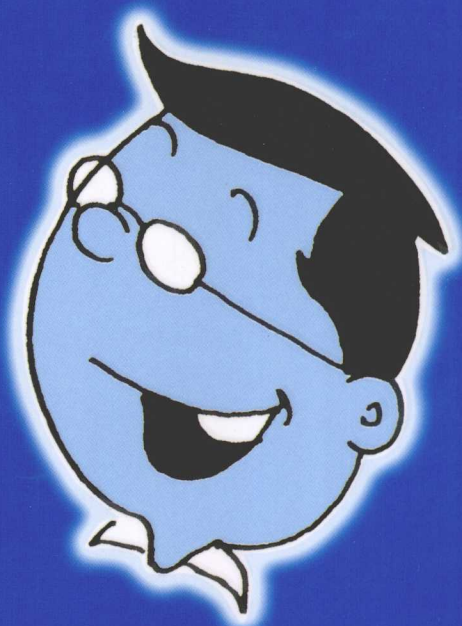
单片机轻松入门系列

MCU

PIC®单片机轻松入门

周 坚 编著



北京航空航天大学出版社

CD-ROM INCLUDED

单片机轻松入门系列

PIC® 单片机轻松入门

周　坚　编著

北京航空航天大学出版社

内 容 简 介

本书以 PIC16 系列单片机的典型芯片为例，详尽介绍了单片机的工作原理、C 语言编程、开发与应用等方面的知识，包括单片机的结构、MPLAB 开发环境、HI－TECH 的 C 语言编程知识、典型接口器件应用等。

本书使用 Proteus 仿真系统作为教学工具；作者为本书写作而开发了硬件实验电路板；随书光盘提供了书中 Proteus 仿真电路的源文件、各例子的源程序以及实验过程与现象的动画等。因此，读者获得的不仅是一本文字教材，更是一个完整的学习环境。

本书结合了作者多年教学、科研实践所获取的经验，融入了作者教学改革的成果，并依据学习者的认知规律来编排内容，充分体现了“以人为本”的指导思想。

图书在版编目(CIP)数据

PIC®单片机轻松入门/周坚编著. —北京：北京航空航天大学出版社，2009.7

ISBN 978－7－81124－612－4

Ⅰ.P… Ⅱ.周… Ⅲ.单片微型计算机 Ⅳ.TP368.1

中国版本图书馆 CIP 数据核字(2009)第 048889 号

PIC®单片机轻松入门

周 坚 编著

责任编辑 孔祥燮 范仲祥

*

北京航空航天大学出版社出版发行

北京市海淀区学院路 37 号(100191) 发行部电话：010－82317024 传真：010－82328026

http://www.buaapress.com.cn E-mail:bhpress@263.net

北京时代华都印刷有限公司印装 各地书店经销

*

开本：787×1 092 1/16 印张：15.25 字数：390 千字

2009 年 7 月第 1 版 2009 年 7 月第 1 次印刷 印数：5 000 册

ISBN 978－7－81124－612－4 定价：29.00 元(含光盘 1 张)

版 权 声 明

商 标 声 明

前 言

随着单片机开发技术的不断发展，目前已有越来越多的人从普遍使用汇编语言过渡到逐渐使用高级语言进行开发。其中主要以C语言为主，市场上几种常见的单片机均有其C语言开发环境。本书将以目前广为流行的Microchip公司PIC单片机为例来学习单片机的C语言编程技术。

简 介

在本书编写以前，作者在多年教学、科研实践以及对单片机课程进行教学改革的基础上，编写了《单片机C语言轻松入门》一书。该书以80C51单片机为例来学习C语言，发行后受到广大读者的欢迎，读者反映该书的确能起到“轻松入门”的作用。本书以PIC单片机为例，延续《单片机C语言轻松入门》一书的风格，带领读者“轻松入门”。通过学习PIC单片机内部结构、C语言的基础知识、Proteus软件的使用及用C语言开发PIC单片机所需的其他相关知识等，最终学会用C语言编写程序。

本书采用“以任务为中心”和C语言体系结构两条主线来编排内容，全书的内容按C语言体系结构来编排，而每一章的内容则采用“以任务为中心”的方式来编排，将C语言编程所需的基本知识，如C语言中的变量、常量、保留字、程序结构、运算符、表达式等知识，结合PIC单片机的结构特点及HI-TECH软件使用方法等，通过一系列的“任务”进行介绍。每个“任务”都包括了一些C语言的知识点。HI-TECH软件的使用、程序调试方法。单片机结构方面及单片机开发中必须了解的其他知识。每个任务都是易于完成的，在完成这些任务后，即可掌握上述各知识点。因此，对于一个已有一定汇编程序编写经验的单片机程序员而言，甚至在学完第1章后，就可以尝试用C语言来改写原来编写过的程序。对于一个刚开始学习单片机的读者来说，则可以同步学到单片机结构、C语言编程及HI-TECH软件使用等各方面的知识。

为了给读者一个完整的练习环境，作者使用Proteus软件设计了一系列的仿真文件，读者可以直接使用这些仿真文件来练习LED显示、键盘操作、数码管显示、串行通信等程序，也可利用学到的Proteus相关知识来完成更多的仿真设计。此外，作者还设计了一块硬件实验实验板，并在随书光盘中提供了该实验板的原理图和印刷线路板图，读者可以使用这一电路板来做一些使用仿真无法完成的练习。

内容安排

第1章是单片机的C语言概述。通过本章的学习，可以了解C语言的基本知识，能够识读一些C语言源程序。

第2章介绍如何建立单片机的C语言学习环境。对于C语言学习而言，一个可供练习的环境非常重要，不同于PC上用的C语言，除了软件实验环境外，单片机的C语言学习还要求有硬件实验环境。本章介绍的是一个具有可操作性的软、硬件实验环境。

第3～5章介绍C语言的数据类型、程序结构和构造数据类型。这一部分知识是C语

言的最基本知识，掌握之后即可进行常用程序的编写工作。

第 6 章介绍单片机的内部结构编程知识，包括 PIC 单片机内部常用的定时器、串行口、CCP 功能和 A/D 转换器的编程方法。

第 7 章介绍函数及其相关知识，包括函数定义、函数调用、全局变量与局部变量、变量的存储方式等内容。

第 8 章介绍常用单片机接口的 C 语言编程，安排了键盘、LED 显示器、I^2C 接口、SPI 接口、实时钟转换、液晶显示器等内容的 C 语言编程实例，并通过这些实例，讲述常用外围电路的 C 语言编程方法，增强读者的实际应用能力。

第 9 章是应用设计举例，引导读者从入门到开发。本章介绍了若干个简单但比较全面的程序，读者可以利用它们来做一些比较完整的“产品”，以便了解使用 C 语言开发项目的完整过程。

本书特点

C 语言语法知识并不难学，使用也不困难，但很多读者的问题在于不知道在什么场合可以使用这些知识。因此，本书在编写时，尽可能为每一个知识点都找到工程实际中的应用实例，以便读者更好地理解相关知识，并尽快将其应用到自己的开发实践中。

使用 C 语言进行嵌入式开发实践性很强，必须通过较多的实践操作才能学好。本书编写时考虑读者的实际情况，在讲解例子时，不假设读者随时可以有老师指导，而是立足于自学。书中不仅使用文字对有关实验过程进行细致的介绍，而且在随书光盘上还大量应用动画形式提供实验过程和效果以供参考，对于部分内容还提供了完整的操作过程的动画记录，保证读者可以无师自通。

作者为本书的写作使用 Proteus 设计了多个仿真文件，并设计了实验电路板。随书光盘提供了作者所设计的 Proteus 仿真文件、书中所有的例子，以及记录使用实验仿真板实验过程的动画等。读者获得的不仅是一本文字教材，更是一个完整的学习环境。

本书安排的例子大部分是由作者编写的，部分是参考其他资料改写而成，全部程序都由作者调试并通过。对于例子的使用说明也尽量详细，力争让读者“看则能用，用则能成”，保证读者在动手的过程中会常常体会到成功的乐趣，而不是常常遇到挫折。

本书第 1、2、7 章由周坚编写；第 5、6 章和第 8、9 章分别由常州轻工职业技术学院龚益民、冷雪锋两位老师编写，他们还负责全书 Proteus 软件相关的绘图、仿真调试等工作；第 3、4 章由常州旅游商贸高等职业技术学校汤欣老师编写；全书由周坚统稿。另外，华旭东、夏爱联参与了部分硬件电路的设计、制作及调试工作；张庆明、史建福等参与了部分程序的调试工作；陈素娣参与了多媒体制作、插图绘制、文字输入、排行等工作，在此表示由衷地感谢。

本书在提供文字教材的同时还通过网络为广大读者提供服务，欢迎读者与作者探讨。

网站：平凡单片机工作室(http://www.mcustudio.com)。

作　者

2009 年 2 月

目 录

第 1 章

单片机的 C 语言概述

随着单片机开发技术的不断发展，目前已有越来越多的人从普遍使用汇编语言到逐渐使用高级语言进行开发，其中主要是以 C 语言为主，市场上几种常见的单片机均有其 C 语言开发环境。本书将以流行的 PIC 单片机为例来学习单片机的 C 语言编程技术。下面首先来介绍有关 C 语言的基本知识。

1.1 C 语言简介

1.1.1 C 语言的产生与发展

C 语言由早期的编程语言 BCPL(Basic Combind Programming Language)发展演变而来。1970 年，美国贝尔实验室的 Ken Thompson 根据 BCPL 语言设计出了 B 语言，并用 B 语言编写了 UNIX 操作系统。1972—1973 年间，贝尔实验室的 D. M. Ritchie 在 B 语言的基础上设计出了 C 语言。

随着微型计算机的日益普及，出现了许多 C 语言版本，由于没有统一的标准，使得这些 C 语言之间出现了一些不一致的地方。为了改变这种情况，美国国家标准研究所(ANSI)为 C 语言制定了一套 ANSI 标准，成为现行的 C 语言标准。

1.1.2 C 语言的特点

C 语言发展非常迅速，成为最受欢迎的语言之一，主要因为它具有强大的功能。归纳起来，C 语言具有下列特点。

1. 与汇编语言相比

(1) C 语言是一种高级语言，具有结构化控制语句。结构化语言的显著特点是代码及数据的分隔化，即程序的各部分除了必要的信息交流外，彼此独立。这种结构化方式可使程序层次清晰，便于使用、维护及调试。

(2) C 语言适用范围大，可移植性好。与其他高级语言一样，C 语言不依赖于特定的 CPU，其源程序具有很好的可移植性。只要某种 CPU 或 MCU 有相应的 C 编译器，就能使用 C 语言进行编程。目前，主流的 CPU 和常见的 MCU 都有 C 编译器。对于嵌入式系统的开发者，这一点显得很重要。现在可供选用的 MCU 品种极多，这些 MCU 各有特点，开

发者在做不同项目时往往需要选用不同型号的MCU，以发挥其特长。如果要熟悉每一种MCU的汇编语言并能写出高质量的程序，并非易事。如果使用C语言编程，借助于C语言的可移植性，只要熟悉所用MCU的特性即可编程，这可节省大量的时间，从而将精力专注于所要解决的问题上面。

作者曾多次将应用程序从80C51单片机移植到PIC单片机上，或者从PIC单片机移植到AVR单片机，甚至32位ARM机上。对于这些内核完全不同的单片机，如果使用汇编语言，则必须完全重写程序，要想在短时间内掌握各种单片机的汇编语言并非易事。但由于这些程序都是用C语言编写的，因此可以进行移植。用户在找到这些单片机的C编译程序后，可一边熟悉这些单片机的内部结构，一边进行移植，在很短的时间内即可将此程序完全移植成功。这其中的绝大部分时间是用来熟悉不同型号单片机的内部结构，理解其硬件用法与其他单片机之间的差异，真正用于改写代码的时间非常少，需要修改的代码量也很少。

2. 与其他高级语言相比

(1) 简洁、紧凑、灵活、方便。C语言一共只有32个关键字及9种控制语句，程序书写自由，主要用小写字母表示。它把高级语言的基本结构和语句与低级语言的实用性有效地结合起来。

(2) 运算符丰富。C语言的运算符包含的范围很广泛，共有34个运算符。C语言把括号、赋值、强制类型转换等都作为运算符处理，从而使C语言的运算类型极其丰富，表达式类型多样化。灵活使用各种运算符，可以实现在其他高级语言中难以实现的运算。

(3) 数据结构丰富。C语言的数据类型有整型、实型、字符型、数组类型、指针类型、结构体类型、共用体类型等，能用来实现各种复杂的数据类型的运算。

(4) C语言语法限制不太严格，程序设计自由度大。例如，对数组下标越界不做检查，由程序编写者自己保证程序的正确；对变量的类型使用比较灵活，整型、字符型等各种变量可通用。

(5) C语言允许直接访问物理地址，可以直接对硬件进行操作。C语言既具有高级语言的功能，又具有低级语言的许多功能，能够像汇编语言一样对位、字节和地址进行操作。

(6) C语言程序生成的代码质量高，程序执行效率高。用C语言编写的程序，编译后一般只比有丰富经验的汇编编程人员所写的汇编程序效率低10%～20%。

3. 关于C语言的学习

很多人对于学习C语言有一种畏惧情绪，因为传说中C语言很难学，这里有必要对此进行一些说明。

C语言难学的这种说法来源于在PC上编程的程序员。早期PC上使用的编程语言主要有BASIC语言、PASCAL语言、数据库(DBASE、FOXBASE等)编程语言等，与这些语言相比，C语言的确是属于“难学”的语言，但必须对为何“难学”进行深入的分析。由于C语言能够完成底层的操作，所以往往被用来编写一些系统软件，而这必然涉及更深奥的计算机基础知识。例如，必须理解ASCII码的知识及数据在内存中如何存放的知识，才能理解为何只要简单的一句“c=c+32;”即可完成大写字母到小写字母的转换工作，而这样的工作在BASIC语言中是用专门的函数来完成的，并不需要使用者了解这些知识。换言之，C语言的难学来自于其要完成的任务。对于单片机编程而言，这些知识本来就属于必须要掌握的，

所以如果编程者已有一定的汇编语言基础，那么这就不是什么难题。作为单片机开发者，必须要接触硬件，理解这些知识，所以这些并不能算“C 语言难学”。

1.2　C 语言入门知识

下面将通过一些实例介绍 C 语言编程的方法，这里采用 PIC 系列单片机的 C 编译器 HI－TECH 软件作为开发环境，关于 HI－TECH 软件的具体用法将在第 2 章作详细介绍。

图 1－1 所示电路使用 PIC16F877A 单片机作为实验用芯片，这款单片机内部有 8 KB 的 Flash ROM，可以反复擦写，并有 ISP 功能，支持在线下载，不需要反复拔、插芯片，非常适于做实验。2.4 节将详细介绍使用这种单片机制作的硬件实验平台。如图 1－1 所示，PIC16F877A 的 PORTB 引脚上接了 8 个发光二极管，下面一些例子的任务是让接在 PORTB 引脚上的发光二极管按要求点亮或熄灭。

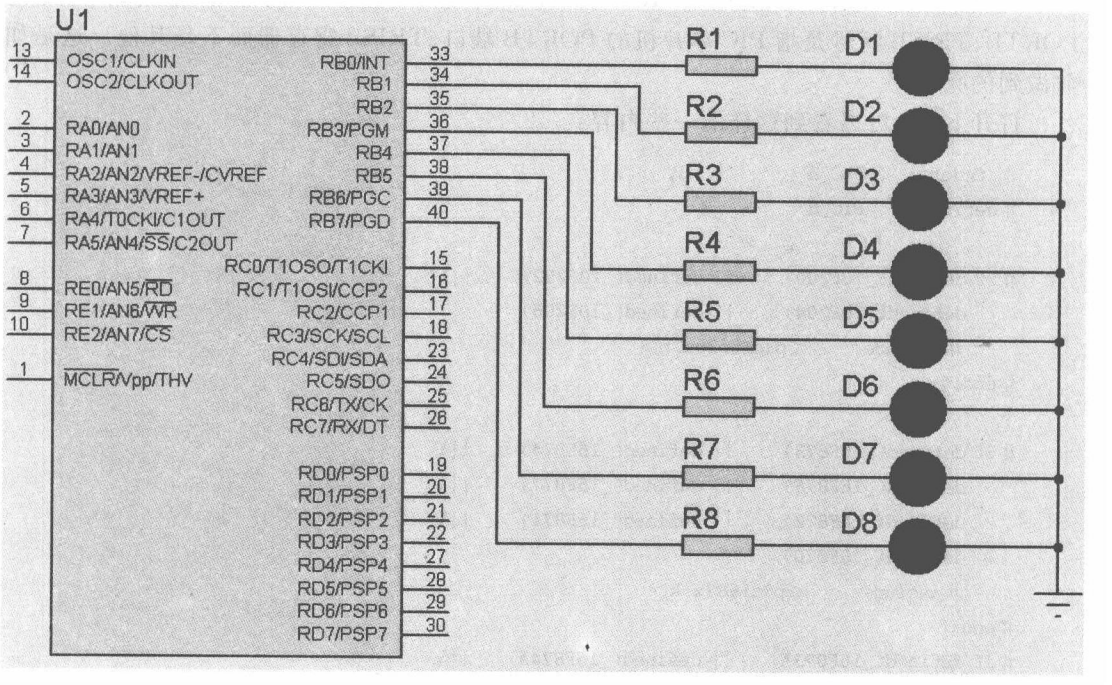


图 1－1　接有 LED 的单片机基本电路*

1.2.1　简单的 C 程序介绍

【例 1－1】 让接在 RB0 引脚上的 LED 发光。

```
#include "pic.h"
__CONFIG(HS&WDTDIS&LVPDIS);
```

* 本书中用 Proteus 软件所生成的图中，有一些器件符号的画法及标称值的写法与国标不一致。为了保持与软件的一致，这类图保持原样未改。

```
//配置文件,设置为HS方式振荡,禁止看门狗,低压编程关闭
void main()
{    TRISB = 0;          //PORTB设置为输出
     PORTB = 0x01;       //RB0输出高电平,其余引脚均为低电平
     for(;;){;}
}
```

这个程序的功能是让接在RB0引脚上的LED点亮。下面来分析这个C语言程序包含了哪些信息。

1. “文件包含”处理

程序的第1行是一个“文件包含”处理。

所谓“文件包含”是指一个文件将另外一个文件的内容全部包含进来,所以看起来这个程序只有6行,但C编译器在处理这段程序时却要处理几十或几百行。这段程序中包含pic.h文件的目的是为了使用PORTB、TRISB符号,即通知C编译器,程序中所写的PORTB、TRISB符号是指PIC单片机的PORTB端口、TRISB寄存器而不是其他。这是如何做到的呢?

打开pic.h可以看到这样的一些内容:

```
#ifndef    _PIC_H
#define    _PIC_H

#if defined(_10F200)    || defined(_10F202)    ||\
    defined(_10F204)    || defined(_10F206)
    #include    <pic10f20x.h>
#endif
⋮
#if defined(_16F873)    || defined(_16F874)    ||\
    defined(_16F876)    || defined(_16F877)    ||\
    defined(_16F872)    || defined(_16F871)    ||\
    defined(_16F870)
    #include    <pic1687x.h>
#endif
#if defined(_16F873A)   || defined(_16F874A)   ||\
    defined(_16F876A)   || defined(_16F877A)
    #include    <pic168xa.h>
#endif
⋮
```

PIC单片机与80C51系列单片机的不同之处在于其包含了一个庞大的系列,这个系列中的很多芯片有其特定的头文件。为了编写程序的方便,HI-TECH编译器给出了一个统一的头文件pic.h,在这个文件中,根据编译环境所定义的器件名称,调入定义这个器件的头文件。在2.2节介绍工程实现时会看到,要编译这段程序,需要先建立一个工程,在建立工程时将定义器件名称为16F877A,相当于满足了下述条件中的defined(_16F877A)部分:

```
#if defined(_16F873A)     || defined(_16F874A)     ||\
    defined(_16F876A)     || defined(_16F877A)
```

因此，在编译程序时会执行：

```
#include     <pic168xa.h>
```

这样的一行语句，从而调入 pic168xa.h 头文件。下面再来看一看这个头文件中又包含了什么样的内容。

pic168xa.h 的内容如下：

```
/*
 *     Header file for the Microchip
 *     PIC 16F873A chip
 *     PIC 16F874A chip
 *     PIC 16F876A chip
 *     PIC 16F877A chip
 *     Midrange Microcontroller
 */

#if defined(_16F874A)     || defined(_16F877A)
#define __PINS_40
#endif

static volatile unsigned char     INDF      @ 0x00;
static volatile unsigned char     TMR0      @ 0x01;
static volatile unsigned char     PCL       @ 0x02;
static volatile unsigned char     STATUS    @ 0x03;
static          unsigned char     FSR       @ 0x04;
static volatile unsigned char     PORTA     @ 0x05;
static volatile unsigned char     PORTB     @ 0x06;
static volatile unsigned char     PORTC     @ 0x07;
#ifdef __PINS_40
static volatile unsigned char     PORTD     @ 0x08;
static volatile unsigned char     PORTE     @ 0x09;
#endif
static          unsigned char     PCLATH    @ 0x0A;
static volatile unsigned char     INTCON    @ 0x0B;
static volatile unsigned char     PIR1      @ 0x0C;
⋮
/* PORTB bits */
static volatile bit RB7 @  (unsigned)&PORTB * 8 + 7;
static volatile bit RB6 @  (unsigned)&PORTB * 8 + 6;
static volatile bit RB5 @  (unsigned)&PORTB * 8 + 5;
static volatile bit RB4 @  (unsigned)&PORTB * 8 + 4;
static volatile bit RB3 @  (unsigned)&PORTB * 8 + 3;
static volatile bit RB2 @  (unsigned)&PORTB * 8 + 2;
```

```
static volatile bit RB1 @   (unsigned)&PORTB * 8 + 1;
static volatile bit RB0 @   (unsigned)&PORTB * 8 + 0;

/* TRISB bits */
static volatile bank1 bit TRISB7 @   (unsigned)&TRISB * 8 + 7;
static volatile bank1 bit TRISB6 @   (unsigned)&TRISB * 8 + 6;
static volatile bank1 bit TRISB5 @   (unsigned)&TRISB * 8 + 5;
static volatile bank1 bit TRISB4 @   (unsigned)&TRISB * 8 + 4;
static volatile bank1 bit TRISB3 @   (unsigned)&TRISB * 8 + 3;
static volatile bank1 bit TRISB2 @   (unsigned)&TRISB * 8 + 2;
static volatile bank1 bit TRISB1 @   (unsigned)&TRISB * 8 + 1;
static volatile bank1 bit TRISB0 @   (unsigned)&TRISB * 8 + 0;
⋮
```


这个头文件很长，以上是部分与 PORTB 有关的定义。从以上定义可以看出，这些符号的定义规定了符号名与地址的对应关系。注意，其中有：

```
static volatile unsigned char PORTB @ 0x06;
```

这样的一行（上文中用黑体表示），即定义 PORTB 与地址 0x06 对应，PORTB 的地址就是 0x06（0x06 是 C 语言中十六进制数的写法）。

2. 配置文件

```
__CONFIG(HS&WDTDIS&LVPDIS);
```

这一行程序称之为配置文件，其目的是为这款单片机进行配置。PIC 单片机内部具有多种功能，可以根据需要来进行配置，以便在不同场合能正确地工作。这种配置工作可以在烧写芯片时手工设置，不过更好的方法是将配置写在程序中，在生成可烧写文件后，就将配置信息也包含在内了。对于大部分的编程器所配套的软件而言，它们能够识别这种信息，从而自动完成配置，避免了手工设置可能带来的错误。

3. main 主函数

每一个 C 语言程序有且只有一个主函数 main，函数后面一定有一对大括号“{}”，在大括号里面书写其他程序。

通过上面的分析了解了部分 C 语言的特性，下面再看一个稍复杂一点的例子。

【例 1-2】 让接在 RB0 引脚上的 LED 闪烁发光。

```
#include "pic.h"
#define uchar unsigned char
#define uint  unsigned int
__CONFIG(HS&WDTDIS&LVPDIS);
//配置文件，设置为 HS 方式振荡，禁止看门狗，低压编程关闭
void mDelay(uint DelayTime)
{    uint temp;
     for(;DelayTime>0;DelayTime--)
     {    for(temp=0;temp<270;temp++)
```

```
        {;}
    }
}
void main()
{   TRISB = 0;              //PORTB 设为输出
    PORTB = 0;
    for(;;)
    {   RB0 = !RB0;         //取反 RB0
        mDelay(1000);
    }
}
```

程序分析：main 函数中的第 1 行暂且不看，第 5 行是“RB0＝！RB0;”，在 RB0 前有一个符号“!”，符号“!”是 C 语言的一个运算符，就像数学中的“＋”、“－”一样，是一种运算符号，意义是“取反”，即将该符号后面的那个变量的值取反。

注意：取反运算只是对变量的值而言的，并不会自动改变变量本身。

可以认为，C 编译器在处理“！RB0”时，将 RB0 的值给了一个临时变量，然后对这个临时变量取反，而不是直接对 RB0 取反，因此取反完毕后，还要使用赋值符号“＝”将取反后的值再赋给 RB0。这样，如果原来 RB0 是低电平（LED 亮），那么取反后，RB0 就是高电平（LED 灭）；反之，如果 RB0 是高电平，取反后，RB0 就是低电平。这条指令被反复地执行，接在 RB0 上的灯就会不断“亮”、“灭”。

该条指令会被反复执行的关键就在于 main 函数中的第 1 行程序“for(;;)”。这里不对此作详细介绍，读者暂时只要知道，这行程序连同其后的一对大括号“{}”构成了一个无限循环语句，一旦程序开始运行，该大括号内的语句会被反复执行，直到断电为止。

第 3 行程序是“mDelay(1000);”。这行程序的功能是延时 1 s 时间。由于单片机执行指令的速度很快，如果不进行延时，灯亮之后马上就灭，灭了之后马上就亮，亮与灭之间的间隔时间非常短，人眼根本无法分辨。

这里，“mDelay(1000)”并不是由 HI－TECH C 提供的库函数，如果在编写其他程序时写上这么一行，就会发现编译通不过。那么这里为什么又能正确编译呢？注意观察，可以发现这个程序中有“void mDelay(…)”开始的一段程序行，可见，mDelay 这个词是编程者自己起的名字，并且为此编写了一段程序，如果读者的程序中没有这么一段程序，那就不能使用“mDelay(1000)”了。有人脑子快，可能马上会想到，可不可以把这段程序复制到其他程序中，然后就可以在那个程序中用“mDelay(1000)”了呢？回答是：当然可以。还有一点需要说明，mDelay 这个名称是由编程者自己命名的，可自行更改，但一旦更改了名称，main()函数中的名字也要作相应的更改。

mDelay 后面有一个小括号，小括号里有数据(1000)，这个 1000 被称为“参数”，用它可以在一定范围内调整延时时间的长短，这里用“1000”来要求延时时间为 1 000 ms，要做到这一点，必须通过自行编写 mDelay 程序来决定。

1.2.2　C 程序的特点

通过上述的几个例子，可以得出一些结论：