HZ BOOKS 华章教育

计 算 机 科 学 丛 书

ELSEVIER 爱思唯尔

嵌入式C编程

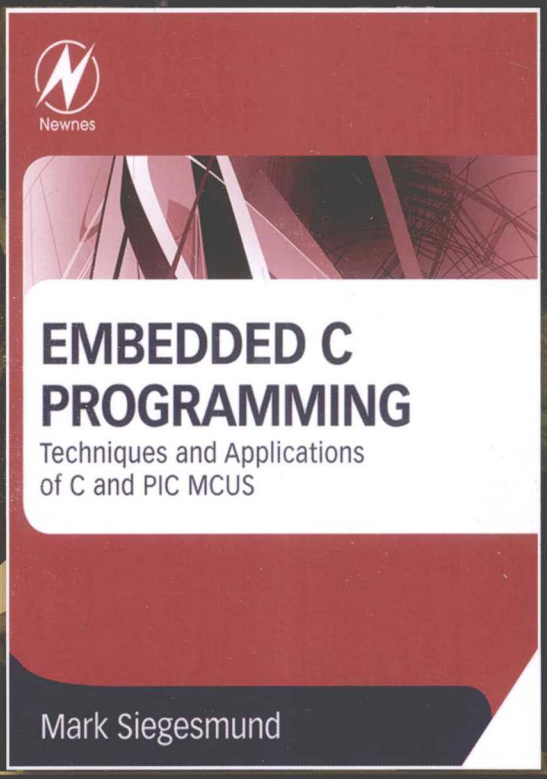
PIC单片机和C编程技术与应用

[美] 马克·西格斯蒙德（Mark Siegesmund） 著

王文峰 袁洪艳 译

Embedded C Programming

Techniques and Applications of C and PIC MCUS



机械工业出版社
China Machine Press

计 算 [redacted] 学 丛 书

嵌入式C编程

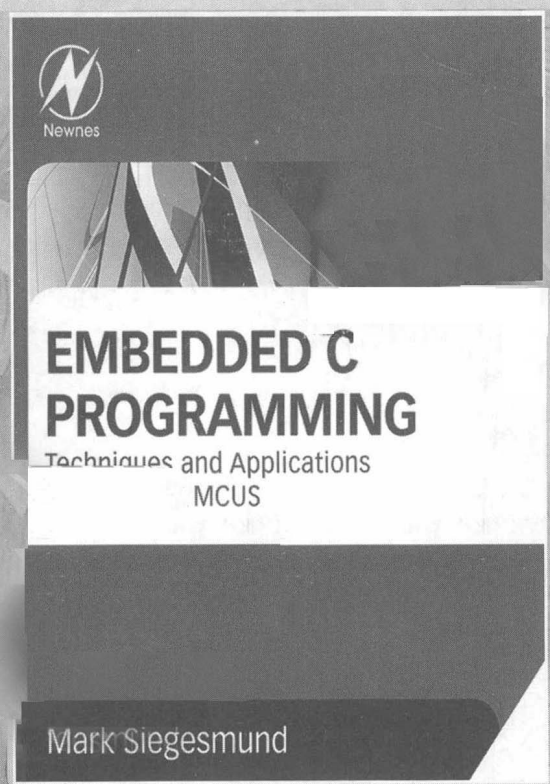
PIC单片机和C编程技术与应用

[美] 马克·西格斯蒙德（Mark Siegesmund） 著

王文峰 袁洪艳 译

Embedded C Programming

Techniques and Applications of C and PIC MCUS



机械工业出版社
China Machine Press

图书在版编目（CIP）数据

嵌入式 C 编程：PIC 单片机和 C 编程技术与应用 /［美］马克·西格斯蒙德（Mark Siegesmund）著；王文峰，袁洪艳 . —北京：机械工业出版社，2017.5
（计算机科学丛书）
书名原文：Embedded C Programming：Techniques and Applications of C and PIC MCUS

ISBN 978-7-111-56444-7

I. 嵌… II. ① 马… ② 王… ③ 袁… III. 单片微型计算机 – C 语言 – 程序设计
IV. TP368.1

中国版本图书馆 CIP 数据核字（2017）第 065380 号

本书版权登记号：图字：01-2015-5220

本书介绍 PIC 单片机的 C 语言程序设计方法，前半部分详细讲解 C 语言的基本概念，后半部分重点关注 PIC 及其外围组件。全书风格简洁清晰，知识点、代码示例、编程练习都紧紧围绕工程实践需求。本书不要求读者具备 C 语言或硬件接口的预备知识，初学者可边学边练逐步精进，而有一定基础的程序员则可从作者分享的实战经验中获益。

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码：100037）
责任编辑：曲　熠　　责任校对：殷　虹
印　　刷：北京瑞德印刷有限公司　　版　　次：2017 年 5 月第 1 版第 1 次印刷
开　　本：185mm×260mm　1/16　　印　　张：18.25
书　　号：ISBN 978-7-111-56444-7　　定　　价：79.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991　88361066　　投稿热线：（010）88379604
购书热线：（010）68326294　88379649　68995259　　读者信箱：hzjsj@hzbook.com

文艺复兴以来，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的优势，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭示了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短的现状下，美国等发达国家在其计算机科学发展的几十年间积淀和发展的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起到积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章公司较早意识到“出版要为教育服务”。自 1998 年开始，我们就将工作重点放在了遴选、移译国外优秀教材上。经过多年的不懈努力，我们与 Pearson，McGraw-Hill，Elsevier，MIT，John Wiley & Sons，Cengage 等世界著名出版公司建立了良好的合作关系，从他们现有的数百种教材中甄选出 Andrew S. Tanenbaum，Bjarne Stroustrup，Brian W. Kernighan，Dennis Ritchie，Jim Gray，Afred V. Aho，John E. Hopcroft，Jeffrey D. Ullman，Abraham Silberschatz，William Stallings，Donald E. Knuth，John L. Hennessy，Larry L. Peterson 等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及珍藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力相助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专门为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近两百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍。其影印版“经典原版书库”作为姊妹篇也被越来越多实施双语教学的学校所采用。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证。随着计算机科学与技术专业学科建设的不断完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都将步入一个新的阶段，我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

华章网站：www.hzbook.com
电子邮件：hzjsj@hzbook.com
联系电话：（010）88379604
联系地址：北京市西城区百万庄南街 1 号
邮政编码：100037

华章科技图书出版中心

无论是嵌入式还是C语言，都有相当数量的巨著。这些书籍会详细介绍每个知识点，非常适合系统学习，但往往篇幅过长。本书结合工程实践，详细介绍了最常用的知识点，简洁却不简单，足以满足工程中的应用。

本书前半部分着重介绍C语言，对没有C语言基础的读者非常实用。每章之后附带的练习和测验能够帮助读者回顾本章内容，并了解这些内容在工程中是如何应用的。

本书后半部分将重点放到CCS C编译器和PIC单片机上。在介绍PIC单片机的外围组件时，作者没有停留在手册层面，而是分不同的领域，详细介绍这些组件的标准和使用方法，并比较了某些组件在不同PIC系列中的特点。最后两章介绍了在实践中非常实用的内联汇编和调试技术，总结了嵌入式系统中常见的几种调试手段，为理论知识到工程实践的进阶做了很好的铺垫。

作者有着非常丰富的工作经验，语言轻松活泼。译者试图在翻译过程中将这种风格传承下来，但限于文字水平，仅能部分体现，抑或有所疏漏。此外，书中出现的术语可能会有多种翻译，如MCU，根据语境会使用“单片机”或者“微控制器”，请读者灵活理解。

译　者

2017年3月

微控制器是将微型计算机的主要部分集成到一个芯片上的单芯片微型计算机。上电后，它会运行内部程序存储器上的程序。内部程序存储器可以使用只读存储器（ROM）或者Flash。我们在各种地方都会发现微型计算机的踪影，如家用电器、玩具、汽车和计算机外设（如键盘或鼠标）。从电池充电器到雷达系统，几乎所有的电子设备中都有微型计算机的身影。

PIC 微控制器的速度很快，在写作本书时已经达到每秒 7000 万次指令，且价格低廉，某些型号的单价在 1 美元以内。由于这些优点，它逐渐成为新的设计方案中最流行的选型，并大量地应用于接口中，如 USB、以太网和模拟信号接口。

C 语言最初是由 AT&T 实验室的 Brian Kernighan 和 Dennis Ritchie 开发的，称为 K&R C，随后在 1989 年被 ANSI 标准化，也就是 C89。从 C 语言中衍生出的新特性催生了 C++ 语言。1998 年，C++ 的 ISO 标准被批准通过。C++ 有些复杂的语言元素，使其无法应用于微控制器而只能用于 PC 这样相对宏观的设备。C 是微控制器编程中最常用的语言。

C 语言在计算机语言中属于高级语言。高级语言使用一种名为编译器的工具将 C 文本文件转换成机器语言文件。

本书的前半部分着重讲述 C 语言。如果读者在这之前已经有些编程经验，这会对阅读本书有些帮助，但这些经验并不是必需的。我们会使用 C 语言元素的正式定义，详细列出微控制器所需要的全部语言特性。本书从第 15 章开始介绍 PIC 微控制器及其外围组件，以及如何在 C 语言中使用这些组件。硬件接口部分相关的基础电子知识有助于读者理解本书，但这些知识对于阅读本书也不是必需的。

在不同的编译器厂商之间或不同的微控制器系列之间会有些不同的 C 语言扩展。本书中出现的不同的处理器或编译器之间可能会产生兼容性问题，这些部分都做了标记。每一章也会分享一些好的编程实践经验以及相应的文档。为了巩固所学的概念，每章都会提供练习和测验。本书中的例子全部可以在 CCS C 编译器中编译通过，这也是 Microchip PIC 系列处理器最常用的编译器。

Mark Siegesmund

目 录

Embedded C Programming: Techniques and Applications of C and PIC MCUS

C 语言概述和程序结构

1.1 C 源代码

下面是一段 C 语言源代码：

```
/*  Chapter One sample C program for
    the CCS C compiler */

#include <e3.h>

void main(void) {
    int i;

    for( i=1; i<=10; i=i+1 ) {
         output_high(PIN_C6);        //Turn green LED on
         delay_ms(500);
         output_low(PIN_C6);         //Turn green LED off
         delay_ms(500);
    }
}
```

这段代码初看起来可能会觉得难以理解，但读完本书并完成书中的实验之后，你就可以轻松地理解它，甚至能理解比它更复杂的程序。读完稍后几章再回过头来看这段程序，你会发现它非常简单。现在，让我们先对它有个整体的认识。

1.2 注释

注释是对代码的说明，可以帮助任何一个阅读代码的人（也包括你自己）理解代码。通常，有两种注释方式：

```
/*    */    注释内容位于 /* 和 */ 之间（可以跨越多行，但不能嵌套使用）
//           注释内容位于 // 和行尾之间（只允许在一行内使用）
```

编译器会忽略所有注释。上面的示例程序同时使用了上述两种类型的注释。

1.3 程序结构

C 程序由编译单元组成，有时也称作翻译单元。一组文件由编译器编译在一起就构成了一个编译单元。本书中大多数例子都只用了一个编译单元。编译单元由全局数据和函数组成。把一段经常使用的代码封装起来，在使用的时候可以直接调用，这就是 C 语言中的函数，在其他编程语言中，这也被称为过程或子程序。函数中的局部数据只能在函数内部定义和使用。

1.4 C 预编译指令

预编译是 C 语言中一个非常有意思的特性。预处理使用工具（预处理器）在编译前先扫

描一遍代码，并对代码做出相应的修改从而生成用来编译的代码。预编译指令由 # 开始，占用一整行。在第 3 章中将会详细介绍它。在上面的例子中，**#include** 指令将文件（e3.h）的整个内容都替换到 **#include** 这一行，然后再来编译。

例如，假设我们创建了一个名为 delay.inc 的文件，并将下面一行代码添加进去：

```
delay_ms(500);
```

这样可以将上面程序中的两行 **delay_ms** 替换成 **#include <delay.inc>**。程序的最终编译结果也和以前没有区别。在编译开始前，预处理器会读取 delay.inc 文件的内容，然后将所有 **#include <delay.inc>** 替换成 **delay_ms(500)**。

预处理器是 C 语言的一个强大特性，可以提高程序的可读性，让我们能够在最大程度上重用代码，并显著提高程序的可维护性。

在上述示例程序中，第一行就是一个预编译指令，用来包含 e3.h 文件。在许多工程和硬件规范定义中，除注释外，程序第一行通常都用一个 **include** 指令来包含某文件。这类文件通常以 .h 为扩展名。在上述例子中，E3 硬件所需的所有定义都在 e3.h 这个文件中。

1.5 函数

接下来，我们看到了一个函数定义“**main**”。所有程序都需要有一个 **main()** 函数，也就是主函数。这是程序开始执行的地方。在本书中，当提到函数名时，我们都会在它的名字后面跟上 **()**，表示这是个函数。

函数名前的 **void** 表示此函数的类型是 **void** 类型（空类型），也就是不返回任何内容，**()** 内的 **void** 表示该函数不需要调用者传入任何内容。花括号 **{}** 是集合符号。所有函数内容都要放到一对花括号内。

第 7 章会详细介绍函数，这里，为了对函数有个初步的印象，先给出了几个例子：

```
x=sin(y);                sin是个有一个参数和一个返回值的函数
x=sin(y*3.1415/180);     参数可以是任意一个表达式
x=180*sin(y)/3.1415;     返回值可以直接用到表达式中
```

1.6 声明

“**int i**”声明了一个以标识符 **i** 作为变量名的数据类型。**int** 表示该变量是一个整数。本例中，**i** 只能在 **main()** 函数中使用。如果声明 **i** 的这一行放到函数之前（在函数外），那么 **i** 也可以在其他函数中使用。变量的使用范围称作变量作用域。作用域将在第 4 章中详细介绍。

1.7 语句和表达式

在上面的例子中 **for** 那一行就是一个语句（statement）。程序在运行时会执行这些语句。该语句包含 3 个表达式。表达式将在第 5 章介绍，语句将在第 6 章介绍。在 **for** 语句中：

```
程序首先执行第一个表达式                                    i=1
然后重复执行下列步骤：
测试第二个表达式，如果结果为假（false）则退出循环          i<=10
执行括号)之后的语句
执行第三个表达式                                          i=i+1
```

在这个例子中，**for** 语句的后面有 4 行语句放在一对大括号 **{}** 内，随着 **i** 从 1 增长到

10，被执行 10 次。当 i 变成 11 的时候，由于条件 11<=10 为假，条件不成立，循环终止。

表达式由常量、变量、运算符和函数调用组合而成。表达式通常都有一个结果（值）。像 + - * / 以及稍微有点特殊的 =，则是一些简单的运算符。

在上面的例子中，由于在 for 循环中需要执行 4 条语句，我们将它们用 { 和 } 包围起来构成一个组，也称为复合语句。大括号中可以包含 0 条或多条语句。如果没有大括号，则只有 output_high() 会被执行 10 次，后面的 3 条语句将在整个循环语句结束之后运行一次。

在这个 for 循环中的 4 行语句每一行都是一个函数调用。我们在程序中并没有定义这些函数，但编译器会将它们编译进去。函数名后的括号告诉编译器这是个函数调用。函数名后面括号内的表达式是要传入函数的数据。在函数调用的地方，它们称为实际参数（argument，也叫实参）；而在函数定义中，称为形式参数（parameter，也叫形参）。

C 语言中，每个有效语句都以分号 ; 结尾。但是需要注意，这里说的是“有效”，而不是“有意义”。例如，下面是一个有效的 C 语言语句：

```
1+2;
```

但是，该语句并没有完成任何工作。有些编译器可能会在这条语句上浪费点时间将其编译一下，但也仅此而已，并没有任何意义。好的编译器在这一行会弹出一个警告，因为这可能是程序员输入错了。

在分号前不写任何语句，我们称之为空语句。空语句什么都不做。

C 语言不像其他一些语言有赋值语句，而是使用赋值运算符 = 来赋值：

```
x=3;
```

x=3 这个表达式包括变量、运算符和常量，再加上 ;，构成了一个完整的语句。赋值语句将右侧（rvalue）的值赋给左侧（lvalue）的变量。

1.8 时间

在 delay_ms 中，ms 表示毫秒（milliseconds）。编程中常用的时间单位有：

ns nanosecond（纳秒） 0.000 000 001s

μs microsecond（微秒） 0.000 001s

ms millisecond（毫秒） 0.001s

其中，100 000 000 μ s = 1s。

1.9 输入准确度

编写 C 源代码时输入的准确度也非常重要。每个标点符号，无论是输入错误还是遗漏，都会导致许多令人头疼的问题，因为你的程序可能无法通过编译。而编译器则是严格地按照你的输入来编译的。

例如，如果 for 语句中的第一个 { 忘记写了，编译器将会在后面的 } 处弹出错误。

语句和声明之后的分号非常重要，编译器通过分号来识别定义或语句的结尾。以 # 开头的预编译指令结尾不需要分号。丢失或者额外多出来的分号或大括号可能产生令人困惑的编译错误。好的 C 语言编辑器会高亮提示大括号和小括号的配对以及相应的语法元素，从而减少输入错误。

1.10 文本格式

编译器会忽略空格、制表位（Tab）、回车等空白字符。格式化代码有助于我们阅读代码。空白字符（缩进），如空格、制表位（Tab）和空行，可以帮助我们更好地组织代码。Tab可以代替多个空格来控制代码缩进。我们注意到，在上面的代码中，函数内的代码都做了缩进，而循环内的代码则再缩进一层。缩进和其他空白字符在代码中属于可选部分，但是我们强烈推荐使用它们来格式化代码。由于编译器并不关心这些字符，也就无所谓正确或者错误的用法。某些公司可能有自己的代码规范，会规定代码的缩进、注释、函数最大行数，以及代码可读性方面的其他要求。

1.11 兼容性

// 注释属于C++风格，并不是所有的C编译器都支持。

大多数C编译器对大小写敏感。例如，`Output_High()`和`output_high()`是两个不同的函数。默认情况下，CCS C编译器对大小写不敏感。要想使其对大小写敏感，必须加上`#case`预编译指令。

`output_high()`和`delay_ms()`等内建函数不在C标准中，它们是CCS C编译器特有的。

1.12 小结

- 程序一个或多个编译（或翻译）单元构成。
- 编译单元的预编译指令在编译之前会首先由预编译解决。
- 编译器会忽略注释和大多数空白字符。
- 一个编译单元就是一个文件，其中包含一些函数和全局数据声明，它们的排列顺序没有关系。
- 函数有局部数据声明和语句，它们被包含在一对大括号内。
- 函数可能有返回值，调用者也可以在调用时传递参数进去。在函数调用时传递的是实参，函数定义时则是形参。
- 一组语句可以放到一个大括号`{}`内，构成一个复合语句。
- 某些语句中可以包含表达式。
- 语句可以是任意一个以分号结尾的表达式。
- 表达式由常量、变量、运算符和函数调用组成，通常用于求值运算。

1.13 练习

目标：了解如何使用编译器和原型开发板。

工具：E3模块、USB连线、PC。

步骤 / 技术流程	笔记
1. 安装并运行编译器，设置硬件。有关问题请参照附录B。	
2. 创建源文件。 `File > New > Source` 输入Ex1-1；IDE会自动为文件添加.c的扩展名。	

（续）

步骤 / 技术流程	笔记
3. 输入本章前面的例子并保存： `File > Save`	
4. 编译代码。 单击屏幕上方菜单栏中的“编译”，会出现编译下拉列表；再单击下拉列表中的编译图标，（期望）源程序将会编译成功。这时会出现输出窗口，显示编译器输出结果。如果没有出现错误（0 errors），就表示成功了，这也是我们期望的结果。	
5. 用 USB 线将 E3 板子和 PC 连接起来。 当 Windows 系统检测到 E3 这个新硬件并提示安装驱动时，搜索目录将指向“c:\program files\PICC\USB Drivers”。在你的计算机上，“program files”的名字可能和这个不一样，酌情修改。 注意，E3 板子是通过 USB 连线供电的。 E3 板子上已经预置了一个程序，方便用户通过 USB 将程序下载到板子上。	
6. 在 IDE 的 Compiler（编译）下拉列表中选择 PROGRAM CHIP 中的 E3 BOOTLOAD。	
7. 如果一切顺利，我们会看到 E3 板子上的绿色 LED 每秒钟闪烁一次。 此时会弹出一个编程窗口（串口监视器），稍后的练习会用到这个窗口，现在我们可以关掉它。	
8. 修改程序，使 LED 每 5s 闪烁一次。 编译，下载，测试。	

目标：进一步熟悉工具的用法。

工具：E3 模块、USB 连线、PC。

步骤 / 技术流程	笔记
1. 了解出错处理，将源代码中的两个 PIN_C6 改成 PIN_C66 并重新编译。 编译器提示两个错误：无法识别 PIN_C66。 双击其中一个错误消息，光标会移动到编译器探测到错误的位置。 更正错误并重新编译。	
2. 删除 `main()` 函数定义中括号内的 `void` 并重新编译。 这次没有编译错误，但是有一个警告（warning）。 “警告”表示编译器可以处理当前的程序，但是怀疑其中有些错误。 这个例子中，编译器觉得你可能忘记给函数写参数了。加入 `void` 的作用是显式地告诉编译器该函数没有参数。 更正错误，重新编译。	
3. 在 VIEW> DATA SHEETS > Other PDF's > E3 Schematic 中查看板子的原理图，找出为什么我们使用 PIN_C6 来点亮绿色 LED。	
4. 修改程序使红色 LED 闪烁，并测试其是否工作。	
5. 修改程序，使其能够判断 LED 点亮时引脚的状态处于低电平还是高电平。	
6. 在编译下拉列表中单击 C/ASM 按钮，查看 C 和汇编的混合文件，也叫 list 文件，使用 .LST 做后缀名。 数一下程序中有几条汇编指令。 数一下每行 C 代码（不包括预编译指令、注释和空行）被翻译成几条汇编指令。有时这种统计数据可以用来判断编译器性能的好坏。 使用“语句”代替之前的“C 代码行”重新计算一个语句转换成多少汇编指令。有人认为使用“平均每个语句有多少条汇编指令”更能表现编译器性能。	
7. 修改程序，使 LED 每 5s 闪烁一次。 编译，下载，验证。	