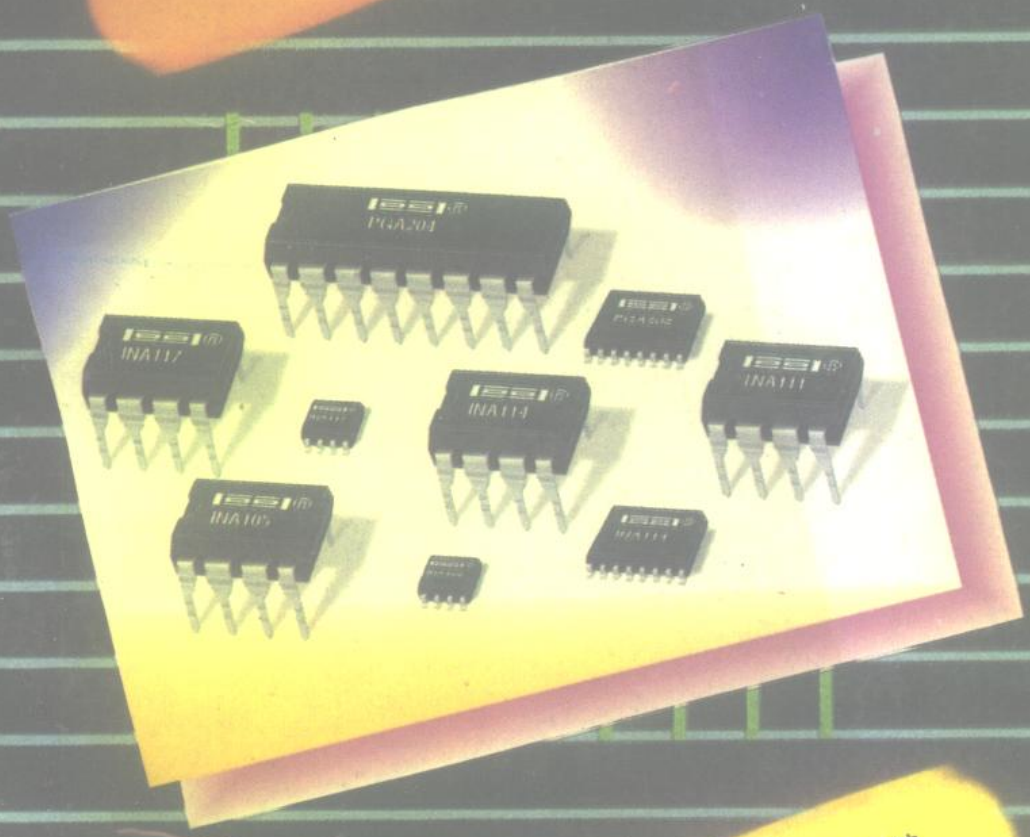
单片机高级程序设计语言

PL/M－51与PL/M－96



陈力钧 刘英 编著

西安电子科技大学出版社

单片机高级程序设计语言

PL/M－51 与 PL/M－96

陈力钧　　刘　英　编著

西安电子科技大学出版社

1994

(陕)新登字 010 号

内 容 提 要

PL/M 语言是国内外使用十分广泛的适用于开发微处理器和微控制器的高级程序设计语言，包括 PL/M－51、PL/M－80、PL/M－96 和 PL/M－86 等，具有编程速度快、模块性好、代码转换率高、易于移植等优点。本书分上、下两篇，系统介绍 PL/M－51 和 PL/M－96 语言的数据类型、程序结构、库函数以及编译操作等内容。另外，还介绍了一些有关的实用软件工具，包括连接程序（RL51 和 RL96）、库管理程序（LIB51 和 LIB96）和格式转换程序 OH 等。书中附有大量典型实例程序，读者可从中获得一些典型的程序设计方法和技巧。

本书可作为单片机技术培训教材，也可作为从事工业控制、智能仪表等领域工作的工程技术人员、大专院校师生学习使用 PL/M 语言的综合性参考书。

单片机高级程序设计语言 PL/M－51 与 PL/M－96

陈力钧　刘　英　编著

责任编辑　霍小齐　云立实

西安电子科技大学出版社出版发行

陕西省军区印刷分厂印刷

陕西省新华书店发行　各地新华书店经售

开本 787×1092　1/16　印张 21　　字数 4934 字

1994 年 9 月第 1 版　1994 年 9 月第 1 次印刷　印数 1－5000 册

ISBN　7－5606－0344－0/TP・0130　　定价：15.50 元

前　言

近年来单片机已广泛应用于自动控制、智能仪表、数据采集和处理、家用电器等多个方面，作为开发单片机程序的主流语言工具 PL/M－51 和 PL/M－96 得到了广大工程技术人员的广泛重视，但是目前介绍 PL/M 语言的资料比较缺乏，难以满足广大读者的需要。因此，我们编写本书，希望能在这方面尽一份微薄之力。

PL/M 语言是 Intel 公司为开发微处理器和微控制器而专门设计的高级程序语言，包括 PL/M－80、PL/M－86、PL/M－96、PL/M－51 等，在计算机系统开发及应用领域使用十分广泛。国外流行的微处理器开发系统一般都配有 PL/M 工具软件供程序设计人员使用，目前国内几种主要的单片机开发系统如 MDS－55 系列、ECI 系列及 DVCC 系列等也都能支持 PL/M 语言程序开发，使用 PL/M 语言开发单片机应用程序的条件已经十分成熟。

与汇编语言相比，PL/M 语言具有程序模块性好，易于编程和维护，程序开发周期短，代码转换率高等优点，因而成为开发单片机程序的得力工具。

全书分上下两篇，分别介绍 PL/M－51 和 PL/M－96 语言的数据类型、程序结构、库程序以及编译操作等内容。另外，还介绍了一些实用程序，包括连接定位程序 RL51 和 RL96、库管理程序 LIB51 和 LIB96、浮点库 FPAL96 和代码转换程序 OH 等。书中附有大量典型应用实例，相信读者会从中得到使用 PL/M 开发单片机的奥妙和真谛。

本书由陈力钧主编并编写 PL/M－96 部分，刘英编写 PL/M－51 部分。

本书的编写得到了西北工业大学董大群教授的大力支持和精心指导，在此特表示感谢。西安电子科技大学的云立实和霍小齐两位老师对本书的出版自始至终给予了热情的扶持和关怀，西北工业大学的王丁、殷雷和鲜敏同志协助我们做了许多工作，在此一并表示感谢。

由于水平有限，加上时间仓促，不足之处在所难免，恳请读者和同行不吝赐教。

作　者

1994 年 5 月于西安

目　录

上篇　PL/M－51 程序设计语言

下篇 PL/M－96 程序设计语言

上篇　PL/M－51 程序设计语言

第一章 概 论

1.1 PL/M - 51 语言

PL/M - 51 语言是 Intel 公司为开发 MCS - 51 单片机的系统软件和应用软件而专门设计的一种高级程序设计语言。

PL/M - 51 编译器是一种软件工具，它把 PL/M - 51 源程序翻译成可重定位的目标代码，这些目标代码可以与其他的 PL/M、汇编语言或别的高级语言程序的浮动目标模块相连接，最终得到可在 MCS - 51 环境下运行的代码。

1.1.1 PL/M - 51 语言的特点

作为开发 MCS - 51 单片机的高级语言工具，PL/M - 51 有许多十分突出的优点：

1. 与汇编语言相比，PL/M 语言更接近人类的思维习惯，因而更易于编写程序。

2. 语句的功能强。完成相同功能的程序，采用 PL/M 语言编写的程序远比汇编程序简洁，所以能大大减少程序出错的可能性，并能显著地缩短软件的开发周期。

3. 代码转换率高。虽然在一些局部问题上 PL/M 程序的代码生成情况不如汇编语言，但是在处理比较复杂的问题时，前者的代码量并不明显地比后者大，甚至优于后者。

4. 程序的可读性、可维护性好。相应地，对于一个汇编语言程序员来说，阅读一段有十几 K 代码量的汇编程序绝对不是一件轻松的事情，更不要说调试和维护了。

5. PL/M - 51 支持模块化程序设计，允许将一个复杂的程序分解成多个小的功能模块，每个模块单独编辑、编译和调试，最后再把这些模块连接起来，形成一个完整的应用程序。与单块程序相比，这样做更容易编程和调试，并且可以实现程序共享，即一个程序的模块可以被其它程序引用，同时也便于用户建立自己的函数库。

6. 支持混合语言编程。程序员可以使用 PL/M - 51、宏汇编语言或其它的高级语言来编写程序的各个模块，然后分别由各自的编译器进行编译、调试，最后用连接程序连接生成一个完整的绝对地址目标文件。混合使用多种语言编程可以发挥各种语言工具的长处，例如用汇编语言来编写程序中实时性要求很高的部分，而用 PL/M - 51 来编写主程序，以发挥该语言流程控制功能强的优点。此外，混合语言编程可以使 PL/M 程序能较好地利用各种软件资源，例如在 PL/M 程序中调用一些供汇编程序使用的浮点库。

7. 通用性好，PL/M 程序在 Intel 的微处器和微控制器之间是可移植的。

8. PL/M - 51 不仅支持 Intel 公司的 MCS - 51 系列单片机，还可用于开发 Philips 公司的 80C31 系列，包括 80C552、80CL410 等。

9. PL/M - 51 提供了丰富的变量类型和数据类型，可以使用布尔变量、字符串、指针、数组和结构等。

10. PL/M - 51 程序能十分方便地进行 I/O 操作、内存访问等与硬件有关的操作。

11. 能直接编写中断服务子程序，用以响应 MCS - 51 产生的各种中断。

12. 具有条件转移、循环、无条件转移等多种流程控制功能，适于编写流程复杂的控制程序。

13. PL/M－51 编译器提供了丰富的编译控制项，程序员可视需要灵活地选择编译方式。

14. 不要求程序员了解 MCS－51 指令集和目标处理器的每一个细节，但是程序员应该知道目标应用系统的存储结构和地址分配。

1.1.2　两种 PL/M－51 语句

PL/M－51 语句分为说明语句和可执行语句两类。其中说明语句包括 DECLARE 语句和 PROCEDURE 语句，前者用来定义变量、指针、数组、结构和标号等，后者用来定义一个过程，这里所说的过程类似于其它高级语言的子程序或函数。可执行语句包括了除 DECLARE 和 PROCEDURE 之外的所有语句：赋值语句、IF 语句、简单 DO 语句、重复 DO 语句、DO WHILE 语句、DO CASE 语句、END 语句、CALL 语句、RETURN 语句、ENABLE 语句和 DISABLE 语句。

下面是一条简单的 DECLARE 语句：

```
DECLARE X BYTE;
```

这条语句引入了一个标识符 X，并把它与一个字节的存储单元的内容联系起来。程序员不必知道这个字节的具体位置，他只需通过标识符 X 来对该字节进行各种操作。

下面的语句是一条可执行语句：

```
X=Y+2;
```

其中标识符 X 和 Y 预先已作说明。编译器根据上面的语句将产生如下的机器代码：

1. 取与标识符 Y 相对应的存储单元的内容；
2. 将该值加 2；
3. 把这个新值传给与标识符 X 相应的存储单元。

在一般情况下，程序员不必考虑这些存储单元的位置，编译器会自动地给变量 X、Y 分配存储单元，并生成用以完成求和、赋值等操作的机器代码。

1.1.3　PL/M－51 程序的结构

PL/M－51 程序由一个或多个模块组成，每个模块又是由一个或多个程序块构成的。

程序块是模块化程序设计的基本单位，包括 DO 程序块和 PROCEDURE 程序块，它们由 DO 语句或 PROCEDURE 语句开始，中间是其它语句，最后以 END 语句结束。程序块可以互相嵌套，以完成复杂的程序功能。

模块是不嵌套在其它程序块的带标号的简单 DO 程序块，它由 DO 语句开始，中间是其它语句，并以 END 语句结束。一个模块单独地存放在一个磁盘文件中，并单独进行编译，编译通过后再由连接定位程序把程序的所有模块连接起来，生成一个完整的绝对地址目标文件。

1.2　PL/M－51 程序的开发过程

本节简要地介绍使用编译器 PLM51.EXE、连接定位器 RL51.EXE 和代码转换程序 OH.

EXE 来开发 MCS－51 单片机软件的过程：

1. 完整地定义问题。

2. 依次规划出硬件及软件的设计方案，一旦完成方案，便可以开始设计硬件。

3. 设计软件。这一步骤包括多个环节：把任务分解成模块、选择算法、选择编程语言等。

4. 使用文本编辑器编写程序。

5. 文件过滤。用过滤程序 FILTER. EXE 对文件进行过滤处理，使之符合 Intel 公司的文本文件格式。

6. 用 PLM51 编译各程序模块，如有错误回到 4 修改程序，然后重新过滤和编译。编译器生成两个输出文件：浮动目标模块 *. OBJ 和列表文件 *. LST。

7. 浮动目标模块连接。连接定位程序 RL51. EXE 把输入的浮动目标文件 *. OBJ 和库文件 *. LIB连接起来，输出用 TO 指定生成一个绝对地址目标文件 *. ABS，同时产生一个列表文件 *. M51。

8. 用 OH. EXE 进行十六进制格式转化，将 *. ABS 转换成 *. HEX 文件，该文件可供单片机开发系统调试使用或用以固化 EPROM。

9. 用开发系统或其它工具调试程序结果，并重复 4 至 8，直到程序能正确运行。

10. 用库管理程序 LIB51. EXE 把有关浮动目标模块装入库中，使之能为其它程序共享。

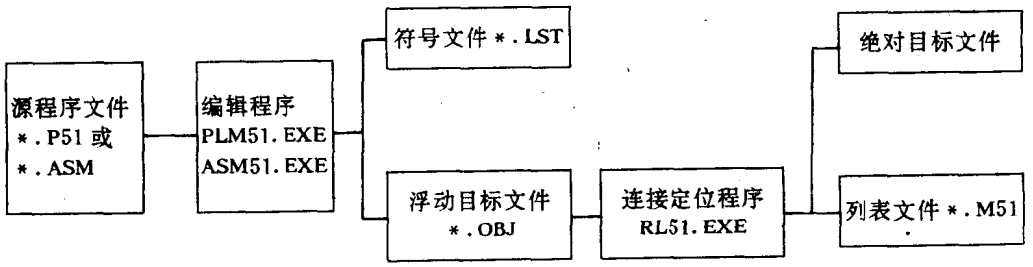


图 1.1 程序编译连接过程的输出

第二章　PL/M－51 程序基础

2.1　语言基本成分

2.1.1　字符集

在 PL/M－51 语言中，合法的字符包括以下几类：

英文字母(除作为字符串常数外，程序大小写不加区分)：

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

数字：0 1 2 3 4 5 6 7 8 9

算术运算符：+　－　*　/　MOD

关系运算符：<　>　=　<=　>=　<>

逻辑运算符：NOT　AND　OR　XOR

分界符：.　(　)　,　:　;　'

特殊字符：　$　_　/*　*/

空字符：　Space(空格)　Tab(制表符)　Carriage－return(回车)　line－feed(换行)

如果一个 PL/M－51 程序含有任何不属于上述集合中的字符，编译程序将按出错处理。

英文字母在程序中采用大小写完全相同，例如 xyz 和 XYZ 可以互相替代。但是，若它们出现在字符串中，则对应于不同的 ASCII 码，例如，'A'的 ASCII 码是 41H，而'a'的 ASCII 码是 61H。

PL/M 程序一般采用大写字母书写。注释大小写字母混和使用。

编译程序把任何断开的空白序列当作一个单一的空白。但对于字符串中的空白不作这样处理。另外，由于编译程序忽略程序中的美元符($)，因而可以在源程序中插入一些美元符($)，以改善程序的可读性。

2.1.2　标识符和保留字

和其它高级语言一样，用来标识变量名、符号常量名、数组名、过程名、文件名的有效字符序列称为标识符(Identifier)。简单地说，标识符就是一个名字。

PL/M－51 语言规定标识符只能由字母、数字、下划线及美元符 $ 四种字符组成，且第一个字符必须是字母。下面是一些合法的标识符：

X，A3，PORT_C，X_A，X$A，XA

在编译程序看来 X$A 和 XA 是相同的，二者可以互换，但 X_A 则表示的是另一个变量，与它们没有任何关系。

保留字是 PL/M－51 语言的一个组成部分，它们有确定的意义，编译程序根据其执行

一定的功能。保留字包括语句标识符(如DECLARE,GOTO,CALL等)、属性(如PUBLIC,EXTERNAL,BASED等)、分割字键(如THEN,ELSE,BY等)、运算符(如OR,AND,MOD等)等等。保留字不能用作标识符

还有一类标识符称为预说明的标识符,它们是PL/M语言提供的内部过程。与保留字不同的是,用户可以用说明语句重新说明这些标识符,但在该说明的作用域(即标识符有效引用的范围)内相应的内部过程不再有效。

保留字及预说明的标识符列表详见附录I.3及I.4。

2.1.3 符号、分界符和空符号

PL/M语句是由符号组成的,每个符号属于下列种类之一:

1. 标识符
2. 保留字
3. 简单分界符:字符集中除英文字母、数字、下划线和美元符$以外的单个PL/M字符:+ - * / < > = . ( ) , : ; '
4. 复合分界符:特定两个字符的某种组合,包括<> <= >= /* */
5. 纯值常数
6. 字符串常数

例如在一个赋值语句中

```
Y=X2*(X1-2)/X3;
```

这里X1、X2、X3和Y是标识符,2是常数,其它符号是简单分界符。

在一个语句中,标识符、保留字和常数必须相互分离,如果在两个标识符、保留字或常数之间没有分界符,则应加入空格作为分界符。如:

```
DECLAREABYTE;
```

应加入空格,即为

```
DECLARE A BYTE;
```

注释也可以当作分界符使用,如:

```
DECLARE A/*RESULT*/BYTE;
```

空符号也可以自由地插入任何符号前、后,而不会改变原PL/M－51语句的意义。如:

```
Y=X2*(X1-2)/X3;
```

等价于

```
Y = X2 * (X1-2)/ X3;
```

2.1.4 注释

在PL/M语言程序正文中可以插入注释,以改善程序的可读性。注释的左、右两端分别使用/*和*/来定界,编译程序将根据该定界符忽略注释部分,而不作为程序正文处理。

注释可以使用任何可打印的ASCII字符以及空格、回车、换行及制表符等。

注释不能出现在字符串常数中,否则会被编译程序认为是字符串的一部分而不予忽略。例如,'AG/*P*/A'将被翻译为41 47 2F 2A 50 2A 2F 41,而不会认作'AGA'(41 47 41)。除此之外,任何空格处都可插入注释。