高等学校教材

MCS—51单片机高级语言PL/M—51程序设计及应用

战 明 刘克金 徐 峰 金智贤 编著

NEUPRESS
东北大学出版社

MCS—51单片机高级语言PL/M—51程序设计及应用

战明　刘克金　徐峰　金智贤　编著

东北大学出版社

（辽）新登字第 8 号

内 容 简 介

本书系统地介绍了 MCS－51 单片机 PL/M－51 高级语言，并在此基础上介绍了 PL/M－51语言的查表、散转、排序、代码转换等程序设计方法，给出了各种典型 I/O 接口的应用程序。此外，还介绍了PL/M－51的模块连接技术，PL/M－51与汇编语言的连接技术，PL/M－51浮点库的使用方法，以及 PL/M－51 的调试方法。这使读者能够掌握 PL/M－51单片机高级语言及其编程和调试技术，使之脱离繁琐的汇编语言开发阶段，进而达到使用高级语言编程、调试的高技术水平。

本书在叙述上深入浅出，实用性很强，适用于自学和作为大专院校及各种培训班的教材，对从事单片机应用开发的技术人员也是颇为有益的读物。

MCS－51 单片机高级语言
PL/M－51 程序设计及应用
战明　刘克金　徐峰　金智贤　编著

东北大学出版社出版发行
（沈阳·南湖）
辽宁地质勘查局地质矿产研究所印刷厂印刷

开本：787×1092 毫米　1/16　印张：14.625　字数：365 千字
1994 年 1 月第 1 版　1994 年 1 月第 1 次印刷
印数：1～1400 册

责任编辑：王金邦　责任校对：王　令
封面设计：唐敏智　责任出版：高志武

ISBN 7-81006-649-8/TP·33　定价：11.00 元

前　言

随着计算机技术的发展，单片机在过程控制及智能化仪表等方面的应用越来越广泛，并起着重要的作用。目前从事单片机开发及应用的技术人员正不断增加，从总体上来看，大多数的系统仍处在汇编语言的开发层次上，这样在一个单片机应用系统的开发过程中，硬件设计由于各种接口及系统扩展比较规范，所以其开发成本相对较低，而软件设计由于各个系统的要求千差万别，使其开发成本相对较高。解决这个问题的根本方法就是使用高级语言开发单片机。

在单片机系统的开发过程中，程序设计和程序调试是一个不可分割的统一过程，不仅要根据要求设计出源程序，而且要在尽可能短的时间内调试出这个程序，人们对这一点的认识日益深刻。目前，大多数的软件设计和调试都处在汇编语言的层次上，对于汇编语言其可调性要比高级语言低得多。本书的目地是帮助读者掌握 PL/M－51 单片机高级语言及其程序设计技术，以使读者脱离繁琐的汇编语言开发阶段，进入用高级语言编程，同时帮助读者了解高级语言开发系统，学习高级语言调试技术，从而达到用高级语言编程，并在高级语言级调试的高技术水平。

PL/M－51 是开发 MCS－51 系列单片机的高级语言，它具有软件开发周期短、可读性好、可靠性高、可维护性高、代码转换效率高等特点。在一般情况下，由 PL/M－51 语言生成的机器代码，不论是代码长度、还是程序运行的速度都优于或不亚于由汇编语言所产生的机器代码。作为高级语言，PL/M－51 易于描述算法，体现人的设计思想。不仅如此，PL/M－51 语言还能象汇编语言那样直接对 CPU 的硬件和外设接口进行操作，从而使软件设计的负担大大减轻。PL/M－51 语言是简单易学的，特别是对于掌握 BASIC 语言和 MCS－51 汇编语言的人来说，很容易理解和掌握。

本书的内容共有十五章，分为三个部分。第一个部分为第一至十一章，介绍 PL/M－51 语言；第二部分为第十二至十四章，介绍 PL/M－51 程序设计及应用程序设计；第三部分为第十五章，介绍 PL/M－51 语言的调试技术。

本书力求深入浅出，不仅介绍了 PL/M－51 语言，而且还详细介绍了其各种程序设计及其接口程序设计的技巧和方法，结合单片机具体应用的硬件环境深入阐述了 PL/M－51 语言在 MCS－51 单片机开发中的用法和典型的实用程序，对从事单片机应用开发的技术人员是一本很有实用价值的教材。本书可作为大专院校及各种培训班的教材，也适合作自学读本。

本书第一、二、三、十四、十五章由刘克金编写，第四、五、六、七、八、九、十一、十二、十三章及附录由战明编写，第十章由徐峰编写。全书由刘克金和金智贤组织，金智贤最后统稿。

由于我们的水平有限和编写时间仓促，书中难免存在许多缺点和错误，敬请读者批评并加以谅解。

编　者

1992.8.15

目　　录

第一章 概 述

1.1 PL/M—51 语言

PL/M—51 是由美国 INTEL 公司设计用于开发 MCS—51 系列单片机的高级语言。它由 AEDIT.EXE，PLM51.EXE，RL51.EXE，PLM51.LIB，OH.EXE 软件组成。

AEDIT.EXE 为全屏幕编辑软件，用于编辑源程序。

PLM51.EXE 是编译软件，编译软件把源程序翻译成可重新定位的目标，编译时编译系统能自动检查用户源程序中的各种语法错误，并在生成的 *.LST 文件中指出错误的位置和类型。

RL51.EXE 是连接软件，它将用户的目标码同 INTEL 公司提供的 PLM51.LIB 库（子程序和算法库）连接，也可同时和其他 PL/M—51 目标码、汇编语言目标码以及他高级语言的目标码进行连接，生成完整的可执行代码。

OH.EXE 软件将用户的可执行代码转换成可调试的程序代码（HEX 格式），然后再用仿真器进行调试。

作为高级语言 PL/M—51 比汇编语言更接近于人类的思维过程，更容易体现人的设计思想。它能够象自然语言一样来描述算法，一个应用软件从构思到代码实现过程，用高级语言要比用汇编语言简捷的多。如表达式 M=1+3 就是将 1 和 3 相加，其结果存储在变量 M 中。不仅如此，PL/M—51 还能象汇编语言那样直接对 CPU 及外设接口的硬件进行操作。例如标识符 P1 可以说明为地址为 8031 的 P1 口的变量，从 P1 口输入一个字节数，可以表示为 M=P1，这条语句的执行结果是将 P1 口的数传递给变量 M。因此，与高级语言相比，PL/M—51 具有硬件控制功能强，使用灵活，用途广泛等特点。

1.2 PL/M—51 特点

1. 简单易学

PL/M—51 语言只有两类语句：一类是说明语句，用于说明变量和过程；另一类是可执行语句，如赋值语句、条件语句和循环语句等。这些语句的功能和 BASIC 语言中相应的语句相似。如

```
P1=02H;                    /* 赋值语句 */
IF M>2 THEN N=P1;          /* 条件语句 */
DO I=0 TO 10;              /* 循环语句 */
```

2. 可读性好

PL/M－51是一种结构化语言，其程序是块式结构。可采用缩进式方法书写源程序，使程序层次清楚，便于理解和阅读。一个完整的程序可由多个模块组成，而每个模块又是由多个程序块构成。

3. 可靠性高

PL/M－51编译软件检查用户程序对单片机的各种资源使用情况，对冲突使用情况或不合理使用情况能提出警告，并能自动为用户程序合理分配内存。

4. 隔离性好

PL/M－51中，既可有全局的静态变量，也可有局部的动态变量，对于全局性的静态变量，其性质和 BASIC 中的变量一样，该变量在整个程序中均有效。对于局部的动态变量，它只有在其所说明的程序块中或过程中有效，在该程序块或过程之外则无效。

局部动态变量的特点是，当由多个人共同编制一个程序中的不同模块时，而不必担心是否使用了相同的标识符而发生冲突。模块化的程序设计增加了程序设计的灵活性和方便性。

5. 兼容性好

PL/M－51可同汇编语言或其他语言的目标模块连接起来，生成可执行代码，如果用户必须使用汇编语言编制一段程序，或是想利用已调好的汇编语言程序块，那么该段汇编可以很方便地连接到 PL/M－51程序目标块中。

6. 具有良好的算法库和库管理能力

PL/M－51语言具有 INTEL 公司所提供的算法库和库管理功能，用户能利用现成的算法库完成加、减、乘、除等运算，并能建立自己的特定算法库，使得编程一劳永逸，提高软件的开发效率。

第二章 PL/M－51编程基础

2.1 PL/M－51程序构成

Pl/M－51 程序是由多个程序行组成，对于每个程序行可分为标号段、语句段、注释段三个部分。标号段的内容为标号，在 PL/M－51中标号可以用作程序块、过程的名称。与汇编语言相同，标号是用来标识语句的位置。语句段的内容可为 PL/M－51可执行语句或说明语句，可以有一个语句，也可有多个语句，各语句间由“;”分隔。注释段的内容是对该语句的功能、作用进行说明，它不能生成可执行代码，只在源程序中提高程序的可读性。对于每一条语句可以有注释语句，也可以没有注释语句，一般是在程序中的每个功能段的开始处对该段程序的功能进行说明。下面为一个 PL/M－51程序。

```
标号段:语句段                          /* 注释段 */
DEMO: DO;
        DECLARE X BYTE;
        DECLARE Y BYTE;
        DECLARE C BYTE;
        X=1;Y=2;C=X+Y;      /* C←X+Y */
    END DEMO;
```

上面程序是由六个程序行组成，第一个程序行的内容为标号 DEMO 和 DO 语句，第二、三、四程序行只有语句段，各有一条说明语句。第五程序行有三条语句和注释内容。该程序的功能是将变量 X 与 Y 相加，其结果存储在变量 C 中。

2.2 PL/M－51字符集

PL/M－51语言程序是由一系列字符组成的，所用的字符集是 ASCII 字符集的子集（见附录 G）。其合法的字符有如下几类：

1. 英文字符

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
a b c d e f g h i j k l m n o p q r s t u v w x y z

2. 数字

0 1 2 3 4 5 6 7 8 9

3. 运算符

① 算术运算符：+　－　*　/
② 关系运算符：<　>　=
③ 点 操 作 符：　.

4. 分界符

(　)　,　:　;　'

5. 特殊字符

$　_　*　/

6. 空字符

Space（空格），Tab（制表符），Carriage－return（回车），Line－feed（换行）。

如果在一个 PL/M－51程序中包含了非上述的任何字符，编译程序将把它们作为错误对待。例如在程序中若用了一个变量名为 A!，则是非法的。

在 PL/M－51中，除了字符串之外，英文字母大小写在程序中的作用是相同的，即xyz和XYZ可以互相替代。语句 DECLARE 也可以写成 declare，其功能是相同的。此外，除了在字符串内容中有所不同，在 PL/M－51语言中，单个的空格和连续的空格是等效的。编译程序把连续的空格也当做一个单一的空格处理。

下面两条语句是等同的

```
DECLARE    (X1, X2)    BYTE;
DECLARE      ( X1,      X2)    BYTE;
```

2.3　标识符、保留字和预说明标识符

2.3.1　标识符

标识符是用来命名变量、过程、符号常数以及语句标号的一组字符。它可由字母、数字、下划线及货币 $ 符号组成，标识符的第一个字符必须是字母。在 PL/M－51中，标识符的长度最多可达31个字符（不包括 $ ）。

对于 PL/M－51语言，编译程序编译时，忽略嵌入标识符内的货币符号 $ ，对于标识符 P8279 $ C 和 P8279C 是等效的。利用这个特点，合理使用 $ 符号可以提高程序的可读性。例如对于标识符

LONGIDENTIFIERNUMBER3

很难表明其含义，但在其中加入一定的 $ 符号后变为

LONG $ IDENTIFIER $ $ $ NUMBER $ 3

可以根据其英文单词的含义反映出该标识符的含义。也就是说使用了 $ 后的标识符其可读性明显高于未使用 $ 符号的标识符。

这里要注意的是，$符号不能作标识符的第一个字符，增加和减少标识符或常数中的$并不改变这个标识符或常数，它们是等效的。

以下是有效标识符和常数：

A	MYNAME	PORT_1	PORT$1	00111001B
X(2)	P8279A	P8255C	PORT1	0011$1001B

其中PORT$1与PORT1是等效的，00111001B和0011$1001B是相等的。但PORT$1与PORT_1是不同的。

2.3.2 保留字

同BASIC语言一样，PL/M－51语言中也有一些保留字，所谓保留字就是在PL/M－51中有专门含义的一组字母和符号。这些保留字有的是语句定义符，有的是操作符。如

DECLARE，DO，END，IF　　为语句定义符，确定语句功能。

AND，OR，MINUS，PLUS　　为操作符，确定操作数操作。

以上都是保留字，全部的保留字见附录C。

由于保留字具有特定的含义，所以用户在程序中不能用保留字作标识符，说明为变量或过程名。但可以作标识符的一部分，如

AB$END，ANDR，DECLARE1等

2.3.3 预说明标识符

预说明标识符是PL/M－51语言提供的内部过程和内部变量的标识符。用户在程序中可以直接使用这些预说明标识符。如

```
Y=SHL (X, 4);    /* SHL是预说明标识符，该语句功能是向左逻辑移4位 */
```

与保留字不同，对于预说明标识符用户可以在程序中用说明语句重新说明这些标识符，但是一旦这样说明之后，这些标识符在所说明的程序块中便失去了原内部过程的作用，而具有新的含义，而在该程序块之外，则仍保持原过程的作用。附录E列出了全部预说明标识符，这里仅介绍预说明标识符的概念，关于其具体的内容将在第九章中介绍。

2.4 符号、分隔符及空白的作用

一条PL/M－51语句是由下列符号组成：

(1) 标识符：用以表示变量、标号和常数。

(2) 保留字：为语句定义符和操作符。

(3) 简单分界符：

```
+  -  *  /  <  >  =  .  (  )  ,  :  ;  '
```

(4) 复合分界符：特定两个字符的某种组合。即

```
<>  <=  >=  /*  */
```

(5) 数常数：如

```
251, 12, 45H
```

(6) 字符串：如 、

'WONT', 'SLASH'

在大多数场合，一个符号的结束点就是下一个符号的开始点。例如在赋值语句中

EXACT=APPROX * (OFFSET-3) /SCALL;

其中 EXACT，APPROX，OFFSET 及 SCALL 都是标识符，用以作为变量。3是数值常数，而其余各字符都是简单分界符。符号 EXACT 的结束点就是=的开始点。

在某些情况下，标识符、保留字和数常数必须由空格符号相互分离，例如，在两个标识符、保留字或数常数之间若没有出现简单或复合分界符时，则应在它们中间插入一个或多个空格作为分隔符。如

DECLAREABYTE;

是一个非法语句，应插入空格为：

DECLARE A BYTE;

注释也可以当做分隔符。如：

DECLARE A /* RESULT */BYTE;

也是合法的语句。

2.5 注　　释

在 PL/M-51源程序中，可以插入适当的注释内容，标明语句的作用、目的和功能，以改善程序的可读性。

注释的内容左边必须以符号/* 开始，右边必须是以符号 */结束。其中间的字符串（任何可打印的 ASCII 码，包括空格、回车换行及制表符）为注释内容。在编译时，编译程序对注释内容予以忽略，不生成可执行代码。

要注意的是注释内容不能嵌入字符串常数之内，因为这样它会成为该串的一部分，编译程序不能将其区分为注释，而视其为正常程序的一部分，不予忽略。例如，对于字符串

'B/* R */DA'

由于/* R */的加入，该字符串将被翻译成其相应的 ASCII 码为

42 2F 2A 52 2A 2F 44 41

而不会被认作

42 44 41

除此而外，注释能出现在空格能够出现的任何地方。即除了标识符、保留字之内的任何地方。

下面是 PL/M-51注释的一个实例：

```
/* This is an example of PL/M-51 */
DO;
  DECLARE X BYTE;
  DECLARE Y BYTE;
  X=1; Y=2;
  Y=Y+X;              /* Y←X+Y */
END;
```

2.6 常　数

所谓常数就是在程序执行期间不能改变值的数。在PL/M－51中，常数可分为数值常数和字符串常数两大类：

1. 数值常数

数值常数可为二进制、八进制、十进制和十六进制数。它们分别由后缀 B，O（或Q），D 和 H 来识别。对不带后缀的数，编译程序将其当作十进制数。如果一个常数内有所用数制内无效的字符，编译程序将指示出错。

例如，对于 PL/M－51，其最大的数值常数为

1111$1111$1111$1111B＝1777Q＝65535＝0FFFFH

要注意的是，对于十六进制的数，若第一个字符是非数字字符，则应在其前加0，以防止编译程序把它看成标识符。

例如，以十六进制数表示163必须写成0A3H 而不是 A3H，否则编译程序会将其错当成一个标识符，而不是一个数。

下面是一些有效的数值常数的例子：

12AH，2，33Q，1010B，55D，0BF3H，65535，777O，3EACH

而下面是一些无效的数值常数的例子：

12A，12AD，11102B，2ADGH

因数12A 不带后缀，则认为是十进制数，而在十进制数中没有 A 符号，所以是错误的。

同样，数12AD 最后的 D 可能作为一个后缀；不过，A 并不是十进制的数字。因此也是非法的．如果打算表示十六进制数，必须最后为 H。

11102B 中，2不是一个有效的二进制数。

2ADGH 中，G 不是一个有效的十六进制数。

对于 PL/M－51，一个数值常数可以是一个 BIT（位），BYTE（字节）或 WORD（字）的值，这取决于它的大小及前后关系。

2. 字符串常数

字符串常数是以单引号括起来的一组可打印的 ASCII 字符。如

'A'，'ABC'

它们也是常数。编译程序将字符串中的字符顺序转换成 ASCII 代码存放到存储器中，字符串中的每一个字符占一个字节，每个字节的最高位（D7位）为零，其余7位是字符的 ASCII 代码。长度为1的字符串占一个字节，长度为2的字符串占两个字节。在字符串中允许有（'），如

'ABC'DEF'

字符串中还允许有空格，但不允许有换行符。在这里要注意，字符串与 PL/M－51语言的其它部分不同，字母大、小写产生的代码值是不同的。这一点与 BASCI 语言一致。

下面是几个字符串及代码的例子。

'A'→ 41H，　　'a'→ 61H