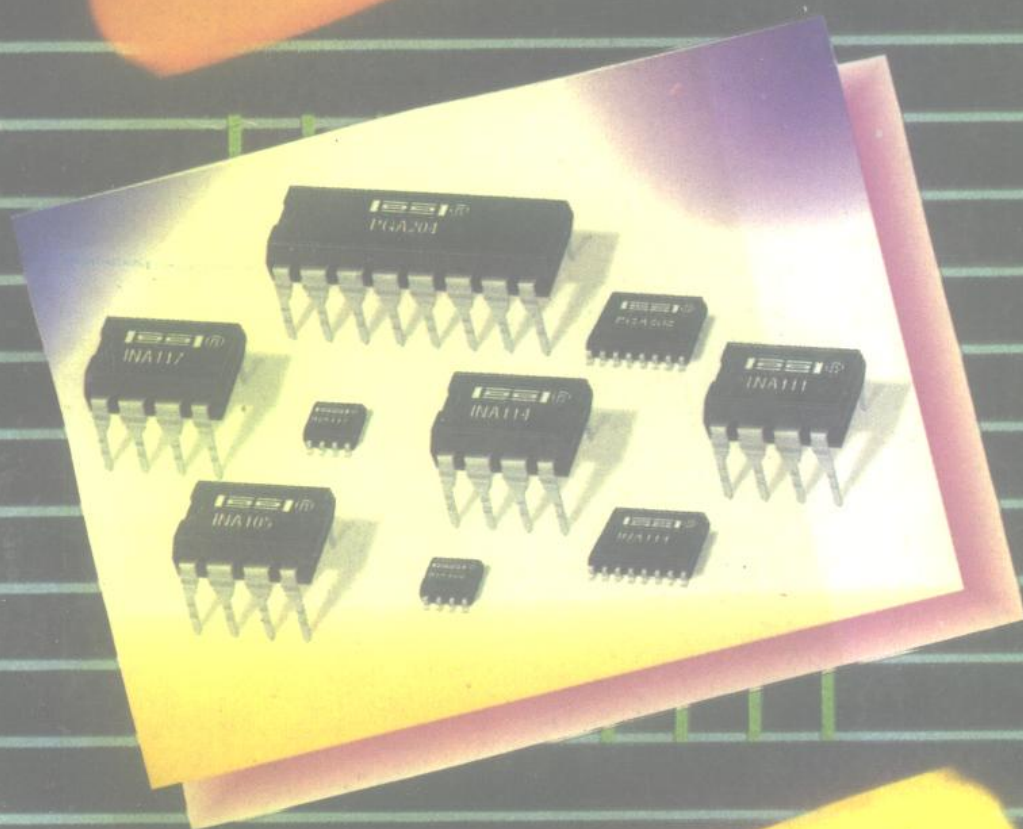


单片机

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

第一章 概论	3	2.6.2 结构	19
1.1 PL/M - 51 语言	3	2.6.3 结构数组	19
1.1.1 PL/M - 51 语言的特点	3	2.6.4 结构内数组	19
1.1.2 两种 PL/M - 51 语句	4	2.6.5 结构数组内数组	20
1.1.3 PL/M - 51 程序的结构	4	2.6.6 数组的隐含长度说明	20
1.2 PL/M - 51 程序的开发过程	4	2.6.7 对变量的引用	20
第二章 PL/M - 51 程序基础	6	2.7 PL/M - 51 程序基本结构	21
2.1 语言基本成分	6	第三章 高级说明语句	23
2.1.1 字符集	6	3.1 概述	23
2.1.2 标识符和保留字	6	3.2 组合说明和因子式说明	23
2.1.3 符号、分界符和空符号	7	3.2.1 组合说明	23
2.1.4 注释	7	3.2.2 因子式说明	23
2.2 PL/M - 51 语句	8	3.3 AT 属性说明	24
2.2.1 语句类型	8	3.4 有基变量	25
2.2.2 简单说明语句	8	3.5 连接属性说明	27
2.3 常数	9	3.6 文字(LITERALLY)说明	28
2.3.1 纯值常数	9	3.7 过程说明	30
2.3.2 字符串常数	9	3.8 存储区及后缀	30
2.4 变量	10	3.9 存储的相邻性	33
2.4.1 变量说明	10	第四章 PL/M - 51 执行语句	34
2.4.2 变量类型及运算	11	4.1 赋值语句	34
2.5 运算与表达式	11	4.1.1 简单赋值语句	34
2.5.1 运算对象	11	4.1.2 隐类型转换	34
2.5.2 表达式类型	13	4.1.3 多重赋值语句	35
2.5.3 算术运算及其表达式	14	4.2 DO 程序块	35
2.5.4 关系运算及其表达式	14	4.2.1 简单 DO 程序块	35
2.5.5 逻辑运算及其表达式	15	4.2.2 DO WHILE 程序块	36
2.5.6 运算优先级	15	4.2.3 循环 DO 程序块	37
2.5.7 常数表达式	16	4.2.4 DO CASE 程序块	38
2.5.8 负数	17	4.3 IF 语句	39
2.6 数组与结构	18	4.3.1 IF 语句	39
2.6.1 数组及下标变量	18	4.3.2 嵌套的 IF 语句	40

4.3.3 顺序 IF 语句	41	7.2.2 BOOLEAN、EXPAND 和 PROPAGATE 函数	65
4.4 语句标号和 GOTO 语句	42	7.3 移位及循环移位函数	66
4.4.1 语句标号	42	7.3.1 逻辑移位函数 SHL 和 SHR	66
4.4.2 GOTO 语句	42	7.3.2 循环移位函数 ROL 和 ROR	66
4.5 CALL 及 RETURN 语句	43	7.4 输入/输出函数	67
4.6 空语句(;)	43	7.5 其它内部过程	67
4.7 ENABLE 和 DISABLE 语句	43	7.5.1 TIME 过程	67
第五章 过程	45	7.5.2 TESTCLEAR 过程	68
5.1 概述	45	7.6 与 8051 硬件有关的运算符及 内部过程	68
5.2 过程说明	45	7.6.1 优化与 8051 硬件标志	68
5.2.1 参数	46	7.6.2 PLUS 及 MINUS 运算符	69
5.2.2 有类型和无类型过程	47	7.6.3 与硬件有关的内部过程	69
5.2.3 从过程退出	47	第八章 PL/M - 51 的编译程序 及控制项	71
5.2.4 过程体	48	8.1 编译及控制简介	71
5.3 过程的属性	49	8.1.1 编译控制项的位置	71
5.3.1 公共(PUBLIC)和外部 (EXTERNAL)属性	49	8.1.2 编译控制项的种类	72
5.3.2 中断(INTERRUPT)	50	8.2 工作文件(WORKFILES)控制	74
5.3.3 开中断(ENABLE)和关中断 (DISABLE)语句	52	8.3 目标文件控制	74
5.3.4 USING 属性	52	8.3.1 目标文件建立(OBJECT/ NOOBJECT)控制	74
5.4 过程的调用	53	8.3.2 调试(DEBUG/NODEBUG)控制	75
5.4.1 CALL 调用	53	8.3.3 代码模式 ROM 控制	75
5.4.2 函数引用	53	8.3.4 优化(OPTIMIZE)控制	75
5.4.3 间接调用	54	8.3.5 寄存器(REGISTERBANK)控制	77
第六章 结构化程序设计	55	8.3.6 中断向量入口(INTVECTOR/ NOINTVECTOR)控制	77
6.1 结构化程序	55	8.4 列表文件内容控制	78
6.2 程序的模块化结构	56	8.4.1 打印输出(PRINT/NOPRINT) 控制	78
6.2.1 模块	56	8.4.2 源程序列表(LIST/NOLIST) 控制	78
6.2.2 模块的连接	57	8.4.3 目标代码、汇编语句列表 (CODE/NOCODE)控制	79
6.3 作用域	57	8.4.4 相互引用表(XREF/NOXREF) 控制	79
6.3.1 作用域	57	8.4.5 符号列表(SYMBOLS/NOSYMBOLS) 控制	79
6.3.2 变量的作用域	58	8.5 列表文件(.LST)格式控制	79
6.3.3 作用域扩展	59	8.5.1 分页(PAGING/NOPAGING) 控制	80
6.4 标号的作用域和对 GOTO 语句的限制	60		
第七章 内部过程	63		
7.1 获取变量信息的内部过程	63		
7.1.1 LENGTH 函数	63		
7.1.2 LAST 函数	64		
7.1.3 SIZE 函数	64		
7.2 显式类型转换	65		
7.2.1 LOW、HIGH 和 DOUBLE 函数	65		

8.5.2 页长(PAGELNGTH)控制	80	9.4.2 连接控制项	96
8.5.3 行宽(PAGEWIDTH)控制	80	9.4.3 定位控制项	97
8.5.4 标题(TITLE)控制	80	9.4.4 结构控制项	98
8.5.5 日期(DATE)控制	81	9.5 与 PL/M-51 有关的应用程序	99
8.5.6 换页(EJECT)控制	81	9.5.1 PLM51.LIB	99
8.6 嵌入源文件控制	81	9.5.2 LIB51.EXE 建库程序	99
8.6.1 嵌入源文件(INCLUDE)控制	81	9.5.3 OH 文件格式转换程序	100
8.6.2 保留/恢复(SAVE/RESTORE) 控制	82	9.5.4 文本文件过滤处理	101
8.7 条件编译控制	82	9.6 与 ASM51 程序交叉使用	102
8.7.1 条件编译(IF/ELSE/ELSE IF/END IF)控制	82	9.6.1 汇编程序调用 PL/M-51 程序	103
8.7.2 条件开关设置 (SET/RESET)控制	84	9.6.2 PL/M-51 程序调用汇编程序	106
8.8 对输出列表文件的一些解释	84	9.7 PL/M-51 程序的编译、连接实例	108
8.8.1 程序清单	84	第十章 PL/M-51 实例程序	113
8.8.2 符号清单及相互访问清单	87	10.1 LED 动态显示	113
8.8.3 编译总结	88	10.2 8051 与 PC 机串行通讯	114
第九章 软件 RL51 及其它有关软件	89	10.3 交通灯控制实例	117
9.1 概述	89	10.4 键盘/LED 显示接口实例	122
9.2 RL51 连接与定位的原理	89	10.5 多模块连接	126
9.2.1 基本功能	89	附录 I	132
9.2.2 选择模块	89	附录 I.1 PL/M-51 的语法	132
9.2.3 段	90	附录 I.2 程序的约束	135
9.2.4 可重新定位段	91	附录 I.3 PL/M-51 保留字	135
9.2.5 给各段分配存储器	91	附录 I.4 预先声明的标识符	136
9.2.6 决定外部访问	92	附录 I.5 PL/M-80 与 PL/M-51 之间 的区别	136
9.2.7 汇编可重新定位的地址	92	附录 I.6 ASC II 代码	138
9.3 RL51 程序使用方法	92	附录 I.7 PL/M-51 与 ASM51 的连接	139
9.3.1 输入文件列表	93	附录 I.8 运行时的中断处理	141
9.3.2 输出文件	94	附录 I.9 处理器的描述符文件	144
9.3.3 控制项列表	94	附录 I.10 PLM51 出错信息	150
9.4 连接定位控制项	94	附录 I.11 RL51 控制项总结	156
9.4.1 列表控制项	94	附录 I.12 RL51 出错信息	159
		附录 I.13 LIB51 命令总结	166
		附录 I.14 LIB51 出错信息	167

下篇 PL/M-96 程序设计语言

第十一章 PL/M-96 编程基础	171	及保留字	172
11.1 概述	171	11.2.1 基本符号	172
11.1.1 PL/M-96 语言及其特点	171	11.2.2 标识符和保留字	173
11.1.2 PL/M-96 程序开发过程	171	11.3 常数	173
11.2 PL/M-96 语言的基本符号、标识符		11.3.1 纯值常数	173

11.3.2 浮点常数	174	(CALL 和 RETURN)	197
11.3.3 字符串常数	174	13.4.2 空语句(;)	197
11.4 简单说明语句	174	13.4.3 开中断(ENABLE)和关中断 (DISABLE)语句	198
11.5 变量和变量类型	175	第十四章 过程、程序块结构 和作用域	199
11.5.1 字节、字和双字变量	175	14.1 过程	199
11.5.2 整型、短整型和长整型	176	14.1.1 过程说明	199
11.5.3 实型和地址型	176	14.1.2 参数	200
11.5.4 隐含类型转换	176	14.1.3 有类型与无类型过程	200
11.6 有基变量	177	14.1.4 从过程中退出	201
11.6.1 有基变量	177	14.1.5 过程体	201
11.6.2 地址引用和有基变量举例	177	14.2 过程的属性	201
11.7 结构和数组	178	14.2.1 公共(PUBLIC)和外部 (EXTERNAL)属性	201
11.7.1 数组和下标变量	178	14.2.2 中断和中断属性	202
11.7.2 结构	179	14.2.3 开中断(ENABLE)和关中断 (DISABLE)语句	203
11.7.3 结构数组和结构内数组	179	14.2.4 可重入性(REENTRANT)	204
11.7.4 对数组与结构的引用	180	14.2.5 间接调用属性(INDIRECTLY- CALLABLE)	205
11.8 表达式和赋值语句	180	14.2.6 中断调用属性(INTERRUPT- CALLABLE)	205
11.8.1 算术表达式	181	14.3 过程的调用	205
11.8.2 关系表达式	181	14.3.1 函数引用	206
11.8.3 逻辑表达式	182	14.3.2 CALL 调用	206
11.8.4 运算的优先级	182	14.3.3 间接过程调用	206
11.8.5 赋值语句	182	14.3.4 调用其它模块的过程	206
第十二章 高级说明语句	184	14.4 PL/M-96 语言的程序块结构	207
12.1 连接属性说明	184	14.5 作用域	207
12.2 AT 属性	185	14.5.1 基本术语	207
12.3 DATA 赋值	186	14.5.2 作用域	209
12.4 语句标号说明	186	14.5.3 标号对 GOTO 语句的限制	209
12.5 LITERALLY 说明	187	第十五章 内部过程和内部变量	212
12.6 FAST 和 SLOW 属性	187	15.1 获取变量信息的内部过程	212
12.7 PL/M-96 程序的基本结构	187	15.1.1 LENGTH 过程	212
第十三章 程序流程控制语句	189	15.1.2 LAST 过程	212
13.1 DO 程序块	189	15.1.3 SIZE 过程	213
13.1.1 简单 DO 程序块	189	15.2 类型转换	213
13.1.2 DO WHILE 程序块	190	15.2.1 LOW、HIGH 和 DOUBLE 过程	214
13.1.3 循环 DO 程序块	192	15.2.2 SHORT 和 EXTEND 过程	215
13.1.4 DO CASE 程序块	194	15.2.3 SIGNED 和 UNSIGNED 过程	216
13.2 条件控制语句(IF 语句)	195	15.2.4 FLOAT 和 FIX 过程	216
13.2.1 IF 语句	195		
13.2.2 嵌套 IF 语句	196		
13.2.3 IF 语句串联	196		
13.3 无条件转移语句(GOTO 语句)	197		
13.4 其它可执行语句	197		
13.4.1 过程调用和返回语句			

15.2.5	ABS 和 IABS 过程	217	16.4.2	设置控制字过程: SET \$ REAL \$ MODE	230
15.3	移位和循环移位过程	218	16.4.3	获取出错字节过程: GET \$ REAL \$ ERROR	230
15.3.1	循环移位 ROL 和 ROR	218	16.4.4	保存 REAL 状态过程: SAVE \$ REAL \$ STATUS	230
15.3.2	逻辑移位 SHL 和 SHR	218	16.4.5	恢复 REAL 状态过程: RESTORE \$ REAL \$ STATUS ...	230
15.3.3	算术移位 SAL 和 SAR	218	16.4.6	浮点运算实例	231
15.4	串处理过程	219	16.5	编写实数异常事件处理程序	231
15.4.1	串复制过程:MOVB 和 MOVW	219	16.6	浮点运算库 FPAL96 的连接	234
15.4.2	串比较过程:CMPB 和 CMPW ...	219	16.7	FPAL96 浮点运算库	234
15.4.3	串搜索过程:FINDB 和 FINDW	220	16.7.1	FPAL96 子程序命令规则	235
15.4.4	SKIPB 和 SKIPW 过程	221	16.7.2	参数传递规则	235
15.4.5	SETB 和 SETW 过程	221	16.7.3	调用 FPAL96 子程序实例	235
15.5	位操作过程	221	16.7.4	FPAL96 的子程序	236
15.5.1	BITSET 过程	221	第十七章 PL/M - 96 程序编译控制 ...	245	
15.5.2	BITCLR 过程	221	17.1	编译概述	247
15.5.3	BITTST 过程	222	17.2	目标文件控制	247
15.5.4	BITCPL 过程	222	17.2.1	优化(OPTIMIZE)控制	247
15.5.5	BITASN 过程	222	17.2.2	寄存器覆盖(REGOVERALY/ NOREGOVERALY)控制	251
15.6	其它内部过程和内部变量	222	17.2.3	快速(FAST)控制	252
15.6.1	MOVE 过程	222	17.2.4	建立目标文件(OBJECT/ NOOBJECT)控制	252
15.6.2	TIME 过程	223	17.2.5	调试(DEBUG/NODEBUG) 控制	253
15.6.3	MEMORY 数组	223	17.2.6	类型(TYPE/NOTYPE)控制	253
15.6.4	STACKPTR 过程	223	17.3	列表选择和列表内容控制	253
15.7	与 MCS - 96 硬件有关的运算符及 内部过程	223	17.3.1	打印输出 (PRINT/NOPRINT)控制	253
15.7.1	代码优化和 MCS - 96 硬件	224	17.3.2	源程序列表 (LIST/NOLIST)控制	254
15.7.2	PLUS 和 MINUS 运算符	224	17.3.3	目标代码 (CODE/NOCODE)控制	254
15.7.3	与硬件有关的内部过程	224	17.3.4	相互引用列表 (XREF/NOXREF)控制	254
第十六章 浮点库 FPAL96. LIB 及 有关过程	225		17.3.5	符号列表(SYMBOLS/NOSYMBOLS) 控制	254
16.1	实数的表示	225	17.4	列表格式控制	255
16.2	REAL 数学部件	226	17.4.1	页长(PAGELNGTH)控制	255
16.3	异常事件及其处理	228	17.4.2	行宽(PAGewidth)控制	255
16.3.1	不合法操作事件	228	17.4.3	标题(TITLE)控制	255
16.3.2	被零除异常事件	228			
16.3.3	上溢异常事件	229			
16.3.4	下溢异常事件	229			
16.3.5	精度降低	229			
16.3.6	非规格化异常事件	229			
16.4	与浮点库有关的内部过程	229			
16.4.1	初始化过程: INIT \$ REAL \$ MATH \$ UNIT ...	229			

17.4.4 换页(EJECT)控制	255	19.1 A/D 转换实例程序	272
17.5 应用实例	256	19.2 波形发生器	275
17.5.1 源程序和汇编代码列表	256	19.2.1 由 PWM 构成的波形发生器	275
17.5.2 符号表和相互引用表	258	19.2.2 由高速输出部件 HSO 构成的 波形发生器	277
17.5.3 编译总结	259	19.3 PC 机与多个 8098 单片机 串行通讯	279
17.6 嵌入源文件控制	259	19.4 键盘/显示实例	285
17.6.1 嵌入源文件(INCLUDE)控制	259	19.5 交通灯模拟控制实例	287
17.6.2 保存/恢复 (SAVE/RESTORE)控制	260	19.6 直接调用浮点运算库 FPAL96 的过程	290
17.7 条件编译控制	260	19.6.1 浮点运算实例	290
17.7.1 条件编译(IF/ELSEIF/ELSE/ENDIF) 控制	260	19.6.2 REAL 数与 DEC 数之间的 转换实例	291
17.7.2 设置条件开关 (SET/RESET)控制	261	19.7 与汇编语言混合编程	291
17.7.3 条件列表 (COND/NOCOND)控制	262	附录 I	295
17.7.4 条件控制举例	262	附录 I.1 PL/M - 96 保留字及预说明 的标志符	295
第十八章 与 PL/M - 96 有关的 应用程序	265	附录 I.2 程序的限制	296
18.1 PL/M - 96 程序的连接	265	附录 I.3 PL/M - 96 语言和汇编语言及 C 语言程序接口	297
18.1.1 DOS 下 RL96 的启动	265	附录 I.4 运行时的中断处理	297
18.1.2 ROM 控制	266	附录 I.5 MCS - 96 I/O 寄存器符号名	298
18.1.3 RAM 控制	267	附录 I.6 出错信息	299
18.1.4 STACKSIZE 控制	267	附录 I.7 RL96 出错信息	310
18.1.5 其它控制	267	附录 I.8 LIB96 出错信息	317
18.1.6 应用举例	267	附录 I.9 零操作数和无穷大操作数 的应用	318
18.2 库管理程序 LIB96.EXE	268	附录 I.10 TPAL96 异常事件与 操作代码	320
18.2.1 启动 LIB96 库管理程序	268	附录 I.11 ASC II 字符表	321
18.2.2 LIB96 命令	268	附录 I.12 MCS - 96 系列汇编语言 指令表	323
18.2.3 命令输入	269	主要参考资料	326
18.2.4 命令说明	269		
18.3 OH 格式转换	271		
18.4 文件过滤程序 FILTER.EXE	271		
18.5 编译连接操作实例	271		
第十九章 PL/M - 96 实例程序	272		

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 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 的微处理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。

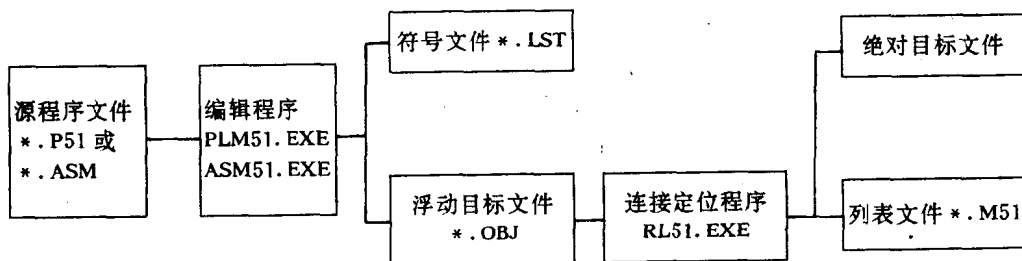


图 1.1 程序编译连接过程的输出

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。