PL/M
程序设计语言及其应用

袁涛　孙腾谌　编著

清　华　大　学　出　版　社

PL / M 程序设计语言及其应用

袁 涛　　孙腾谌

清 华 大 学 出 版 社

内容简介

PL／M 语言是目前流行的一种计算机高级语言，它主要用于单片微机和一般微机开发，尤其是 16 位单片微机开发的得力工具。本书详细介绍了 PL／M 语言，包括变量类型、说明语句、过程说明、可执行语句、作用域和结构化程序设计、内部过程、浮点运算库及有关过程、与硬件有关的操作、PL／M 语言和汇编语言及 C 语言的交叉使用。书中不仅叙述了 PL／M 语言本身，还详细叙述了如何进行编译和连接，包括各种编译、连接控制项的使用。书中还专门给出了包括编译、连接操作在内的完整程序实例。本书以 PL／M－96 为主，同时叙述了 PL／M－86 和 PL／M－51。

本书通俗易懂，很适合初学者使用。本书可作为高等院校和培训班的教材或参考书，也适合从事单片微机和一般微机开发的科技人员和自学读者使用。

PL／M 程序设计语言及其应用

袁 涛　孙腾谌

☆

清华大学出版社出版

北京　清华园

世界知识印刷厂印装

新华书店总店科技发行所发行

☆

开本：787×1092 1／16　印张：12.5　字数：320 千字

1990 年 8 月第 1 版　1990 年 8 月第 1 次印刷

印数：0001—4000

ISBN 7-302-00737-3／TP · 245

定价：7.90 元

前　言

PL／M 语言是目前流行的一种计算机高级语言，它主要用于单片计算机和一般微型计算机。具体有 PL／M-96、PL／M-86、PL／M-51 等。它们的区别主要是编译、连接程序不同，从而生成不同的机器代码。例如，PL／M-96 编译、连接程序所生成的是 MCS-96 系列（含 8098）16 位单片机的机器代码，PL／M-86 编译、连接程序所生成的是 8086／8088 机器代码，PL／M-51 编译、连接程序所生成的是 MCS-51 系列单片机的机器代码。PL／M 语言编译、连接程序可以直接在普通的 IBM PC／XT／AT／286／386 及其兼容机上运行。

PL／M 语言程序设计快、可读性好、可靠性高、代码转换质量高。一般情况下，由 PL／M 语言程序生成的机器代码，不论是代码长度，还是程序运行速度都优于或不亚于人工直接用汇编语言编写的程序质量。完成同样的任务，使用 PL／M 高级语言比用汇编语言可提高速度 5～10 倍，在调试阶段更容易体会这一点。PL／M 语言的特点是同时兼有高级语言和汇编语言的优点，还能象汇编语言那样直接利用 CPU 的硬件特性进行程序设计。PL／M 语言简单、易学，是一种结构化程序设计语言。一个完整的程序可由一个或多个模块（单独编译的程序块）组成。PL／M 语言目标模块还可以同汇编语言、C 语言目标模块连接组成一个完整程序。

目前在单片机应用领域，尤其是 16 位单片机应用领域，PL／M 语言越来越受到人们的重视。市场上大量出现的 Intel 公司 MCS-96 系列中的 8098 单片机，功能强（具有 16 位 CPU、定时器、22 微秒带采样保持器的 10 位 4 路 A／D 转换器、高速输入 HSI、高速输出 HSO、可调脉宽输出 PWM、串行通讯口等）、速度快（除了使用 16 位运算外，还解决了 8 位机的"瓶颈"问题，片内 232 个寄存器都可作为累加器使用，大大减少了数据传送指令的使用）、价格低（明显低于单独器件组合后的价格），使人们将单片机应用的重点由 8 位机转向了 16 位机。PL／M 语言是开发 16 位单片机的得力工具。

书中以 PL／M-96 为主进行叙述。为了避免重复，对 PL／M-86 和 PL／M-51 仅叙述它们与 PL／M-96 不同的部分，相同部分可根据提示阅读 PL／M-96 中的有关内容就可以了。PL／M-96、PL／M-86 和 PL／M-51 之所以有一些不同之处，主要是由于硬件功能和操作的不同引起的。

书中不仅叙述了语言本身，还详细叙述了如何进行编译、连接，包括各种编译、连接控制项的使用，这对如何用好 PL／M 语言是很关键的内容。

本书适用于 PL／M 语言的初学者。对已掌握一定基础的人，可以先阅读附录，然后根据自己掌握的程度，选择阅读书中内容。如果已掌握了 PL／M-86，在学习 PL／M-96 时，可以先阅读 PL／M-86 一章，了解 PL／M-96 与 PL／M-86 区别之后，仅需阅读部分章节。

本书第 11 章由孙腾谌执笔，其它各章由袁涛执笔。

在成书过程中，得到了北京东方计算机研究所的大力支持，提供了 ECI UD98 高级语言交叉窗口调试器，为书中程序实例的调试节省了很多时间。成书过程中，魏峰同志协助做

了许多工作。本书由清华大学许道荣教授审阅。在此，向以上单位和同志表示衷心感谢。

由于编者水平有限，且时间仓促，书中难免存在不少缺点和错误，恳请读者批评指正。

编　者

1990.2

于清华大学自动化系

目　录

第一章　PL / M 语言基本知识 …… 1
1.1 概述 …… 1
1.1.1 PL / M 语言及其特点 …… 1
1.1.2 使用 PL / M 语言的必要性 …… 2
1.1.3 书中的约定 …… 3
1.2 字符集、标识符、保留字和预说明的标识符 …… 3
1.2.1 字符集 …… 3
1.2.2 标识符、保留字和预说明的标识符 …… 4
1.2.3 符号、分界符和空符号的作用 …… 4
1.2.4 注释 …… 5
1.3 常数 …… 5
1.3.1 纯数常数 …… 5
1.3.2 浮点常数 …… 6
1.3.3 字符串 …… 6
1.4 简单说明语句 …… 7
1.5 变量、变量类型、数据类型 …… 8
1.5.1 标量变量和变量 …… 8
1.5.2 类型 …… 8
1.5.3 字节(BYTE)、字(WORD)和双字(DWORD)变量 …… 9
1.5.4 整型(INTEGER)、短整型(SHORTINT)和
长整型(LONGINT)变量 …… 9
1.5.5 实型(REAL)变量 …… 10
1.5.6 地址型(ADDRESS)变量 …… 10
1.5.7 "点"运算符(.)和地址引用 …… 10
1.5.8 FAST 和 SLOW 属性 …… 11
1.5.9 隐含类型转换 …… 11
1.6 运算、表达式及规则 …… 13
1.6.1 运算对象 …… 13
1.6.2 表达式 …… 14
1.6.3 算术运算及其表达式 …… 14
1.6.4 关系运算及其表达式 …… 15
1.6.5 逻辑运算及其表达式 …… 16
1.6.6 表达式计算 …… 17
1.6.7 常数表达式计算 …… 18

1.7 数组和结构 …… 22
1.7.1 数组说明 …… 22
1.7.2 下标变量 …… 22
1.7.3 结构 …… 23
1.7.4 结构数组 …… 24
1.7.5 结构内数组 …… 24
1.7.6 结构数组内数组 …… 24
1.7.7 数组的隐含长度说明 …… 25
1.8 对变量(包括数组和结构)的引用 …… 25
1.8.1 完全限定的变量引用 …… 26
1.8.2 非限定的变量引用 …… 26
1.8.3 部分限定的变量引用 …… 26
1.9 有基变量 …… 26
1.9.1 有基变量 …… 26
1.9.2 有基变量和地址引用应用举例 …… 27
1.10 高级说明语句 …… 28
1.10.1 概述 …… 28
1.10.2 连接属性说明(PUBLIC 和 EXTERNAL)——扩展作用域 …… 28
1.10.3 AT 属性说明 …… 30
1.10.4 DATA 赋值 …… 31
1.10.5 语句标号说明 …… 32
1.10.6 文字(LITERALLY)说明及用途 …… 33
1.10.7 组合说明语句 …… 34
1.11 存储的相邻性 …… 34

第二章 PL / M-96 可执行语句 …… 36
2.1 赋值语句 …… 36
2.1.1 赋值语句 …… 36
2.1.2 多次赋值语句 …… 36
2.1.3 内嵌赋值语句 …… 37
2.2 DO 程序块 …… 37
2.2.1 简单 DO 程序块 …… 37
2.2.2 DO WHILE 程序块 …… 38
2.2.3 循环 DO 程序块 …… 39
2.2.4 DO CASE 程序块 …… 41
2.3 条件(IF)语句 …… 42
2.3.1 IF 语句 …… 42
2.3.2 嵌套 IF 语句 …… 43
2.3.3 顺序 IF 语句 …… 44
2.4 语句标号和 GOTO 语句 …… 45

2.5 其它可执行语句 …… 46
2.5.1 调用(CALL)和返回(RETURN)语句 …… 46
2.5.2 空语句(;) …… 46
2.5.3 开中断(ENABLE)和关中断(DISABLE)语句 …… 47

第三章 过程 …… 48
3.1 概述 …… 48
3.2 过程说明 …… 48
3.2.1 参数 …… 49
3.2.2 有类型过程和无类型过程 …… 50
3.2.3 从过程的转出 …… 50
3.2.4 过程体 …… 51
3.3 过程的属性 …… 52
3.3.1 公共(PUBLIC)和外部(EXTERNAL)属性 …… 52
3.3.2 中断和中断属性(INTERRUPT) …… 53
3.3.3 ENABLE(开中断)和 DISABLE(关中断)语句 …… 54
3.3.4 重入性和 REENTRANT(重入)属性 …… 54
3.3.5 INDIRECTLY-CALLABLE(可间接调用)属性 …… 55
3.3.6 INTERRUPT-CALLABLE(可中断调用)属性 …… 55
3.4 过程的调用 …… 55
3.4.1 函数引用 …… 55
3.4.2 CALL 调用 …… 56
3.4.3 间接过程调用 …… 56
3.4.4 调用其它模块中的过程 …… 56

第四章 作用域和结构化程序 …… 58
4.1 结构化程序 …… 58
4.2 程序模块 …… 58
4.2.1 程序的模块化结构 …… 58
4.2.2 程序模块之间的连接 …… 58
4.3 作用域 …… 59
4.3.1 几个基本术语(层和内含) …… 59
4.3.2 作用域 …… 60
4.4 标号作用域和对 GOTO 语句的限制 …… 62
4.4.1 标号的作用域 …… 62
4.4.2 对 GOTO 语句的限制 …… 63

第五章 内部过程和内部变量 …… 65
5.1 获取变量信息的内部过程 …… 65
5.1.1 LENGTH 过程 …… 65

5.1.2 LAST 过程 …… 66
5.1.3 SIZE 过程 …… 66
5.2 类型转换 …… 67
5.2.1 LOW、HIGH 和 DOUBLE 过程 …… 67
5.2.2 SHORT 和 EXTEND 过程 …… 67
5.2.3 SIGNED 和 UNSIGN 过程 …… 69
5.2.4 FLOAT 和 FIX 过程 …… 70
5.2.5 ABS 和 IABS 过程 …… 70
5.3 移位和循环移位过程 …… 71
5.3.1 循环移位过程:ROL 和 ROR …… 71
5.3.2 逻辑移位过程:SHL 和 SHR …… 71
5.3.3 代数移位过程:SAL 和 SAR …… 72
5.4 串处理过程 …… 72
5.4.1 MOVB 和 MOVW 过程 …… 73
5.4.2 CMPB 和 CMPW 过程 …… 73
5.4.3 FINDB 和 FINDW 过程 …… 74
5.4.4 SKIPB 和 SKIPW 过程 …… 74
5.4.5 SETB 和 SETW 过程 …… 75
5.5 位操作过程 …… 75
5.5.1 BITSET 过程 …… 75
5.5.2 BITCLR 过程 …… 76
5.5.3 BITTST 过程 …… 76
5.5.4 BITCPL 过程 …… 76
5.5.5 BITASN 过程 …… 76
5.6 其它内部过程和内部变量 …… 76
5.6.1 MOVE 过程 …… 76
5.6.2 TIME 过程 …… 77
5.6.3 MEMORY 数组 …… 77
5.6.4 STACKPTR 变量 …… 77

第六章 与 MCS-96 硬件有关的 PL / M-96 运算符和内部过程 …… 78
6.1 优化和 MCS-96 硬件标志 …… 78
6.2 PLUS 和 MINUS 运算符 …… 78
6.3 与硬件有关的 PL / M-96 内部过程 …… 79

第七章 浮点运算库及有关过程 …… 80
7.1 实型(REAL)数的表示 …… 80
7.2 REAL 数学部件 …… 81
7.3 REAL 运算中的例外状态 …… 83
7.3.1 不合法操作例外 …… 84

7.3.2 非规格化操作例外 …… 84
7.3.3 被零除例外 …… 84
7.3.4 上溢例外 …… 84
7.3.5 下溢例外 …… 85
7.3.6 精度降低例外 …… 85
7.4 与浮点运算库有关的内部过程 …… 85
7.4.1 初始化(INIT$REAL$MATH$UNIT)过程 …… 85
7.4.2 设置控制字(SET$REAL$MODE)过程 …… 86
7.4.3 获取出错字节(GET$REAL$ERROR)过程 …… 86
7.4.4 保存 REAL 状态(SAVE$REAL$STATUS)过程 …… 86
7.4.5 恢复 REAL 状态(RESTORE$REAL$STATUS)过程 …… 87
7.4.6 浮点运算库中的开平方等过程 …… 87
7.5 如何编写处理实数例外的过程 …… 88
7.6 浮点运算库(FPAL96)连接 …… 90

第八章 PL / M-96 的编译和连接 …… 92
8.1 编译程序控制 …… 92
8.2 目标文件控制 …… 94
8.2.1 优化(OPTIMIZE)控制 …… 94
8.2.2 寄存器覆盖(REGOVERLAY / NOREGOVERLAY)控制 …… 102
8.2.3 FAST(快速)控制 …… 104
8.2.4 建立目标文件(OBJECT / NOOBJECT)控制 …… 105
8.2.5 DEBUG / NODEBUG(调试)控制 …… 106
8.2.6 TYPE / NOTYPE(类型)控制 …… 106
8.3 列表选择和列表内容控制 …… 107
8.3.1 PRINT / NOPRINT(打印输出)控制 …… 107
8.3.2 LIST / NOLIST(源程序列表)控制 …… 107
8.3.3 CODE / NOCODE(目标代码)控制 …… 107
8.3.4 XREF / NOXREF(相互引用列表)控制 …… 108
8.3.5 SYMBOLS / NOSYMBOLS(符号列表)控制 …… 108
8.4 列表格式控制 …… 108
8.4.1 PAGELENGTH(页长)控制 …… 108
8.4.2 PAGEWIDTH(行宽)控制 …… 109
8.4.3 TITLE(标题)控制 …… 109
8.4.4 EJECT(换页)控制 …… 109
8.5 应用实例 …… 109
8.5.1 源程序和汇编代码列表 …… 109
8.5.2 标识符和相互引用列表部分 …… 112
8.5.3 编译概要 …… 113
8.6 嵌入源文件控制 …… 113

8.6.1 嵌入源文件(INCLUDE)控制 …… 113
8.6.2 保存／恢复(SAVE／RESTORE)控制 …… 114
8.7 条件编译控制 …… 114
8.7.1 IF／ELSE／ELSEIF／ENDIF(条件编译)控制 …… 114
8.7.2 SET／RESET(设置条件开关)控制 …… 115
8.7.3 COND／NOCOND(条件列表)控制 …… 117
8.8 用户程序目标模块的连接 …… 118
8.8.1 使用 RL96 的一般格式 …… 118
8.8.2 ROM 控制 …… 119
8.8.3 RAM 控制 …… 119
8.8.4 STACKSIZE 控制 …… 119
8.8.5 其它控制 …… 120
8.9 编译、连接操作实例 …… 120
8.10 PL／M 语言和汇编语言及 C 语言程序接口 …… 121

第九章 程序设计实例 …… 122
9.1 样本程序 1 (分类程序) …… 122
9.2 样本程序 2 (使用过程的分类程序) …… 124
9.3 样本程序 3 (计算一组数据的均值、残差、方差、标准差) …… 126
9.4 样本程序 4 (PL／M 语言程序中调用汇编语言程序) …… 128
9.5 样本程序 5 (A／D 转换、D／A 转换) …… 129
9.5.1 A／D 转换程序 …… 129
9.5.2 D／A 转换(使用高速输出 HSO)程序 …… 130
9.5.3 在中断服务程序中使用 HSO 进行 D／A 转换 …… 130
9.6 样本程序 6 (模块化结构程序) …… 131

第十章 PL／M-86 …… 135
10.1 PL／M-86 独有的内容 …… 135
10.1.1 PL／M-86 独有的运算符和保留字 …… 135
10.1.2 PL／M-86 独有的内部过程 …… 136
10.1.3 PL／M-86 编译程序独有的控制项 …… 140
10.1.4 PL／M-86 与 PL／M-96 的其它不同 …… 145
10.2 PL／M-96 独有内容 …… 146

第十一章 PL／M-51 …… 148
11.1 引言 …… 148
11.2 存储空间及后缀 …… 148
11.2.1 MAIN(片内直接寻址)后缀 …… 148
11.2.2 AUXILIARY(片外数据存储空间)后缀 …… 148
11.2.3 REGISTER(寄存器)后缀 …… 148

11.2.4 IDATA(片内间接寻址)后缀 …… 149
11.2.5 CONSTANT(程序存储空间)后缀 …… 149
11.3 数据类型及基变量 …… 149
11.3.1 位变量 …… 149
11.3.2 基变量 …… 149
11.3.3 字变量 …… 150
11.4 内嵌赋值(PL／M-96 独有) …… 150
11.5 过程和中断 …… 150
11.5.1 有类型过程 …… 150
11.5.2 INTERRUPT(中断)属性 …… 150
11.5.3 REENTRANT(重入)属性(PL／M-96 独有) …… 150
11.5.4 USING 属性 …… 150
11.6 内部过程 …… 151
11.6.1 PL／M-51 和 PL／M-96 共有的内部过程 …… 151
11.6.2 PL／M-51 独有的内部过程 …… 151
11.6.3 PL／M-96 独有的内部过程 …… 151
11.6.4 与 MCS-51 硬件标志有关的过程 …… 152
11.6.5 Intel 实用程序库 UTIL51.LIB …… 153
11.6.6 ECI 浮点运算程序库 FPAL51.LIB …… 157
11.7 编译控制项 …… 157
11.7.1 PL／M-51 和 PL／M-96 共有的编译控制项 …… 158
11.7.2 PL／M-96 独有的编译控制项 …… 159
11.7.3 PL／M-51 独有的编译控制项 …… 159
11.8 连接定位控制项 …… 160
11.8.1 列表控制项 …… 160
11.8.2 连接控制项 …… 162
11.8.3 定位控制项 …… 163
11.8.4 结构控制 …… 164
11.8.5 覆盖控制 …… 164
11.9 PL／M-51 与 ASM-51 连接 …… 165
11.9.1 调用顺序 …… 165
11.9.2 过程的结尾 …… 166
11.9.3 从有类型过程回送的值 …… 167

附录 …… 168
附录 A PL／M-96 出错信息 …… 168
A.1 PL／M-96 源程序错误 …… 168
A.2 命令错误 …… 174
A.3 输入／输出错误 …… 175
A.4 内存不足错误 …… 175

A.5 编译程序故障错误 …… 175
附录 B PL／M 特殊字符 …… 175
附录 C 程序限制 …… 176
附录 D PL／M 语言保留字 …… 176
D.1 PL／M-96 保留字 …… 177
D.2 PL／M-86 保留字 …… 177
D.3 PL／M-51 保留字 …… 177
附录 E PL／M 语言预说明的标识符 …… 178
E.1 PL／M-96 预说明的标识符 …… 178
E.2 PL／M-86 预说明的标识符 …… 178
E.3 PL／M-51 预说明的标识符 …… 179
附录 F MCS-96 I／O 寄存器符号名 …… 179
附录 G ASCII 字符表 …… 180
附录 H MCS-96 系列汇编语言指令表 …… 181
H.1 指令系统简表 …… 181
H.2 指令操作码和执行时间 …… 184

参考文献 …… 188

第一章　PL / M 语言基本知识

1.1　概　　述

1.1.1　PL / M 语言及其特点

PL / M 语言是一种高级语言，它由美国 Intel 公司设计，主要用于 Intel 公司生产的单片机和微处理器系统软件和应用软件的开发。作为高级语言，它更接近和体现人的设计思想。PL / M 语言不仅具有一般高级语言的特点，而且还能象汇编语言那样直接利用 CPU 的硬件特性进行程序设计。因而，与其它高级语言相比，功能多，用途更广泛，尤其在 16 位单片机应用领域更受到人们的普遍重视。

PL / M 语言的特点主要体现在以下几方面:

1. 简单、易学。PL / M 语言的语句可分为两类：一类是说明语句，用于说明变量和过程（过程类似于其它语言中的子程序，但功能更强)；另一类是可执行语句，如赋值语句。

2. 可读性好。PL / M 语言属于结构化语言，其程序是块式结构，层次清晰，便于理解和阅读。一个完整的程序可由多个单独编译的模块组成，每个模块可由多个程序块组成，程序块可以互相嵌套。

3. 占用内存容量小，运行速度快。若程序长度接近或超过 2K 字节时，其占用内存大小和运行速度甚至可优于一般人直接用汇编语言编写的程序。

4. 可靠性高。用 PL / M 语言编写的程序容易达到正确的目的。

5. 可维护性好。既便于修改和增添，有利于将来扩充和开发，也便于发现程序中的错误。

6. 能够使用与实际问题更接近的数据类型和数据结构。例如，布尔变量，字符，数组，结构，有符号和无符号整数，浮点数，位操作等。

7. 作用域概念和规则，增强了程序编写的灵活性。例如，可以多人共同编制一个程序，而不必担心是否使用了相同的标识符。

8. 程序设计速度快，开发成本低（投入人力、财力少)，周期短，效益高。一条 PL / M 语句相当于多条汇编语句，且程序出错可能性小，必然有前面的优点。

9. PL / M 语言可与汇编及其它高级语言程序连接生成一组目标码。

10. PL / M 语言程序库可由用户增加和删改。

PL / M 语言到机器代码的转换主要由编译程序完成. 经过编译后生成可重定位的机器代码文件，最终由连接程序将多个模块连接定位。可重定位的目标模块（机器代码）可由 PL / M 语言、汇编语言或其它高级语言程序生成。编译程序提供一个输出清单、错误信息、一定数量的控制功能，以帮助进行程序开发和调试。程序编译后可得到列表文件，其扩展名为 .LST，程序连接后可得到有关列表文件，其扩展名为 .M96。列表文件给出了各种有关的信息。PL / M 编译程序主要特点如下:

1. 结构化编程。
2. 兼容性好。可与其它语言程序生成的目标模块相连接。
3. 支持多种数据类型及逻辑、算术、关系等多种运算。还可使用有基变量。
4. 支持多种数据结构，如数组、结构（可有不同类型）、数组和结构的结合。
5. 支持中断管理，使用 INTERRUPT 过程。
6. 丰富的编译控制，这增加了程序编译灵活性。这些控制包括:
   (1) 优化。
   (2) 条件编译。
   (3) 将磁盘上的 PL／M 源文件引入到程序中。
   (4) 符号交叉引用。
   (5) 选择列出 PL／M 语言程序对应的汇编语言程序和机器代码。
7. 有 4 级代码自动优化功能，优化内容为:
   (1) 常数表达式的结合。例如，用左移代替乘 2。
   (2) 机器码优化；除去不必要的分支；重复使用完全相同的代码；除去不可能到达的代码。
   (3) 片内寄存器覆盖。
   (4) 有基变量优化处理。
   (5) 尽可能使用短跳转。
8. 提供了作为 PL／M 语言组成部分之一的内部过程。除了类型转换、串处理、位操作内部过程外，还提供了访问硬件标志的过程。
9. 详细的错误检查。PL／M 编译程序发现程序或结合错误，则提供很详细的错误信息。这对初学者是非常有益的。

1.1.2 使用 PL／M 语言的必要性

PL／M 语言，尤其是用于开发 MCS-96 系列单片机的 PL／M-96，其优越性已在微机开发中明显体现出来了。从 80 年代中期开始，国际上单片机应用就进入了 16 位单片机时代，我国目前也在逐步跟上国际微机应用发展的步伐。在诸多单片机中，功能强、应用广的 Intel 公司的 MCS-96 系列单片机占有一定的优势。16 位单片机的高性能，不仅在于其字长，且硬件功能大大加强（片内 RAM，片内 A／D，片内高速 I／O，片内串行通讯口，片内定时器等），而价格却很低（比使用单独的器件价格要低得多）。16 位单片机的机器指令远远比 8 位机丰富，指令功能强，指令字节数也相应加长，最长的有 7 字节指令，这使我们编写程序时觉得很灵活，但也有不利因素，即在程序较长时，我们很难靠人工来充分利用各种指令的功能，很难使整个程序水平达到最优。对此，PL／M 语言可以很好地予以解决。

当今国际上科技发展速度加快，要想跟上世界科技发展速度，只靠增加工作时间是不行的，必须使用先进的技术和手段。在微机开发应用领域更是如此。对此，使用 PL／M 语言开发微机（尤其是单片机），可大大缩短开发周期。例如，国内某单位，使用 PL／M-96 及相应开发装置，3 个多月时间完成了软件、硬件设计和调试（包含 35K 字节程序及显示、键盘、A／D、输出和 CPU 板等），并送到了现场运行，且学习、掌握 PL／M-96 语言和 8098 单片机也在这 3 个多月时间之内。若使用汇编语言，想如此迅速完成这样的课题，几

乎是不可能的。

使用 PL／M 语言的工作效率高，其生成的机器代码质量也是高水平的。下面仅举一例就可说明这个问题。某单位对 35K 程序使用计算机进行优化，仅寄存器覆盖一项优化，就使程序长度减少 3K。若由人工完成这项工作，除了要花费大量时间外，且很难做得理想，也很容易由此引起其它错误。程序越长，使用 PL／M 语言的优越性越明显。

使用 PL／M 语言可显著增加程序可靠性。这一点与其它高级语言一致。

1.1.3 书中的约定

在 PL／M 语言中，使用英文字母的大写和小写完全相同。

叙述语句时，方括号[ ]括起来的内容是任选项，即省去该项目，其语句的语法仍是正确的。例如，

END [标号]；

这条语句方括号中内容可以省略，这里方括号中内容只是为了增加程序可读性。此语句也可以如下表示：

END；

书内程序中的…表示省略了一个语句序列，以免烦琐。

书中除了第十章和第十一章外，都是以 PL／M-96 为例叙述的，而实际上，除了在第十章和第十一章所指出的区别外，PL／M-96、PL／M-86、PL／M-51 是相同的。对于掌握了其中之一的读者，只要在这两章了解它们的区别之后，阅读有关章节就可以了。实际上，PL／M-96、PL／M-86、PL／M-51 之间的区别主要是由 MCS-96、8086／8088、MCS-51 硬件上的不同决定的。其中的 PL／M-96 也适用于开发 80C196。

1.2 字符集、标识符、保留字和预说明的标识符

1.2.1 字符集

PL／M 语言程序是由字符组成的。合法的字符分为如下几类：

英文字母大小写彼此不加区分（仅在字符串常数中例外），这些字母是：

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)

在 PL／M 语句中，一些字符的名称和作用介绍如下，其它将在后面章节中陆续介绍，也可参看附录。

符号	名　称	作　　用
.	at 符	引用地址
$	美元符号	①控制行首符号。 ②数和标识符中加入 $，是为了改善可读性，并不改变原数和标识符含义。例如，程序并不区分变量 XB 和 X$B，认为它们是同一个变量。
:=	赋值号	内嵌赋值时的赋值号。
_	下划线	可作标识符中的符号。
/*	注释前分界符	每个注释以此分界符开始。
*/	注释后分界符	每个注释以此分界符结束。

除在字符串常数中以外，任何空符号作用相同，任何未断开的一串空符号与单个空符号相同。除在字符串常数中以外，大小写字母意义相同，例如 XAB 与 xab 是可互换的。

如果一个 PL/M 程序包含了不在上述全部字符中的任何字符，编译程序将其视为错误。

1.2.2 标识符、保留字和预说明的标识符

标识符是用来命名变量、过程、宏说明和标号等。一个标识符最长可有 31 个字符。每个标识符第一个字符必须是英文字母，其余字符可以是字母、数字或下划线_。

在标识符中，美元符号 $ 总是被编译程序略去。例如 X$Y 与 XY 被认为是同一个标识符。在标识符中使用美元符 $ 可以改善可读性。例如，GET$REAL$ERROR 的可读性优于 GETREALERROR。在标识符中不允许用美元符 $ 作为第一个字符。下面是合法的标识符的例子：

X	NAME	INPUT_A1	INPUT$A1
Y(2)	WAT	DA212	INPUTA1

其中 INPUT$A1 与 INPUTA1 相同，可以互换，而与 INPUT_A1 不同。

保留字不能用作标识符（参见附录 D），它们已有确定的意义，实际上是 PL/M 语言的一部分。例如，DECLARE，DATA，DO，IF，EOF 等。

应注意，EOF 作为 PL/M-96 的一条特殊语句，表示一个模块（可单独编译的程序块）的结束。但对编译结果不产生影响。

预说明的标识符是 PL/M 语言提供的内部过程和内部变量（见第六章）。预说明的标识符与保留字的不同，在于可以用说明语句重新说明这些标识符，且在这个说明的作用域内就不能再使用相应的内部过程和内部变量了。附录 E 列出了预说明的标识符。

1.2.3 符号、分界符和空符号的作用

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、下横线和美元符号$以外的单个PL/M字符（单个 ASCII 字符）：+ - * / < > = . ( ) , : ; '
4. 复合分界符：特定两个字符的某种组合。即 <> <= >= := /* */