PL/M-51

单片机程序设计语言

綦希林　陈嘉庆　编

科　学　普　及　出　版　社

PL/M-51 单片机程序设计语言

蔡希林　陈嘉庆　编

科学普及出版社

北　京

(京)新登字 026 号

图书在版编目(CIP)数据

PL/M-51 单片机程序设计语言/綦希林，陈嘉庆编，—北京:科学普及出版社,1993.3

ISBN 7—110—02941—7

Ⅰ.P…

Ⅱ.①綦… ②陈…

Ⅲ.①电子计算机—程序设计 ②电子计算机—程序语言

Ⅳ.TP312

科学普及出版社出版

北京海淀区白石桥路 32 号 邮政编码:100081

新华书店北京发行所发行 各地新华书店经售

北京市京精印刷厂

*

开本:787×1092 毫米 1/16 印张:8.5 字数:100 千字

1993 年 8 月第 1 版 1993 年 8 月第 1 次印刷

印数:1—2000 册 定价:11.00 元

前　言

当前单片机的应用已经广泛深入到工农业、科学技术、国防建设和人民生活的各个领域。据不完全统计，现在国内每年消耗的单片机数量已超过百万片。随着单片机应用的日益普及，单片机应用技术也在不断提高。目前，国内应用最广泛的当属 INTEL 公司的 MCS-51 系列单片机。MCS-51 系列单片机有多种语言支持，其中包括 PL/M-51 高级语言和 ASM-51 汇编语言。MCS-51 系列单片机现广泛应用在各种工矿企业的生产第一线，已成为工业测控系统和企业技术设备改造的优选机型。为了帮助广大用户开发单片机的应用程序，同时也为了配合已经出版的《多国单片机实用技术》和《ASM-51 单片机程序设计语言》两本书，我们又编印了《PL/M-51 单片机程序设计语言》以满足广大读者的需求。

对于从事 MCS-51 系列单片机应用的读者和在开发系统上进行单片机应用程序调试的用户，本书将会对您的工作给予很大的帮助。它将成为单片机应用领域中不可缺少的工具书。

在本书的校对过程中，得到了北京三环电子有限公司唐铭新和魏世萌同志的大力支持。在此特向他们致以诚挚的谢意。

由于编者的水平有限，加之时间仓促，书中难免有不妥之处，恳请广大读者批评指正。

编者

1993 年 3 月

目　录

第一章　概述

第二章　PL/M-51 程序基础

第三章　声明

第四章　数据类型及基变量

第五章　表达式及赋值

第六章　结构与数组

第七章　程序控制语句

第八章　示范程序 1

第九章　程序块结构、作用域以及使用期规则

第十章　过程及中断

第一章　概述

本章介绍 PL/M-51 语言以及使用这种语言开发应用软件的过程。

1.1 产品的定义

PL/M 是一种高级语言，用于为各种微型处理器和微型控制器编写程序。它由 Intel 公司设计，在系统和应用领域内的计算机软件设计中得到了广泛应用。

PL/M-51 编译程序是一种软件工具，它把 PL/M-51 源程序翻译成可以重新定位的目标模块，这些目标模块可以与其他 PL/M、汇编语言或别的高级语言代码模块连接起来。该编译程序提供一个输出列表、出错信息以及一些编译控制功能，以帮助程序的开发和调试。

编译之后，可以使用软件开发实用程序 RL51 及 LIB51。使用在线仿真器或开发系统调试应用程序，或把程序写到 EPROM 中。

1.2 PL/M-51 语言

使用高级语言　高级语言比起较低级的语言（如汇编语言）更接近于人类的思维过程。高级语言对于从概念至代码的翻译步骤比低级的语言要少些；相对更容易书写，且写得更快。由于高级语言程序产生错误的机会较少，也容易改正。

以高级语言编写的程序比那些以较低级语言书写的更容易阅读和理解，从而也容易修改。因此，开发高级语言程序只用较短时间。在开发和维护方面支付的费用也较低。

此外，高级语言程序容易从一种处理器移植到另一种处理器上。

如果首次使用 PL/M-51 高级语言，还应当知道高级语言与汇编语言在编程上有什么区别。使用一种高级语言时：

- 不必知道当前使用的处理器指令集，但必须了解它们的存贮器结构。
- 不必了解目标处理器的细节（如寄存器分配或为每项数据指定字节数），编译程序会自动处理。
- 使用类似于英语的关键字及短语
- 能够把许多操作（包括算术和逻辑操作）结合到表达式中去；因此，仅仅用一条语句执行整个操作顺序。
- 能够使用与实际问题更加接近的数据类型和数据结构。例如，在 PL/M-51 内能够用布尔变量、字符、数组，以及其他数据结构代替位、字节或字来编程。

高级语言的编程思路不同于汇编语言，而不用汇编语言，事实上采用高级语言编程更接近于人在规划总体系统设计时所使用的思路。

为什么采用 PL/M　当今有许多高级编程语言。有些高级语言早在 PL/M 语言之前就已

问世。那么 PL/M 与其他高级语言有什么不同？它的先进性表现在什么地方？它适合于什么应用环境？

以下是 PL/M 的若干特点：

- 具有程序块结构和控制构造，有助于结构化编程。
- 包含各种服务程序用于像结构数组以及有基指针动态变量那样的数据结构。
- 是一种类型化的语言，即程序在编译时作数据类型检查以帮助查找程序中的逻辑错误。
- 数据结构服务程序和控制语句设计成逻辑式的，因此 PL/M 对于系统编程表达算法是一种优良语言。
- 控制结构使程序无差错并较易校核。
- Intel 微型计算机的一种标准语言，因此 PL/M 程序在 Intel 处理器之间是可移植的。

PL/M 是为程序员（通常为系统程序员）设计的，他需要利用微处理器的特性，像间接寻址(BASED)及直接 I/O，以对全系统资源达到最佳的利用。

PL/M 与早先的语言像FORTRAN、BASIC以及COBOL有许多不同。PL/M比BASIC有更多的功能，比FORTRAN（最适于科学应用）或 COBOL（为事务数据处理所设计）任何一种来说是一种有更加通用目标的语言。此外，PL/M 以其类型及程序块结构更优于这些语言。

1.3 两种 PL/M-51 语句

PL/M-51 两种语句：声明语句和可执行语句。

声明语句的一个简单例子是：DECLARE WIDTH BYTE；

这条声明语句定义了标识符WIDTH，并把它与一个字节（8 位）存贮器内容联系起来。不必知道该字节的位置（即它在存贮器中的实际地址）。通过使用WIDTH名就可简单地访问到该字节的内容。

一个可执行语句的例子是：

CLEARANCE＝WIDTH＋2；

这条可执行语句有两个标识符：CLEARANCE及WIDTH。这两个标识符必须在这条可执行语句之前作出声明，然后从存贮器中找到WIDTH地址，对该值加 2，结果放入CLEARANCE存贮器地址中。

多数场合，PL/M-51 程序员不必考虑存贮器位置。CLEARANCE及WIDTH都是变量，由赋值语句把表达式WIDTH＋2的值送入变量CLEARANCE。编译程序自动地生成访问正确的存贮类型组成的数据所必须的全部机器代码、对该表达式求值，并把结果存入适当的位置。

一组用来执行一种功能的语句，即一个子程序，可以声明成一个过程：

ADDER—UPPER：PROCEDURE（BETE）；

之后为过程的语句。PL/M-51 的这个程序块将在程序的其他地方被引用。一个过程可以传送若干参数，并回送一个数值。当一个过程执行完成后，回到调用该过程的下面的语句。这是模块程序构造的主要特性。

1.4 模块化结构

PL/M-51 是一种结构化语言。PL/M-51 程序中的每一个语句都包含在一个相应的程序块中（一个程序块是一组语句，它由一个 DO 语句或一个过程声明打头，并用 END 语句结束）。PL/M-51 的编译单位是模块，模块是带标号的简单 DO 程序块；因此，一个模块必须由一个带标号的 DO 语句开始，并用一个 END 语句结束。模块中间（在该 DO 程序块内部）其他语句提供组成该程序的数据定义和进程。这些语句是该程序块的一部分，包含或嵌套在该程序块内。一个模块能包含其他的程序块，但它本身永远不能被别的程序块所包含。

(DO 程序块被称为简单的，是由于它只是四种 DO 程序块中的一种；其余三种要在本手册的后面讲解)。

PL/M-51 应用程序可以由一个或多个模块组成，它们被分别编译。每一个模块由一个或多个程序块组成。PL/M-51 有两种程序块：DO 程序块和过程程序程序块。

过程程序块是以过程声明开始（如节 1.3 所示）END 语句结尾的一组语句。其他的声明和可执行语句放在两者之间。

定义一个程序块实际是对该过程要执行的每一种对象作出进一步的声明。由于它以后才能执行，因此过程程序块被认为是声明的另一种形式。

程序块嵌套及变量的作用域：引言

有些程序块把其他程序块整个包含进来，如下面的例子所示。

例 1

```
Start: DO;
       DECLARE (A, B, C, D, E, F, G, H, L) BYTE;
       A=17;
       C=B+D;
       middle: DO;
               DECLARE (J, K) BYTE;
               E=F+G;
               H=J+K+A;
       END  middle;
       Last: L=H+C;
END start;
```

例 2

```
start: DO;
         DECLARE (A, B, C, D, E, F, G, L) BYTE;
         A=17;
         C=B+D;
         middle: DO;
                   DECLARE (H, J, K, L) BYTE;
                   E=F+G;
```

```
        H=J+K+A;
    END middle;
    last: B=H+C; /*This is an error since H undeclared at outer
                   levle*/
END start;
```

MIDDLE 程序块完全包含在 START 程序块内；MIDDLE 叫做“被嵌套于 START 程序块内”。

START 程序块叫做“外层程序块”。外层用来指 START 块内而不在 MIDDLE 块内的各语句。例如，A=，C=，及 B=开始的语句都在例 1 及 2 中表示程序块的外层。

PL/M-51 允许每个程序块独立于其他程序块，任何在外层声明过的名字能够在一个被嵌套的程序块内重新声明，使它具有新的意义和数值。如果在外层声明过的名字未被重新声明，它们便保持原有位置及内容。

因此在 MIDDLE 块内的 A 仍然为 17。在一个被嵌套了的程序块内所声明的变量，仅在该程序块执行中有它局部的含义，进入该程序块外边的语句，这些变量便失去它们局部的意义。

因此，如果 H 仅在 MIDDLE 内声明，如例 2 所示，它的值在标号为“last”的语句内是未知的，该语句将是无效的。如果 H 也是 START 内声明，last 内所用的值将具有它外层的意义，而与 MIDDLE 内的结果无关。它们仅在 START 中声明而不在 MIDDLE 中声明的条件下才是相同的，如例 1 一样。

当写一个程序块时，不必担心相同的标识符是否已经用于别的程序块。

嵌套程序块、外层和内层的概念是编制 PL/M-51 程序的关键。例如，一个程序模块必须遵守这样的规则：只有一个模块的最外层有可执行的语句。那个模块叫做主模块（有时叫做主程序）。所有其它模块的最外层只含定义过程程序块及其他的声明，见下面各节中的讨论。

本书所讨论的多数规则，包括刚才涉及的内容，与为第五个名字定义和保存明确的含义、地址以及数值有关系。这种一致性必须在每一个程序块内以及各分程序块之间通信时成立。

1.5 可执行的语句

下面是全部 PL/M-51 可执行语句的列表及讨论的章次：

赋值语句	第五章
GOTO 语句	第七章
IF 语句	第七章
简单 DO 语句	第七章
重复 DO 语句	第七章
DO WHILE 语句	第七章
DO CASE 语句	第七章
END 语句	第十章
CALL 语句	第十章
RETURN 语句	第十章

下面的段落给出某些可执行语句的简单叙述，或许有助于更加了解 PL/M-51 以及对后续章节中详细叙述有所帮助。

赋值语句　我们已经介绍过赋值语句，它是PL/M-51的基础。尽管它的形式相当简单，但在一个赋值语句内的表达式可能是相当复杂，而且有相当数量的运算，这在第五章内将会见到。

赋值语句的最简单形式是：

identifier＝expression；

其中：

identifier（标识符）是一个变量名。

对 expression（表达式）求解，其结果为变量的值。在第五章内给出这种形式的变量。

IF 语句　下面是 IF 语句的一个例子：

```
IF  WEIGHT〈MINWT  THEN
  COUNT＝COUNT＋1；
  ELSE
  COUNT＝0；
```

这条语句被分成四个容易分辨的语句行，从而更加好读。如第二章要解释的那样，空白（空格，横表，回车，注释和换行）可以自由地插到语句元素之间而不会改变其含义。

WEIGHT，MINWT，和 COUNT 是先前声明过的变量。该 IF 语句例子有三部分：

- IF 部分包含有保留字 IF 及条件 WEIGHT〈MINWT
- THEN 部分包含有保留字 THEN 及语句 COUNT＝COUNT＋1
- ELSE 部分包含有保留字 ELSE 及语句 COUNT＝0

如果IF语句中的IF部分的条件成立，THEN部分中的语句将被执行。否则，ELSE部分中的语句将被执行。

在上例中，如果 WEIGHT 的值比 MINWTA 的值小，则 COUNT 的值加 1。否则，值 0 将赋于 COUNT。

IF 语句的 ELSE 部分是任选的。第七章中有对 IF 语句完整的叙述。

DO 及 END 语句　DO 及 END 语句用于构造 DO 程序块。一个 DO 程序块起始于一个语句并用一个对应的 END 语句结束。

PL/M-51 有四种 DO 语句，用来构造四种 DO 程序块。

简单 DO 程序块用一个简单的 DO 开头，并且（像所有的 DO 程序块一样）可以用在能使用单一语句的任何地方。下面是一例子，表示一个简单 DO 程序块代替 IF 语句中 THEN 部分的一个单语句：

```
IF  TMP〉＝4  THEN
    DO；
        INCR＝INCR＊2；
        COUNT＝COUNT＋INCR；
    END；
ELSE
```

```
    COUNT=0
```

这个例子允许在条件为真时执行两个或多个可执行语句。

一个重复 DO 语句引入一个重复 DO 程序块，使该程序块内的可执行语句被反复执行。下面是重复 DO 语句的例子。

```
DO  J=0  TO  9;
  VTCTOR (J) =O;
END
```

此处

J 是预先声明为 BYTE 或 WORD 变量（在第三～五章内详细讨论）。

VTCTOR 是至少有 10 个元素的数组。

该赋值语句被执行 10 次，J 值从 0 开始每次加 1 直到 9。由于 J 是用来作为在赋值语句中 VECTOR 的下标，因此这个重复 DO 程序块对从元素 0 至元素 9 的各个 VTCTOR 元素赋以 0 值。

DO WHILE 语句包含一个条件（像 IF 语句中的 IF 部分条件一样）。如果条件为真，该程序块内的可执行语句被反复妨行。

在下面的例子中，DO WHILE 程序块用来比较数组（TABLE）的各元素，直到一个元素值大于 LEVEL 变量的值。

```
I=0
DO WHILE TABLE (I) <=LEVEL;
    I=I+1
END;
```

TABLE是预先声明了的数组，LEVEL和 I 是已声明的变量。I 初值为 0，而后用作TABLE的下标。I 每次加 1，执行DO WHILE语句时用不同的TABLE元素与LEVEL比较。当找到一个大于LEVEL的元素时，DO WHILE语句中的条件不再成立，该程序块便不再重复，于是程序跳转到END语句后面的语句。此时，I 的值是TABLE中不大于LEVEL的第一个元素的下标。

DO CASE 程序块由 DO CASE 语句引入，使用表达式去选择要执行的语句。在下面的例子中，假设 DO CASE 语句中的表达式 TST-1 是 0 至 3 中的一个数值。

```
DO CASE TST-1
    RED=0;
    BLUE=0;
    CREEN=0;
    GREY=0;
END;
```

如果表达式为 0，执行第一个赋值语句，0 赋于 RED。如果表达式为 1，执行第二个赋值语句，0 赋于 BLUE。表达式为 2 或 3 将对 GREEN 或 GREY 依次赋于 0。

1.6 内部过程

PL/M-51 有许多内部过程。这些过程提供如移位、循环、数据类型变换以及测试与设置等功能。第十一章叙述内部过程。

1.7 表达式

正如前面提及，PL/M-51 表达式由操作数和操作符组成，类似于通常的代数表达式。

操作数包括常数（如378或105）或变量。操作符包括＋（加）、－（减）、*（乘）/（除）、MOD（算术求模）。

像代数表达式一样，PL/M-51 表达式中的元素可以用括号组合。

1.8 程序开发过程

PL/M-51 编译程序及运行时的库是微计算机系统的开发手段中不可缺少的一部分。

软件开发过程的步骤如下：

1. 完整地定义问题。
2. 依次规划出硬件和软件的设计方案。一旦这一步完成了，便可以开始设计硬件。
3. 设计系统软件。这个步骤由若干环节组成，包括把任务分解成模块，选择编程语言以及算法。
4. 使用文本编辑程序编制程序。
5. 用 PL/M-51 编译程序翻译 PL/M 应用程序。
6. 使用文本编辑程序修改编译时出现的错误；而后重新编译。
7. 把编译后的目标模块与 PLM51.LIB 连接并用 RL51 定位成绝对目标代码。
8. 用开发系统或其他工具测试程序结果、并重复步骤 6 至 8 直到程序能正确地执行。

第二章　PL/M-51 编程基础

编写 PL/M-51 程序的格式是自由的，这意味着语句放在输入行的什么地方是无所谓的，空格能够自由地插在程序元素之间。

2.1 PL/M-51 字符集

PL/M-51 中用的字符集是 ASCII 字符集的子集。如下：

ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz

0123456789

以及特殊字符

= . / ( ) + − , ' : ; $ — < >

和空白或空格，加上横表、回车和换行符。

字符串在 2.4 节中讨论，注释在 2.5 节中讨论，除此之外这一节中的规则适用于 PL/M-51 程序当中的任何场合。

如果一个 PL/M-51 程序包含了不在上述集合中的任何字符，编译程序按错误处理。

除了在串内容中有所不同外，大小写字母是无区别的。例如，xyz 和 XYZ 可以互相替代。在本手册内，全部 PL/M-51 代码以大写字母出现，以便与解释的文字相区别。

除了在串内容中有所不同外，空白是不区分的。编译程序把任何断开的空白序列当做一个单一的空白。

特殊字符及其组合在 PL/M-51 程序中具有特定的意义。

表 2-1 提供了特殊字符及其组合的词汇及用法。

2.2 标识符及保留字

标识符用来定义变量、过程、符号常数以及语句标号。标识符的长度可以多达 31 个字符。第一个字符必须是字母，其余可以为字母、数字或下横杠（-）或货币号（$）。

编译程序忽略嵌入的货币号，用它可以自 由地改进标识符或常数的可读性（但 $ 不可用作第一个字符）。一个包含货币号的标识符或常数与删除了货币号的同样的标识符完全相当。

有效标识符的例子是：

INPUT—COUNT

X

GAMM

LONGIDENTIFIERNUMBER3

LONG $ $ $ IDETIFER $ $ $ NUMBER $ $ $ $

INPUT $ COUNT

INPUTCOUNT

表 2—1　PL/M-51 特殊字符

符　号	号　　称	用　　途
=	等号	两种不同的用法
		（1）赋值操作符
		（2）关系测试操作符
.	点号	两种不同的用法
		（1）构造成分的限定条件
		（2）寻址操作符
/	斜杠	除号
/*		注释开始分隔符
*/		注释结束分隔符
(	左括号	表项，下标及某些表达式的左分隔符
)	右括号	表项，下标及某表达式的右分隔符
+	加	加号或一元加操作符
-	减	减或一元减操作符
'	撇号	串分隔符
*	星号	乘操作符，隐维数说明符
<	小于	关系测试操作符
>	大于	关系测试操作符
<=	小于或等于	关系测试操作符
=>	大于或等于	关系测试操作符
<>	不等	关系测试操作符
:	冒号	标号分隔符
;	分号	语句分隔符
,	逗号	列表元素分隔符
_	下横杠	在标识符中有效字符
$	货币符	在标识符中有无效字符

最后这两个标识符（在编译程序来看）是相同的，因此可以互换，但不同于第一个例子。

某些保留字不可用作标识符，他们是 PL/M-51 语言起作用的部分。附录 C 列出这些保留字。

PL/M-51 也有一组叫做内部过程的预先声明了的标识符。可以按自己的意愿声明这些名字，但是，具有相同名字的内部过程成为不可访问的了。附录 D 列出这些标识符。

2.3 记号，分隔符和空白的用法

正如英语句子是由字组成一样，PL/M-51 语句是由记号组成。第一个记号属于下面分类之一：

- 标识符
- 保留字
- 简单定界符（全部特殊字符，除下横杠和货币号外，都是简单定界符）
- 复合定界符——下面是两个特殊字符的几种复合：