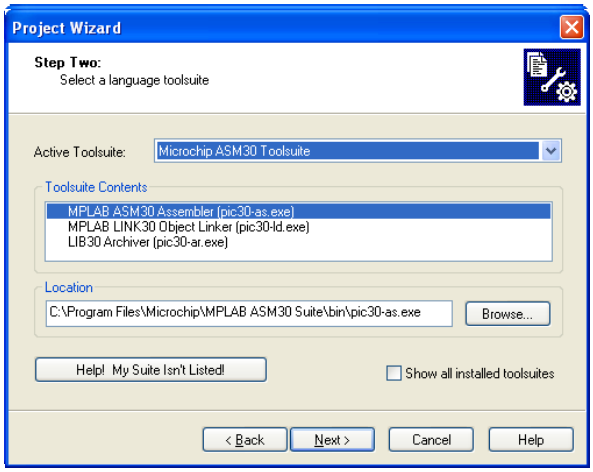
单击 **Next** 继续。

© 2007 Microchip Technology Inc.

步骤 2——选择语言工具包

下一屏幕（图 3-2）允许您选择工具包。从下拉菜单中选择 “Microchip ASM30 Toolsuite”（Microchip ASM30 工具包）。

图 3-2：　　PROJECT WIZARD——步骤 2



检查汇编器和链接器的可执行文件是否位于以下位置：

汇编器：

```
C:\Program Files\Microchip\MPLAB ASM30 Suite\bin\pic30-as.exe
```

链接器：

```
C:\Program Files\Microchip\MPLAB ASM30 Suite\bin\pic30-ld.exe
```

归档器：

```
C:\Program Files\Microchip\MPLAB ASM30 Suite\bin\pic30-ar.exe
```

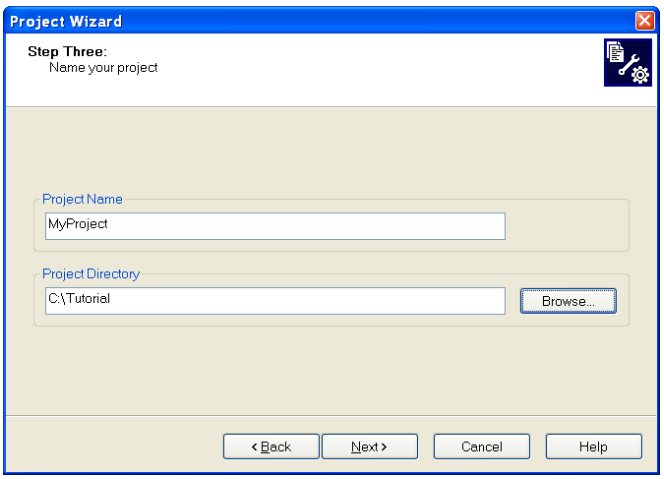
上述工具的位置都是假定最新版本的 MPLAB IDE 是按照默认设置安装的。

位置为空白的工具包旁边会有一个红 “X”。如果有红 “X”，则选中该工具包并单击 **Browse** （浏览）按钮来设置位置。选择了工具包且位置正确时，单击 **Next** 继续。

步骤 3——为项目命名

下一屏幕（图 3-3）允许您为项目命名。

图 3-3：　　PROJECT WIZARD——步骤 3

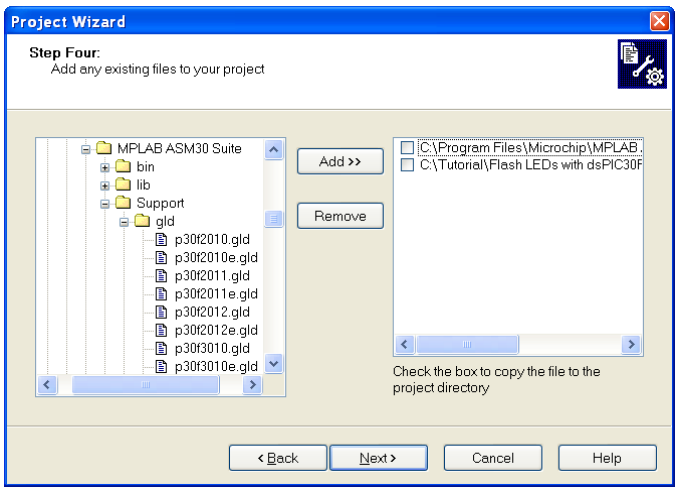


输入 “MyProject”作为项目名称，并找到 （或输入） C:\Tutorial 作为项目目录。单击 **Next** 继续。

步骤 4——向项目添加文件

下一屏幕 （图 3-4）允许您向项目添加文件。

图 3-4： PROJECT WIZARD——步骤 4



选择 Flash LEDs with dsPIC30F6014.s 文件并单击 **Add>>** （添加）将该文件添加到项目中。

导航到 C:\Program Files\Microchip\MPLAB ASM30 Suite\Support\gld 文件夹。选择 p30f6014.gld 文件并单击 **Add>>** 将其添加到项目中。现在，项目中应有两个文件。

单击 **Next** 继续。出现 Summary （摘要）屏幕时，单击 **Finish** （完成）。

注：	如果是 dsPICDEM 入门演示板，请选择 Flash LEDs with dsPIC30F6012.s 和 p30f6012.gld 文件。如果是 dsPICDEM 28 引脚入门演示板，请选择 Flash LED with dsPIC30F2010.s 和 p30f2010.gld 文件。如果是 dsPICDEM 2 开发板，请选择 Flash LED with dsPIC30F4011.s 和 p30f4011.gld 文件。

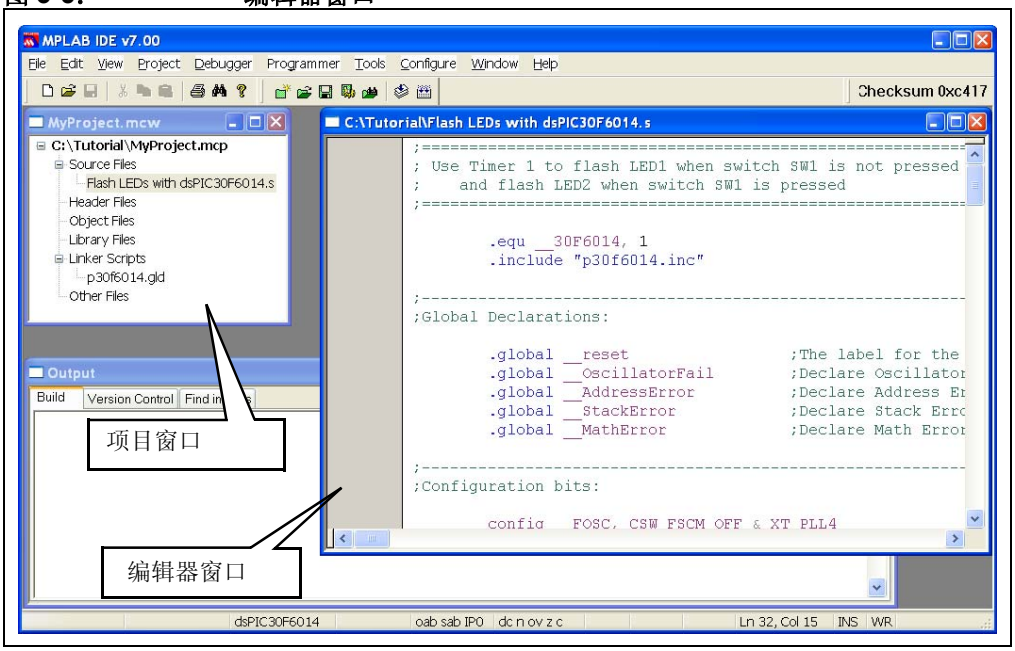
Project Wizard 完成后，MPLAB IDE 项目窗口将显示 Source Files（源文件）目录中的 Flash LEDs with dsPIC30F6104.s 文件和 Linker Scripts（链接描述文件）目录中的 p30f6014.gld 文件。在**第 9 章 “MPLAB LINK30 链接器”**中将对 .gld 文件进行更为深入的介绍。

如果您忘记向项目添加文件了，也无需重启 Project Wizard。只需右击项目树中的目录，从下拉菜单中选择 Add Files （添加文件）并找到要添加的文件即可。通过右击文件名并选择 **Remove** （删除），可删除文件。

现在 MPLAB IDE 已创建了一个项目文件 MyProject.mcp 和一个工作区文件 MyProject.mcw （见图 3-5）。双击项目窗口中的 Flash LEDs with dsPIC30F6014.s 文件将其打开。该文件将显示在编辑器窗口中。

© 2007 Microchip Technology Inc.

图 3-5：　　编辑器窗口



3.3.2　　编辑器

MPLAB IDE 中的编辑器提供了多种功能，使得代码编写过程更为顺畅。这些功能包括：

- 语法突出显示
- 查看并打印行号
- 在所有项目文件中或单个文件内进行搜索
- 添加书签并跳转到指定行
- 双击错误消息转到代码行
- 块注释
- 大括号匹配
- 可更改字体和字体大小

语法突出显示是 MPLAB IDE 的一个特别有用的功能。代码元素如指令、伪指令和寄存器等可以用不同的颜色和字体显示。这使得您可以轻松地解释代码并更快的发现错误。

3.4 编译代码

3.4.1 汇编和链接

编译项目包括两个步骤。第一步为汇编或编译过程，在此过程中每个源文件转换为目标文件（后缀名为 .o），包含操作码或 dsPIC 指令。这些目标文件可用于构造库（作为代码模块添加到其他项目中），或者用于生成最终 hex 文件（用于编程 dsPIC 器件）。

编译过程的第二步为链接。在链接阶段，不同对象和库文件中的所有 dsPIC 指令和变量将根据链接描述文件提供的存储器映射保存到存储器中。

链接器创建以下两个文件：

1. .hex 文件，列出了要保存到 dsPIC 器件程序存储器、 EEPROM 和配置存储器中的数据。
2. .cof 或 COFF （Common Object File Format，公共目标文件格式）文件，包含调试源代码所需的其他信息。

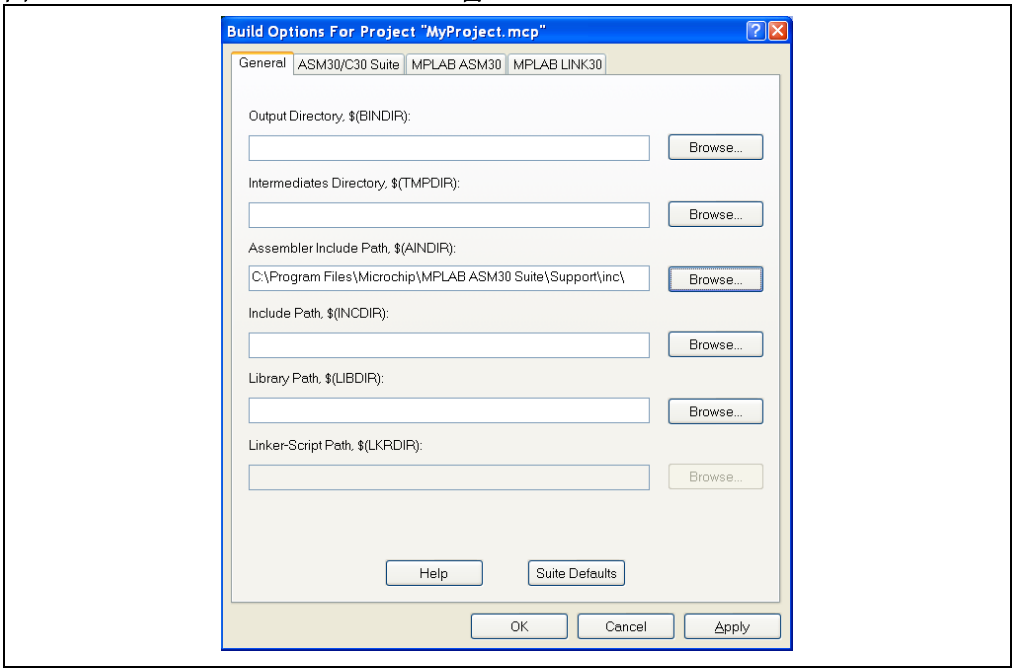
3.4.2 头文件

编译前，必须告知 MPLAB IDE 何处可找到头文件。在靠近 Flash LEDs with dsPIC30F6014.s 文件的顶部，可看到以下信息行：

```
.include "p30f6014.inc"
```

p30f6014.inc 文件包含通过名称而不是毫无意义的数字来引用特殊功能寄存器位所需的符号信息。要使 MPLAB IDE 知道可在何处找到这些文件，应使用 *Project>Build Options>Project* 菜单来打开 Build Options （编译选项）窗口，如图 3-6 所示。然后，单击 “Assembler Include Path, $(AINDIR):” 字段旁边的 **Browse** 按钮。

图 3-6： BUILD OPTIONS 窗口



找到 C:\Program Files\Microchip\MPLAB ASM30 Suite\Support\inc 文件夹并单击 **Select** （选择）。此目录是 MPLAB IDE 保存它所支持的所有 dsPIC 器件的头文件的地方。

最后，单击 **OK** 保存信息。现在可以编译项目了。

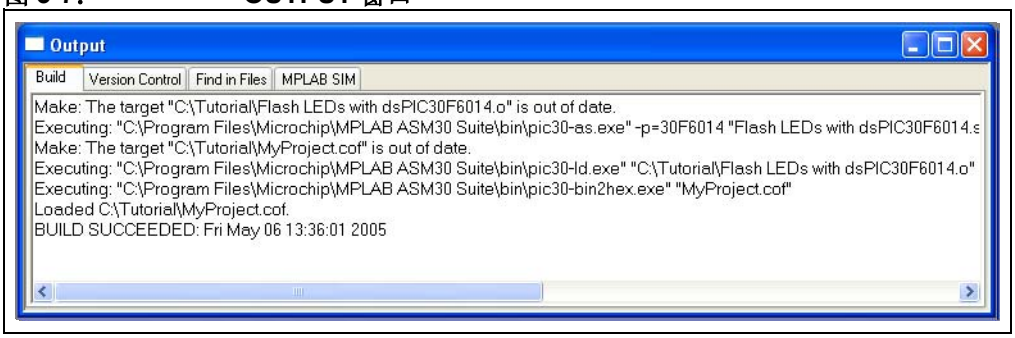
© 2007 Microchip Technology Inc.

3.4.3 编译项目

要编译项目，可使用 *Project>Make* 菜单。编译结果将显示在 Output （输出）窗口中，并指示编译成功，如图 3-7 所示。

根据项目编译选项的设置，可在编译结果中看到 “Memory Usage Report”（存储器使用报告）。

图 3-7： OUTPUT 窗口



3.4.4 配置位

代码包括配置位设置，由配置伪指令指定。可使用 *Configure>Configuration Bits* 菜单在 Configuration Bits （配置位）窗口中查看此设置。这些设置可以更改。单击 Setting （设置）栏中的文本可编辑这些设置。

3.4.5 下一步——调试

成功编译项目后，即可调试代码。有几个工具可供使用。如果希望使用软件模拟器进行调试，请继续**第 4 章 “MPLAB SIM 软件模拟器”**查看该软件模拟器的教程。如果希望使用在线调试器 （MPLAB ICD2），可转至**第 5 章 “MPLAB ICD 2 在线调试器”**。要使用 MPLAB ICE 4000 在线仿真器，请转至**第 6 章 “MPLAB ICE 4000 在线仿真器”**。

注：

© 2007 Microchip Technology Inc.

第 4 章 MPLAB SIM 软件模拟器

4.1 MPLAB SIM 概述

现在，您是不是想测试代码，但又不想安装任何硬件？那么 MPLAB SIM 软件模拟器就是专为您准备的。 MPLAB SIM 软件模拟器已完全集成到 MPLAB IDE 环境中。该软件模拟器不需要您花费昂贵的开销就可以模拟代码在硬件上的执行。还可以免费测试外部输入和外设事务，以及查看处理器内的信号。

但是， MPLAB SIM 软件模拟器也存在一些缺点。它不能对任何真实信号做出反应，也不能产生任何真实信号。它不能发出蜂鸣、点亮 LED 或与其他处理器进行交互。尽管如此，它在开发代码和调试故障方面还是给您提供了极大的灵活性。

MPLAB SIM 软件模拟器具有以下功能：

- 修改代码并立即重新执行
- 注入外部激励到模拟处理器
- 在特定时间点载入寄存器值

dsPIC 器件具有与其他外设复用的 I/O 引脚，因此引脚有多个名字。软件模拟器只识别在标准器件头文件中指定为有效 I/O 引脚的引脚名。所以，您应参阅器件的头文件（如 `p30F6014.inc` 或 `p30F6014.h`）以确定正确的引脚名。

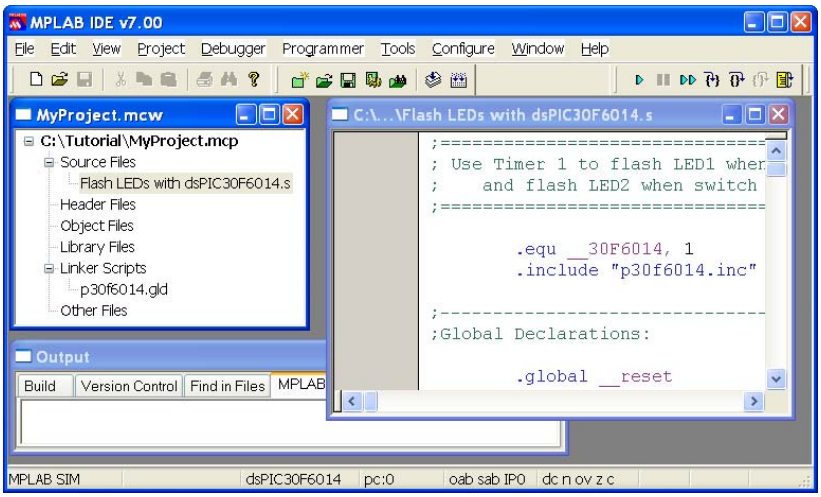
本章将讨论如何使用此软件模拟器。首先，您将打开在**第 3.3 节 “创建项目”**中创建的项目。如果尚未创建项目，请现在参考该章节创建一个项目。您将使用该软件模拟器单步运行代码、创建断点、使用跑表功能并应用激励。

> **注：** 如果您创建的项目是针对 dsPICDEM 入门演示板、 dsPICDEM 28 引脚入门演示板或 dsPICDEM 2 开发板的，则也可使用该项目并按照本章的说明进行操作。它们的代码非常类似。

4.2 打开项目

如果尚未打开项目，则通过选择 *File>Open Workspace* 并浏览到 `C:\Tutoial\MyProject.mcw` 打开在**第 3.3 节 “创建项目”**中创建的工作区。这样，在项目窗口的标题栏上应显示工作区名称，在项目窗口的顶部显示项目名称。使用 *Project>Make* 菜单编译项目，确保它是最新的。

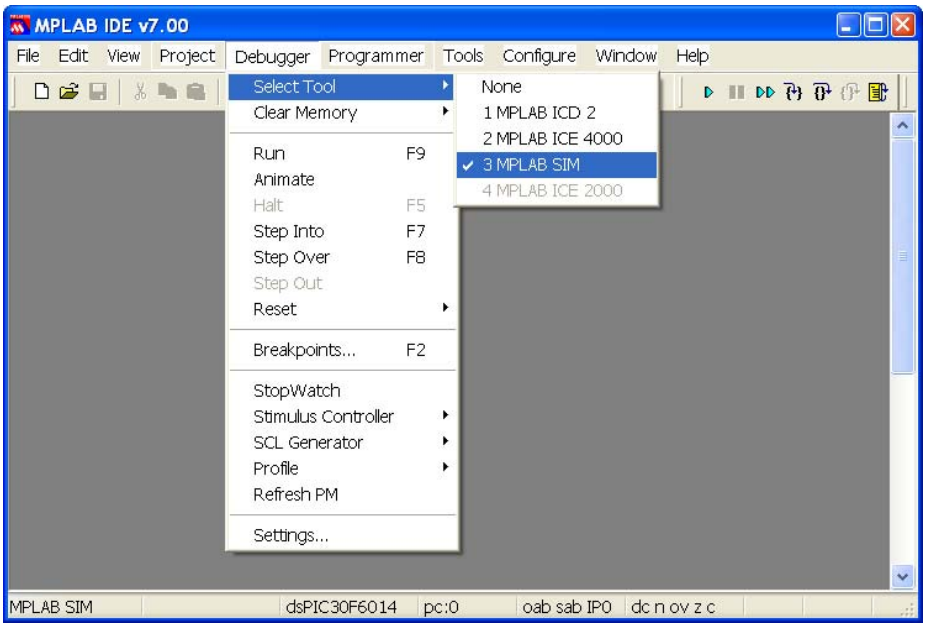
图 4-1： 项目工作区



4.3 选择软件模拟器

使用 *Debugger>Select Tool>MPLAB SIM* 菜单选择 MPLAB SIM 软件模拟器，如图 4-2 所示。这样做的同时，也将软件模拟器操作添加到了菜单和工具栏中。标准调试选项以及 Stopwatch （跑表）、 Stimulus Controller （激励控制器）和 SCL Generator （SCL 发生器）也都添加到 MPLAB IDE 中的 Debugger （调试器）菜单中。

图 4-2： 调试工具选择菜单



© 2007 Microchip Technology Inc.

4.4 复位代码

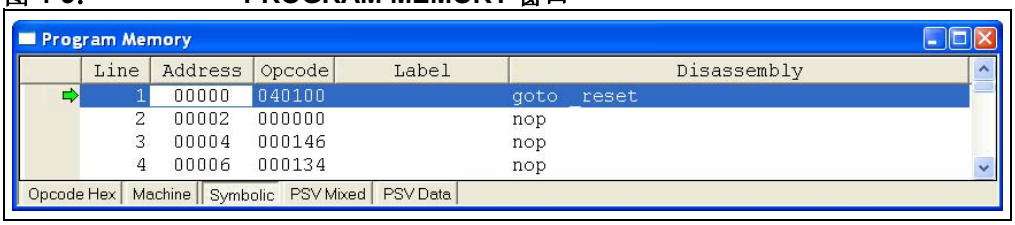
当选择 MPLAB SIM 时，程序计数器将自动设置为零，即复位向量。可通过 *Debugger>Reset>Processor Reset* 菜单来模拟上电复位。在 MPLAB IDE 屏幕底部的状态栏中显示文本 “pc:0”，这表示程序计数器为 0。

这里存在四种类型的复位，可从 Debugger 菜单中选择：

- $\overline{\text{MCLR}}$ Reset （$\overline{\text{MCLR}}$ 复位）
- Watchdog Timer Reset （看门狗定时器复位）
- Brown-out Reset （欠压复位）
- Processor (Power-on) Reset （处理器 （上电）复位）

使用 *View>Program Memory* 菜单打开 Program Memory （程序存储器）窗口 （图 4-3），并单击该窗口底部的 **Symbolic** （符号）选项卡。左边空白处的绿色箭头指向包含地址 0 处代码的第一行。

图 4-3： PROGRAM MEMORY 窗口



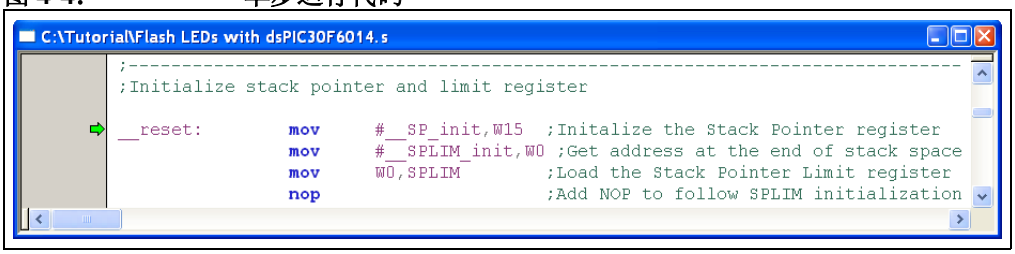
地址 0 处包含一条 `goto __reset` 指令，该指令由链接器自动添加，未包含在源代码文件中。

由于所有其他指令都包含在源代码中，所以此时可以关闭 Program Memory 窗口。

4.5 单步运行代码

现在可以使用软件模拟器单步运行源代码了。使用 *Debugger>Step Into* 菜单来单步运行 `goto __reset` 指令。绿色箭头现在指向 `Flash LEDs with dsPIC30F6014.s` 文件中可执行代码的第一行。

图 4-4： 单步运行代码



使用 *Debugger>Step Into* 菜单继续单步运行其余代码。有几种不同的方法可单步执行代码，它们都位于 *Debugger* 菜单下：Step Into （单步运行）、 Step Over （单步跳过）和 Animate （连续单步运行）。

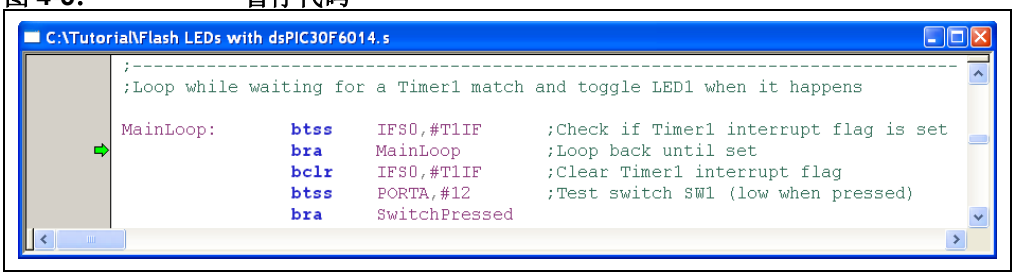
- Step Into 命令执行当前指令后暂停。如果当前指令是对子程序的调用，则程序计数器将改为指向被调用函数的起始地址。
- Step Over 命令执行下一个程序计数器单元前的所有代码。对于大多数指令而言，它与 Step Into 功能相同。但对于 CALL 指令，此命令执行完被调用子程序后返回。
- Animate 命令连续单步运行代码。该命令相当于不断地执行 Step Into 操作，直到选择了 Halt （暂停）按钮。

4.6 运行代码

选择 *Debugger>Run* 来运行应用程序。状态栏将显示 “Running...”（运行 ...）字样和一个小进度条。除了前进的进度条外屏幕上无任何变化。因为程序计数器正在更改，故无法显示当前执行点，所以左边空白处的绿色箭头变为透明。

选择 *Debugger>Halt* 来停止执行程序。执行 Halt 命令后，MPLAB IDE 将更新其窗口，并用绿色箭头指示当前执行点 （图 4-5）。

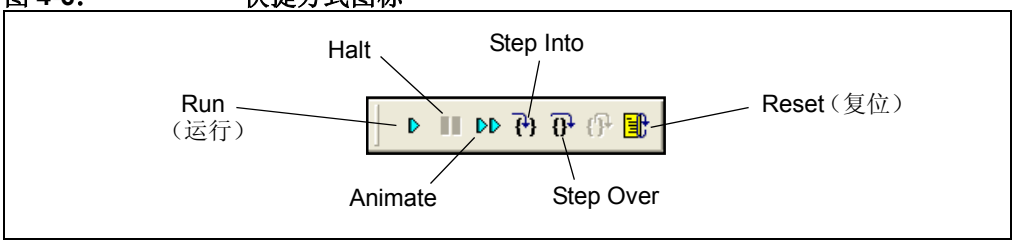
图 4-5： 暂停代码



4.7 DEBUG 工具栏和热键

Debug （调试）工具栏提供了控制软件模拟器的快捷方式图标，如图 4-6 所示。

图 4-6： 快捷方式图标



当选择 MPLAB SIM 软件模拟器时， Debug 工具栏将自动打开。为了方便操作，可将此工具栏拖到桌面的任一位置。单击相应工具栏图标，可运行、暂停、连续单步运行、单步运行、单步跳过或复位程序。

MPLAB SIM 还使用以下功能键来快速访问其主要调试功能：

<F5> Halt
<F6> Reset
<F7> Single Step （单步运行）
<F9> Run

右击源代码的某行可使用其他功能。其中最重要的功能是 Set Breakpoint （设置断点）和 Run to Cursor （运行到光标）。

如果工具栏未显示，则可通过 *View>Toolbars>Debug* 菜单使能。

© 2007 Microchip Technology Inc.

4.8 断点

软件模拟器允许您设置断点——在需要暂停执行的代码处设置断点。您可以在源代码窗口、 Program Memory 窗口或 Disassembly （反汇编）窗口中直接设置断点。

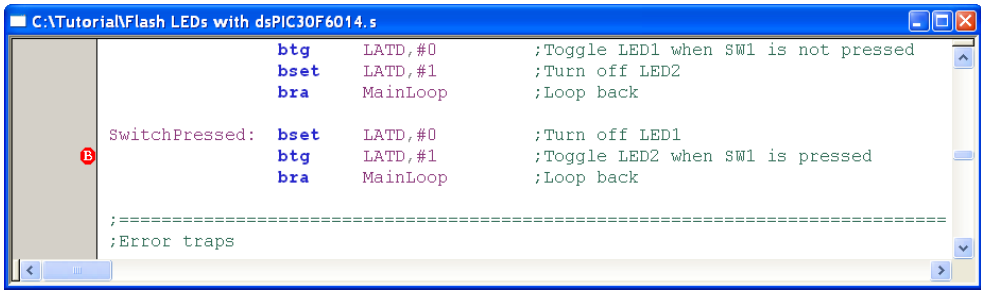
在本示例中，在翻转 PORTD 的位来点亮 LED 的代码处设置断点。向下滚动到第 100 行，可找到 `btg LATD,#1` 指令。注意，行号显示在状态栏中。

> **注：** 如果使用 `Flash LED with dsPIC30F6012.s` 文件，应在第 100 行 `btg LATD,#5` 指令处设置断点。如果使用 `Flash LED with dsPIC30F2010.s` 文件，应在第 91 行 `btg LATD,#0` 指令处设置断点。如果使用 `Flash LED with dsPIC30F4011.s` 文件，应在第 103 行 `btg LATB,#1` 指令处设置断点。

单击该代码行将光标移至当前行，然后右击鼠标，从弹出菜单中选择 Set Breakpoint。双击代码行也可设置断点 （如果已用 *Edit>Properties* 菜单启用了该选项）。

按下 <F9> 或使用 *Debugger>Run* 菜单运行程序。程序执行到断点所在行 （即翻转 I/O 引脚状态）后暂停。暂停后，绿色箭头与红色停止标记重叠 （图 4-7）。

图 4-7： 在断点暂停

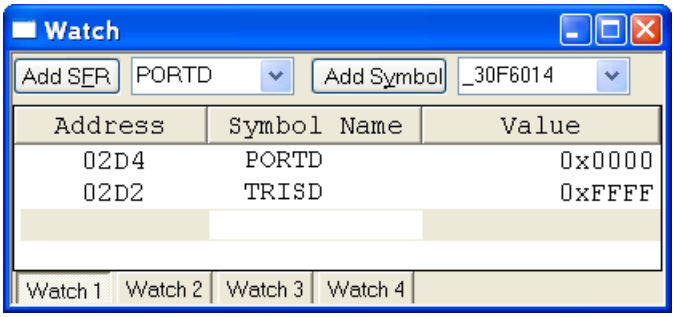


组合使用 <F7> 单步运行键和 <F9> 运行键。注意，每次代码执行时，遇到断点都会暂停。

4.9 WATCH 窗口

有几种方法可在软件模拟器中查看存储器。 Watch 窗口是最有效的方法之一。使用 *View>Watch* 菜单打开 Watch 窗口 （图 4-8）。

图 4-8： WATCH 窗口



在窗口顶部的 **Add SFR** （添加 SFR）（特殊功能寄存器）选择框中输入 “PORTD”。单击 **Add SFR** 将其添加到 Watch 窗口列表中。也可直接在 Watch 窗口中输入寄存器名称；请以相同的方式添加 TRISD。

现在 Watch 窗口中应列有 2 个特殊功能寄存器。每栏分别表示符号的 “Address”（地址）、“Symbol Name”（符号名称）和 “Value”（值）。

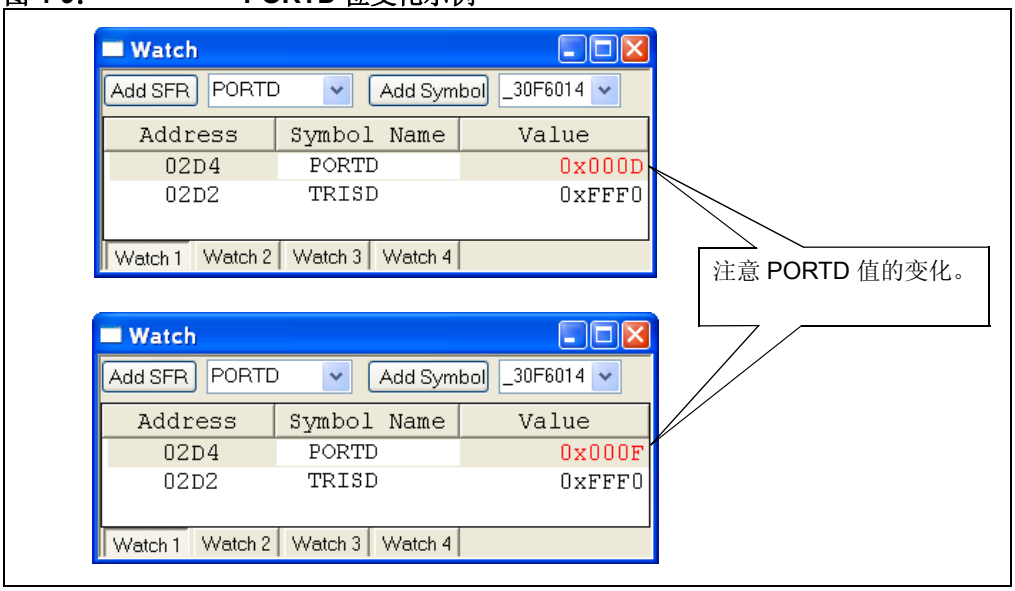
按下 <F6> 复位代码。注意， TRISD 的值为 0xFFFF，这是复位后 TRISD 寄存器的状态。按下 <F9> 运行代码。代码执行并在先前设置的断点处暂停。注意，此时 TRISD 寄存器的值变为 0xFFF0。通过对 TRISD 执行写操作可设置 I/O 端口方向，可在 Watch 窗口中看到这一变化。注意，每次执行 Step 或 Halt 命令时，更改过的值以红色显示，而未更改的值以黑色显示。

注： 如果使用 `Flash LED with dsPIC30F6012.a` 文件，则 TRISD 将变为 0xFF0F。如果使用 `Flash LED with dsPIC30F2010.s` 文件，则 TRISD 将变为 0xFFFE。如果使用 `Flash LED with dsPIC30F4011.a` 文件，则 TRISD 将变为 0xFFFC。

注意 Watch 窗口中 PORTD 的状态，然后按下 <F6> 再次运行程序。每次代码运行并在断点处暂停时， PORTD 中的位就会发生变化 （见图 4-9）。这表示代码通过翻转该引脚来点亮或熄灭 LED。

© 2007 Microchip Technology Inc.

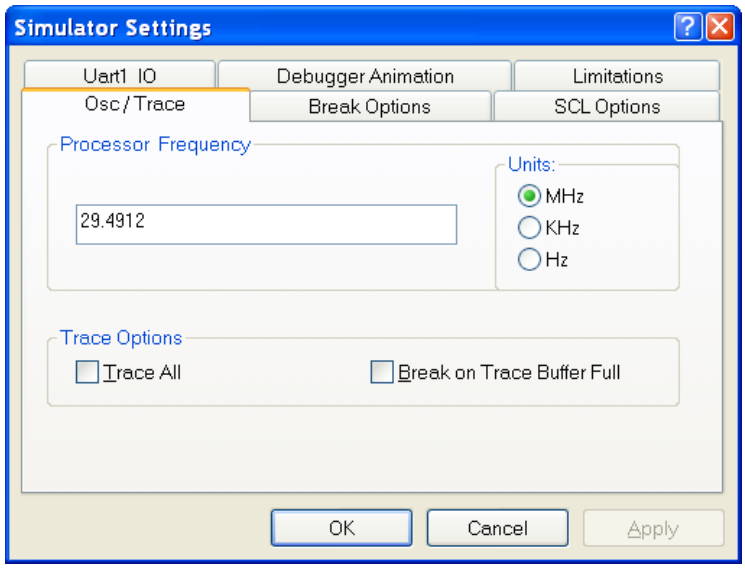
图 4-9：　　PORTD 位变化示例



4.10　软件模拟器设置

在 MPLAB IDE 环境中，使用 *Debugger>Settings* 菜单可设置软件模拟器的各种功能。必需设置振荡器速度，以进行时序测量。

图 4-10：　　SIMULATOR SETTINGS 对话框



选择 **Osc/Trace** （振荡器 / 跟踪）选项卡，然后将 “Processor Frequency”（处理器频率）设置为 29.4912 MHz。选择 **Trace All** （跟踪所有）复选框。稍后将使用到此跟踪功能。

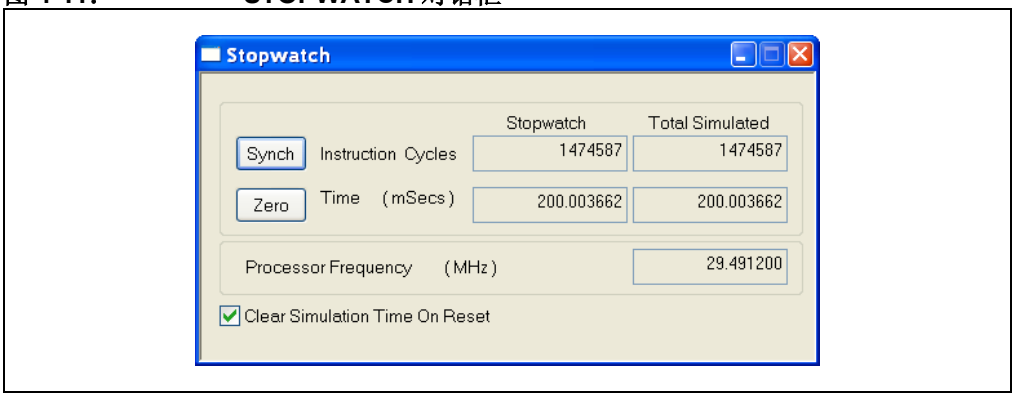
dsPICDEM 1.1 通用开发板装有 7.3728 MHz 的晶振，代码通过配置位选择 4 x PLL 选项，这使得频率变成了原来的 4 倍。也就是说处理器的工作频率为 29.4912 MHz。每个指令周期包括四个时钟周期，因此指令执行速度为 7.3728 MIPS。

注： 如果使用 `Flash LED with dsPIC30F6012.s` 文件，则将处理器频率设为 16 MHz。其他演示文件使用 29.4912 MHz 晶振频率。

4.11 跑表

使用跑表功能可测量两个事件之间的执行时间。单击 *Debugger>Stopwatch* 打开 Stopwatch。跑表将跟踪已执行的指令周期数，同时还显示这些周期所占用的时间。它使用先前输入的 “Processor Frequency”来计算时间。

图 4-11： STOPWATCH 对话框



现在尝试做一个简单的练习。按 <F6> 来复位代码然后按 <F9> 来运行。代码执行到先前在**第 4.8 节 “断点”**中设置的断点处暂停。注意 “Stopwatch”和 “Total Simulated”（共模拟）栏都显示 200 ms。代码将 Timer1 周期寄存器设置为 1/5 秒（200 ms），并在检测到定时器周期的代码行设置断点。

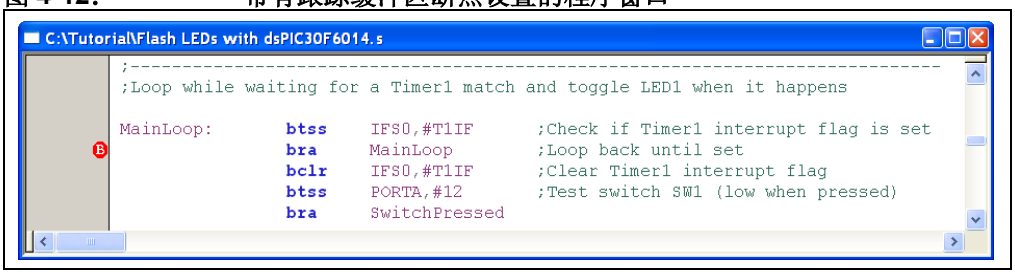
© 2007 Microchip Technology Inc.

4.12 跟踪缓冲区

跟踪缓冲区是 MPLAB SIM 软件模拟器的一个便捷功能。在 *View>ICE Trace* 菜单下可找到该功能。

跟踪缓冲区持有已执行指令的列表。它可存储 65,000 条以上的指令。跟踪缓冲区在后台运行，无需显式使能。一有代码执行，跟踪缓冲区就会将其记录下来。

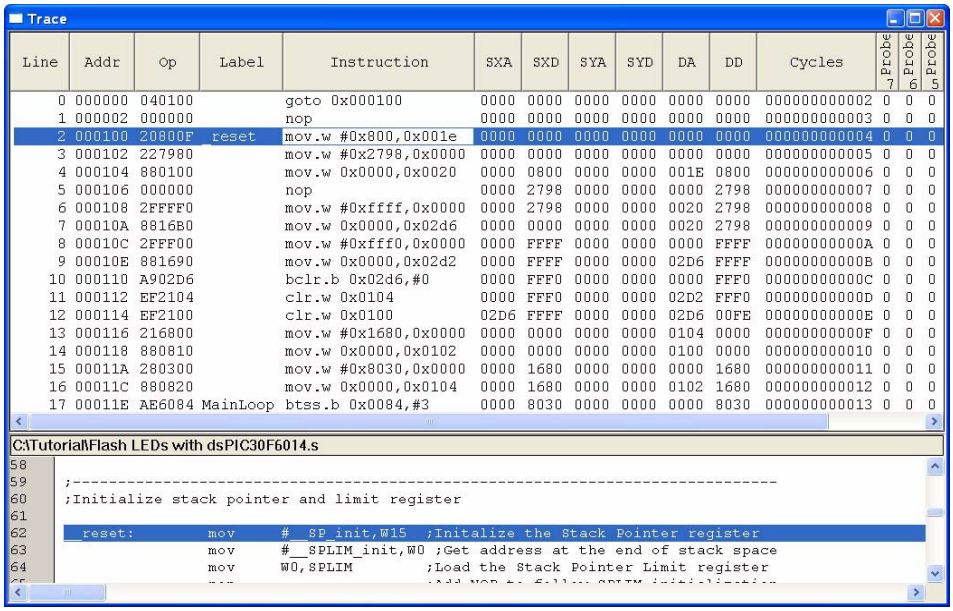
图 4-12： 带有跟踪缓冲区断点设置的程序窗口



在以下简单实验中，请按以下步骤执行数行代码并在 Trace 窗口中查看它们：

1. 在程序窗口中，在 `bra MainLoop` 指令处设置断点，即 `MainLoop:` 地址标号后 （见图 4-12）。
2. 按 <F6> 复位代码，然后按 <F9> 运行。代码将在断点处暂停。
3. 选择 *View>ICE Trace* 显示跟踪缓冲区。所有已执行的指令均显示在 Trace 窗口，从地址 0 处的 `goto __reset` 开始 （见图 4-13）。如果 Trace 视图为空白，则应检查一下 complex trigger setting （复杂触发设置），看看哪一个筛选跟踪没有选择。
4. 滚动到 Trace 窗口的最末端，查看最后一条已执行的指令。注意， Trace 窗口记录到 `btss IFS0,#T1IF`，即断点的前一条指令，然后停止。
5. 比较跟踪缓冲区和源代码，将发现断点前的所有指令均显示在 Trace 窗口中。注意，如果在 Trace 视图中选择了某一行，则在窗口的底部将突出显示相关的源代码行，如图 4-13 所示。

图 4-13： TRACE 窗口



© 2007 Microchip Technology Inc.

第 5 章 MPLAB ICD 2 在线调试器

5.1 MPLAB ICD 2 概述

MPLAB ICD 2 是一款开发级编程器和在线调试器。尽管不如在线仿真器那样功能强大，但它还是有很多优点的。

- ICD 2 允许在真实的目标芯片上执行代码。
- 可以全速运行，或单步执行指令。
- 可以实时查看和修改寄存器内容，以及在源代码中设置断点。

该工具价格低廉。

5.1.1 安装 USB 驱动程序

MPLAB ICD 2 有一个 USB 接口。但是在使用该接口之前必须先安装 USB 驱动程序。MPLAB ICD 2 也可使用串行端口连接，但使用 USB 接口会更加方便快捷。安装 USB 驱动程序的说明在 MPLAB IDE 安装目录下：

`C:\Program Files\Microchip\MPLAB IDE\ICD2\Drivers\ezicd2.htm`

如果使用 Windows® XP 或 Windows® 2000 以外的操作系统，那么该说明文件可能有些不同。请在 Drivers 文件夹中查找相应的 `.htm` 文件。

5.1.2 打开项目

本章包含了演示如何使用 MPLAB ICD 2 的教程。首先，您将打开在**第 3.3 节 “创建项目”**的教程中创建的项目。如果尚未创建项目，请现在参考该章节创建一个项目。

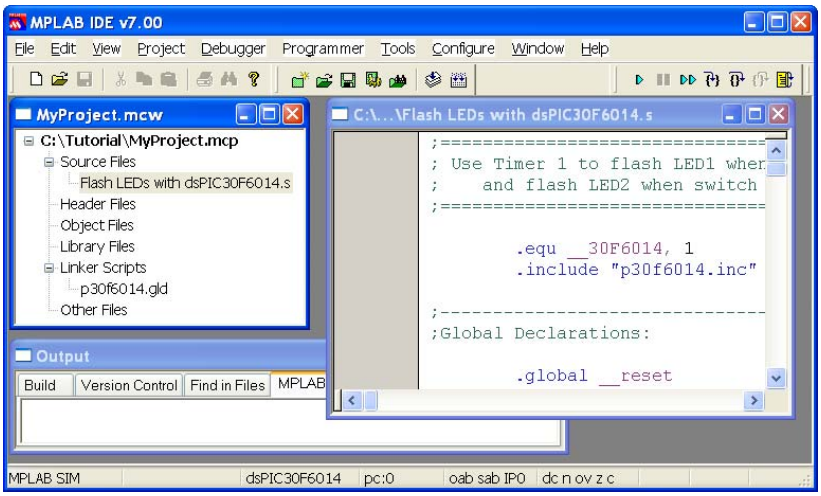
项目代码需要在 dsPICDEM 1.1 通用开发板上运行。所以要继续该教程您还需要一个演示板。

> **注：** 如果您创建的项目是针对 dsPICDEM 入门演示板、 dsPICDEM 28 引脚入门演示板或 dsPICDEM 2 开发板的，则也可在演示板上使用该项目并按照本节的说明进行操作。它们的代码非常类似。如果使用 dsPICDEM 2 开发板，则需要确保插入了正确的跳线 （H6-M ALL、 H12-M D3 和 H12-M D4）。

打开在**第 3.2 节 “项目和工作区”**中创建的工作区 （选择 *File>Open Workspace* 并浏览到 `C:\Tutorial\MyProject.mcw`）。这时，在项目窗口的标题栏上应显示工作区名称，在项目窗口的顶部显示项目名称。

编译项目 （*Project>Make* 菜单），确保它是最新的。

图 5-1：　　教程项目和工作区

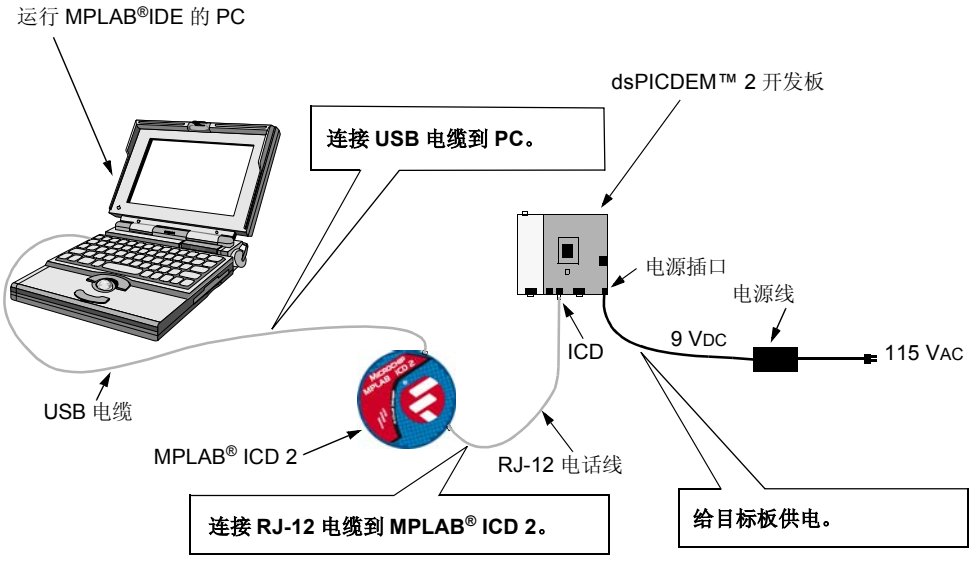


5.2　设置 MPLAB ICD 2

使用 USB 电缆连接 PC 到 MPLAB ICD 2 上，如图 5-2 所示。给目标板供电，使用 RJ-12 （电话线型）电缆连接 MPLAB ICD 2 到目标板上。

注：　请确保 MPLAB ICD 2 在连接到目标板之前已经通过 USB 电缆连接到 PC 上。

图 5-2：　　连接到开发板的 MPLAB® ICD 2



使能 MPLAB ICD 2 为调试器（使用 *Debugger>Select Tool>MPLAB ICD 2* 菜单）。当选择 MPLAB ICD 2 时，标准调试操作以及 MPLAB ICD 2 特有调试操作被添加到 Debugger 菜单和工具栏中。在菜单中选择了 MPLAB ICD 2 时， Output 窗口可能会出现以下警告消息：

“ICDWarn0030:MPLAB ICD 2 is about to download a new operating system”
（ICDWarn0030：MPLAB ICD 2 将下载一个新的操作系统）

© 2007 Microchip Technology Inc.

不必惊慌。当执行下列操作时出现这种情况很正常：当处理器系列首次与 MPLAB ICD 2 一起使用，或者在处理器之间切换时。有时还会出现其他错误信息，这与 MPLAB ICD 2 的设置有关。

如果首次使用 MPLAB ICD 2，则运行 ICD 2 Setup Wizard （ICD 2 设置向导）。使用 *Debugger>MPLAB ICD 2 Setup Wizard* 菜单可以运行该向导。它使得 MPLAB ICD 2 的设置变成一件简单的事情。

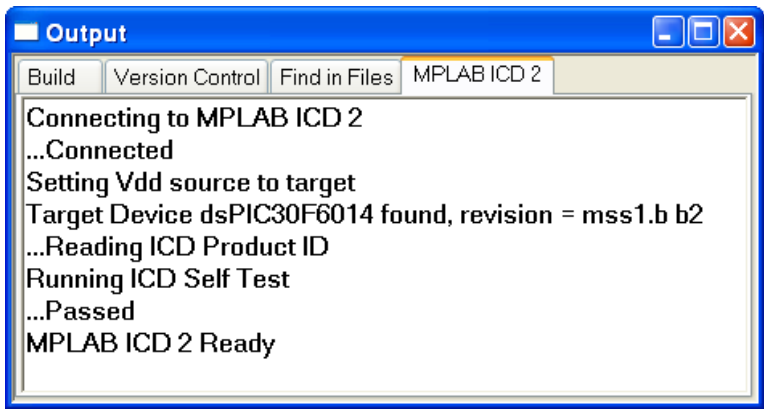
在该向导中选择下列选项：

- 在 Welcome 屏幕上，单击 **Next**。
- 选择 **USB** 作为 **Com Port** （公共端口），单击 **Next**。
- 选择 **Target has own power supply** （目标板有自己的电源），单击 **Next。**
- 选中 **MPLAB IDE automatically connects to the MPLAB ICD 2** （MPLAB IDE 自动连接到 MPLAB ICD 2），单击 **Next**。
- 选中 **MPLAB IDE automatically downloads the required operating system** （MPLAB IDE 自动下载需要的操作系统），单击 **Next**。
- 在 Summary 屏幕上，单击 **Finish** （完成）。

就是这么简单。MPLAB ICD 2 现在已设置完毕，可供使用。您可以使用 *Debugger>Settings* 菜单检查设置并进行修改。

现在可以将 MPLAB ICD 2 连接到目标板 （使用 *Debugger>Connect* 菜单）。Output 窗口应显示 MPLAB ICD 2 已连接并标识出目标板上的 dsPIC30F6014 器件，如图 5-3 所示。

图 5-3： OUTPUT 窗口中的连接状态



您的硅片版本可能不同，但 Output 窗口都会显示已找到的正确目标器件。

注： 如果使用 dsPICDEM 入门演示板，则将找到 dsPIC30F6012 器件。如果使用 dsPICDEM 28 引脚入门演示板，则将找到 dsPIC30F2010 器件。如果使用 dsPICDEM 2 开发板，则将找到 dsPIC30F4011 器件。

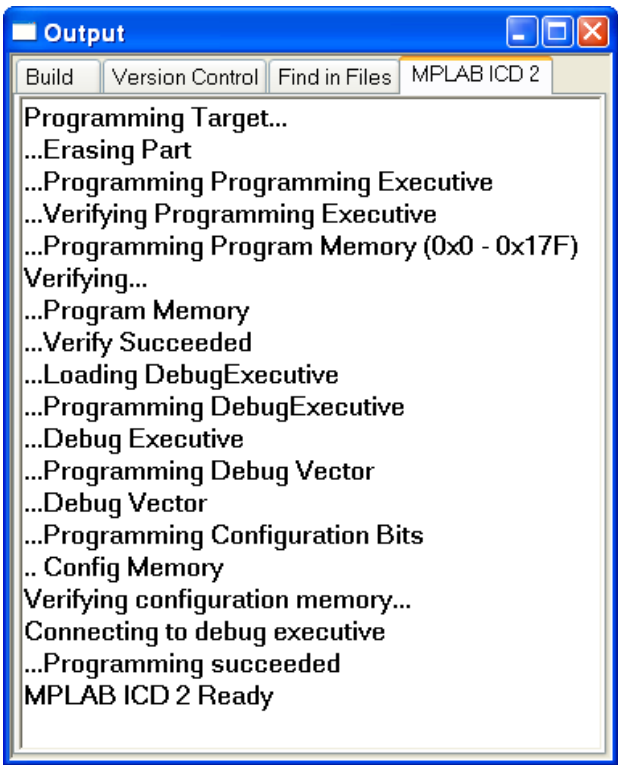
如果出现信息“ICDWarn0020: Invalid target device id”（ICDWarn0020：无效目标器件 id），则表示很可能存在连接或电源问题。双击文本 “ICDWarn0020”可获取更多帮助信息，但是您可能需要检查连线和电源。

5.3 对 dsPIC 器件编程

在进行任何调试之前，必须对器件进行编程。使用 *Debugger>Program* 菜单对器件进行编程。

Output 窗口显示编程和校验过程的每个步骤。如果出现任何错误消息或警告，请双击消息编号以获取更多帮助信息。

图 5-4： OUTPUT 窗口中的编程状态



当状态显示 “MPLAB ICD 2 Ready”（MPLAB ICD 2 就绪）时，表示 dsPIC30F6014 目标芯片正在运行调试执行代码，此代码允许您单步运行、设置断点、查看寄存器以及执行其他调试任务。

值得注意的是， dsPIC 器件在调试执行代码的控制下运行，如果没有 MPLAB ICD 2 的命令它不会运行该代码。如果稍后您希望代码能自动运行，您可以将 MPLAB ICD 2 设置为编程器，而不是调试器。设置方式类似，但需要使用 *Programmer* （编程器）菜单，而不是 *Debugger* 菜单。

5.4 复位代码

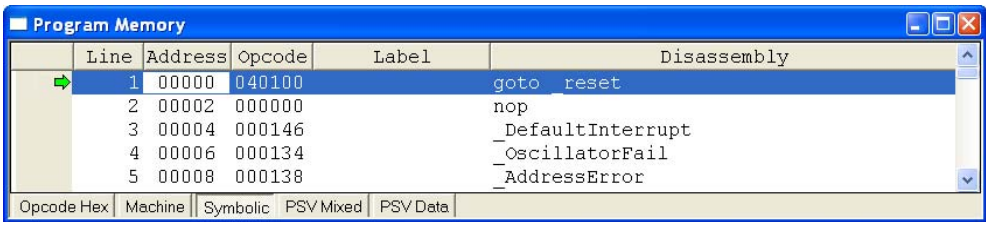
在将 MPLAB ICD 2 编程为调试器后，程序计数器将自动设置为 0 （即复位向量）。您可以通过 Debugger>Reset>Processor Reset 菜单强制执行复位操作。在 MPLAB IDE 屏幕底部的状态栏中显示文本 “pc:0”，这表示程序计数器为 0。

使用 *View>Program Memory* 菜单打开 Program Memory 窗口，并单击该窗口底部的 **Symbolic** 选项卡。左边空白处的绿色箭头指向包含地址 0 处代码的第一行，如图 5-5 所示。

© 2007 Microchip Technology Inc.

地址 0 处包含一条 goto __reset 指令，该指令由链接器自动添加，未包含在源代码文件中。

图 5-5：　　PROGRAM MEMORY 窗口



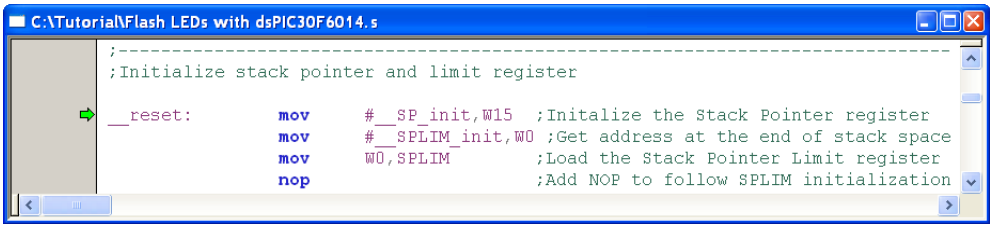
由于所有其他指令都包含在源代码中，所以此时可以关闭 Program Memor 窗口。

5.5　单步运行代码

现在可以使用 MPLAB ICD 2 单步运行源代码了。

使用 *Debugger>Step Into* 菜单来单步运行 goto __reset 指令。绿色箭头现在指向 Flash LEDs with dsPIC30F6014.s 文件中可执行代码的第一行。

图 5-6：　　PROGRAM MEMORY 窗口中单步执行的结果



使用 *Debugger>Step Into* 菜单继续单步运行代码。*Debugger* 菜单还允许您单步运行、单步跳过和连续单步运行代码：

- Step Into 命令执行当前指令后暂停。如果当前指令是对子程序的调用，则程序计数器将改为指向被调用函数的起始地址。
- Step Over 命令执行下一个程序计数器单元前的所有代码。对于大多数指令而言，它与 Step Into 功能相同。但对于 CALL 指令，此命令执行完被调用子程序后返回。
- Animate 命令连续单步运行代码。该命令相当于重复执行 Step Into 操作，直到选择了 Halt 按钮。

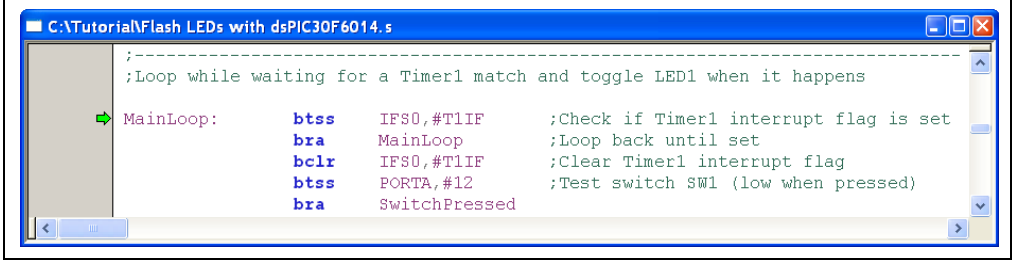
5.6 运行代码

选择 *Debugger>Run* 菜单来运行应用程序。 MPLAB IDE 工作区状态栏将显示“Running…”字样和一个进度条。

除了前进的进度条之外， MPLAB IDE 屏幕上无任何变化。因为程序计数器在变化，故无法显示当前执行点，所以左边空白处的绿色箭头变成透明。因为代码正在演示板上的 dsPIC 器件中运行，故演示板上的 LED 不断闪烁。

要停止程序执行，请选择 *Debugger>Halt* 菜单。执行 Halt 命令后， MPLAB IDE 将更新其窗口，并用绿色箭头指示当前执行点，如图 5-7 所示。

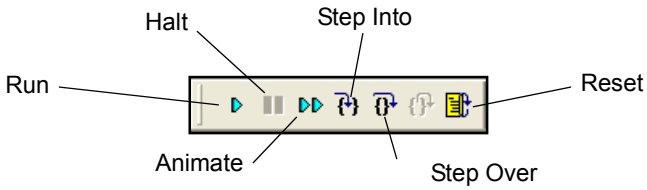
图 5-7： 程序窗口中的暂停位置



5.7 DEBUG 工具栏和热键

Debug 工具栏提供了控制 MPLAB ICD 2 的快捷方式图标，如图 5-8 所示。

图 5-8： 快捷方式图标



当选择 MPLAB ICD 2 时， Debug 工具栏将自动打开。为了方便操作，可将此工具栏拖到桌面的任一位置。单击相应的工具栏图标，可运行、暂停、连续单步运行、单步运行、单步跳过或复位程序。

MPLAB ICD 2 还可使用以下功能键来快速访问其主要调试功能：

<F5> Halt
<F6> Reset
<F7> Single Step
<F9> Run

右击源代码的某行可使用其他功能。其中最重要的功能是 Set Breakpoint 和 Run to Cursor。

如果工具栏未显示，则可通过 *View>Toolbars>Debug* 菜单使能。

© 2007 Microchip Technology Inc.

5.8 断点

MPLAB ICD 2 允许您设置断点——在需要暂停执行的代码处设置断点。您可以直接在源代码窗口、 Program Memory 窗口或 Disassembly 窗口中直接设置断点。

在本示例中，在通过翻转 PORTD 中的位来点亮 LED 的代码处设置断点。向下滚动到第 95 行，找到 `btg LATD,#0` 指令。注意，行号显示在状态栏中。

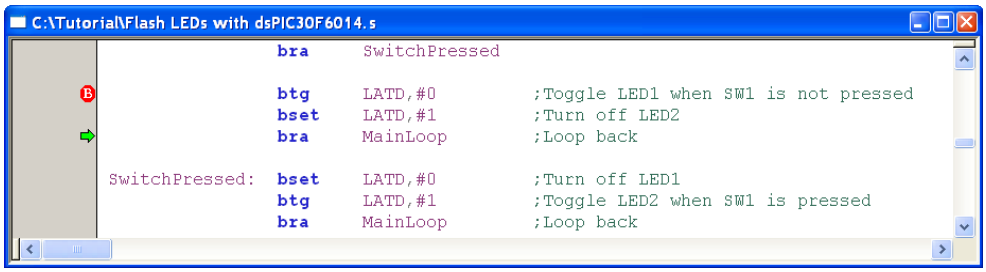
> **注：** 如果使用 `Flash LED with dsPIC30F6012.s` 文件，请将断点设置在第 95 行的 `btg LATD,#4` 指令处。如果使用 `Flash LED with dsPIC30F2010.s` 文件，请将断点设置在第 90 行的 `bclr IFS0,#T1IF` 指令处。如果使用 `Flash LED with dsPIC30F4011.s` 文件，请将断点设置在第 98 行的 `btg LATB,#0` 指令处。

右击该代码行，然后从弹出菜单中选择 Set Breakpoint。

也可通过双击该代码行来设置断点。此选项可能需要通过 ***Edit>Properties*** 菜单启用。

现在按下 <F9> 或选择 ***Debugger>Run*** 菜单。这时程序执行到此断点所在行 （即翻转 I/O 引脚状态）后暂停。注意，演示板上的 LED 在每次代码暂停时都会发生变化。程序暂停后，绿色箭头指向断点后的第二条指令，如图 5-9 所示。

图 5-9： 显示断点的程序窗口



在 dsPIC 器件上进行在线调试的局限性之一就是遇到断点暂停时会产生 skew。当硬件检测到断点地址时，断点后的下一条指令已经被执行了。其实，这一问题并不难解决，只需调节断点位置即可。但是，如果断点后是一条转移指令，这样做就会产生混淆。

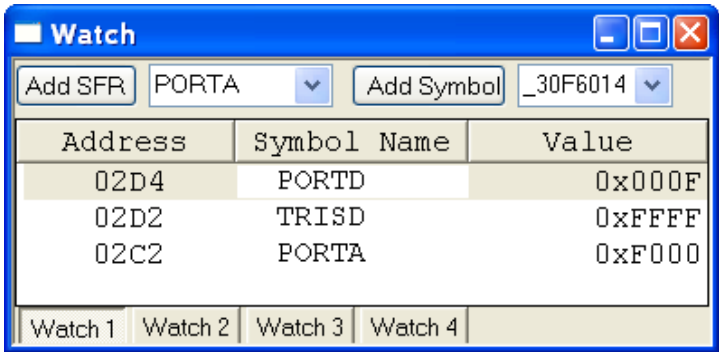
组合使用单步运行热键 <F7> 和运行键 <F9>。注意，每次代码执行时，都会在断点后暂停。注意，按下开发板上的开关 SW1 可解除暂停。

> **注：** 如果使用 dsPICDEM 入门演示板，请按开关 S1。如果使用 dsPICDEM 2 开发板，请按开关 S5。 dsPICDEM 28 引脚入门演示板则没有开关。

5.9 WATCH 窗口

使用 MPLAB ICD 2 时，有几种方法可查看存储器。 Watch 窗口 （图 5-10）是最有效的方法之一。 Watch 窗口允许您指定不同编程条件下想要观察的存储单元。打开 Watch 窗口，请选择 *View>Watch* 菜单。

图 5-10： WATCH 窗口



例如，如需添加 PORTD 到 Watch 窗口，可在该窗口顶部的 **Add SFR** （特殊功能寄存器）选项框中输入 “PORTD”，然后单击 **Add SFR** 将其添加到 Watch 列表。

也可直接在 Watch 窗口中输入寄存器名称。

请以相同的方式添加 TRISD 和 PORTA。现在 Watch 窗口中应列有 3 个特殊功能寄存器。每栏分别表示符号的 “Address”、“Symbol Name”和 “Value”。

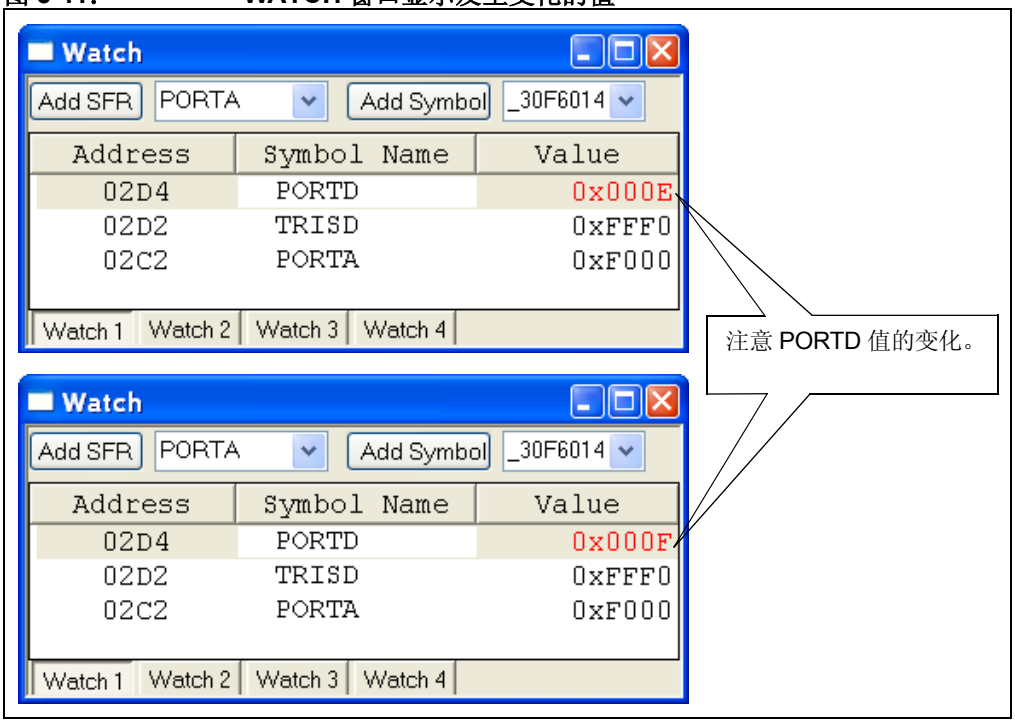
按下 <F6> 复位代码。注意， TRISD 的值为 0xFFFF，这是复位后 TRISD 寄存器的状态。按下 <F9> 运行代码。代码将执行并在先前设置的断点处暂停。注意，此时 TRISD 寄存器的值变为 0xFFF0。通过对 TRISD 执行写操作可设置 I/O 端口方向，可在 Watch 窗口中看到这一变化。注意，每次执行 Step 或 Halt 命令时，更改过的值以红色显示，而未更改的值以黑色显示。

注： 如果使用 `Flash LED with dsPIC30F6012.s` 文件，则 TRISD 将变为 0xFF0F。如果使用 `Flash LED with dsPIC30F2010.s` 文件，则 TRISD 将变为 0xFFFE。如果使用 `Flash LED with dsPIC30F4011.s` 文件，则 TRISD 将变为 0xFFFC。

注意 Watch 窗口中 PORTD 的状态，然后按下 <F6> 再次运行程序。每次代码运行并在断点处暂停时， PORTD 中的位就会发生变化。这表示代码通过翻转该引脚来点亮或熄灭 LED。

© 2007 Microchip Technology Inc.

图 5-11：　　WATCH 窗口显示发生变化的值



分别在按下和释放开关 SW1 的情况下按下 <F7> 单步执行代码。注意观察按下开关时 PORTA 将如何变化。该开关位于 PORTA 的 RA12 引脚上，且可在 Watch 窗口中查看该输入引脚的状态。

注：　如果使用 dsPICDEM 入门演示板，请按开关 S1 并在 Watch 窗口中观察 PORTC 的状态。该开关位于 PORTC 的 RC13 引脚上。如果使用 dsPICDEM 2 开发板，请按开关 S5 并在 Watch 窗口中观察 PORTE 的状态。该开关位于 PORTE 的 RE8 引脚上。

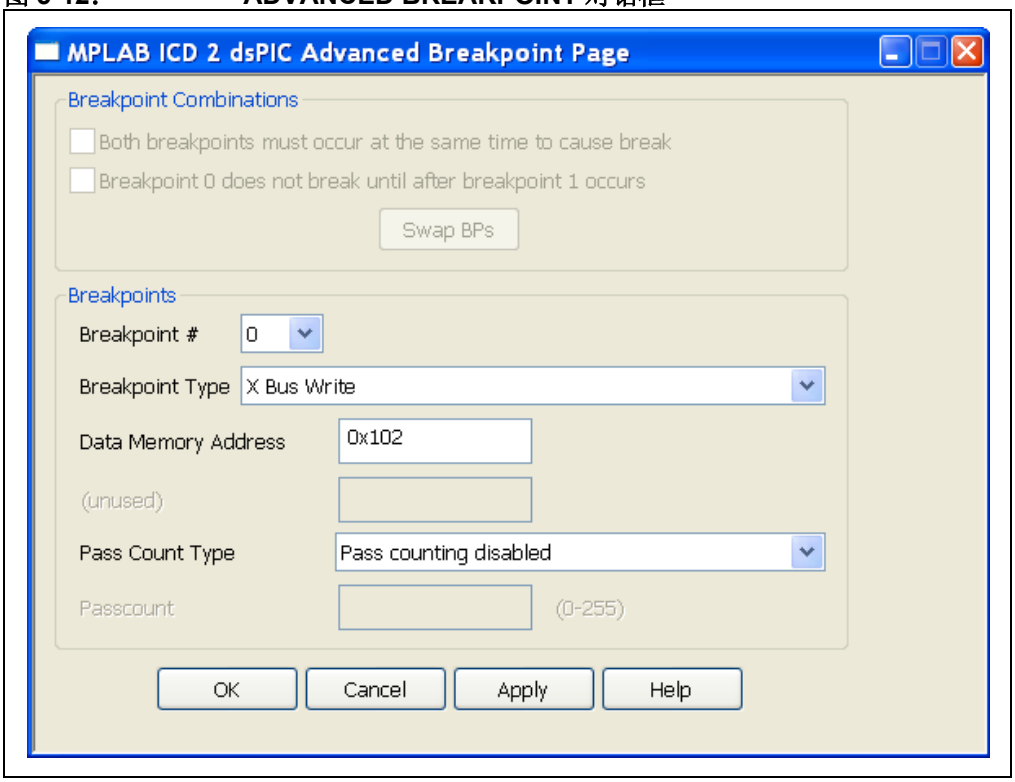
5.10 高级断点

您可设置 MPLAB ICD 2 的高级断点功能以在一组复杂条件下暂停。它有助于捕捉代码中的异常情况。在本例中，设置一个高级断点，以便在向 PR1 寄存器写入数据时暂停。由于代码中有一条 `mov W0,PR1` 指令，因此该功能可以实现。要查找 PR1 寄存器的地址，可查看 `p30f6014.gld` 链接描述文件并找到代码行 `PR1 = 0x0102;` 即可。

通过 *Debugger>Advanced Breakpoints* 菜单打开 Advanced Breakpoint （高级断点）对话框 （图 5-12）。有两个高级断点可用，但是您将只使用断点 “0”。在 Breakpoint Type （断点类型）中选择 “X Bus Write”（X 总线写），以检测对地址单元 0x0102（在 X 数据空间中）的写操作。实际上，除非是执行非常特殊的 DSP 操作，否则所有数据存储器都是通过 X 总线来操作的。

设置 Data Memory Address （数据存储器地址）为 “0x102”，并保持选择 “Pass counting disabled”（禁止次数计数）。单击 **OK** 保存高级断点设置。

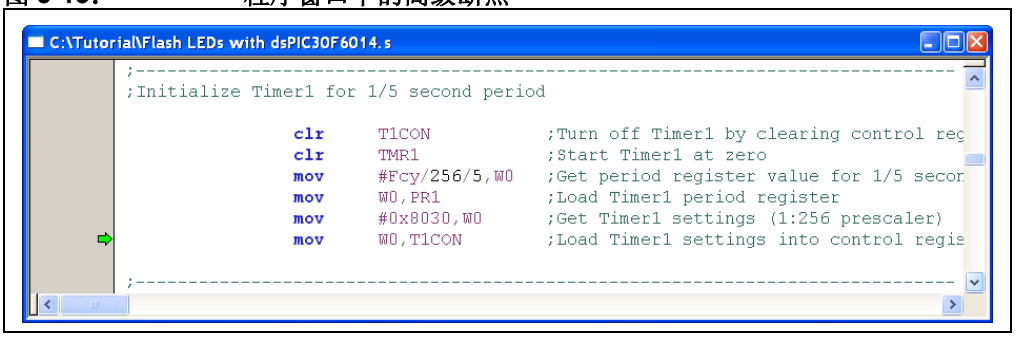
图 5-12： ADVANCED BREAKPOINT 对话框



现在，按下 <F6> 复位代码，并按下 <F9> 运行代码。注意，代码执行暂停且绿色箭头指向写 PR1 寄存器后的第二条指令，如图 5-13 所示。上文提到的 skew 将导致多运行两个周期。

© 2007 Microchip Technology Inc.

图 5-13：　　程序窗口中的高级断点



此演示是高级断点的一个非常简单的应用。也可在读写特定存储单元中的特定数据时暂停。还可在对程序存储器中的单元进行表读表写操作，以及对程序存储器进行简单的取值操作时暂停。由于有次数计数器，因此也可在事件发生若干次后才暂停，或在某一事件发生后再执行若干个指令周期后才暂停。

这里提供了两个独立的高级断点，它们可一起使用，这样只有在两个独立事件都发生的情况下才能触发暂停。总之，高级断点提供了非常强大的调试功能，而这些功能通常只有昂贵的仿真器才有。如果您学会了如何使用这些选项，您就会发现它们的功能有多强大。

注：

© 2007 Microchip Technology Inc.

第 6 章 MPLAB ICE 4000 在线仿真器

6.1 MPLAB ICE 4000 概述

MPLAB ICE 4000 是一个针对 PIC18 和 dsPIC 器件设计的在线仿真器 （ICE）。它为执行期间的指令和数据路径提供了全速仿真和可视性。

除基本的运行、暂停、单步运行和软件断点功能外， MPLAB ICE 4000 还提供了高级功能，如指令 / 地址数据监视、指令数据跟踪、复杂触发和代码覆盖。 MPLAB ICE 4000 还可与您的目标板连接充当在线处理器，或者单独使用，以调试程序。

图 6-1： MPLAB® ICE 4000 在线仿真器

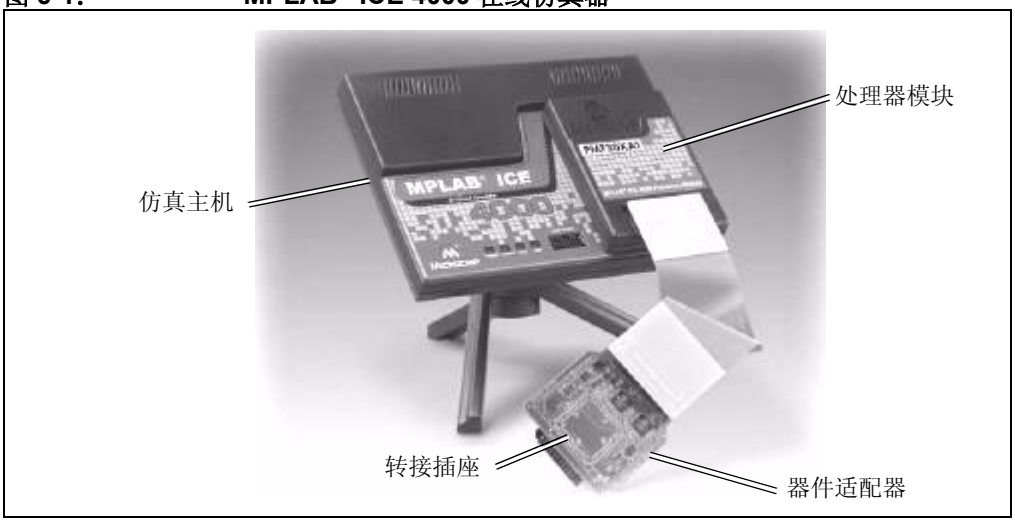


图 6-1 给出了 MPLAB ICE 4000 的主要组件。

- ICE 4000 仿真器主机是 PC 和处理器模块间的接口。它包含的硬件可从处理器模块读取数据并将数据发送回 PC，也可从 PC 获取命令和其他各种数据并将其发送到处理器模块。
- 处理器模块是实际仿真特定器件的组件。
- 器件适配器是一个可更换的装置，它可将仿真器连接到目标应用系统。
- 转接插座允许将器件适配器连接到支持表面贴装型器件的目标应用板上。

6.1.1 安装 USB 驱动程序

MPLAB ICE 4000 具有一个 USB 接口。在使用 MPLAB ICE 4000 前，必须先安装 USB 驱动程序。安装 USB 驱动程序的说明位于 MPLAB IDE 安装目录中：

```
C:\Program Files\Microchip\MPLAB IDE\ICE 4000\Drivers\ezice4k.htm
```

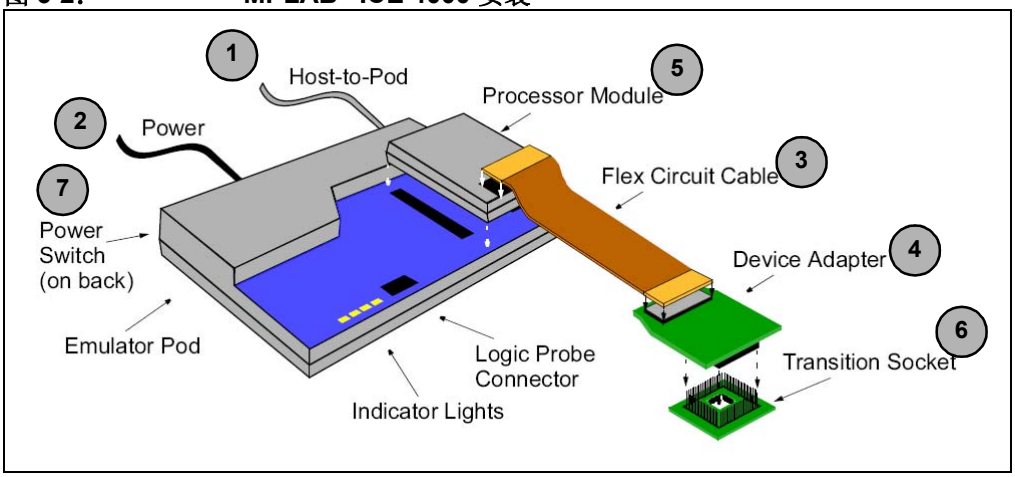
如果使用 Windows XP 或 Windows 2000 以外的操作系统，那么说明文件可能有所不同。请在 Drivers 文件夹中选择相应的 `.htm` 文件。

注： 安装 USB 驱动程序，必须先插入并启动 MPLAB ICE 4000。启动 ICE 4000 前，请按照 `ezice4k.htm` 文件中的指示进行操作。

6.1.2 连接 MPLAB ICE 4000 硬件

请按照图 6-2 所示连接仿真器和演示板。

图 6-2： MPLAB® ICE 4000 安装



1. 使用 USB 电缆连接 MPLAB ICE 4000 主机到 PC。
2. 将电源连接到仿真器，但不要打开。
3. 将扁平柔性电缆标记为 “Emulation Module”（仿真模块）的那一端连接到处理器模块。
4. 将扁平柔性电缆标记为 “Device Adapter”（器件适配器）的那一端连接到器件适配器。
5. 将处理器模块插入仿真器主机顶部的连接器中。
6. 将器件适配器插入到演示板的引脚上。如果没有引脚，将需要一个转接插座。
7. 依次打开仿真器主机和演示板的电源。

© 2007 Microchip Technology Inc.

6.2 打开项目

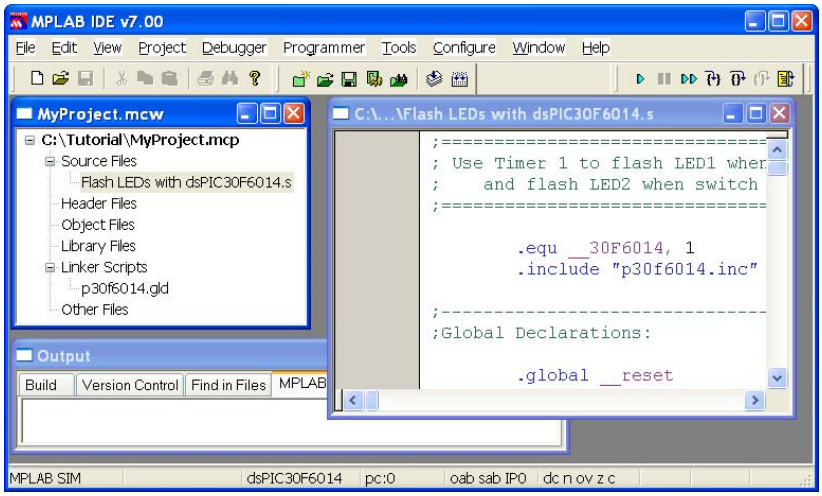
本章提供了演示如何使用 MPLAB ICE 4000 的教程。首先，您将打开在**第 3.3 节 “创建项目”**的教程中创建的项目。如果尚未创建项目，请现在参考上述章节创建一个项目。该项目代码需运行在 dsPICDEM 1.1 通用开发板上。所以您需要一个演示板来继续本教程。

> **注：** 如果您创建的项目是针对 dsPICDEM 入门演示板、 dsPICDEM 28 引脚入门演示板或 dsPICDEM 2 开发板的，则也可使用该演示板上的项目并按照本节的说明进行操作。它们的代码非常类似。

如果未打开项目，请通过选择 *File>Open Workspace* 并浏览到 `C:\Tutorial\MyProject.mcw` 打开在**第 4 章 “MPLAB SIM 软件模拟器”**中创建的工作区。这时在项目窗口的标题栏上显示工作区名称，而在项目窗口的顶部显示项目名称，如图 6-3 所示。

编译项目 （*Project>Make* 菜单），确保它是最新的。

图 6-3： 项目窗口



6.3 特殊仿真器器件

ICE 4000 处理器模块（例如， PMF30XA1）使用了一个超集仿真器芯片，它能够仿真所有 dsPIC30F 器件。这使得操作更为简便，且使得所需的处理器模块的数量减至最少。但是，该仿真器芯片中存在一个固定的 X/Y 存储器边界。

如**第 1.2.3 节 “数据存储器”**所述， dsPIC 器件具有两个用于 DSP 指令的独立的可寻址数据存储空间。这两个 X 和 Y 存储区之间的边界是固定的，但其位置因器件不同而异。当您使用一些数据存储空间较小的诸如 dsPIC30F2010 或 dsPIC30F4011 之类的器件时，有必要修改地址映射，以便使用 ICE 4000 上的不同存储单元。而且在必要时， MPLAB IDE 会出现警告，如图 6-4 所示。

图 6-4：　　XY 数据边界警告

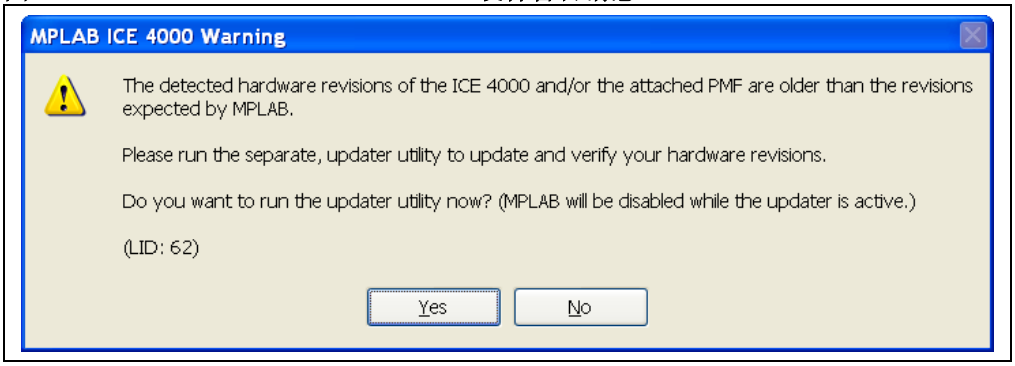
MPLAB IDE 通过为要仿真的器件提供不同的部件编号，来支持 ICE 4000 的不同存储器映射。例如，要仿真 dsPIC30F2010，可打开 *Configure>Select Device* 菜单，然后选择 dsPIC30F2010e 器件，而不是普通的 dsPIC30F2010 器件。类似地，向项目中添加的是 `p30f2010e.gld` 链接描述文件，而不是 `p30f2010.gld` 文件。**第 3.3 节“创建项目”**说明了如何设置项目。您可能发现某些器件，如 dsPIC30F6014 和 dsPIC30F6012，和仿真器芯片具有相同的 XY 边界，而不需要特殊的部件编号。

6.4　选择 MPLAB ICE 4000

将 MPLAB ICE 4000 用作调试器（使用 *Debugger>Select Tool>MPLAB ICE 4000* 菜单）。将标准调试操作以及 MPLAB ICE 4000 特有的调试操作都添加到 Debugger 菜单和工具栏中。

在菜单中选择 MPLAB ICE 4000 后，MPLAB IDE 可能马上弹出如图 6-5 所示的警告消息。不必惊慌。第一次使用 MPLAB ICE 4000，或者安装新版本时，出现这种情况是很正常的。

图 6-5：　　MPLAB® ICE 4000 硬件警告消息



如果出现此警告：

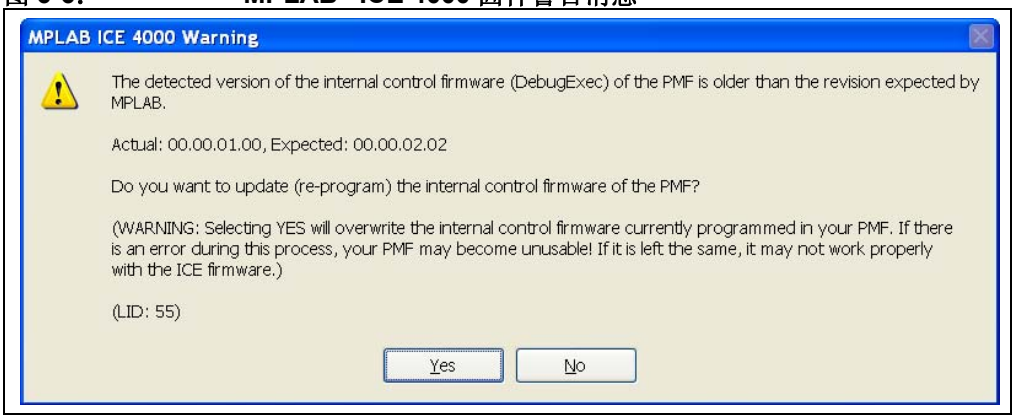
1. 单击 **Yes** 以更新 MPLAB ICE 4000。
2. 出现 ICE4K Update（ICE4K 更新）窗口时，单击 **UpdateICE4K**。
3. 更新完成后，关闭窗口。

ICE 4000 更新程序使用最新版本配置 MPLAB ICE 4000 中的可编程逻辑。

更新完 MPLAB ICE 4000 硬件后，MPLAB IDE 可能会弹出另一警告消息框要求更新处理器模块的固件，如图 6-6 所示。

© 2007 Microchip Technology Inc.

图 6-6：　　MPLAB® ICE 4000 固件警告消息



如果出现此警告，请单击 **Yes** 更新固件。

注：　更新完成后，需要重新选择 MPLAB ICE 4000 作为调试器 （通过 *Debugger>Select Tool>MPLAB ICE 4000* 菜单）。

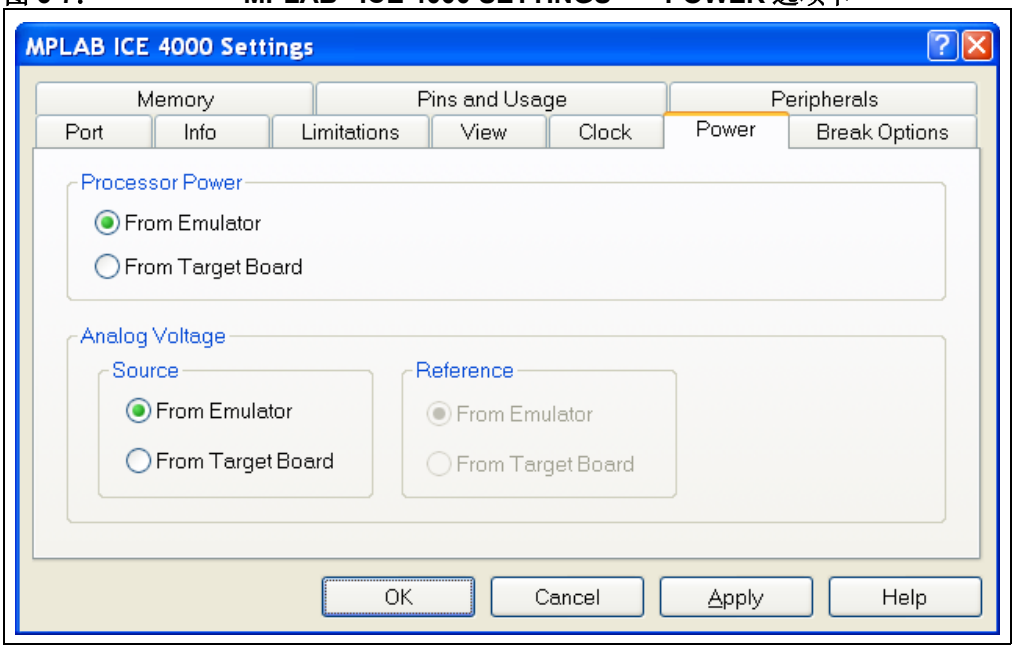
仿真器的初始化可能需要 1 分钟。在初始化过程中，将听到继电器的滴答声。初始化完成后就可以使用 MPLAB ICE 4000 了。

6.5　MPLAB ICE 4000 设置

在使用 MPLAB ICE 4000 进行调试之前，必须确保它已经正确设置：

1. 打开 MPLAB ICE 4000 Settings （MPLAB ICE 4000 设置）对话框 （通过 *Debugger>Settings* 菜单）。
2. 选择 **Power** （电源）选项卡并检查所有选项是否均设置为 **From Emulator （从仿真器）**（见图 6-7）。

图 6-7：　　MPLAB® ICE 4000 SETTINGS——POWER 选项卡

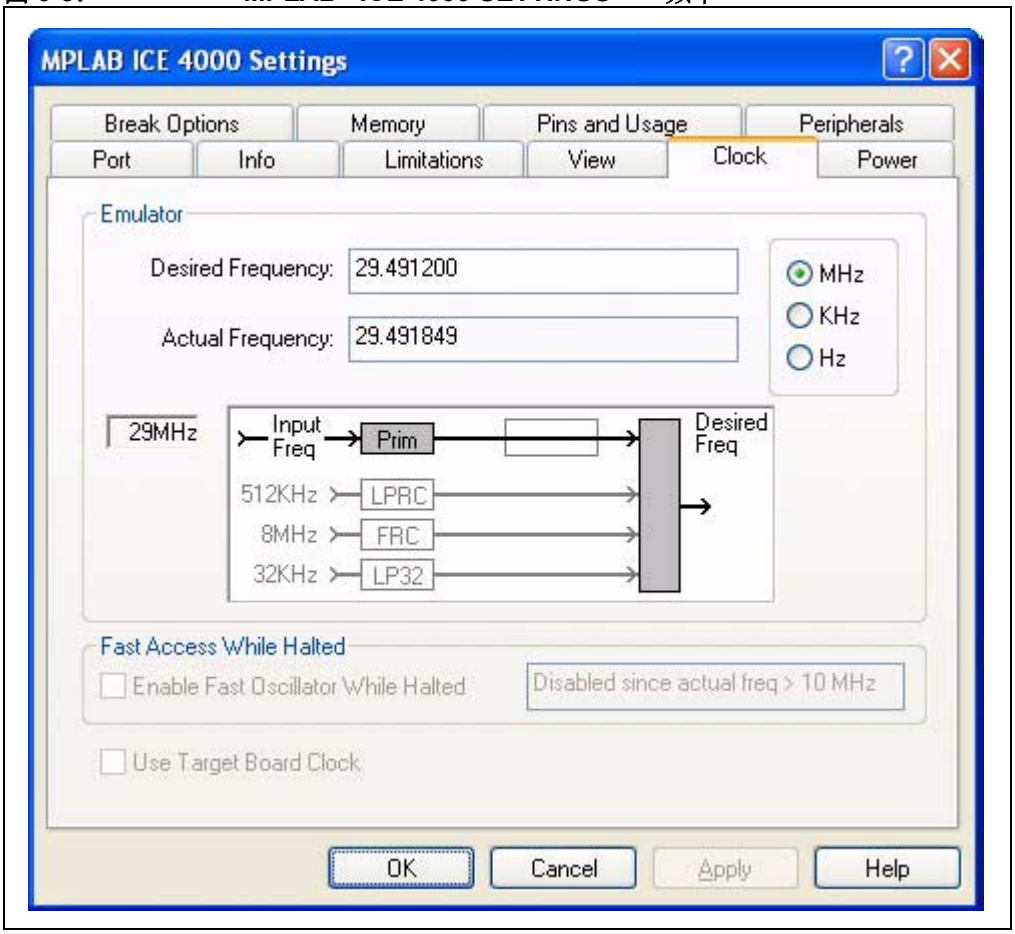


3. 选择 **Clock** （时钟）选项卡并将 Desired Frequency （目标频率）设置为 29.4912 MHz （图 6-8）。

注：	如果使用 `Flash LED with dsPIC30F6012.s` 文件，则将处理器频率设置为 16 MHz。其他演示板使用的处理器频率为 29.4912 MHz。

4. 单击 **Apply** （应用），然后单击 **OK**。

图 6-8： MPLAB® ICE 4000 SETTINGS——频率



dsPICDEM 1.1 通用开发板使用 7.3728 MHz 的晶振，代码通过配置位选择 4x PLL 选项，这使得频率变成了原来的 4 倍。

因此，处理器以 29.4912 MHz 的频率运行。因为每个指令周期包括 4 个时钟周期，所以指令周期频率为 7.3728 MHz。

注意， **Use Target Board Clock** （使用目标板时钟）复选框为灰显。如果也将仿真器设置为使用目标电源，则只能使用演示板上的目标时钟。

© 2007 Microchip Technology Inc.

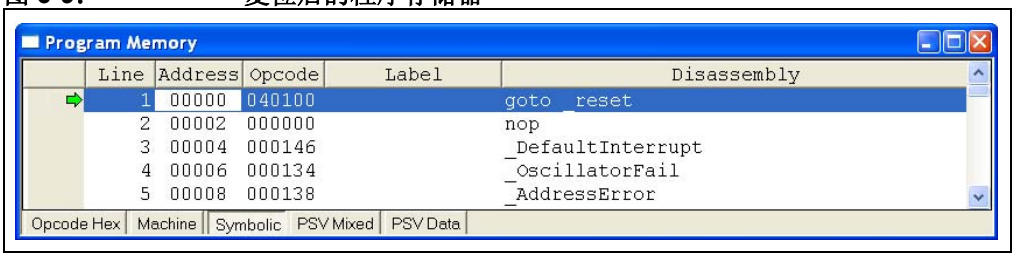
6.6 复位代码

将 MPLAB ICE 4000 选择为调试器后，程序计数器将自动设置为 0 （即复位向量）。MPLAB IDE 屏幕底部的状态栏中显示文本 “pc:0”，这表明程序计数器为 0。

可使用 *Debugger>Reset>Processor Reset* 菜单强制复位。

使用 *View>Program Memory* 菜单打开 Program Memory 窗口并单击窗口底部的 **Symbolic** 选项卡 （见图 6-9）。左边空白处的绿色箭头指向包含地址 0 处代码的第一行。地址 0 处包括一条 `goto __reset` 指令，该指令由链接器自动添加，未包含在源代码文件中。

图 6-9： 复位后的程序存储器

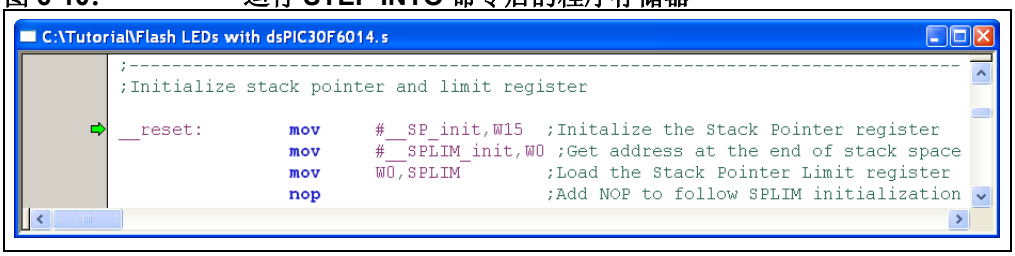


由于所有其他指令都包含在源代码中，所以此时可以关闭 Program Memory 窗口。

6.7 单步运行代码

现在可以使用 MPLAB ICE 4000 单步运行源代码了。使用 *Debugger>Step Into* 菜单以单步运行 `goto __reset` 指令。绿色箭头现在指向 `Flash LEDs with dsPIC30F6014.s` 文件中可执行代码的第一行，如图 6-10 所示。

图 6-10： 运行 STEP INTO 命令后的程序存储器



使用 *Debugger>Step Into* 菜单继续单步运行其余代码。 *Debugger* 菜单还允许您单步运行、单步跳过和连续单步运行代码。

- Step Into 命令执行当前指令后停止。如果当前指令是对子程序的调用，则程序计数器更改为被调用函数的起始地址。
- Step Over 命令执行下一个程序计数器单元前的所有代码。对于大多数指令而言，它与 Step Into 功能相同。但对于 CALL 指令，此命令执行完被调用子程序后返回。
- Animate 命令连续单步运行代码。该命令相当于重复执行 Step Into 操作，直到选择了 Halt 按钮。

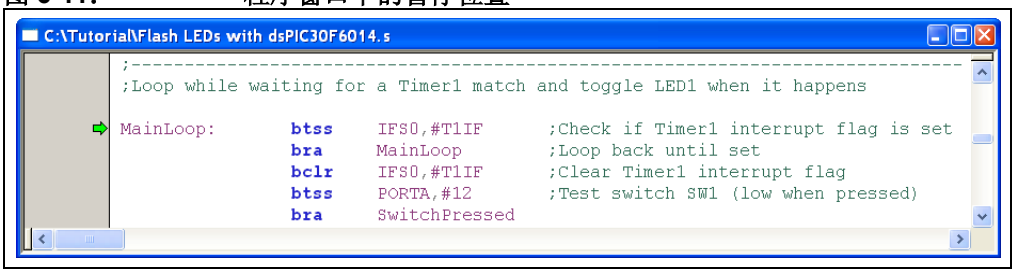
6.8 运行代码

运行应用程序 （选择 *Debugger>Run*）。 MPLAB IDE 工作区状态栏将显示“Running…”字样和一个进度条。

除了前进的进度条外， MPLAB IDE 屏幕上无任何变化。因为程序计数器在变化，故无法显示当前执行点，所以左边空白处的绿色箭头变为透明。因为代码正在演示板上的 dsPIC 器件中运行，故演示板上的 LED 不断闪烁。

停止程序执行 （选择 *Debugger>Halt*）。执行 Halt 命令后， MPLAB IDE 将更新其窗口，并用绿色箭头指示当前执行点，如图 6-11 所示。

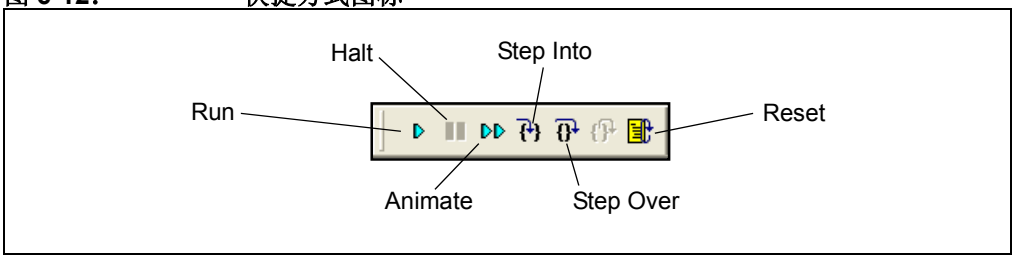
图 6-11： 程序窗口中的暂停位置



6.9 Debug 工具栏和热键

Debug 工具栏提供了控制 MPLAB ICE 4000 的快捷方式图标，如图 6-12 所示。

图 6-12： 快捷方式图标



选择 MPLAB ICE 4000 后， Debug 工具栏将自动打开（为了方便操作，可将此工具栏拖到桌面的任一位置）。单击相应图标以运行、暂停、连续单步运行、单步运行、单步跳过或复位程序。

MPLAB ICE 4000 还使用以下功能键快速访问其主要调试功能：

<F5> Halt
<F6> Reset
<F7> Single Step
<F9> Run

右击源代码行将显示一个功能菜单，可在此选择其他功能。其中最重要的功能是 Set Breakpoint 和 Run to Cursor。

如果工具栏未显示，则可通过 *View>Toolbars>Debug* 菜单使能。

© 2007 Microchip Technology Inc.

6.10 断点

MPLAB ICE 4000 允许您设置断点——在需要暂停执行的代码处设置断点。您可在源代码窗口、 Program Memory 窗口或 Disassembly 窗口中直接设置断点。

在本示例中，在翻转 PORTD 中的位来点亮 LED 的代码处设置断点。向下滚动到第 95 行，可找到 `btg LATD,#0` 指令。注意，行号显示在状态栏中。

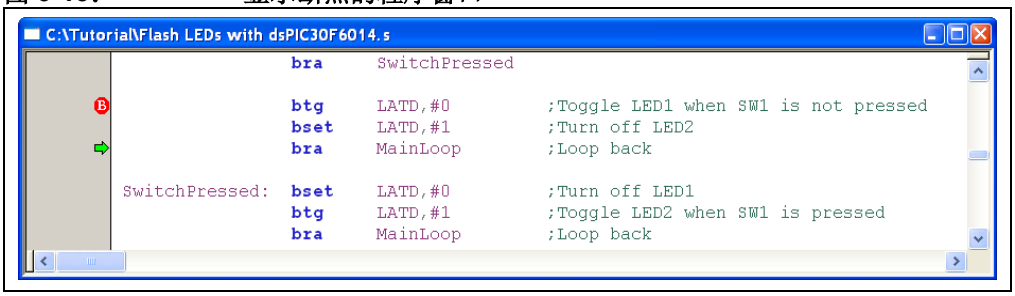
> **注：** 如果使用 `Flash LED with dsPIC30F6012.s` 文件，请在第 95 行的 `btg LATD,#4` 指令处设置断点。如果使用 `Flash LED with dsPIC30F2010.s` 文件，请在第 90 行的 `bclr IFS0,#T1IF` 指令处设置断点。如果使用 `Flash LED with dsPIC30F4011.s` 文件，请在第 98 行的 `btg LATB,#0` 指令处设置断点。

右击该代码行，然后从弹出菜单中选择 Set Breakpoint。

也可通过双击该行设置断点。此选项可能需要通过 *Edit>Properties* 菜单启用。

按下 <F9> 或选择 *Debugger>Run* 菜单运行程序。程序执行到此断点所在行 （即翻转 I/O 引脚状态）后暂停。注意，演示板上的 LED 在每次代码暂停时发生变化。程序暂停后，绿色箭头指向断点后的第二条指令，如图 6-13 所示。

图 6-13： 显示断点的程序窗口



在 dsPIC 器件上进行在线调试的局限性之一就是遇到断点停止时会产生 skew。当硬件检测到断点地址时，断点后的下一条指令已经被执行了。其实，这一问题并不难解决，只需调节断点位置即可。但是，如果断点后是一条转移指令，这样做就会产生混淆。

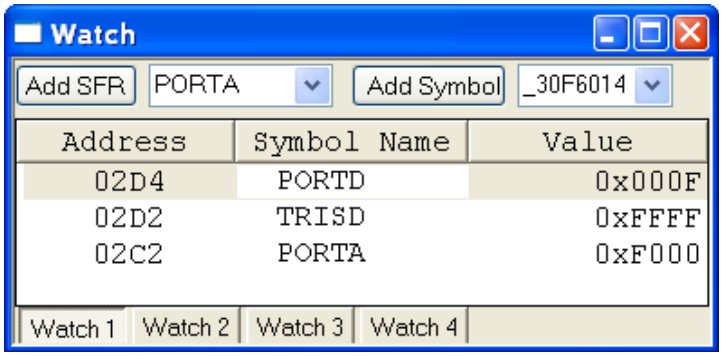
组合使用单步执行键 <F7> 和运行键 <F9>。注意，每次代码执行时，都会在断点后停止。注意，按下开发板上的开关 SW1 可解除暂停。

> **注：** 对于 dsPICDEM 入门演示板，请使用开关 S1。对于 dsPICDEM 2 演示板，请使用开关 S5。 dsPICDEM 28 引脚入门演示板上没有开关。

6.11 WATCH 窗口

使用 MPLAB ICE 4000 时，有几种方法可查看存储器。Watch 窗口（图 6-14）是最有效的方法之一。 Watch 窗口允许您指定不同编程条件下想要观察的存储单元。打开 Watch 窗口，请选择 *View>Watch* 菜单。

图 6-14： WATCH 窗口



例如，如需将 PORTD 添加到 Watch 窗口，可在该窗口顶部的 **Add SFR** （特殊功能寄存器）选择框中输入 “PORTD”。然后单击 **Add SFR** 将其添加到 Watch 列表。

也可直接在 Watch 窗口中输入寄存器名称。

请以相同的方式添加 TRISD 和 PORTA。现在 Watch 窗口中应列有 3 个特殊功能寄存器。每栏分别表示符号的 “Address”、“Symbol Name”和 “Value”。

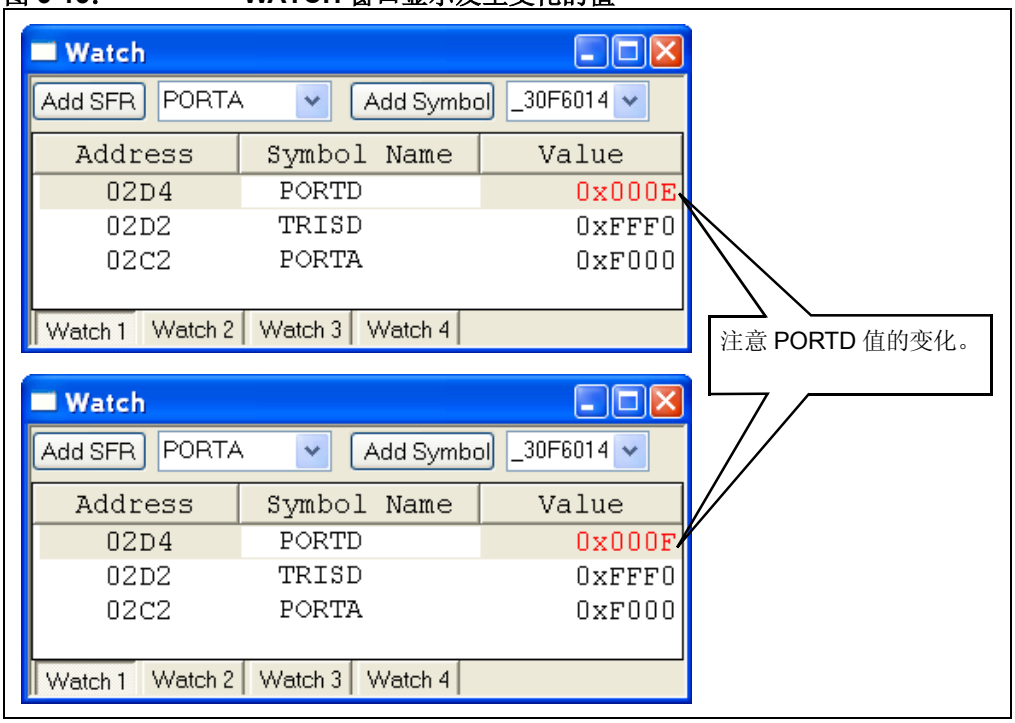
按下 <F6> 复位代码。注意， TRISD 的值为 0xFFFF。这是复位后 TRISD 寄存器的状态。按下 <F9> 运行代码。代码将执行并在先前设置的断点处暂停。注意，此时 TRISD 寄存器的值变为 0xFFF0。通过对 TRISD 执行写操作，可设置 I/O 端口方向。可在 Watch 窗口中看到这一变化。注意，每次执行 Step 或 Halt 命令时，更改过的值以红色显示，而未更改的值以黑色显示。

注： 如果使用 `Flash LED with dsPIC30F6012.s` 文件，则 TRISD 将变为 0xFF0F。如果使用 `Flash LED with dsPIC30F2010.s` 文件，则 TRISD 将变为 0xFFFE。如果使用 `Flash LED with dsPIC30F4011.s` 文件，则 TRISD 将变为 0xFFFC。

注意， Watch 窗口中 PORTD 的状态，然后按下 <F6> 再次运行程序。每次代码运行并在断点处暂停时， PORTD 中的位就会发生变化。这表示代码通过翻转该引脚来点亮或熄灭 LED。

© 2007 Microchip Technology Inc.

图 6-15：　　WATCH 窗口显示发生变化的值



分别在按下和释放开关 SW1 的情况下按下 <F7> 单步执行代码。注意观察按下开关时 PORTA 将如何变化。该开关位于 PORTA 的 RA12 引脚上，且可在 Watch 窗口中查看该输入引脚的状态。

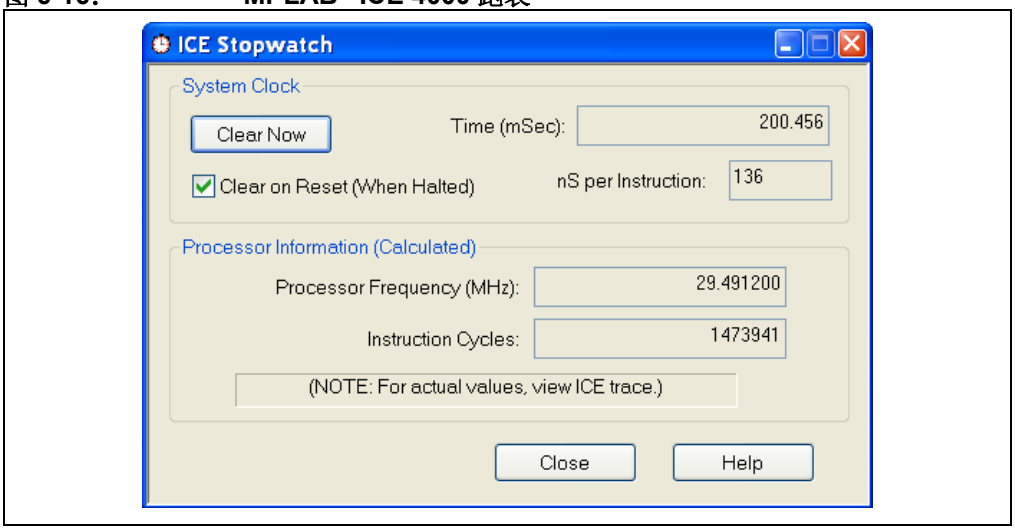
注：　如果使用 dsPICDEM 入门演示板，请按开关 S1 并在 Watch 窗口中观察 PORTC 的状态。该开关位于 PORTC 的引脚 RC13 上。如果使用 dsPICDEM 2 开发板，请按开关 S5 并在 Watch 窗口中观察 PORTE 的状态。该开关位于 PORTE 的引脚 RE8 上。

6.12 跑表

可使用 MPLAB ICE 4000 的跑表功能测量两个事件之间的执行时间。跑表将跟踪已执行的指令周期数和这些周期消耗的时间。它通过在设置中输入的 “Processor Frequency”来计算时间。

选择 *Debugger>Stopwatch* 菜单可打开 Stopwatch。

图 6-16： MPLAB® ICE 4000 跑表



现在请完成以下简单练习：

1. 选中 **Clear on Reset** （复位时清零）。此选项将确保执行复位操作后跑表将从零开始计数。
2. 在源代码窗口中单击并按 <F6> 复位代码，然后按 <F9> 运行。代码应执行到先前在**第 6.10 节 “断点”**中设置的断点处暂停。
3. 注意， **Time (mSec):** 的值为 200 ms。代码将 Timer1 周期寄存器设置为 1/5 秒（200 ms）。在检测到定时器周期的代码处设置断点。
4. 请按 <F9> 运行。代码再次在断点处暂停。
5. 注意，现在跑表时间为 400 ms，这是因为又过了一个定时器周期。

可按下 **Clear Now** （现在清零）按钮随时从零开始计时。这样就可以灵活轻松地计算单个循环和函数的执行时间。

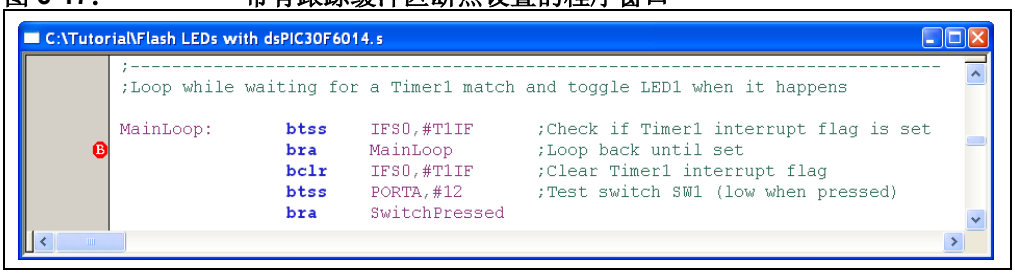
© 2007 Microchip Technology Inc.

6.13 跟踪缓冲区

跟踪缓冲区是 MPLAB ICE 4000 的一个便捷功能。在菜单 *View>ICE Trace* 下可找到此功能。

跟踪缓冲区保存已执行指令的列表。它可存储 65,000 条以上的指令。跟踪缓冲区在后台运行，且无需显式使能。一有代码执行，跟踪缓冲区就会将其记录下来。

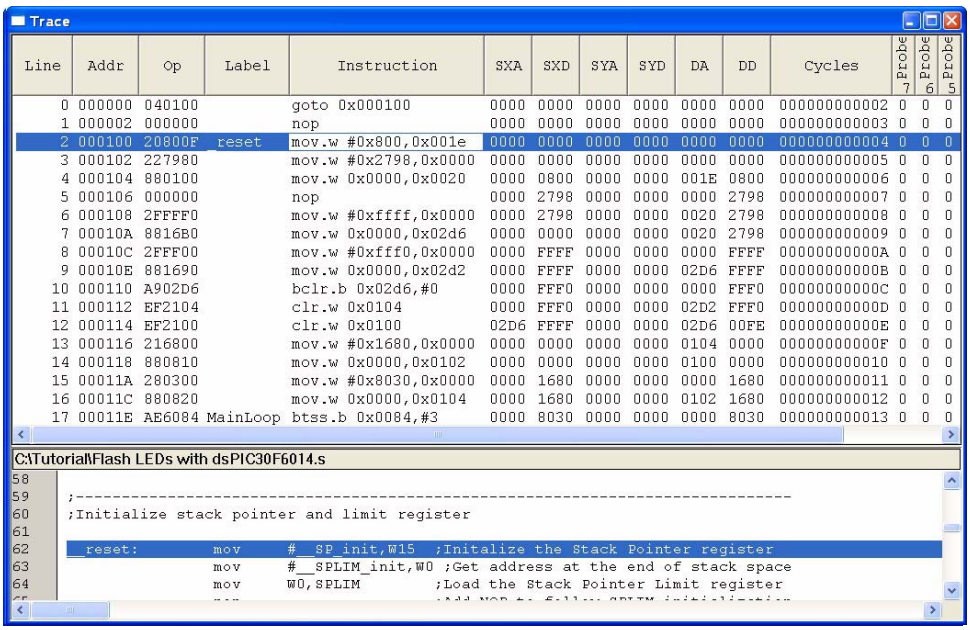
图 6-17：　　带有跟踪缓冲区断点设置的程序窗口



在以下简单实验中，请按以下步骤执行数行代码并在 Trace 窗口中查看它们：

1. 在程序窗口中，在 `bra MainLoop` 指令处设置断点，即 `MainLoop:` 地址标号之后 （见图 6-17）。
2. 按 <F6> 复位代码，然后按 <F9> 运行。代码将在断点处暂停。
3. 选择 *View>ICE Trace* 显示跟踪缓冲区。所有已执行的指令均显示在 Trace 窗口中，从地址 0 处的 `goto __reset` 指令开始 （见图 6-18）。如果 Trace 视图为空白，则应检查 **Complex Trigger Settings** （见**第 6.14 节 “复杂触发”**），看看哪一个筛选跟踪没有选择。
4. 滚动到 Trace 窗口的最下面，查看最后一条已执行的指令。注意 Trace 窗口记录到 `btss IFS0,#T1IF` 指令，即断点的前一条指令，然后停止。
5. 比较跟踪缓冲区和源代码，将发现断点前的所有指令均显示在 Trace 窗口中。注意，如果在 Trace 视图中选择了某一行，则在窗口的底部将突出显示相关的源代码行，如图 6-18 所示。

图 6-18： TRACE 窗口



6.14 复杂触发

复杂触发功能允许您设置 MPLAB ICE 4000，使之可在一组复杂的条件下暂停。此功能有助于捕捉代码中的异常情况。例如，可设置一个复杂触发，以便在向 PR1 寄存器写入数据时暂停。

由于代码中有一条 `mov W0,PR1` 指令，因此上述功能可以实现。要查找 PR1 寄存器的地址，查看 `p30f6014.gld` 链接描述文件并找到行 `PR1 = 0x0102;` 即可。

使用 *Debugger>Complex Triggers and Code Coverage* 菜单打开 MPLAB ICE 4000 Analyzer （MPLAB ICE 4000 分析程序）对话框。此对话框提供了以下选项卡：

- **Complex Trigger Settings**——用于最多 4 个事件 （如，程序存储器或数据存储器读或写操作）同时发生时触发暂停。也可在 **Logic Probes:**（逻辑探针）输入上检测到逻辑信号时暂停。
- **Trigger In/Out Settings** （触发输入 / 输出设置）——用于触发单个外部触发输入信号，并输出一个脉冲，用于触发示波器或其他数据捕捉器件。
- **Code Coverage** （代码覆盖）——用于跟踪已使用的程序存储器指令。它对于查看测试时是否执行了所有代码非常有用。
- **Internal Triggers** （内部触发）——dsPIC 处理器模块中内置的其他触发电路，它允许多个事件同时发生时触发暂停。

选择 **Complex Trigger Settings** 选项卡并按照图 6-19 和表 6-1 中的说明进行设置。

© 2007 Microchip Technology Inc.

图 6-19： MPLAB® ICE 4000 ANALYZER 对话框

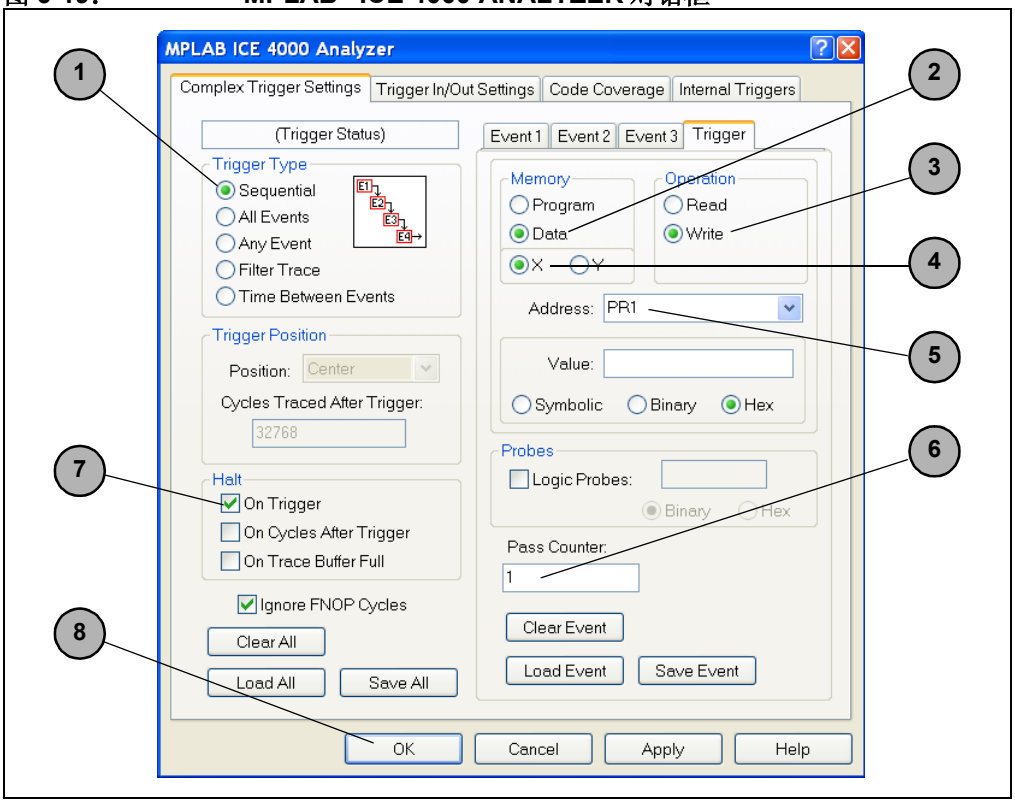
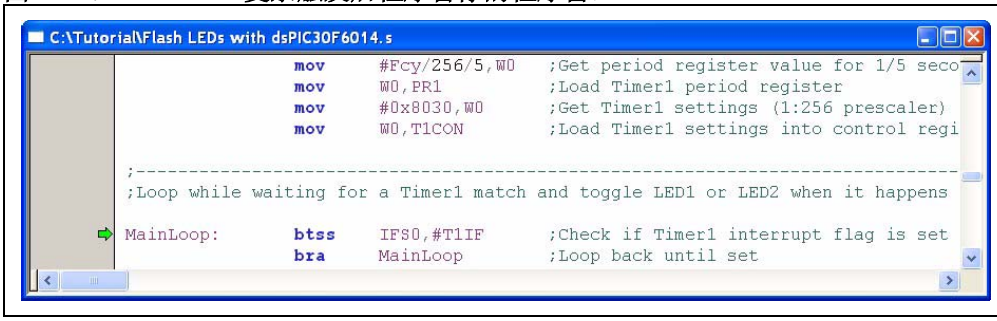


表 6-1： 复杂触发设置

参考	设置
1	确保 **(Trigger Status)** （触发状态）设置为 **Sequential** （顺序）。
2	设置 **Memory** （存储器）类型为 **Data** （数据）。
3	设置 **Operation** （操作）类型为 **Write** （写）（因为我们希望检测到对数据存储器地址 PR1 的写操作）
4	在 **Memory** 类型下选择 **X** （dsPIC® 器件中所有数据存储器的写操作均是对 X 存储空间的）
5	设置 **Address:** 为 “PR1”
6	保留 **Pass Counter:** （次数计数器）为 “1”
7	设置 **Halt** 类型为 **On Trigger** （触发时）
8	单击 **OK** 保存 **Complex Trigger Settings**

请按 <F6> 复位，然后按 <F9> 运行代码。注意，写 PR1 寄存器的指令执行后，代码将暂停几个周期，如图 6-20 所示。

图 6-20：　　复杂触发后程序暂停的程序窗口



此演示只是复杂触发的一个非常简单的应用。可在读写数据或程序存储单元中的特定数据时暂停。也可结合次数计数器，使其在所需事件发生若干次后，或者事件发生后再执行若干个指令周期后才暂停。

这里提供有 4 个独立的触发事件，它们可一起使用，因此事件必须按照指定顺序发生后才能暂停。复杂触发提供了非常强大的调试功能。

© 2007 Microchip Technology Inc.

第 7 章 MPLAB ASM30 汇编器

7.1 MPLAB ASM30 汇编器概述

在了解了如何创建和编译项目以及如何使用工具进行模拟或调试后，我们将花费一点时间来学习如何编写代码。既然 MPLAB IDE 中包括了 MPLAB ASM30 汇编器，我们将只讨论使用此语言工具时的一些要点。

MPLAB IDE 汇编器基于一个源代码开放的 GUN 软件，一些用户可能对此软件比较熟悉。此汇编器对源代码文件中的指令和伪指令进行解释以生成目标代码。链接器用于将目标代码转换为一个最终输出 （Hex）文件以编程器件 （见**第 9 章 “MPLAB LINK30 链接器”**）。

在 dsPIC 器件中，指令在运行时执行。指令是 dsPIC 处理器的本机语言。但是，dsPIC 指令集不在本文档的讨论范围内。欲知有关 dsPIC 指令集的详细信息，请参见《dsPIC30F 程序员参考手册》（DS70157B_CN）。

伪指令在编译时由汇编器解释，用于定义存储器的段、初始化常量、声明和定义符号（变量和标号等）以及替换文本等等。文档 《MPLAB® ASM30、 MPLAB® LINK30 和实用程序用户指南》（DS51317F_CN）列出了一些伪指令及其用法。每条伪指令之前必须有一个句点 （“.”）。

我们将讨论一些最常用的伪指令，这样在编写代码时就知道需要哪些伪指令了。其中一些伪指令已在前述教程的代码示例中使用过了。

> **注：** 本章的讨论基于 MPLAB ASM30 汇编器 1.31 版。部分信息会随新版本的发布而过时。

7.1.1 指令和伪指令的一般格式

指令和伪指令的一般格式如下：

```
[label:]     instruction[operands]   [; comment]
[label:]     directive[arguments]    [; comment]
```

标号用于标记在代码中的位置。链接时，为标号分配存储地址；标号的定义可以以“.”（句点）开始，但必须以 “:”（冒号）结束。

指令使用操作数来提供源和目标信息。操作数包括：

- **立即数**——可以是十六进制、八进制、二进制或十进制值。所有立即数必须以符号 “#”开头。
- **寄存器和存储器地址** ——可以是工作寄存器、累加器、通用寄存器 （General Purpose Registers， GPR）和特殊功能寄存器 （SFR）。
- **条件代码** ——即状态位，如 Z （零位）或 C （进位），在条件转移指令中用作操作数。

参数与操作数相似，伪指令使用它提供源和目标信息。

表 7-1 概括了指令和伪指令的语法规则。

表 7-1： 语法规则

字符	说明	用法
.	句点	开始伪指令或标号
:	冒号	结束标号
#	井号	开始立即数
;	分号	开始单行注释
/*		开始多行注释
*/		结束多行注释

7.2 常用伪指令

下面列出了一些常用伪指令。它们在 dsPIC 模板文件中以示例的形式给出，该模板文件位于以下目录：

`C:\Program Files\Microchip\MPLAB ASM30 Suite\Support\templates\assembly`

前五条伪指令也可在教程的**第 4 章 “MPLAB SIM 软件模拟器”**（在该章我们学习了如何创建和编译项目）中找到。尽管教程中没有列出其他五条伪指令，但您也可能会用到它们。

`.equ`	赋值给符号
`.include`	包括另外一个文件到当前文件中
`.global`	使符号全局可见
`.text`	开始一段可执行代码
`.end`	结束文件内汇编
`.section`	开始一段代码或数据 （程序存储器或者数据存储器中）
`.space`	分配段内空间
`.bss`	添加变量到未初始化的数据段
`.data`	添加变量到已初始化的数据段
`.hword`	声明段内数据字
`.palign`	对齐段内代码
`.align`	对齐段内数据

.equ

.equ 是汇编源文件的常用伪指令之一。 .equ 伪指令用于定义符号并给其赋值。在例 7-1 中， .equ 伪指令将立即数 7372800 赋值给符号 FCY。在此， FCY 是一个常数，在整个代码中表示指令周期频率。

例 7-1： .equ

```
; 与程序有关的常数 （代码中使用的立即数）
   .equ   FCY, #7372800        ; 指令周期频率 （Osc x PLL / 4）
;==========================================================================
```

.include

.include 伪指令使用时添加指定文件的内容到汇编源代码中，如例 7-2 所示。它的常见用法之一就是添加标准处理器头文件中的定义。

例 7-2： .include

```
    .equ    30F6014, 1
   .include “p30f6014.inc.”
;--------------------------------------------------------------------------
```


© 2007 Microchip Technology Inc.

.global

使用 .global 伪指令可使在当前文件内定义的标号用于其他文件。在例 7-3 中，因为 __reset 符号被声明为全局符号，所以链接器可将其用作跳入和跳出复位向量的地址。您需要使用 __reset: 标号来标示代码的开始，并将其包括在项目的目标文件中（该目标文件来自汇编器、编译器或库文件）。

例 7-3： .global

```
; 全局声明
  .global __reset              ; 第一行代码的标号
  .global __OscillatorFail     ; 声明振荡器故障陷阱程序标号
  .global __AddressError       ; 声明地址错误陷阱程序标号
```

.text

这是 .section 伪指令的一个特例。 .text 伪指令用于通知汇编器该指令之后的代码将存入程序存储器的可执行段内 （见例 7-4）。

例 7-4： .text

```
; 代码开始
  .text                        ; 代码段开始
```

.end

.end 伪指令用于标示汇编源文件的结束 （见例 7-5）。

例 7-5： .end

```
  .end                         ; 此文件的代码结束
```

.section

.section 伪指令声明存储器段。此段可位于 RAM 或程序存储器中，具体由伪指令后面的属性确定。在例 7-6 中，名称为 MyDataSection 的段位于未初始化的 Near 数据存储区中。名称为 MyOtherSection 的段位于 Y 数据存储区中。《MPLAB® ASM30、MPLAB® LINK30 和实用程序用户指南》（DS51317F_CN）和 "*dsPIC30F Language Tools Quick Reference Card* （DS51322）中有段类型的完整列表。

例 7-6： .section

```
;RAM 变量
      .section MyDataSection, bss, near
Var1:  .space 1              ; 分配空间给变量 （以字节为单位）
      .section MyOtherSection, ymemory
Array1:.space 20             分配空间给数组 （以字节为单位）
```

.space

.space 伪指令指示汇编器保留当前段的空间。在例 7-7 中，保留了一个字节的存储空间给 Var1 变量。

例 7-7： .space

```
;RAM 变量
        .section MyDataSection, bss, near
Var1:  .space 1              ; 分配空间给变量 （以字节为单位）
```

.bss

.bss 伪指令是 .section 伪指令的一个特例。它将未初始化的数据变量添加到一个未初始化的数据段中。在例 7-8 中， Var2 存入一个未初始化的数据存储器。

例 7-8： .bss

```
;RAM 变量
      .bss
Var2:  .space 2              ; 分配空间给变量 （以字节为单位）
```

.data

.data伪指令是.section伪指令的一个特例。它将已初始化的数据变量添加到一个已初始化的数据段中。在例 7-9 中，数组 MyRAM 存入数据存储器，汇编器把数据 0x1111、 0x2222 和 0x3333 存入程序存储器段。

例 7-9： .data

```
; 已初始化的 RAM 变量
      .data
MyRAM: .hword 0x1111, 0x2222, 0x3333
```

要使用已初始化的数据，将正确的启动代码添加到项目中以将数据复制到 RAM，这一点很重要。运行时启动模块在运行时库文件 libpic30.a 中。此文件位于 pic30_tools\lib 文件夹中。

欲知运行时库中启动模块功能的更多信息，请参阅 《MPLAB® ASM30、 MPLAB® LINK30 和实用程序用户指南》（DS51317F_CN）。

.hword

.hword 伪指令声明段内已初始化的数据字。它也可声明程序存储器中的常数。在例 7-10 中， MyData 数组被存入程序存储器。数据字 0x0002、 0x0003 和 0x0005 保存在程序存储器的相邻字中。由于程序存储器为 24 位宽，故各字的高字节将为 0x0。

例 7-10： .hword

```
        .palign 2            对齐下一字到两字节边界
MyData:.hword 0x0002, 0x0003, 0x0005
```


© 2007 Microchip Technology Inc.

.palign

.palign 伪指令对齐程序存储器段内的数据。在例 7-11 中，变量 MyData 将从一个偶数地址 （可被 2 整除）开始。

例 7-11： .palign

```
        .section .myconstbuffer, “x”
        .palign 2               对齐下一字到两字节边界
MyData:.hword 0x0002, 0x0003, 0x0005
```

.align

.align 伪指令对齐段内的数据。在例 7-12 中，变量 Array3 将从可被 8 整除的地址开始。当使用模寻址功能或 dsPIC30F 处理器时， .align 伪指令特别有用。

例 7-12： .align

```
        .bss
        .align 8
Array3:.space 6             分配空间给变量 （以字节为单位）
```

7.3 代码示例

学习了伪指令及其格式后，我们来看一个示例，看看它们是如何工作的。该示例对前述章节使用过的 Flash LEDs with dsPIC30F6014.s 文件中的代码进行了解释。

> **注：** Flash LEDs with dsPIC30F6012.s、Flash LED with dsPIC30F2010.s 和 Flash LEDs with dsPIC30F4011.s 等其他教程文件都与之非常相似，以下描述均适用。

7.3.1 代码描述

Flash LEDs with dsPIC30F6014.s 文件以注释开始。在此文件中，注释说明了许可协议和程序的作用 （根据开关 SW1 的状态使 LED 闪烁）。注释总是以分号 （;）开始，或用具有 C 风格的块注释 （/* */）表示。

注释后紧跟 __30F6014 标号定义，以允许头文件检查是否使用了正确的处理器。将标准头文件包括在内以定义各 SFR 中的所有位。

例 7-13：

```
;=============================================================================
; 使用 Timer 1 计数，未按开关 SW1 时，使 LED1 闪烁
;                   按下开关 SW1 时，使 LED2 闪烁
;=============================================================================

   .equ     __30F6014, 1
   .include  "p30f6014.inc"
```

下一代码段包含 __reset 标号和各种错误陷阱标号的全局声明。这使得链接器可以确定要存入复位向量处的 goto 指令的正确地址，以及要存入错误陷阱程序的中断向量表的地址。（有关错误陷阱和中断向量表的更多信息，请参阅**第 1.2.10 节 “中断”**）。

例 7-14：

```
;-------------------------------------------------------------------------
; 全局声明

        .global __reset                 ; 第一行代码的标号
        .global __OscillatorFail        ; 声明振荡器故障陷阱程序标号
        .global __AddressError          ; 声明地址错误陷阱程序标号
        .global __StackError            ; 声明堆栈错误陷阱程序标号
        .global __MathError             ; 声明数学错误陷阱程序标号
```

在下一代码段中，定义配置位以使用正确的振荡器模式和看门狗定时器设置等编程处理器。配置寄存器地址（如 __FOSC 和 __FWDT）在链接描述文件 p30f6014.gld 中定义。配置寄存器值（如 CSW_FSCM_OFF 和 XT_PLL4）在处理器头文件 p30f6014.inc 中定义。

例 7-15：

```
;-------------------------------------------------------------------------------
; 配置位

        config __FOSC, CSW_FSCM_OFF & XT_PLL4
        config __FWDT, WDT_OFF
        config __FBORPOR, PBOR_OFF & BORV_27 & PWRT_16 & MCLR_EN
        config __FGS, CODE_PROT_OFF
```

给 FCY 标号赋值，从而可以轻松地更改频率。只需修改一行即可使代码能够适用于不同的振荡器选项。

例 7-16：

```
;-------------------------------------------------------------------------------
; 与程序有关的常数（代码中使用的立即数）

    .equ   FCY, #7372800            ; 指令周期频率（Osc x PLL / 4）
```

.text 伪指令通知汇编器接下来的代码应存入默认代码段。

例 7-17：

```
;=============================================================================
; 代码开始

    .text                           ; 代码段开始
```


© 2007 Microchip Technology Inc.

链接器将 __reset: 看作标准标号，并添加代码使程序在复位后跳转到该标号。必须将此标号声明为全局标号以供链接器使用。

分配了所有 RAM 变量后，链接器找到最大可用堆栈空间，并将起始地址分配给 __SP_init 标号。代码将此标号载入堆栈指针寄存器 W15，这样便建立了软件堆栈。注意 “#”符号表示一个立即数。

链接器还提供可用堆栈空间的结束地址，代码将此值从 __SPLIM_init 标号载入堆栈指针极限寄存器 （SPLIM）。这样便建立了用于堆栈溢出的错误检查。如果堆栈指针 W15 等于 SPLIM 中的地址，将发生堆栈错误。注意此操作是用两条指令完成的。先将 __SPLIM_init 值载入 W0 寄存器，然后将此值移入 SPLIM 寄存器。不可能将 16 位立即数和 13 位 Near 存储区地址编码到一条 24 位指令中。

例 7-18：

```
;-------------------------------------------------------------------------------
; 初始化堆栈指针极限寄存器

__reset:  mov    #__SP_init, W15      ; 初始化堆栈指针寄存器
          mov    #__SPLIM_init, W0    ; 获取堆栈空间的结束地址
          mov    W0, SPLIM            ; 装载堆栈指针极限寄存器
          nop                         ; 在 SPLIM 初始化后添加一条 NOP 指令
```

设置了堆栈后，代码将初始化 I/O 端口以驱动 PORTD 上的 LED。 LED 位于 PORTD 上的 bit 0 至 bit 3，当这些引脚被拉为低电平时被点亮。代码随后会将端口锁存寄存器 LATD 中的相应位置 1，这样，当 I/O 引脚被转换为输出时， LED 将熄灭。代码随后将清零端口三态寄存器 TRISD 中的相同位，这样， I/O 引脚被转换为输出。最后，代码清零 LATD 的 bit 0 以点亮一个 LED。

例 7-19：

```
;-------------------------------------------------------------------------------
; 初始化 PORTD 上的 LED 输出引脚 bit 0-3

          mov    #0xffff, W0          ; 初始化 LED 引脚数据为关闭状态
          mov    W0, LATD
          mov    #0xfff0, W0          ; 设置 LED 引脚为输出
          mov    W0, TRISD
          bclr   LATD, #0             ; 点亮 LED1
```

将 Timer1 的周期初始化为 1/5 秒，这样 LED 的闪烁将比较适宜。代码将 Timer1 控制寄存器 T1CON 清零以停止定时器，并将 Timer1 计数寄存器 TMR1 清零使之从零开始计数。

Timer1 周期寄存器 PR1 载入 1/5 秒内的计数值。由于我们已指定了指令频率 （FCY）、预分频比 （256）和时间因数 （1/5 秒），汇编器将计算此值。预分频器对递增定时器的时钟速率进行分频。

最后，对 Timer1 控制寄存器 T1CON 执行写操作以启动定时器，并使用一个预分频比为 1:256 的内部时钟源。

例 7-20：

```
;-------------------------------------------------------------------------------
; 将 Timer1 的周期初始化为 1/5 秒

          clr    T1CON            ; 控制寄存器清零以关闭 Timer1
          clr    TMR1             ; Timer1 从 0 开始计数
          mov    #FCY/256/5, W0   ; 获得 1/5 秒对应的周期寄存器值
          mov    W0, PR1          ; 装载 Timer1 周期寄存器
          mov    #0x8030, W0      ; 设置 Timer1 （预分频比为 1:256）
          mov    W0, T1CON        ; 载入 Timer1 的设置到控制寄存器
```

主代码循环从标号 MainLoop: 开始。测试 IFS0 寄存器中的 Timer1 中断标志位 T1IF 以确定定时器计数是否已达到了周期寄存器的值。如果周期没有结束，则代码跳转回 MainLoop。

Timer1 将 T1IF 位置 1 后，代码将跳过转移指令继续运行以使 LED 闪烁。清零 T1IF 位以检测定时器下一周期是否结束。

开关 SW1 与 RA12 引脚连接，因此代码将测试 PORTA 寄存器的 bit 12 以确定是否按下了开关 SW1。如果开关已被按下，则执行转移指令以翻转 LED2 ； 否则，代码将跳过转移指令并翻转 LED1。

LED1 与引脚 RD0 连接，因此翻转 LATD 的 bit 0 以点亮或熄灭 LED1。LED1 点亮时，通过清零 LATD 的 bit 1 可熄灭 LED2。注意，当 I/O 端口被用作输入时，将使用 PORTx 寄存器；当端口用作输出时，将使用 LATx 寄存器。更改了任何一个 LED 后，代码将跳转回 MainLoop。

例 7-21：

```
;-------------------------------------------------------------------------------
; 循环并等待 Timer1 匹配，发生匹配时翻转 LED1 或 LED2

MainLoop:      btss  IFS0, #T1IF          ; 检查 Timer1 中断标志位是否置 1
               bra   MainLoop             ; 不断循环，直到置 1 为止
               bclr  IFS0, #T1IF          ; Timer1 中断标志位清零
               btss  PORTA, #12           ; 测试开关 SW1 （按下时为低电平）
               bra   SwitchPressed

               btg   LATD, #0             ; 未按下 SW1 时，翻转 LED1
               bset  LATD, #1             ; 熄灭 LED2
               bra   MainLoop             ; 循环返回

SwitchPressed:bset   LATD, #0             ; 熄灭 LED1
               btg   LATD, #1             ; 按下 SW1 时，翻转 LED2
               bra   MainLoop             ; 循环返回 MainLoop
```


© 2007 Microchip Technology Inc.

错误陷阱程序紧跟代码的其他部分。如果因灾难性错误（例如，振荡器故障或转移到不存在的存储器）而导致代码运行失败，硬件会将执行切换到相应错误陷阱程序。每个程序都有一个全局地址标号，如 __OscillatorFail:，链接器将使用这些地址创建中断向量表。每个错误陷阱程序将点亮一个 LED 并无限制的循环。

例 7-22：

```
;=============================================================================
;错误陷阱

;-----------------------------------------------------------------------------
;振荡器故障错误陷阱程序

    .text                               ;代码段开始
_OscillatorFail:
          bclr   LATD, #3               ;点亮 LED4
          bra    __OscillatorFail       ;发生振荡器故障时进入无限循环

;-----------------------------------------------------------------------------
;地址错误陷阱程序

__AddressError:
          bclr   LATD, #3               ;点亮 LED4
          bra    __AddressError         ;发生地址错误时进入无限循环
```

错误陷阱程序后，有一个 .end 伪指令，指示此文件中没有其他要汇编的代码。

例 7-23：

```
    .end                                ;此文件中的代码结束
```

汇编器总是生成需要进行链接的目标文件。如需了解 LINK30 链接器和它如何从目标文件获得代码和数据以及创建最终输出文件的信息，请跳至**第 9 章 “MPLAB LINK30 链接器”**。如果要使用 MPLAB C30 编译器，请继续**第 8 章 “MPLAB C30 C 编译器”**。

注：

© 2007 Microchip Technology Inc.

第 8 章 MPLAB C30 C 编译器

8.1 MPLAB C30 C 编译器概述

很多为 dsPIC 器件编写的软件都是用 “C”语言实现的。 MPLAB C30 C 编译器是一个符合 ANSI 标准的 C 编译器，它使您能为 dsPIC 数字信号控制器编写统一的模块化代码，这些代码比汇编语言编写的代码更容易移植和理解。除了 “C”语言本身的优点，MPLAB C30 C 编译器所提供的库也使得它成为一个功能强大的编译器。例如，浮点数、三角函数、滤波器和 FFT 算法都很难用汇编语言实现。但是通过 DSP、外设和标准数学函数库，就可以很轻松地调用这些程序。“C”语言的模块化功能还降低了函数间交互的可能性。

本章目的不在于描述 C 语言的细节，而是为您提供快速入门和熟练运行该编译器所必需的知识。有关 MPLAB C30 C 编译器操作的详细信息，请参见 《MPLAB® C30 C 编译器用户指南》（DS51284F_CN）。

注： 本章的讨论基于 MPLAB C30 C 编译器 1.30 版。部分信息会随新版本的发布而过时。

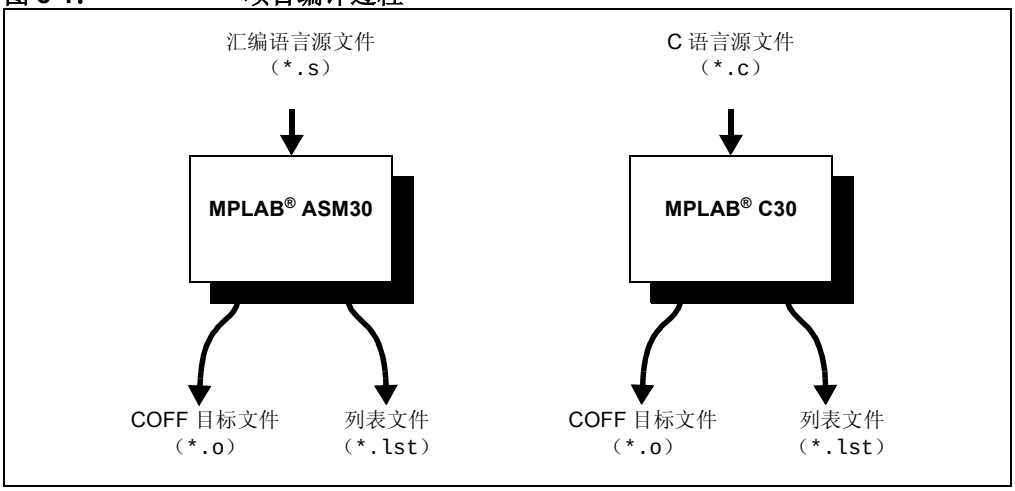
8.2 MPLAB C30 C 编译器项目

已在**第 3 章 “MPLAB 集成开发环境”**中了解到 MPLAB 项目，但这些项目仅使用了汇编语言源文件 （.s）。现在您将了解到 MPLAB C30 C 编译器项目与之非常相似，但是它使用的是 C 语言源文件 （.c）和归档库文件 （.a）。

以前介绍过，编译汇编语言项目是一个两步过程。汇编语言源文件 （.s）被单独汇编成目标文件 （.o），然后再将目标文件链接起来生成输出文件 （.hex 和 .cof），如图 8-1 所示。

编译 MPLAB C30 C 编译器项目也是一个两步过程，先将 “C”语言源文件 （.c）编译为目标文件，然后再将目标文件链接起来生成输出文件。

图 8-1： 项目编译过程



除了 “C”文件外，项目还可能包括与目标文件一起进行链接的库文件。库文件是由预编译目标文件创建的，实际上是那些无需编译即可用到项目中的函数。

项目还可包含 LINK30 链接器的一个链接描述文件。有关链接器的更多信息，请阅读**第 9 章 “MPLAB LINK30 链接器”**。

8.3 使用 PROJECT WIZARD 创建项目

如果您尚未使用过 Project Wizard，请阅读**第 3 章 “MPLAB 集成开发环境”**以了解 MPLAB IDE 中的项目和工作区。我们将跳过重复部分直接进入 MPLAB C30 C 编译器项目的创建流程中。

首先，安装 MPLAB C30 C 编译器。本教程后续部分假设 MPLAB C30 C 编译器已安装在默认位置 `C:\Program Files\Microchip\MPLAB C30`。如果将其安装在其他位置，请在本教程中相应调整路径。

注： 具体环境中工具所安装的位置可能与此处给出的位置有所不同。

在开始之前，为本教程的项目文件创建一个文件夹。文件夹 `C:\Tutorial` 会用于下列指令中。如果您已在先前的教程中创建了此文件夹，则只需将新文件添加到此文件夹中即可。将 `Flash LEDs with dsPIC30F6014.c` 文件复制到 `C:\Tutorial` 文件夹中。这些文件与本文档一同提供。如果该文件是从光盘复制来的，则文件属性为只读。若需要编辑文件，切记要更改其属性。

注： 如果您拥有 dsPICDEM 入门演示板，则可使用 `Flash LEDs with dsPIC30F6012.c` 文件代替，它包含的代码与上述文件的代码极为相似。如果是 dsPICDEM 28 引脚入门演示板，可使用 `Flash LED with dsPIC30F2010.c` 文件。如果是dsPICDEM 2 开发板，可使用`Flash LED with dsPIC30F4011.c` 文件。

现在，启动 MPLAB IDE 并使用 *File>Close Workspace* 菜单关闭任何已打开的工作区。 MPLAB IDE 中的 Project Wizard 是创建新项目的一种简单方法。使用 *Project>Project Wizard* 菜单启动 Project Wizard。当出现 Welcome 屏幕时，单击 **Next>** 继续。

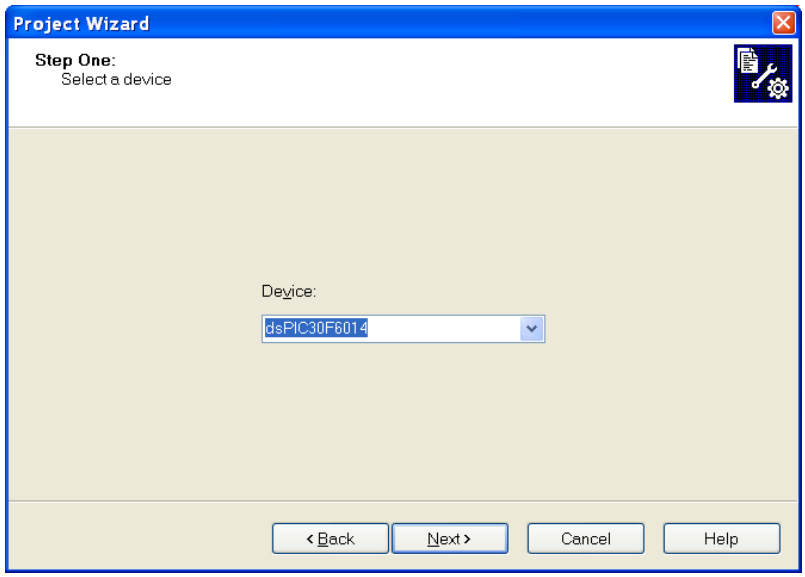
步骤 1——选择器件

Project Wizard 允许您选择要使用的 dsPIC 器件，如图 8-2 所示。从下拉菜单中选择“dsPIC30F6014”，然后单击 **Next>** 继续。

注： 如果使用其他演示板，请相应选择 dsPIC30F6012、 dsPIC30F2010 或 dsPIC30F4011 器件。

© 2007 Microchip Technology Inc.

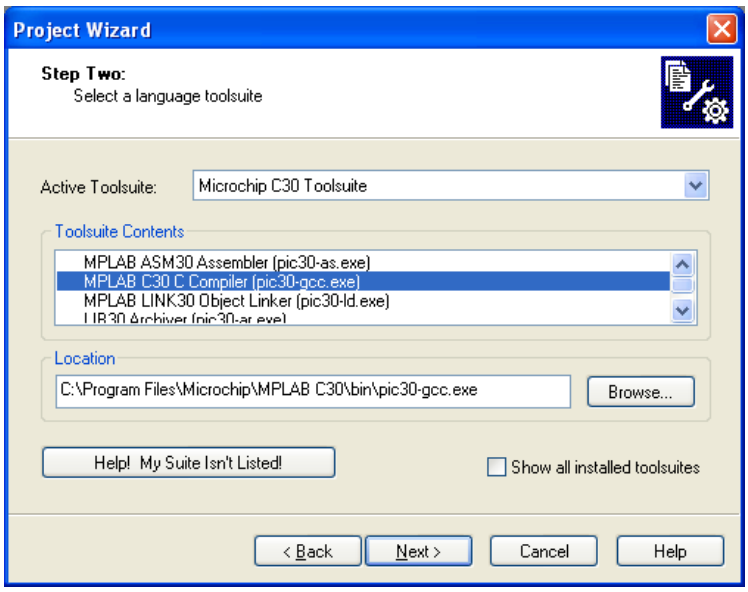
图 8-2：　　PROJECT WIZARD——步骤 1



步骤 2——选择语言工具包

下一屏幕（图 8-3）允许您选择工具包。从下拉菜单中选择 “Microchip C30 Toolsuite”（Microchip C30 工具包）。

图 8-3：　　PROJECT WIZARD——步骤 2



检查汇编器、编译器和链接器的可执行文件是否位于以下位置：

汇编器：　C:\Program Files\Microchip\MPLAB C30\bin\pic30-as.exe
编译器：　C:\Program Files\Microchip\MPLAB C30\bin\pic30-gcc.exe
链接器：　C:\Program Files\Microchip\MPLAB C30\bin\pic30-ld.exe
归档器：　C:\Program Files\Microchip\MPLAB C30\bin\pic30-ar.exe

上述工具的位置都是假设 MPLAB C30 C 编译器是按照默认设置安装的。

注： 工具包位置旁边有红 “X”，表示信息丢失。

如果工具包旁边有红 “X”，则选中该工具包并单击 **Browse** 按钮来设置位置。选择了工具包且位置正确时，单击 **Next** 继续。

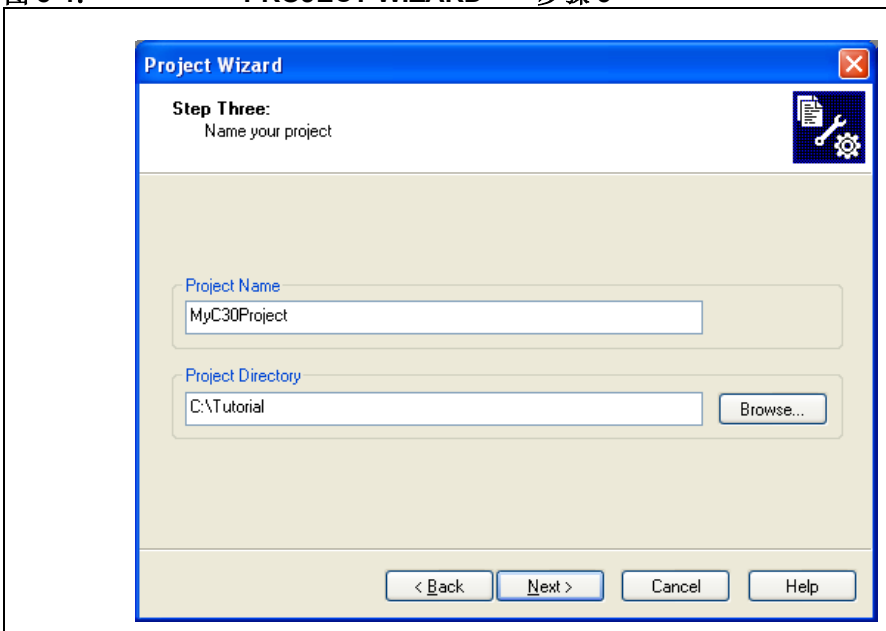
步骤 3——为项目命名

下一屏幕 （图 8-4）允许您为项目命名。

输入 “MyC30Project”作为项目名称，并找到 （或输入） `C:\Tutorial` 作为项目目录。

单击 **Next>** 继续。

图 8-4： PROJECT WIZARD——步骤 3



步骤 4——向项目添加文件

下一屏幕允许您向项目添加文件。

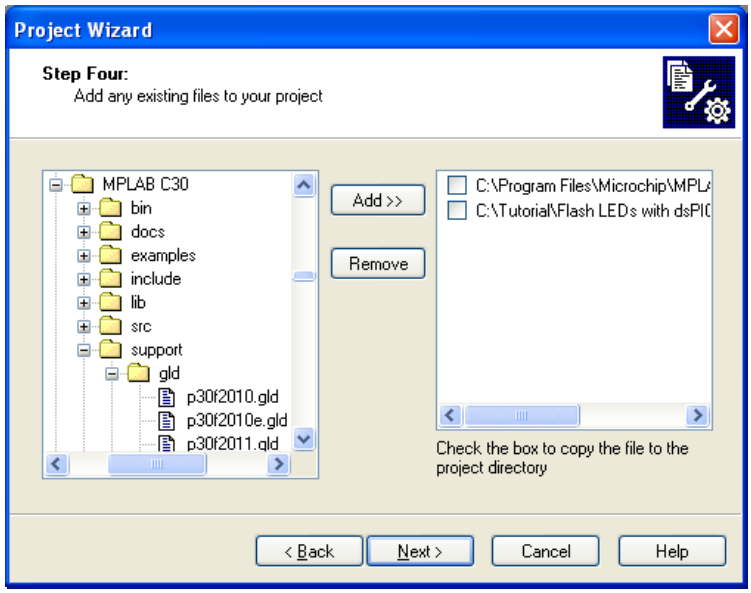
从 `Tutorial` 文件夹中选择 `Flash LEDs with dsPIC30F6014.c` 文件，并单击 **Add>>** 以将其添加到项目。

导航到 `C:\Program Files\Microchip\MPLAB C30\Support\gld` 文件夹。选择 `p30f6014.gld` 文件并单击 **Add>>** 以将其添加到项目。

现在，项目中应该有两个文件，如图 8-5 所示。单击 **Next>** 继续。

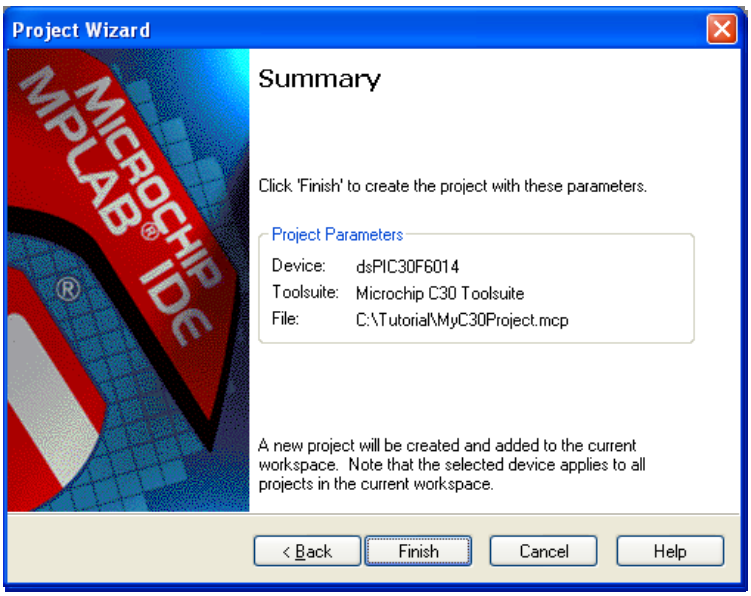
© 2007 Microchip Technology Inc.

图 8-5：　　PROJECT WIZARD——步骤 4



Project Wizard Summary（项目向导摘要）屏幕显示您刚才设置的 Project Parameters（项目参数），如图 8-6 所示。

图 8-6：　　PROJECT WIZARD SUMMARY 屏幕



单击 **Finish**。

注：　如果使用 dsPICDEM 入门演示板，请选择适用于 dsPIC30F6012 的文件。如果使用 dsPICDEM 28 引脚入门演示板，请选择适用于 dsPIC30F2010 的文件。如果使用 dsPICDEM 2 开发板，请选择适用于 dsPIC30F4011 的文件。

设置完 Project Wizard 之后， MPLAB IDE 项目窗口将显示 Source Files 目录中的 Flash LEDs with dsPIC30F6104.c文件和Linker Scripts目录中的p30f6014.gld 文件。**第 9 章 “MPLAB LINK30 链接器”**中将对 GLD 文件进行更加深入的介绍。

如果您忘记向项目添加文件了，也无需重启 Project Wizard。只需右击项目树中的目录，从下拉菜单中选择 “Add Files”并找到要添加的文件即可。通过右击文件名并选择 **Remove**，可删除文件。

现在 MPLAB IDE 已创建了一个项目文件 MyC30Project.mcp 和一个工作区文件 MyC30Project.mcw。双击项目窗口中的 Flash LEDs with dsPIC30F6014.c 文件将其打开。该文件将显示在编辑器窗口中。

8.4 设置编译选项

在编译项目之前，需要设置编译选项。 MPLAB IDE 使用这些设置来查找文件、生成调试信息、控制优化和创建诊断文件。

使用 *Project>Build Options>Project* 菜单来告知 MPLAB IDE 头文件的路径。当出现 Build Options 屏幕时，单击 “Include Path”（头文件路径）的 **Browse** 按钮找到 C:\Program Files\Microchip\MPLAB C30\support\h。单击 **OK** 添加此路径，如图 8-7 所示。

图 8-7： PROJECT BUILD OPTIONS 对话框

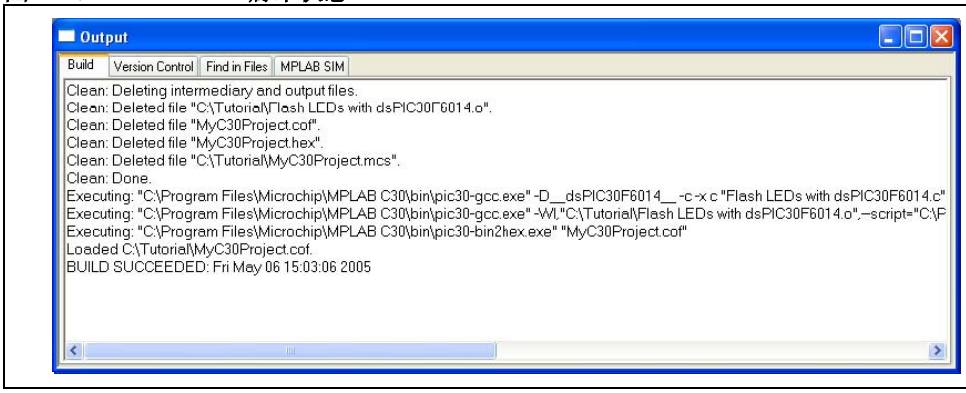
© 2007 Microchip Technology Inc.

8.5 编译项目

现在可以编译项目了。选择 *Project>Make* 菜单。编译结果将显示在 Output 窗口中，并指示编译已完成，如图 8-8 所示。

根据项目编译选项的设置，可在编译结果中看到 Memory Usage Report。

图 8-8： 编译状态



此时已成功编译了项目，您可以运行或调试代码了。您可以返回到**第 4 章 “MPLAB SIM 软件模拟器”**、**第 5 章 “MPLAB ICD 2 在线调试器”**或**第 6 章 “MPLAB ICE 4000 在线仿真器”**。

8.6 语言功能

8.6.1 ANSI C 标准

MPLAB C30 C 编译器是一个 ANSI C 编译器，它具有许多扩展功能可支持 dsPIC 器件的特有性能和功能。该编译器包含一个完整的 ANSI C 标准库，且使用一种可生成高效紧凑代码的优化。

8.6.2 标准头文件

标准头文件 （例如代码示例中用到的 `p30f6014.h`）经常通过 `#include` 语句包含在每个 “C” 文件中。该头文件包含了所有特殊功能寄存器 （SFR）及其位的声明，以便在代码中使用。在本示例中，链接器从链接描述文件 `p30f6014.gld` 中获得 SFR 的地址。

8.6.3 __attribute__ keyword

MPLAB C30 C 编译器使用 `__attribute__ keyword` （注意双下划线前缀和后缀）来指定编译器所特有的、无法由标准 “C” 语法实现的函数和变量操作。它可用于定义段、控制如何使用程序存储器、优化函数、指定中断函数等等。它类似于一些其他编译器 （例如， MPLAB C18 C 编译器）中所用的 `#pragma` 伪指令。

8.7 代码示例

对此编译器有了深入的了解，现在您可以参考下面的示例来了解其工作原理。此处对本教程中使用的 Flash LEDs with dsPIC30F6014.c 文件中的代码示例进行了解释。

> **注：** Flash LEDs with dsPIC30F6012.c、Flash LED with dsPIC30F2010.c和Flash LED with dsPIC30F4011.c 等其他教程文件与之非常类似，以下描述均适用。

文件以注释 （以 “/*” 或 “//” 开头）开始 （见例 8-1）。接下来是标准头文件，用于定义所有的特殊功能寄存器（SFR）。MPLAB C30 C 编译器支持的所有 dsPIC 控制器都各有一个头文件。

例 8-1： 注释和头文件引用

```
/*****************************************************************************
*                                    软件许可协议                                *
*                                                                               *
*   Microchip Technology Incorporated（以下称为 “本公司”）在此为 dsPIC 单片机提供的 *
*   软件旨在向本公司客户提供专门用于 Microchip dsPIC 产品的软件。本软件为本公司及 / 或其供 *
*   应商所有，并受到适用的版权法保护。版权所有。使用时违反前述约束的用户可能会依法受到刑事  *
*   制裁，并可能由于违背本许可的条款和条件而承担民事责任。                          *
*                                                                               *
*   本软件是按 “现状”提供的。任何形式的保证，无论是明示的、暗示的或法定的，包括但不限于    *
*   有关适销性和特定用途的暗示保证，均不适用于本软件。对于在任何情况下，因任何原因造成的    *
*   特殊的、附带的或间接的损害，本公司概不负责。                                *
*                                                                              *
*                                                                              *
*                                                                              *
*                                                                             *
*****************************************************************************/
//============================================================================
// 使用 Timer 1 计数，未按开关 SW1 时，使 LED1 闪烁；
//                  按下开关 SW1 时，使 LED2 闪烁
//============================================================================

#include "p30F6014.h"
```

使用宏定义配置位以声明配置位设置，如例 8-2 所示。这可确保将配置设置包含在 hex 输出文件中。

例 8-2： 配置位

```
//----------------------------------------------------------------------------
// 配置位

_FOSC(CSW_FSCM_OFF & XT_PLL4);
_FWDT(WDT_OFF);
_FBORPOR(PBOR_OFF & BORV_27 & PWRT_16 & MCLR_EN);
_FGS(CODE_PROT_OFF);
```


© 2007 Microchip Technology Inc.

定义处理器频率以便稍后在代码中设置定时器。用 #define 语句为符号 FCY 赋值，如例 8-3 所示。

例 8-3： 处理器频率定义

```
//------------------------------------------------------------------------------
// 与程序有关的常数

#define FCY 7372800          // 指令周期频率 （Osc x PLL / 4）
```

可执行代码从 Main() 程序开始，该程序根据 ANSI “C”标准返回一个整数值。值得注意的是，链接器将添加调用 Main 程序的启动代码。 Main 程序的第一部分 （例 8-4）通过初始化输出锁存器和设置 LED 引脚为输出，来设置 LED 的 I/O 引脚。

例 8-4： 开始 MAIN 程序

```
//------------------------------------------------------------------------------
// 主程序
// 设置 LED 和定时器，等待若干个定时器周期后，使两个 LED 中的一个闪烁

int main(void)
{
  LATD = 0xffff;             // 初始化 LED 引脚数据为关闭状态
  TRISD = 0xfff0;           // 设置 LED 引脚 （PORTD 的 bit 0- 3）为输出
  LATDbits.LATD0 = 0;       // 点亮 LED1 （低电平有效）
```

下列代码将 Timer1 的周期初始化为 1/5 秒，并启动该定时器，如例 8-5 所示。

例 8-5： 初始化定时器

```
  T1CON = 0;              // 关闭 Timer1 并将设置清零
  TMR1 = 0;              // Timer1 从 0 开始计数
  PR1 = FCY/256/5;       // 获得 1/5 秒对应的周期寄存器值
  T1CON = 0x8030;        // 启动 Timer1，其预分频比为 1:256
```

Main() 程序内部的 while(1) 循环确保代码执行不跳出 main 且不会停止。在无限循环中，代码一直运行，直到 Timer1 中断标志位置 1 （见例 8-6）。当定时器的值与周期寄存器中的值匹配时，中断标志位由硬件置 1，即使未允许中断。

例 8-6： WHILE 循环

```
   while(1)                    // 无限循环
   {
      while(!IFS0bits.T1IF){} // 等待一个定时器周期
```

在定时器超时后，代码将中断标志位清零以备稍后再次进行测试。测试 RA12 输入引脚以确定是否按下开关 SW1。根据该输入引脚的状态，一个 LED 会点亮而另一个 LED 熄灭。如例 8-7 所示，等待定时器和翻转 LED 的代码将无止尽地重复。

例 8-7：　　MAIN 程序

```
        IFS0bits.T1IF = 0;          // 定时器标志位清零，以用于下一周期
        if(PORTAbits.RA12)          // 检查是否按下 SW1
        {
            LATDbits.LATD0 ^= 1;    // 未按下 SW1 时，翻转 LED1
            LATDbits.LATD1 = 1;     // 熄灭 LED2
        }
        else
        {
            LATDbits.LATD0 = 1;     // 熄灭 LED1
            LATDbits.LATD1 ^= 1;    // 按下 SW1 时，翻转 LED2
        }
    }
}                                   //main() 结束
```

下面为错误陷阱程序，如例 8-8 所示。如果因灾难性错误（例如，振荡器故障或转移到不存在的存储器）而导致代码运行失败，硬件会将执行切换到相应错误陷阱程序。每个程序都有一个函数名，如 _OscillatorFail，以被链接器识别并用于创建中断向量表。错误陷阱程序点亮 LED 并无限制的循环。

例 8-8：　　错误陷阱程序

```
//============================================================================
// 错误陷阱
//----------------------------------------------------------------------------
// 振荡器故障错误陷阱程序

void _ISR _OscillatorFail(void)
{
    LATDbits.LATD3 = 0;             // 点亮 LED4
    while(1);                       // 无限地等待
}
//----------------------------------------------------------------------------
// 地址错误陷阱程序

void _ISR _AddressError(void)
{
    LATDbits.LATD3 = 0;             // 点亮 LED4
    while(1);                       // 无限地等待
}
//----------------------------------------------------------------------------
// 堆栈错误陷阱程序

void _ISR _StackError(void)
{
    LATDbits.LATD3 = 0;             // 点亮 LED4
    while(1);                       // 无限地等待
}
//----------------------------------------------------------------------------
// 数学（算术）错误陷阱程序

void _ISR _MathError(void)
{
    LATDbits.LATD3 = 0;             // 点亮 LED4
    while(1);                       // 无限地等待
```

本简单代码示例和教程旨在介绍使用 MPLAB C30 C 编译器的基本知识。注意，编译器生成目标文件，而链接器使用目标文件将代码和变量放入存储器并生成输出文件。欲了解关于链接器的更多信息，请继续**第 9 章 “MPLAB LINK30 链接器”**。

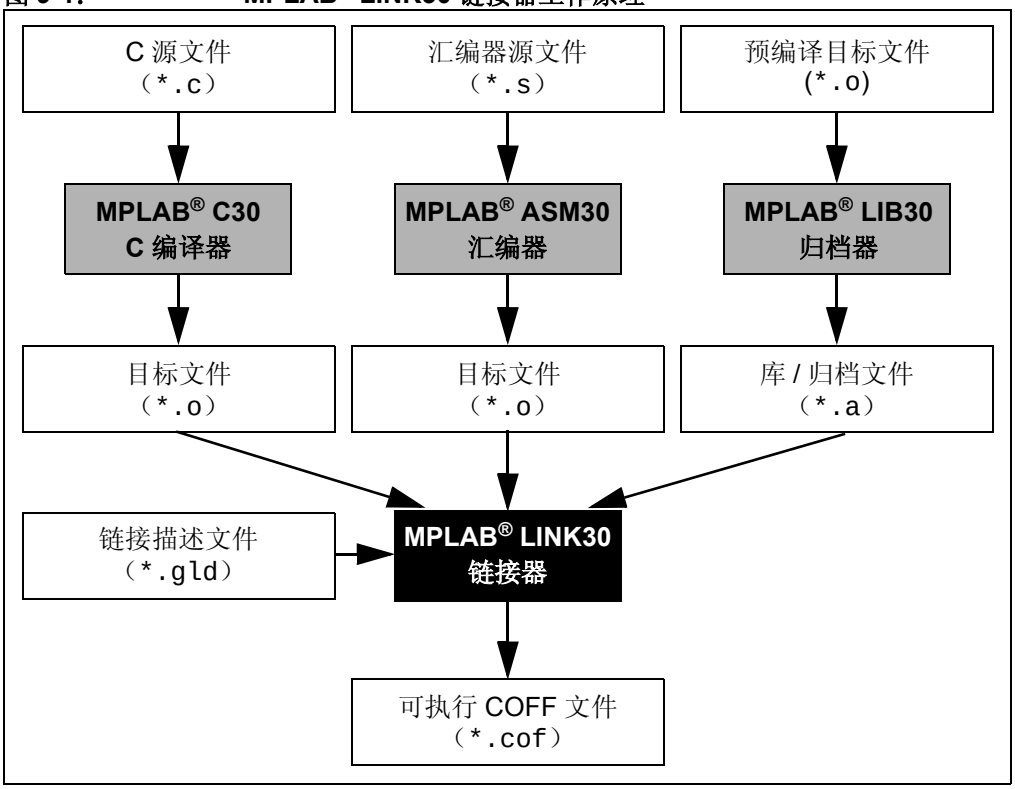
© 2007 Microchip Technology Inc.

第 9 章 MPLAB LINK30 链接器

9.1 MPLAB LINK30 链接器概述

MPLAB LINK30 链接器翻译目标文件和归档 （库）文件并将它们合并到运行 dsPIC 器件的可执行程序中。目标文件由 MPLAB ASM30 汇编器和 MPLAB C30 C 编译器创建。归档文件由 MPLAB® LIB30 归档器 / 库管理器创建。如图 9-1 所示，这些文件经过翻译并链接在一起以组成一个公共目标文件格式 （COFF）的文件，该文件实际控制 dsPIC 器件。

图 9-1： MPLAB® LINK30 链接器工作原理



该链接器主要是将项目中所有编译和汇编后的文件链接在一起，以形成一个可执行文件，该文件可编程到器件中，或用于模拟或者仿真。 Hex 文件和映射文件由 COFF 文件生成。

注： 本章的讨论基于 MPLAB LINK30 链接器 1.31 版。部分信息会随新版本的发布而过时。

9.2 链接描述文件

链接器使用链接描述文件以获得存储器段的位置，以及特定器件上所实现的存储器范围。链接描述文件支持中断向量表结构和软件堆栈分配。它还分配特殊功能寄存器（SFR）的地址。

链接描述文件具有以下信息类型，下面我们将以 p30f6014.gld 文件为例进行简要讨论：

- 输出文件格式和入口点
- 存储区信息
- 基本存储器地址
- 输入 / 输出段映射
- 对 Near 和 X 数据存储区进行范围校验
- 中断向量表
- SFR 地址

9.2.1 输出文件格式和入口点

链接描述文件最开始的几行定义了输出格式、处理器系列和入口点：

例 9-1：

```
/*
** p30f6014 的链接描述文件
*/

OUTPUT_FORMAT(“coff-pic30”)
OUTPUT_ARCH(“pic30”)
EXTERN(__resetPRI)
EXTERN(__resetALT)
ENTRY(__reset)
```

注意入口标号为 __reset。如果代码中有名为 __reset 的全局标号，代码将从这里开始执行。

© 2007 Microchip Technology Inc.

9.2.2 存储区信息

链接描述文件的下一部分定义器件的各种存储区。该部分的信息告知链接器目标器件上有多少存储区可用。在链接过程中添加段时，将对每个存储区进行范围校验。如果有区域溢出，就会报告一个链接错误。

例 9-2：

```
/*
** 存储区
*/
MEMORY
(
   data (a!xr)      : ORIGIN = 0x800,   LENGTH = 8096
   program (xr)     : ORIGIN = 0x100,   LENGTH = ((48K * 2) - 0x100)
   reset            : ORIGIN = 0,       LENGTH = (4)
   ivt              : ORIGIN = 0x04,    LENGTH = (62 * 2)
   aivt             : ORIGIN = 0x84,    LENGTH = (62 * 2)
   __FOSC           : ORIGIN = 0xF80000,LENGTH = (2)
   __FWDT           : ORIGIN = 0xF80002,LENGTH = (2)
   __FBORPOR        : ORIGIN = 0xF80004,LENGTH = (2)
   __CONFIG4        : ORIGIN = 0xF80006,LENGTH = (2)
   __CONFIG5        : ORIGIN = 0xF80008,LENGTH = (2)
   __FGS            : ORIGIN = 0xF8000A,LENGTH = (2)
   eedata           : ORIGIN = 0x7FF000,LENGTH = (4096)
)
```

9.2.3 基本存储器地址

这部分链接描述文件定义链接器为代码或数据分配的段的起始地址。每个基址定义为一个符号，该符号用于指定接下来的段映射中的加载地址。

例 9-3：

```
/*
** 基本存储器地址——程序存储区
*/
__RESET_BASE  = 0:      /* 复位指令                                    */
__IVT_BASE    = 0X04;   /* 中断向量表                                  */
__AIVT_BASE   = 0x84;   /* 备用中断向量表                              */
__CODE_BASE   = 0x100;  /* 句柄、用户代码和库代码                      */


/*
** 基本存储器地址——数据存储区
*/
__SFR_BASE    = 0;      /* 存储器映射的 SFR                             */
__DATA_BASE   = 0x800;  /* X 和通用数据存储区                           */
__YDATA_BASE  = 0x1800; /* 用于 DSP 指令的 Y 数据存储区                 */
```

9.2.4 输入 / 输出段映射

段映射是链接描述文件的核心。它定义了将输入段映射到输出段的方式。注意输入段为源代码中定义的应用程序的各个部分，而输出段是由链接器创建的。通常，几个输入段可合并到一个输出段中。

例如，假设某个应用程序由 5 个不同的函数组成，并且每个函数都是在不同的源文件中定义的。这些源文件总共将生成 5 个输入段。链接器会将这些输入段合并到一个输出段中。只有输出段才具有绝对地址。输入段始终是可重定位的。

如果任何输入或输出段为空，对于链接的应用程序不会有额外的存储开销。大多数应用程序仅使用出现在段映射中的许多段中的很少一部分。

以下是段映射启动的方式：

例 9-4：

```
/*
======================== 段映射 ============================
*/

SECTIONS
(
```

考虑例 9-5 中程序存储器的首段。程序存储器以地址零开始 （__RESET_BASE 在例 9-3 中定义为基本存储器地址），且在中断向量表开始前保留了一个双字指令的空间。将数据装入该段可形成一条双字 GOTO __reset 指令。您可在 《dsPIC30F 程序员参考手册》（DS70157B_CN）中查看 GOTO 指令的编码，以了解该指令是如何构造的。

例 9-5：

```
/*
======================== 程序存储区 =====================
*/

/*
** RESET 指令
*/
.reset __RESET_BASE:
   (
       SHORT(ABSOLUTE(__reset));
       SHORT(0x04);
       SHORT(ABSOLUTE(__reset) >> 16) & 0x7F;
       SHORT(0);
) >reset
```


© 2007 Microchip Technology Inc.

.text 段收集来自所有应用程序输入文件的可执行代码输入段，并将它们放入一个输出段中。定义输入段的顺序以保证 MPLAB C30 C 编译器正常工作。例如 .handle 段用于函数指针，并且第一个被加载。随后加载的是库段 .libc、.libm 和 .libdsp。数学函数库在中间，以便从标准 “C”语言库和 DSP 库中对其进行有效调用。其他库位于代码的剩余部分。

例 9-6：

```
/*
** 用户代码和库代码
*/
.text __CODE_BASE:
   {
       *(.handle);
       *(.libc) *(.libm) *(.libdsp); /* 以此顺序放在一起 */
       *(.lib*);
       *(.text);
   } >program
```

段映射的其余部分接下来定义所有不同类型的程序存储器、 RAM、 EEPROM 以及配置存储器段。

注： 也可在程序存储器和数据存储器中创建用户自定义的输出段。下面的示例说明了如何在链接描述文件中实现上述操作。

9.2.5 对 Near 和 X 数据存储区进行范围校验

将范围校验表达式包括进来是为了对 X 数据存储空间和 Near 数据存储空间进行范围校验。在存储区满时，将对所有其他段进行范围校验。如果任何段超过其分配的存储区，就会报告一个链接错误。

注意， X 数据存储空间限制随器件不同而不同，而 Near 数据存储空间固定为 8 KB，或者地址 0x2000。

例 9-7：

```
/*
** 计算 X 和 Near 数据空间的溢出
*/
__X_OVERFLOW      =       (((__exdata  !=  __bxdata) &&
                           (__exdata    >   __YDATA_BASE)) ?
                           (__exdata    -   __YDATA_BASE) : 0);
__NEAR_OVERFLOW   =       (((__endata  !=  __bndata) &&
                           (__endata    >   0x2000)) ?
                           (__endata    -   0x2000) : 0);
```

9.2.6 中断向量表

在靠近链接描述文件末尾的第二个段映射中定义了主中断向量表和备用中断向量表。下面以振荡器故障陷阱为例对中断向量表进行了简要的说明：

如果定义了符号 __OscillatorFail，则使用该符号的地址；否则使用符号 __DefaultInterrupt 的地址。这意味着如果没有提供中断程序，将会调用默认的中断处理程序。如果应用程序没有提供默认的中断处理程序 （即名为 __DefaultInterrupt 的函数），链接器将自动生成一个。最简单的默认中断处理程序为 RESET 指令。

例 9-8：

```
/*
** 中断向量表的段映射
*/
SECTIONS
{

/*
** 主中断向量表
*/
.ivt __IVT_BASE:
   {
     LONG(DEFINED(__ReservedTrap0)   ? ABSOLUTE(__ReservedTrap0):
          ABSOLUTE(__DefaultInterrupt));
     LONG(DEFINED(__OscillatorFail)  ? ABSOLUTE(__OscillatorFail):
          ABSOLUTE(__DefaultInterrupt));
     LONG(DEFINED(__AddressError)    ? ABSOLUTE(__AddressError) :
          ABSOLUTE(__DefaultInterrupt));
     LONG(DEFINED(__StackError)      ? ABSOLUTE(__StackError)   :
          ABSOLUTE(__DefaultInterrupt));
     LONG(DEFINED(__MathError)       ? ABSOLUTE(__MathError)    :
          ABSOLUTE(__DefaultInterrupt));
     .
     .
     .
   } >ivt
```


© 2007 Microchip Technology Inc.

9.2.7 SFR 地址

特殊功能寄存器 （SFR）的绝对地址被定义为一系列的符号。包含两种版本的 SFR 地址：带或不带有前导下划线。这是为了使 “C”和汇编语言编程人员能在引用 SFR 时使用相同的名称。按照约定， C 编译器会在每个标号前添加一个前导下划线。

例 9-9：

```
/*============================================================================
**         寄存器定义
** （数据空间中的内核和外设寄存器）
**============================================================================
**
**============================================================================
**
**         dsPIC 内核寄存器定义
**
**
**============================================================================
 WREG0 = 0x0000;
_WREG0 = 0x0000;
 WREG1 = 0x0002;
_WREG1 = 0x0002;
    .
    .
    .
 CAN1  = 0x0300;
_CAN1  = 0x0300;
 CAN2  = 0x03C0;
_CAN2  = 0x03C0;

/*============================================================================
** 数据空间中所需的 SFR 定义结束
*============================================================================*/
```

数据可能比想象中所需要的多，但不要被链接器及其链接描述文件所钳制，这一点非常重要。链接器只是按照程序将代码和变量放入可用的存储器中。

我们希望 《dsPIC30F 数字信号控制器入门用户指南》能帮助您适应使用 dsPIC30F 器件和 Microchip 开发工具。祝您在 dsPIC 设计中一切顺利。

注：

© 2007 Microchip Technology Inc.

附录 A dsPICDEM 1.1 通用开发板代码

软件许可协议

Microchip Technology Incorporated （以下称为 “本公司”）提供的软件旨在向本公司客户提供专门用于本公司生产的产品的软件。

本软件为本公司及 / 或其供应商所有，并受到适用的版权法保护。版权所有。使用时违反前述约束的用户可能会依法受到刑事制裁，并可能由于违背本许可的条款和条件而承担民事责任。

本软件是按 “现状”提供的。任何形式的保证，无论是明示的、暗示的或法定的，包括但不限于有关适销性和特定用途的暗示保证，均不适用于本软件。对于在任何情况下，因任何原因造成的特殊的、附带的或间接的损害，本公司概不负责。

A.1 FLASH LED WITH dsPIC30F6014.s

本附录包含 dsPICDEM 1.1 通用开发板的代码示例。本教程示例是为 MPLAB ASM30 汇编器和 MPLAB C30 C 编译器编写的。

```
;=============================================================================  ;
;                                                                              ;
;                           软件许可协议                                       ;
;                                                                              ;
;    Microchip Technology Incorporated（以下称为 “本公司”）在此为 dsPIC 单片机提供     ;
;    的软件旨在向本公司客户提供专门用于 Microchip dsPIC 产品的软件。本软件为本公司及 /     ;
;    或其供应商所有，并受到适用的版权法保护。版权所有。使用时违反前述约束的用户可能会依法     ;
;    受到刑事制裁，并可能由于违背本许可的条款和条件而承担民事责任。                  ;
;                                                                              ;
;    本软件是按 “现状”提供的。任何形式的保证，无论是明示的、暗示的或法定的，包括但不限于     ;
;    有关适销性和特定用途的暗示保证，均不适用于本软件。对于在任何情况下，因任何原因造成的     ;
;    特殊的、附带的或间接的损害，本公司概不负责。                                ;
;                                                                              ;
;=============================================================================  ;
;
; 使用 Timer 1 计数，未下开关 SW1 时，使 LED1 闪烁，
;                   按下开关 SW1 时，使 LED2 闪烁
;
;=============================================================================

      .equ __30F6014, 1
      .include "p30f6014.inc"

;-----------------------------------------------------------------------------
```

```
; 全局声明

        .global __reset                         ; 第一行代码的标号
        .global __OscillatorFail                ; 声明振荡器故障陷阱程序标号
        .global __AddressError                  ; 声明地址错误陷阱程序标号
        .global __StackError                    ; 声明堆栈错误陷阱程序标号
        .global __MathError                     ; 声明数学错误陷阱程序标号

;-------------------------------------------------------------------------------
; 配置位

        config __FOSC, CSW_FSCM_OFF & XT_PLL4
        config __FWDT, WDT_OFF
        config __FBORPOR, PBOR_OFF & BORV_27 & PWRT_16 & MCLR_EN
        config __FGS, CODE_PROT_OFF

;-------------------------------------------------------------------------------
; 与程序相关的常数 （代码中使用的立即数）

        .equ          FCY, #7372800             ; 指令周期频率 （Osc x PLL / 4）

;===============================================================================
; 代码开始

        .text                                   ; 代码段开始

;-------------------------------------------------------------------------------
; 初始化堆栈指针极限寄存器

__reset:      mov    #__SP_init, W15            ; 初始化堆栈指针寄存器
              mov    #__SPLIM_init, W0          ; 获取堆栈空间的结束地址
              mov    W0, SPLIM                  ; 装载堆栈指针极限寄存器
              nop                               ; 在 SPLIM 初始化后添加一条 NOP 指令

;-------------------------------------------------------------------------------
; 初始化 PORTD 上的 LED 输出引脚 bit 0-3

              mov    #0xffff, W0                ; 初始化 LED 引脚数据为关闭状态
              mov    W0, LATD
              mov    #0xfff0, W0                ; 将 LED 引脚设置为输出
              mov    W0, TRISD
              bclr   LATD, #0                   ; 点亮 LED1

;-------------------------------------------------------------------------------
; 将 Timer1 的周期初始化为 1/5 秒

              clr    T1CON                      ; 控制寄存器清零以关闭 Timer1
              clr    TMR1                       ; Timer1 从 0 开始计数
              mov    #FCY/256/5, W0             ; 获得 1/5 秒对应的周期寄存器值
              mov    W0, PR1                    ; 装载 Timer1 周期寄存器
              mov    #0x8030, W0                ; 设置 Timer1 （预分频比为 1:256）
              mov    W0, T1CON                  ; 装载 Timer1 设置到控制寄存器

;-------------------------------------------------------------------------------
; 循环并等待 Timer1 匹配，发生匹配时翻转 LED1 或 LED2

MainLoop:     btss   IFS0, #T1IF                ; 检查 Timer1 中断标志位是否置 1
              bra    MainLoop                   ; 不断循环，直到置 1 为止
              bclr   IFS0, #T1IF                ; Timer1 中断标志位清零
              btss   PORTA, #12                 ; 测试开关 SW1 （按下时为低电平）
              bra    SwitchPressed
```


© 2007 Microchip Technology Inc.

```
            btg    LATD, #0                ; 未按下 SW1 时，翻转 LED1
            bset   LATD, #1                ; 熄灭 LED2
            bra    MainLoop                ; 循环返回

SwitchPressed:bset   LATD, #0              ; 熄灭 LED1
            btg    LATD, #1                ; 按下 SW1 时，翻转 LED2
            bra    MainLoop                ; 循环返回 MainLoop

;=============================================================================
; 错误陷阱

;-----------------------------------------------------------------------------
; 振荡器故障错误陷阱程序

       .text                                ; 代码段开始
__OscillatorFail:
            bclr   LATD, #3                ; 点亮 LED4
            bra    __OscillatorFail        ; 发生振荡器故障时进入无限循环

;-----------------------------------------------------------------------------
; 地址错误陷阱程序

__AddressError:
            bclr   LATD, #3                ; 点亮 LED4
            bra    __AddressError          ; 发生地址错误时进入无限循环

;-----------------------------------------------------------------------------
; 堆栈错误陷阱程序

__StackError:
            bclr   LATD, #3                ; 点亮 LED4
            bra    __StackError            ; 发生堆栈错误时进入无限循环

;-----------------------------------------------------------------------------
; 数学 （算术）错误陷阱程序

__MathError:
            bclr   LATD, #3                ; 点亮 LED4
            bra    __MathError             ; 发生数学错误时进入无限循环

;=============================================================================

       .end                                 ; 此文件中的代码结束
```

A.2 FLASH LED WITH dsPIC30F6014.c

```
/*********************************************************************************
*                                软件许可协议                                    *
*                                                                                *
*   Microchip Technology Incorporated（以下称为 “本公司”）在此为 dsPIC 单片机提  *
*   供的软件旨在向本公司客户提供专门用于 Microchip dsPIC 产品的软件。本软件为本公司及 /  *
*   或其供应 商所有，并受到适用的版权法保护。版权所有。使用时违反前述约束的用户可能会依  *
*   法受到刑事制裁，并可能由于违背本许可的条款和条件而承担民事责任。           *
*                                                                                *
*   本软件是按 “现状”提供的。任何形式的保证，无论是明示的、暗示的或法定的，包括但不限于  *
*   有关适销性和特定用途的暗示保证，均不适用于本软件。对于在任何情况下，因任何原因造成的  *
*   特殊的、附带的或间接的损害，本公司概不负责。                               *
*                                                                                *
*                                                                                *
*                                                                                *
*                                                                                *
*********************************************************************************
*
* 使用 Timer 1 计数，未按开关 SW1 时，使 LED1 闪烁，
*                  按下开关 SW1 时，使 LED2 闪烁
*
*********************************************************************************/

#include "p30F6014.h"

//---------------------------------------------------------------------------
// 配置位

_FOSC(CSW_FSCM_OFF & XT_PLL4);
_FWDT(WDT_OFF);
_FBORPOR(PBOR_OFF & BORV_27 & PWRT_16 & MCLR_EN);
_FGS(CODE_PROT_OFF);

//---------------------------------------------------------------------------
// 与程序有关的常数

#define   FCY    7372800             // 指令周期频率 （Osc x PLL / 4）

//===========================================================================
// 主程序
// 设置 LED 和定时器，等待若干个定时器周期后，使两个 LED 中的一个闪烁

int main(void)
{
    LATD = 0xffff;                    // 初始化 LED 引脚数据为关闭状态
    TRISD = 0xfff0;                   // 设置 LED 引脚 （PORTD 的 bit 0-3）为输出
    LATDbits.LATD0 = 0;               // 点亮 LED1（低电平有效）

    T1CON = 0;                        // 关闭 Timer1 并将设置清零
    TMR1 = 0;                         // Timer1 从 0 开始计数
    PR1 = FCY/256/5;                  // 获得 1/5 秒对应的周期寄存器值
    T1CON = 0x8030;                   // 启动 Timer1，其预分频比为 1:256

    while(1)                          // 无限循环
```


© 2007 Microchip Technology Inc.

```
    {
        while(!IFS0bits.T1IF){}                 // 等待一个定时器周期

        IFS0bits.T1IF = 0;                      // 定时器标志位清零，以用于下一周期
        if(PORTAbits.RA12)                      // 检查是否按下 SW1
        {
            LATDbits.LATD0 ^= 1;                // 未按下 SW1 时，翻转 LED1
            LATDbits.LATD1 = 1;                 // 熄灭 LED2
        }
        else
        {
            LATDbits.LATD0 = 1;                 // 熄灭 LED1
            LATDbits.LATD1 ^= 1;                // 按下 SW1 时，翻转 LED2
         }
    }
}                                               // main() 结束

//============================================================================
// 错误陷阱

//----------------------------------------------------------------------------
// 振荡器故障错误陷阱程序

void _ISR _OscillatorFail(void)
{
    LATDbits.LATD3 = 0;                         // 点亮 LED4
    while(1);                                   // 无限地等待
}

//----------------------------------------------------------------------------
// 地址错误陷阱程序

void _ISR _AddressError(void)
{
    LATDbits.LATD3 = 0;                         // 点亮 LED4
    while(1);                                   // 无限地等待
}

//----------------------------------------------------------------------------
// 堆栈错误陷阱程序

void _ISR _StackError(void)
{
    LATDbits.LATD3 = 0;                         // 点亮 LED4
    while(1);                                   // 无限地等待
}

//----------------------------------------------------------------------------
// 数学 （算术）错误陷阱程序

void _ISR _MathError(void)
{
    LATDbits.LATD3 = 0;                         // 点亮 LED4
    while(1);                                   // 无限地等待
}
```

注：

© 2007 Microchip Technology Inc.

附录 B dsPICDEM 入门演示板代码

B.1 FLASH LED WITH dsPIC30F6012.s

此附件包含 dsPICDEM 入门演示板的代码示例。该示例是为 MPLAB ASM30 汇编器和 C30 C 编译器编写的。

```
;=============================================================================  ;
;                                                                               ;
;                           软件许可协议                                        ;
;                                                                               ;
;   Microchip Technology Incorporated（以下称为 “本公司”）在此为 dsPIC 单片机提供  ;
;   的软件旨在向本公司客户提供专门用于 Microchip dsPIC 产品的软件。本软件为本公司及 /  ;
;   或其供应商所有，并受到适用的版权法保护。版权所有。使用时违反前述约束的用户可能会依法  ;
;   受到刑事制裁，并可能由于违背本许可的条款和条件而承担民事责任。                ;
;                                                                               ;
;   本软件是按 “现状”提供的。任何形式的保证，无论是明示的、暗示的或法定的，包括但不限于  ;
;   有关适销性和特定用途的暗示保证，均不适用于本软件。对于在任何情况下，因任何原因造成的  ;
;   特殊的、附带的或间接的损害，本公司概不负责。                                ;
;                                                                               ;
;=============================================================================  ;
;
; 使用 Timer 1 计数，未按下开关 S1 时，使 LED RD4 闪烁；
;                    按下开关 S1 时，使 LED RD5 闪烁
;
;=============================================================================

        .equ __30F6012, 1
        .include "p30f6012.inc"

;-----------------------------------------------------------------------------
; 全局声明

        .global __reset                 ; 第一行代码的标号
        .global __OscillatorFail        ; 声明振荡器故障陷阱程序标号
        .global __AddressError          ; 声明地址错误陷阱程序标号
        .global __StackError            ; 声明堆栈错误陷阱程序标号
        .global __MathError             ; 声明数学错误陷阱程序标号

;-----------------------------------------------------------------------------
; 配置位

        config __FOSC, CSW_FSCM_OFF & XT_PLL4
        config __FWDT, WDT_OFF
        config __FBORPOR, PBOR_OFF & BORV_27 & PWRT_16 & MCLR_EN
        config __FGS, CODE_PROT_OFF
```

```
;-------------------------------------------------------------------------------
; 与程序有关的常数 （代码中使用的立即数）

       .equ   FCY, #4000000                    ; 指令周期频率 （Osc x PLL / 4）

;===============================================================================
; 代码开始

       .text                                   ; 代码段开始

;-------------------------------------------------------------------------------
; 初始化堆栈指针极限寄存器

__reset:      mov    #__SP_init, W15           ; 初始化堆栈指针寄存器
              mov    #__SPLIM_init, W0         ; 获取堆栈空间的结束地址
              mov    W0, SPLIM                 ; 装载堆栈指针极限寄存器
              nop                              ; 在 SPLIM 初始化后添加一条 NOP 指令

;-------------------------------------------------------------------------------
; 初始化 PORTD 上的 LED 输出引脚 bit 4-7

              mov    #0xff0f, W0               ; 初始化 LED 引脚数据为关闭状态
              mov    W0, LATD
              mov    #0xff0f, W0               ; 设置 LED 引脚为输出
              mov    W0, TRISD
              bset   LATD, #4                  ; 点亮 LED RD4

;-------------------------------------------------------------------------------
; 将 Timer1 的周期初始化为 1/5 秒

              clr    T1CON                     ; 控制寄存器清零以关闭 Timer1
              clr    TMR1                      ; Timer1 从 0 开始计数
              mov    #FCY/256/5, W0            ; 获得 1/5 秒对应的周期寄存器值
              mov    W0, PR1                   ; 装载 Timer1 周期寄存器
              mov    #0x8030, W0               ; 设置 Timer1 （预分频比为 1:256）
              mov    W0, T1CON                 ; 载入 Timer1 的设置到控制寄存器

;-------------------------------------------------------------------------------
; 循环并等待 Timer1 匹配，发生匹配时翻转 LED1

MainLoop:     btss   IFS0, #T1IF               ; 检查 Timer1 中断标志位是否置 1
              bra    MainLoop                  ; 不断循环，直到置 1 为止
              bclr   IFS0, #T1IF               ; Timer1 中断标志位清零
              btss   PORTC, #13                ; 测试开关 S1 （按下时为低电平）
              bra    SwitchPressed

              btg    LATD, #4                  ; 翻转 LED RD4
              bclr   LATD, #5                  ; 熄灭 LED RD5
              bra    MainLoop                  ; 循环返回

SwitchPressed:bclr   LATD, #4                  ; 熄灭 LED RD4
              btg    LATD, #5                  ; 翻转 LED RD5
              bra    MainLoop                  ; 循环返回 MainLoop
```


© 2007 Microchip Technology Inc.

```
;=============================================================================
; 错误陷阱

;-----------------------------------------------------------------------------
; 振荡器故障错误陷阱程序

       .text                                ; 代码段开始
__OscillatorFail:
              bclr   LATD, #7               ; 点亮 LED RD7
              bra    __OscillatorFail       ; 发生振荡器故障时进入无限循环

;-----------------------------------------------------------------------------
; 地址错误陷阱程序

__AddressError:
              bclr   LATD, #7               ; 点亮 LED RD7
              bra    __AddressError         ; 发生地址错误陷阱时进入无限循环

;-----------------------------------------------------------------------------
; 堆栈错误陷阱程序

__StackError:
              bclr   LATD, #7               ; 点亮 LED RD7
              bra    __StackError           ; 发生堆栈错误时进入无限循环

;-----------------------------------------------------------------------------
; 数学 （算术）错误陷阱程序

__MathError:
              bclr   LATD, #7               ; 点亮 LED RD7
              bra    __MathError            ; 发生数学错误时进入无限循环

;=============================================================================

       .end                                 ; 此文件中的代码结束
```

B.2 FLASH LED WITH dsPIC30F6012.c

```
/*********************************************************************************
*                                软件许可协议                                    *
*                                                                                *
*   Microchip Technology Incorporated（以下称为 “本公司”）在此为 dsPIC 单片机提  *
*   供的软件旨在向本公司客户提供专门用于 Microchip dsPIC 产品的软件。本软件为本公司及 /  *
*   或其供应 商所有，并受到适用的版权法保护。版权所有。使用时违反前述约束的用户可能会依  *
*   法受到刑事制裁，并可能由于违背本许可的条款和条件而承担民事责任。           *
*                                                                                *
*   本软件是按 “现状”提供的。任何形式的保证，无论是明示的、暗示的或法定的，包括但不限于  *
*   有关适销性和特定用途的暗示保证，均不适用于本软件。对于在任何情况下，因任何原因造成的  *
*   特殊的、附带的或间接的损害，本公司概不负责。                               *
*                                                                                *
*                                                                                *
*                                                                                *
*                                                                                *
*********************************************************************************
*
*  使用 Timer 1 计数，未按下开关 S1 时，使 LED RD4 闪烁
*                      按下开关 S1 时，使 LED RD5 闪烁
*
*********************************************************************************  /

#include "p30F6012.h"

//---------------------------------------------------------------------------
// 配置位

_FOSC(CSW_FSCM_OFF & XT_PLL4);
_FWDT(WDT_OFF);
_FBORPOR(PBOR_OFF & BORV_27 & PWRT_16 & MCLR_EN);
_FGS(CODE_PROT_OFF);

//---------------------------------------------------------------------------
// 与程序有关的常数

#define   FCY    4000000             // 指令周期频率 （Osc x PLL / 4）

//===========================================================================
// 主程序
// 设置 LED 和定时器，等待若干个定时器周期后，使两个 LED 中的其中一个闪烁

int main(void)
{
    LATD = 0xff0f;                   // 初始化 LED 引脚数据为关闭状态
    TRISD = 0xff0f;                  // 设置 LED 引脚为输出
    LATDbits.LATD4 = 1;              // 点亮 LED RD4

    T1CON = 0;                       // 关闭 Timer1 并将设置清零
    TMR1 = 0;                        // Timer1 从 0 开始计数
    PR1 = FCY/256/5;                 // 获得 1/5 秒对应的周期寄存器值
    T1CON = 0x8030;                  // 启动 Timer1，其预分频比为 1:256

    while(1)                         // 无限循环
```


© 2007 Microchip Technology Inc.

```
    {
        while(!IFS0bits.T1IF){}                 // 等待一个定时器周期

        IFS0bits.T1IF = 0;                      // 定时器标志位清零，以用于下一周期
        if(PORTCbits.RC13)                      // 检查是否按下 S1 （按下时为低电平）
        {
            LATDbits.LATD4 ^= 1;                // 未按下 S1 时，翻转 LED RD4
            LATDbits.LATD5 = 0;                 // 熄灭 LED RD5
        }
        else
        {
            LATDbits.LATD4 = 0;                 // 熄灭 LED RD4
            LATDbits.LATD5 ^= 1;                // 按下 S1 时，翻转 LED RD5
        }
    }
}                                               // main() 结束

//============================================================================
// 错误陷阱

//----------------------------------------------------------------------------
// 振荡器故障错误陷阱程序

void _ISR _OscillatorFail(void)
{
    LATDbits.LATD7 = 0;                         // 点亮 LED RD7
    while(1);                                   // 无限地等待
}

//----------------------------------------------------------------------------
// 地址错误陷阱程序

void _ISR _AddressError(void)
{
    LATDbits.LATD7 = 0;                         // 点亮 LED RD7
    while(1);                                   // 无限地等待
}

//----------------------------------------------------------------------------
// 堆栈错误陷阱程序

void _ISR _StackError(void)
{
    LATDbits.LATD7 = 0;                         // 点亮 LED RD7
    while(1);                                   // 无限地等待
}

//----------------------------------------------------------------------------
// 数学 （算术）错误陷阱程序

void _ISR _MathError(void)
{
    LATDbits.LATD7 = 0;                         // 点亮 LED RD7
    while(1);                                   // 无限地等待
}
```

注：

© 2007 Microchip Technology Inc.

附录 C dsPICDEM 28 引脚入门演示板代码

C.1 FLASH LED WITH dsPIC30F2010.s

此附件包含 dsPICDEM 28 引脚入门演示板的代码示例。该示例是为 MPLAB ASM30 汇编器和 C30 C 编译器编写的。

```
;=============================================================================  ;
;                                                                               ;
;                              软件许可协议                                     ;
;                                                                               ;
;   Microchip Technology Incorporated（以下称为 “本公司”）在此为 dsPIC 单片机提供   ;
;   的软件旨在向本公司客户提供专门用于 Microchip dsPIC 产品的软件。本软件为本公司及 /  ;
;   或其供应商所有，并受到适用的版权法保护。版权所有。使用时违反前述约束的用户可能会依法  ;
;   受到刑事制裁，并可能由于违背本许可的条款和条件而承担民事责任。                ;
;                                                                               ;
;   本软件是按 “现状”提供的。任何形式的保证，无论是明示的、暗示的或法定的，包括但不限于  ;
;   有关适销性和特定用途的暗示保证，均不适用于本软件。对于在任何情况下，因任何原因造成的  ;
;   特殊的、附带的或间接的损害，本公司概不负责。                                ;
;                                                                               ;
;=============================================================================  ;
;
; 使用 Timer 1 使 LED 闪烁
;
;=============================================================================

        .equ __30F2010, 1
        .include "p30f2010.inc"

;-----------------------------------------------------------------------------
; 全局声明

        .global __reset                         ; 第一行代码的标号
        .global __OscillatorFail                ; 声明振荡器故障陷阱程序标号
        .global __AddressError                  ; 声明地址错误陷阱程序标号
        .global __StackError                    ; 声明堆栈错误陷阱程序标号
        .global __MathError                     ; 声明数学错误陷阱程序标号

;-----------------------------------------------------------------------------
; 配置位

        config __FOSC, CSW_FSCM_OFF & XT_PLL4
        config __FWDT, WDT_OFF
        config __FBORPOR, PBOR_OFF & BORV_27 & PWRT_16 & MCLR_EN
        config __FGS, CODE_PROT_OFF
```

```
;-------------------------------------------------------------------------
; 与程序有关的常数 （代码中使用的立即数）

       .equ   FCY, #7372800              ; 指令周期频率 （Osc x PLL / 4）

;=========================================================================
; 代码开始

       .text                             ; 代码段开始

;-------------------------------------------------------------------------
; 初始化堆栈指针极限寄存器

__reset:      mov    #__SP_init, W15      ; 初始化堆栈指针寄存器
              mov    #__SPLIM_init, W0    ; 获取堆栈空间的结束地址
              mov    W0, SPLIM            ; 装载堆栈指针极限寄存器
              nop                         ; 在 SPLIM 初始化后添加一条 NOP 指令

;-------------------------------------------------------------------------
; 初始化 PORTD 上的 LED 输出引脚 bit 0

              mov    #0xfffe, W0          ; 初始化 LED 引脚数据为关闭状态
              mov    W0, LATD
              mov    #0xfffe, W0          ; 设置 LED 引脚为输出
              mov    W0, TRISD
              bset   LATD, #0             ; 点亮 LED

;-------------------------------------------------------------------------
; 初始化 Timer1 的周期为 1/5 秒

              clr    T1CON                ; 清零控制寄存器以关闭 Timer1
              clr    TMR1                 ; Timer1 从 0 开始计数
              mov    #FCY/256/5,W0        ; 获得 1/5 秒对应的周期寄存器值
              mov    W0, PR1              ; 装载 Timer1 周期寄存器
              mov    #0x8030,W0           ; 设置 Timer1（预分频比为 1:256）
              mov    W0, T1CON            ; 载入 Timer1 的设置到控制寄存器

;-------------------------------------------------------------------------
; 循环并等待 Timer1 匹配，发生匹配时翻转 LED1

MainLoop:     btss   IFS0, #T1IF          ; 检查 Timer1 中断标志位是否置 1
              bra    MainLoop             ; 不断循环，直到置 1 为止
              bclr   IFS0, #T1IF          ; Timer1 中断标志位清零
              btg    LATD, #0             ; 触发 LED
              bra    MainLoop             ; 循环返回

;=========================================================================
; 错误陷阱

;-------------------------------------------------------------------------
; 振荡器故障错误陷阱程序

       .text                              ; 代码段开始
__OscillatorFail:
              bclr   LATD, #0             ; 点亮 LED
              bra    __OscillatorFail     ; 发生振荡器故障时进入无限循环

;-------------------------------------------------------------------------
; 地址错误陷阱程序
```


© 2007 Microchip Technology Inc.

```
__AddressError:
              bclr   LATD, #0             ; 点亮 LED
              bra    __AddressError       ; 发生地址错误陷阱时进入无限循环

;-------------------------------------------------------------------------------
; 堆栈错误陷阱程序

__StackError:
              bclr   LATD, #0             ; 点亮 LED
              bra    __StackError         ; 发生堆栈错误时进入无限循环

;-------------------------------------------------------------------------------
; 数学 （算术）错误陷阱程序

__MathError:
              bclr   LATD, #0             ; 点亮 LED
              bra    __MathError          ; 发生数学错误时进入无限循环

;===============================================================================

       .end                               ; 本文件中的代码结束
```

C.2 FLASH LED WITH dsPIC30F2010.c

```
/*********************************************************************************
*                                软件许可协议                                    *
*                                                                                *
*   Microchip Technology Incorporated（以下称为 “本公司”）在此为 dsPIC 单片机提  *
*   供的软件旨在向本公司客户提供专门用于 Microchip dsPIC 产品的软件。本软件为本公司及 /  *
*   或其供应 商所有，并受到适用的版权法保护。版权所有。使用时违反前述约束的用户可能会依  *
*   法受到刑事制裁，并可能由于违背本许可的条款和条件而承担民事责任。           *
*                                                                                *
*   本软件是按 “现状”提供的。任何形式的保证，无论是明示的、暗示的或法定的，包括但不限于  *
*   有关适销性和特定用途的暗示保证，均不适用于本软件。对于在任何情况下，因任何原因造成的  *
*   特殊的、附带的或间接的损害，本公司概不负责。                               *
*                                                                                *
*                                                                                *
*                                                                                *
*                                                                                *
*********************************************************************************
*
*  使用 Timer 1 使 LED 闪烁
*
*********************************************************************************/

#include "p30F2010.h"

//--------------------------------------------------------------------------------
// 配置位

_FOSC(CSW_FSCM_OFF & XT_PLL4);
_FWDT(WDT_OFF);
_FBORPOR(PBOR_OFF & BORV_27 & PWRT_16 & MCLR_EN);
_FGS(CODE_PROT_OFF);

//--------------------------------------------------------------------------------
// 与程序有关的常数

#define   FCY    7372800             // 指令周期频率 （Osc x PLL / 4）

//================================================================================
// 主程序
// 设置 LED 和定时器，等待若干个定时器周期后，使两个 LED 中的一个闪烁

int main(void)
{
   LATD = 0xfffe;                    // 初始化 LED 引脚数据为关闭状态
   TRISD = 0xfffe;                   // 设置 LED 引脚为输出
   LATDbits.LATD0 = 1;               // 点亮 LED

   T1CON = 0;                        // 关闭 Timer1 并将设置清零
   TMR1 = 0;                         // Timer1 从 0 开始计数
   PR1 = FCY/256/5;                  // 获得 1/5 秒对应的周期寄存器值
   T1CON = 0x8030;                   // 启动 Timer1，其预分频比为 1:256
```


© 2007 Microchip Technology Inc.

```
    while(1)                            // 无限循环
    {
        while(!IFS0bits.T1IF){}         // 等待一个定时器周期
        IFS0bits.T1IF = 0;              // 定时器标志位清零，以用于下一周期
        LATDbits.LATD0 ^= 1;            // 触发 LED
    }
}                                       // main() 结束

//============================================================================
// 错误陷阱

//----------------------------------------------------------------------------
// 振荡器故障错误陷阱程序

void _ISR _OscillatorFail(void)
{
    LATDbits.LATD0 = 1;                 // 点亮 LED
    while(1);                           // 无限地等待
}

//----------------------------------------------------------------------------
// 地址错误陷阱程序

void _ISR _AddressError(void)
{
    LATDbits.LATD0 = 1;                 // 点亮 LED
    while(1);                           // 无限地等待
}

//----------------------------------------------------------------------------
// 堆栈错误陷阱程序

void _ISR _StackError(void)
{
    LATDbits.LATD0 = 1;                 // 点亮 LED
    while(1);                           // 无限地等待
}

//----------------------------------------------------------------------------
// 数学（算术）错误陷阱程序

void _ISR _MathError(void)
{
    LATDbits.LATD0 = 1;                 // 点亮 LED
    while(1);                           // 无限地等待
}
```

注：

© 2007 Microchip Technology Inc.

附录 D dsPICDEM 2 开发板代码

D.1 FLASH LED WITH dsPIC30F4011.s

本附录包含了 dsPICDEM 2 开发板的代码示例。该示例是为 MPLAB ASM30 汇编器和 C30 C 编译器编写的。

```
;=============================================================================  ;
;                                                                               ;
;                                软件许可协议                                   ;
;                                                                               ;
;   Microchip Technology Incorporated（以下称为“本公司”）在此为 dsPIC 单片机提供  ;
;   的软件旨在向本公司客户提供专门用于 Microchip dsPIC 产品的软件。本软件为本公司及 /  ;
;   或其供应商所有，并受到适用的版权法保护。版权所有。使用时违反前述约束的用户可能会依法  ;
;   受到刑事制裁，并可能由于违背本许可的条款和条件而承担民事责任。              ;
;                                                                               ;
;   本软件是按“现状”提供的。任何形式的保证，无论是明示的、暗示的或法定的，包括但不限于  ;
;   有关适销性和特定用途的暗示保证，均不适用于本软件。对于在任何情况下，因任何原因造成的  ;
;   特殊的、附带的或间接的损害，本公司概不负责。                                ;
;                                                                               ;
;=============================================================================  ;
;
; 使用 Timer 1 计数，未按开关 S5 时，使 LED D3 闪烁；
;                    按下开关 S5 时，使 LED D4 闪烁。
; 必须正确安装跳线 H6-M ALL、 H12-M D3 和 H12-M D4
;
;=============================================================================

        .equ __30F4011, 1
        .include "p30f4011.inc"

;-----------------------------------------------------------------------------
; 全局声明

        .global __reset                 ; 第一行代码的标号
        .global __OscillatorFail        ; 声明振荡器故障陷阱程序标号
        .global __AddressError          ; 声明地址错误陷阱程序标号
        .global __StackError            ; 声明堆栈错误陷阱程序标号
        .global __MathError             ; 声明数学错误陷阱程序标号
```

```
;-------------------------------------------------------------------------------
; 配置位

       config __FOSC, CSW_FSCM_OFF & XT_PLL4
       config __FWDT, WDT_OFF
       config __FBORPOR, PBOR_OFF & BORV_27 & PWRT_16 & MCLR_EN
       config __FGS, CODE_PROT_OFF

;-------------------------------------------------------------------------------
; 与程序有关的常数 （代码中使用的立即数）

       .equ   FCY, #7372800                  ; 指令周期频率 （Osc x PLL / 4）

;===============================================================================
; 代码开始

       .text                                 ; 代码段开始

;-------------------------------------------------------------------------------
; 初始化堆栈指针极限寄存器

__reset:      mov    #__SP_init, W15         ; 初始化堆栈指针寄存器
              mov    #__SPLIM_init, W0       ; 获取堆栈空间的结束地址
              mov    W0, SPLIM               ; 装载堆栈指针极限寄存器
              nop                            ; 在 SPLIM 初始化后添加一条 NOP 指令

;-------------------------------------------------------------------------------
; 初始化 PORTD 上的 LED 输出引脚 bit 0-3

              mov    #0x0000, W0             ; 初始化 LED 引脚数据为关闭状态
              mov    W0, LATB
              mov    #0x0003, W0             ; 设置 LED 引脚为数字引脚，而非模拟引脚
              mov    W0, ADPCFG
              mov    #0xfffc, W0             ; 设置 LED 引脚为输出
              mov    W0, TRISB
              bset   LATB, #0                ; 点亮 LED D3

;-------------------------------------------------------------------------------
; 初始化 Timer1 的周期为 1/5 秒

              clr    T1CON                   ; 控制寄存器清零以关闭 Timer1
              clr    TMR1                    ; Timer1 从 0 开始计数
              mov    #FCY/256/5,W0           ; 获得 1/5 秒对应的周期寄存器值
              mov    W0, PR1                 ; 装载 Timer1 周期寄存器
              mov    #0x8030,W0              ; 设置 Timer1 （预分频比为 1:256）
              mov    W0, T1CON               ; 载入 Timer1 的设置到控制寄存器

;-------------------------------------------------------------------------------
; 循环并等待 Timer1 匹配，发生匹配时翻转 LED1 或 LED2

MainLoop:     btssIFS0, #T1IF               ; 检查 Timer1 中断标志位是否置 1
              bra    MainLoop               ; 不断循环，直到置 1 为止
              bclr   IFS0, #T1IF            ; Timer1 中断标志位清零
              btss   PORTE, #8              ; 测试开关 S5 （按下时为低电平）
              bra    SwitchPressed

              btg    LATB, #0               ; 未按下 S5 时，翻转 LED D3
              bclr   LATB, #1               ; 熄灭 LED D4
              bra    MainLoop               ; 循环返回
```


© 2007 Microchip Technology Inc.

```
SwitchPressed:bclr   LATB, #0            ; 熄灭 LED D3
              btg    LATB, #1            ; 按下 S5 时，翻转 LED D4
              bra    MainLoop            ; 循环返回 MainLoop

;=============================================================================
; 错误陷阱

;-----------------------------------------------------------------------------
; 振荡器故障错误陷阱程序

       .text                            ; 代码段开始
__OscillatorFail:
              bset   LATB, #1           ; 点亮 LED D4
              bra    __OscillatorFail   ; 发生振荡器故障时进入无限循环

;-----------------------------------------------------------------------------
; 地址错误陷阱程序

__AddressError:
              bset   LATB, #1           ; 点亮 LED D4
              bra    __AddressError     ; 当发生地址错误时进入无限循环

;-----------------------------------------------------------------------------
; 堆栈错误陷阱程序

__StackError:
              bset   LATB, #1           ; 点亮 LED D4
              bra    __StackError       ; 当发生堆栈错误时进入无限循环

;-----------------------------------------------------------------------------
; 数学 （算术）错误陷阱程序

__MathError:
              bset   LATB,#1            ; 点亮 LED D4
              bra    __MathError        ; 当发生数学错误时进入无限循环

;=============================================================================

       .end                             ; 此文件中的代码结束
```

D.2 FLASH LED WITH dsPIC30F4011.c

```
/*********************************************************************************
*                                软件许可协议                                    *
*                                                                                *
*   Microchip Technology Incorporated（以下称为 “本公司”）在此为 dsPIC 单片机提    *
*   供的软件旨在向本公司客户提供专门用于 Microchip dsPIC 产品的软件。本软件为本公司及 /  *
*   或其供应 商所有，并受到适用的版权法保护。版权所有。使用时违反前述约束的用户可能会依  *
*   法受到刑事制裁，并可能由于违背本许可的条款和条件而承担民事责任。                *
*                                                                                *
*   本软件是按 “现状”提供的。任何形式的保证，无论是明示的、暗示的或法定的，包括但不限于  *
*   有关适销性和特定用途的暗示保证，均不适用于本软件。对于在任何情况下，因任何原因造成的  *
*   特殊的、附带的或间接的损害，本公司概不负责。                                  *
*                                                                                *
*                                                                                *
*                                                                                *
*                                                                                *
*********************************************************************************
*
* 使用 Timer 1 计数，未按开关 S5 时，使 LED D3 闪烁;
*                    按下开关 S5 时，使 LED D4 闪烁。
* 必须正确安装跳线 H6-M ALL、 H12-M D3 以及 H12-M D4
*
*********************************************************************************/

#include "p30F4011.h"

//---------------------------------------------------------------------------
// 配置位

_FOSC(CSW_FSCM_OFF & XT_PLL4);
_FWDT(WDT_OFF);
_FBORPOR(PBOR_OFF & BORV_27 & PWRT_16 & MCLR_EN);
_FGS(CODE_PROT_OFF);

//---------------------------------------------------------------------------
// 与程序有关的常数

#define   FCY    7372800             // 指令周期频率 （Osc x PLL / 4）

//===========================================================================
// 主程序
// 设置 LED 和定时器，等待若干个定时器周期后，使两个 LED 中的一个闪烁

int main(void)
{
    static int MyVar1, MyVar2, MyVar3, MyVar4, MyVar5;

    MyVar1 = 1000;
    MyVar2 = 20;
    MyVar3 = MyVar1 + MyVar2;
    MyVar4 = MyVar1 * MyVar2;
    MyVar5 = MyVar4 / MyVar3;
```


© 2007 Microchip Technology Inc.

```
    LATB = 0x0000;                  // 初始化 LED 引脚数据为关闭状态
    TRISB = 0xfffc;                 // 设置 LED 引脚为输出
    LATBbits.LATB0 = 1;             // 点亮 LED D3

    T1CON = 0;                      // 关闭 Timer1 并将设置清零
    TMR1 = 0;                       // Timer1 从 0 开始计数
    PR1 = FCY/256/5;                // 获得 1/5 秒对应的周期寄存器值
    T1CON = 0x8030;                 // 启动 Timer1，其预分频比为 1:256

    while(1)                        // 无限循环
    {
        while(!IFS0bits.T1IF){}     // 等待一个定时器周期

        IFS0bits.T1IF = 0;          // 定时器标志位清零，以用于下一周期
        if(PORTEbits.RE8)           // 检查 S5 是否被按下
        {
            LATBbits.LATB0 ^= 1;    // 未按下 S5 时，翻转 LED D3
            LATBbits.LATB1 = 0;     // 熄灭 LED D4
        }
        else
        {
            LATBbits.LATB0 = 0;     // 熄灭 LED D3
            LATBbits.LATB1 ^= 1;    // 按下 S5 时，翻转 LED D4
        }
    }
}                                   // main() 结束

//============================================================================
// 错误陷阱

//----------------------------------------------------------------------------
// 振荡器故障错误陷阱程序

void _ISR _OscillatorFail(void)
{
    LATBbits.LATB1 = 1;             // 点亮 LED D4
    while(1);                       // 无限地等待
}

//----------------------------------------------------------------------------
// 地址错误陷阱程序

void _ISR _AddressError(void)
{
    LATBbits.LATB1 = 1;             // 点亮 LED D4
    while(1);                       // 无限地等待
}

//----------------------------------------------------------------------------
// 堆栈错误陷阱程序

void _ISR _StackError(void)
{
    LATBbits.LATB1 = 1;             // 点亮 LED D4
    while(1);                       // 无限地等待
}
```

```
//------------------------------------------------------------------------------
// 数学 （算术）错误陷阱程序

void _ISR _MathError(void)
{
   LATBbits.LATB1 = 1;              // 点亮 LED D4
   while(1);                        // 无限地等待
}
```


© 2007 Microchip Technology Inc.

索引

© 2007 Microchip Technology Inc.

X

Y

Z

全球销售及服务网点

美洲

公司总部 Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 1-480-792-7200
Fax: 1-480-792-7277
技术支持：
http://support.microchip.com
网址：www.microchip.com

亚特兰大 Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

波士顿 Boston
Westborough, MA
Tel: 1-774-760-0087
Fax: 1-774-760-0088

芝加哥 Chicago
Itasca, IL
Tel: 1-630-285-0071
Fax: 1-630-285-0075

达拉斯 Dallas
Addison, TX
Tel: 1-972-818-7423
Fax: 1-972-818-2924

底特律 Detroit
Farmington Hills, MI
Tel: 1-248-538-2250
Fax: 1-248-538-2260

科科莫 Kokomo
Kokomo, IN
Tel: 1-765-864-8360
Fax: 1-765-864-8387

洛杉矶 Los Angeles
Mission Viejo, CA
Tel: 1-949-462-9523
Fax: 1-949-462-9608

圣克拉拉 Santa Clara
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

加拿大多伦多 Toronto
Mississauga, Ontario,
Canada
Tel: 1-905-673-0699
Fax: 1-905-673-6509

亚太地区

亚太总部 Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

中国 – 北京
Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

中国 – 成都
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

中国 – 福州
Tel: 86-591-8750-3506
Fax: 86-591-8750-3521

中国 – 香港特别行政区
Tel: 852-2401-1200
Fax: 852-2401-3431

中国 – 南京
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

中国 – 青岛
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

中国 – 上海
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

中国 – 沈阳
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

中国 – 深圳
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

中国 – 顺德
Tel: 86-757-2839-5507
Fax: 86-757-2839-5571

中国 – 武汉
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

中国 – 西安
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

台湾地区 – 高雄
Tel: 886-7-536-4818
Fax: 886-7-536-4803

台湾地区 – 台北
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

台湾地区 – 新竹
Tel: 886-3-572-9526
Fax: 886-3-572-6459

亚太地区

澳大利亚 Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

印度 India - Bangalore
Tel: 91-80-4182-8400
Fax: 91-80-4182-8422

印度 India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

印度 India - Pune
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

日本 Japan - Yokohama
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

韩国 Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

韩国 Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 或
82-2-558-5934

马来西亚 Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

马来西亚 Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

菲律宾 Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

新加坡 Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

泰国 Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

欧洲

奥地利 Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

丹麦 Denmark-Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

法国 France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

德国 Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

意大利 Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

荷兰 Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

西班牙 Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

英国 UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820

10/05/07

© 2007 Microchip Technology Inc.