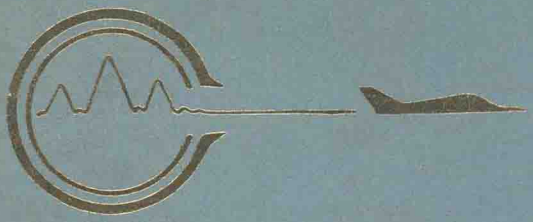


PL/M—86 用户指南
用于 8086 开发系统

手册号　121636—001

第二十二册

PL/M—86 用户指南
用于 8086 开发系统

手册号　121636—001
第二十二册

翻译：张　纶
　　刘昊天
校对：韩苏南

航空工业部第五七四厂

前　言

本书主要介绍 PL/M-86 语言与 PL/M-86 编译程序的使用。第一章定义了“模块”，内容包括模块，程序块，说明语句和一些 PL/M-86 基本规则的定义及示意图，并简要介绍了大多数可执行语句。

第二章定义了各种常量、变量或标号的说明语句，并给出了示意图。（虽然也提到了过程，但第十章才进行详细介绍。）本章还首次引入了数据类型。

第三章讨论了 PL/M-86 程序使用的字符，保留字，常量和基本成分并附有示意图和实例。

第四章开始讨论 PL/M-86 程序用各种数据类型进行的各种算术运算。本章还介绍了地址引用，利用地址引用可访问某个地址，而不是访问存在某个地址中的数值。我们还给出了应用实例。

第五章详细介绍了算术运算，其中涉及表达式的计算及赋值语句中的隐式数据类型转换。在该章结尾处用示意图说明了有效表达式的构成，还列表说明了表达式中允许的类型混用的情形。

第六章给出了数组和结构的定义，说明及其实例，并讨论了部分限定引用。

第七章的主题是 DO 程序块，IF 结构和 GOTO 语句。并提及第九、十章分别介绍的作用域与过程。

第八章是一个实例程序。

第九章又再次讨论程序块结构与说明，这是为了扩充几条作用域的规则，并介绍 PUBLIC 和 EXTERNAL 连接属性。本章还讨论了所有可以使用这些属性的标识符的有关规则。

第十章详细说明了第一章已经提到的有类型与无类型过程。本章章末的示意图总结了过程说明的正确结构。

第十一章说明并图示了实现类型转换，串处理及其它几种操作的内部变量的过程。

第十二章叙述了与某些 8086 硬件和标志有关的 PL/M-86 内部过程。

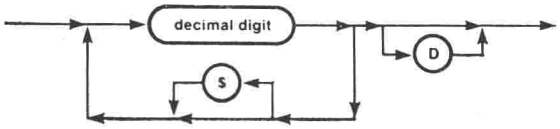
第十三章讨论了 REAL 型数及其赋值和运算：如程序中使用了浮点运算的话，就需要了解本章介绍的过程和限制。

其余各章说明并讨论了编译程序的调用及其控制项。这些控制项的作用有：按一定格式给出编译程序的输出文件，加入其它输入源文件，产生可调试的目标代码以及选择模块的分段类型（8086 内存的划分方法）。控制既可嵌放在源文件中，也可跟在编译程序的调用命令后面。

PL/M-86 编程手册的路轨图表示法

本书中的许多 PL/M-86 语句和命令都是用“路轨”语法图形来表示的，图中说明了各种语言成分是如何有效地组合在一起的。

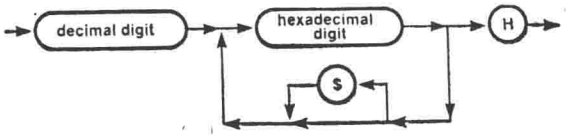
这些图都是从左侧的前头开始，经过不同的园框或椭园框后，于右侧的箭头处结束。我们可根据图中的路径组成合法的句子。主路径上的成分或选择项是不能更动的。侧路径上则是可选的或可重复的成分，它们可从主路径上删除。下面举几个例子进行说明：



无符号十进制常数

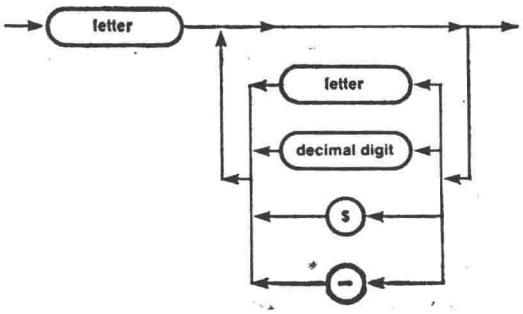
十进制数字是指0,1,2,3,4,5,6,7,8,9中的任何一个数。

该图表示要写一个有效的十进制常数必须从一个十进制数字开始，然后就可以退出去或循环回去并加上更多的数字;或者在数字间加入美元符号以提高可读性;或在数后加D，以表示十进制数。因为无后缀的数即假定为十进制数,所以 D 是任选的。



无符号 16 进制常数

十六进制数字指的是一个十进制数字或为 A, B, C, D, E, F 其中之一。这个图表明了写一个十六进制数必须从十进制数字开始,以 H（代表 16 进制数）结尾。然而在结束前可以加入更多的 16 进制数字或美元符号

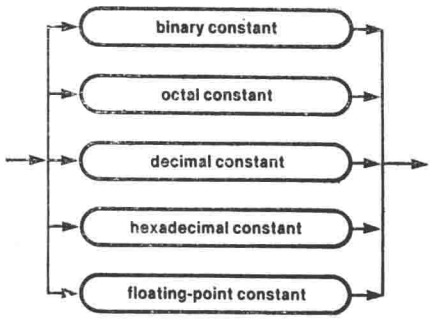


标识符

字母是 A 至 Z 之间的大、小写字母。一个有效的标识符必须从字母开始，加上更多的字母,最多可加到 31 个(不包括美元符号)。美元符号和下横线是用于改善程序的可读性的。

本手册中，“标识符”和“名”这两个词是同义的。

语法图中还可含其它语法图的名，用小写字母表示。以与用大写字母表示的保留字相区别。这种复合图的完整含义,可以通过用具体图形取代图名来理解。比如,无符号的数值常数图包括了十进制和十六进制常数图。



无符号数值常数

图中的小写名还可由用户程序里所使用的名来代替。在第十一章里将出现象“NEW $ BYTE” 这样的名，表示必须用 BYTE 型量来替换；“BYTEOR$WORD”则告诉你 BYTE 型 WORD 型名均可使用。

大写的名是 PL/M-86 的保留字。在程序里只能用同名的小写字母形式替换它们。给出保留字的地方要打入分号。

有关的出版资料

下面给出一些在阅读本手册时还应参考的其他 Intel 资料。每本手册名的后面是手册号，手册题目下面是内容介绍。

Intellec Series III Product Overview, 121575

该手册简要说明了 Intellec 系列 III 开发系统及其支持的硬、软件。在这本篇幅不多的手册中，还介绍了与系列 II 有关的手册并附有词汇表。

Intellec Series III Console Operating Instructions, 121609

Intellec Series III Pocket Reference, 121610

这是系列 III 控制台的使用说明，并包括了驻留监控程序的说明。前者进行了全面的说明，后者则是简要说明。

Intellec Series III Programmer Reference Manual, 121618

该手册说明了在系列 III 的两种微机环境（8080/8085 和 8086）下用户程序是怎样调用系统子程序的。

ISIS-II CREDIT (CRT-Based Text Editor) User's Guide, 9800902

ISIS-II CREDIT (CRT-Based Text Editor) Pocket Reference, 9800903

这两本手册是 CREDIT，即系列 III 的 CRT 文本编辑程序的使用说明。前者是详细说明，后者则是简述，可供快速查阅。

iAPX86, 88 系列服务程序用户指南，用于 8086 开发系统。

该手册介绍在 8086 环境下使用服务程序 LINK86, LOCK86, LIB86 和 OH86，将经过编译或汇编的程序构造成可执行的程序。

8086/8087/8088 宏汇编语言参考手册，用于 8086 开发系统

8086/8087/8088 宏汇编操作说明，用于 8086 开发系统

8086/8087/8088Macro Assembly Language Reference, 121674

这些手册说明了在 8086 环境下怎样使用 8086/8087/8088 宏汇编语言及汇编程序。

语言参考手册对汇编语言做了全面的叙述；操作说明手册详细介绍了汇编程序的使用；为了能快速查阅，袖珍参考手册则提供了有关的简要说明。在用汇编语言编写子程序时需要查阅这些手册。

ICE-86 In-Circuit Emulator Operating Instructions for ISIS-II Users, 9800714

ICE-86 Pocket Reference, 9800838

ICE-88 In-Circuit Emulator Operating Instructions for ISIS-II Users, 9800949

ICE-88 Pocket Reference, 9800950

这些手册给出了开发硬、软件时使用 ICE-86 和 ICE-88 在线仿真器的说明。操作手册是使用说明，袖珍参考手册是为快速查阅而准备的简要说明。如果使用 ICE-86 或 ICE-88 仿真器的话就需要阅读其中相应的手册。

目　录

第一章　概　述

模块之间的引用：LINK 过程……(1)
内存地址分配：LOCATE 过程 ……(1)
PL/M-86 语句的两种类型 ……(4)
一条规则……(4)
程序块：PL/M-86 程序的基本结构 ……(5)
　块嵌套和变量作用域：简介……(6)
　说明语句的位置……(7)
可执行语句……(8)
　赋值语句……(9)
　IF 语句 ……(9)
　DO 和 END 语句……(9)
　内部过程和变量……(11)
　表达式……(11)
　输入和输出……(11)

第二章　说明与类型

变量说明语句……(14)
　类型……(14)
　举例……(14)
　结果……(15)
初始化……(15)
　隐含长度说明符……(17)
　执行常量的名：DATA 的使用……(18)
编译常量(正文替换)：LITERALLY 的使用……(18)
标号名的说明……(19)
　结果……(20)
说明语句的组合……(20)
过程说明……(21)

第三章　PL/M-86 程序的基本成份

PL/M-86 字符集 ……(23)
标识符和保留字……(24)

标记、分隔符和空格的使用……(25)
常数……(25)
纯数常数……(26)
浮点常数……(27)
字符串……(27)
注解……(28)

第四章　算术运算、数据类型和有基变量

WORD 和 BYTE 变量：无符号运算……(29)
INTEGER（整型）变量：带符号算术运算……(29)
REAL 变量：浮点运算……(29)
POINTER 变量和位置引用 ……(30)
@算符……(30)
用地址引用存贮字符串与常数……(31)
“点”算符……(31)
有基变量……(31)
地址引用与有基变量……(33)
内存的相邻存贮……(33)
AT 属性……(33)

第五章　表达式和赋值语句

运算数……(36)
常数……(36)
变量和地址引用……(37)
子表达式……(37)
组合运算数……(37)
算术算符……(37)
十、一、* 和/算符 ……(38)
MOD（模）算符 ……(38)
关系算符……(39)
逻辑算符……(39)
表达式的计算……(40)
算符的优先级：对表达式的分析……(40)
组合运算数类型……(42)
关系算符的限制……(42)
运算数的计算顺序……(43)
算术运算的选择：规则小结……(43)
特殊情况：常数表达式……(44)
赋值语句……(45)

隐含类型转换……(46)
常数表达式……(46)
多重赋值语句……(47)
内嵌赋值语句……(47)

第六章　结构和数组

数组……(49)
下标变量……(49)
结构……(50)
结构数组……(51)
结构内的数组……(51)
结构内带数组的结构数组……(51)
数组和结构的引用……(52)
全限定变量引用……(52)
非限定和部分限定变量引用……(52)

第七章　流程控制语句

DO 和 END 语句：DO 程序块……(54)
简单 DO 程序块……(55)
DO CASE 程序块……(56)
DO WHILE 程序块……(58)
循环 DO 程序块……(58)
IF 语句……(60)
嵌套的 IF 语句……(61)
串连 IF 语句……(63)
GOTO 语句……(64)
HALT 语句……(64)
CALL 和 RETURN 语句……(64)

第八章　实例程序 1

插入分类算法……(66)

第九章　程序块结构和作用域

程序块内认可的名……(70)
对多重说明语句的限制……(71)
扩展的作用域：PUBLIC 和 EXTERNAL 属性……(72)
标号作用域与对 GOTO 语句的限制……(73)

第十章　过　程

过程说明……(77)
参数……(78)
有类型与无类型过程……(79)
过程的调用——函数引用及 CALL 语句 ……(80)
间接过程调用……(81)
退出过程：RETURN 语句……(81)
过程体……(82)
属性：PUBLIC、EXTERNAL、INTERRUPT 和 REENTRANT……(83)
中断和INTERRUPT 属性……(84)
用 CALL 语句调用中断过程 ……(85)
可重入性和 REENTRANT 属性……(85)
实例程序 2 ……(86)

第十一章　内部过程和内部变量

取变量的有关信息……(90)
LENGTH 函数 ……(90)
LAST 函数……(91)
SIZE 函数 ……(91)
显式的类型和值的转换……(91)
LOW、HIGH 和 DOUBLE 函数……(92)
FLOAT 函数 ……(93)
FIX 函数……(93)
INT 函数……(94)
SIGNED 函数……(94)
UNSIGN 函数……(94)
ABS 和 IABS 函数……(95)
移位和循环移位函数……(95)
循环移位函数，ROL 和 ROR ……(96)
逻辑移位函数，SHL 和 SHR……(96)
代数移位，SAL 和 SAR ……(96)
输入和输出……(97)
INPUT 和 INWORD 函数 ……(97)
OUTPUT 和 OUTWORD 数组 ……(97)
串处理过程……(98)
MOVB 和 MOVW 过程 ……(98)
MOVRB 和 MOVRW 过程 ……(98)
CMPB 和 CMPW 函数……(99)
FINDB/FINDW 和 FINDRB/FINDRW 函数…… (100)
SKIPB/SKIPW 和 SKIPRB/SKIPRW 函数 …… (100)

XLAT 过程 …… (101)
SETB 和 SETW 过程 …… (101)
其它各种内部过程和变量 …… (101)
MOVE 过程 …… (101)
TIME 过程 …… (102)
MEMORY 数组 …… (102)
STACKPTR 和 STACKBASE …… (102)
LOCKSET 函数 …… (103)
与中断相关的过程 …… (104)
SET$INTERRUPT 过程 …… (104)
INTERRUPT$PTR 函数 …… (104)
CAUSE$INTERRUPT 语句 …… (104)

第十二章 与 8086 硬件有关的 PL/M-86 内部过程

优化与 8086 硬件标志 …… (105)
PLUS 和 MINUS 运算符 …… (105)
进位循环移位内部过程 …… (105)
DEC 过程 …… (106)
CARRY、SIGN、ZERO 和 PARITY 内部过程 …… (106)

第十三章 浮点运算：REAL 数的数学机构

PEAL 值的表示方法 …… (107)
REAL 参数的传递和堆栈约定 …… (109)
REAL 数的数学机构 …… (109)
REAL 数字运算中的异常状态 …… (111)
无效操作异常 …… (111)
非规格化操作数异常 …… (112)
除零异常 …… (122)
上溢异常 …… (113)
下溢异常 …… (113)
精度异常 …… (113)
INIT$REAL$MATH$UNIT 过程 …… (114)
SET$REAL$MODE 过程 …… (114)
GET$REAL$ERROR 过程 …… (114)
保存和恢复 REAL 状态 …… (114)
SAVE$REAL$STATUS 过程 …… (115)
死锁 …… (115)
REAL 中断处理过程的编写 …… (116)
使用浮点运算时的连接 …… (118)

第十四章　编译程序的控制

编译程序控制简介 …… (121)
WORKFILES 控制 …… (123)
LEFTMARGIN 控制 …… (123)
目标文件控制 …… (124)
INTVECTOR/NOINTVECTOR …… (124)
OVERFLOW/NOOUERFLOW …… (124)
OPTIMIZE …… (125)
OPTIMIZE (0) …… (125)
OPTIMIZE (1) …… (125)
OPTIMIZE (2) …… (125)
OPTIMIZE (3) …… (129)
OBJECT/NOOBJECT …… (133)
DEBUG/NODEBUG …… (134)
TYPE/NOTYPE …… (134)
程序大小的控制 …… (134)
SMALL …… (135)
COMPACT …… (135)
MEDIUM …… (135)
LARGE …… (135)
RAM/ROM 控制 …… (135)
列表选择和列表内容控制 …… (136)
PRINT/NOPRINT …… (136)
LIST/NOLIST …… (136)
CODE/NOCODE …… (137)
XREF/NOXREF …… (137)
IXREF/NOIXREF …… (137)
SYMBOLS/NOSYMBOLS …… (138)
列表格式控制 …… (138)
PAGING/NOPAGING …… (138)
PAGELENGTH …… (138)
PAGEWIDTH …… (139)
TITLE …… (139)
SUBTITLE …… (139)
EJECT …… (139)
程序列表实例 …… (139)
符号与相互对照的列表 …… (141)
编译概况 …… (142)

加进源文件的控制 …… (142)
INCLUDE …… (142)
SAVE/RESTORE …… (142)
条件编译控制 …… (143)
IF/ELSE/ELSEIF/ENDIF …… (143)
SET/RESET …… (144)
CND/NOCOND …… (145)

第十五章 PL/M-86 编译程序及有关文件的使用

编译程序的调用 …… (148)
文件的使用 …… (148)
输入文件 …… (148)
输出文件 …… (149)
编译程序工作文件 …… (149)

第十六章 目标模块的节和程序大小的控制

8086 内存概念 …… (150)
目标模块的节 …… (150)
代码节 …… (151)
常数节 …… (151)
数据节 …… (151)
堆栈节 …… (151)
内存节 …… (152)
SMALL 控制 …… (152)
与 PL/M-80 的兼容性 …… (153)
COMPACT 控制 …… (153)
MEDIUM 控制 …… (153)
在 MEDIUM 的编程设计限制 …… (154)
LARGE 控制 …… (154)
在 LARGE 控制下的编程设计限制 …… (155)

第十七章 出错信息

PL/M-86 源程序出错 …… (156)
致命的命令和控制错误 …… (175)
致命的输入/输出错误 …… (175)
致命的内存贮器不足错误 …… (176)
致命的编译程序故障错误 …… (176)

附录 A PL/M-86 语言语法 …… (177)

附录 B　程序限制 ……(191)
附录 C　PL/M-86 的保留字 ……(192)
附录 D　PL/M-86 预说明的标识符 ……(193)
附录 E　PL/M-80 和 PL/M-86……(194)
附录 F　字符对照表……(196)
附录 G　IXREF 程序 ……(198)
附录 H　PL/M-86 的分段 ……(202)
附录 I　运行时的过程以及汇编语言的连接 ……(204)
附录 J　运行时的中断处理 ……(207)

第一章 概 述

PL/M-86 是为 Intel 8086 或 8088 微处理机进行系统编程和应用编程而设计的一种高级语言。

说它高级是因为它不是接近于机器或汇编语言而是更接近英语和代数公式。汇编语言的每条语句最多产生一条机器语言指令。而一条 PL/M-86 语句可由编译程序翻译成一条 8086 指令或一个完整的指令序列，或不产生任何指令，如分配存贮区语句等。

一个 PL/M-86 程序实际上是一个本手册所叙述的 PL/M-86 语言的指令集。遵循一定的规则写成的指令集叫模块，该模块可用 PL/M-86 编译程序加工成目标代码(机器语言)。原来的指令集叫源码。

一个程序经常分成一些单个的模块。工作时作为一个整体，但编写、编译和调试都是分开来进行的。PL/M-86 设有一些方式和控制，有助于实现这一过程。

模块之间的引用：LINK 过程

模块之间通过使用彼此的变量和过程互相引用。这时必须把它们连接在一起，才能使在一个模块内定义的地址在另一个模块中也能用。做这项工作的是一个叫作 LINK 86 的 Intel 程序。它读进所有的目标码，并将所有的地址引用定位。因为所需要的地址在引用它们的某些模块之外，所以这一过程叫做解外部引用，定义所在的模块应把这些地址说明成“公共”[public]。

内存地址分配：LOCATE 过程

一般说来，编译用户程序时产生的地址与模块头相关。这就意味着说明的变量在从头数的第 4 个字节时，地址应是 4。因为每个模块单独进行编译，所以在分模块编程时很多各自使用的变量可以是同一个相对地址。这样的相对地址与 8086 内存使用的物理地址不同，后者叫做绝对地址。

在编译过程中给 PL/M-86 程序分配的不是最终的绝对地址而是在 8086 内存空间中“可重新定位”的地址。重新定位意味着在把程序装入内存时将重新分配地址，改变编译时分配的相对地址。正如上面提到的那样，有时把大程序分成更易管理，更易理解的模块是有好处的。为了得到可重新定位的各模块，用户必须在预先不知道最终绝对地址的情况下编写程序。然后由 PL/M-86 编译程序产生各模块，这些模块无论最终分配到那里都能工作。

这样，编程员不用事先了解内存中目标码的最终分配就可以着手建立模块。这就是说，不必考虑很多影响因素就能完成要做的工作。而且这种并行开发的产品以后能得益于新的事实，新的资源或一种新的观点。

通常调用一个叫做 LOC 86 (8086定位程序)的 Intel 服务程序,在 LINK 86 解决了模块之间的所有引用后,将相对地址译为绝对地址。用 LOC 86 进行这一复杂的工作要比自己做容易多了。 LOC 86 所产生的结果是一个具有绝对地址的完整程序,它的所有标号与变量都有特定的、明确的绝对内存地址。

图1—1举例说明了一个分为两个模块的 PL/M-86 程序。该程序包括许多未定义的字与结构,这些将在以下几章进行说明。这里只是想让你了解,一个分成两个模块的程序是什么样子的。

```
                    SORTMODULE:DO;/*Beginning of module*/
SORTPROC:PROCEDURE(PTR,COUNT,RECSIZE,KEYINDEX)PUBLIC;
         DECLARE PTR POINTER,(COUNT,RECSIZE,KEYINDEX)WORD;

         /*Parameters:
         PTR is pointer to first record
         COUNT is number of records to be sorted.
         RECSIZE is number of bytes in each record-maximum is 128.
         KEYINDEX is byte position within each record of a BYTE scalar
         to be used as sort key.*/

                 DECLARE RECORD BASED PTR(1) BYTE,
                         CURRENT (128) BYTE,
                         (I,J)WORD;
         SORT:   DO J = 1 TO COUNT-1;
                         CALL MOVB(@RECORD(J*RECSIZE),@CURRENT,RECSIZE);
                 I=J;
         FIND:         DO WHILE I>0 AND
                          RECORD((I-1)*RECSIZE + KEYINDEX)>CURRENT(KEYINDEX);
                          CALL MOVB(@RECORD((I-1)*RECSIZE),
                                                  @RECORD(I*RECSIZE), RECSIZE);

                          I=I-1;
         END FIND;
         CALL MOVB(@CURRENT,@RECORD(I*RECSIZE),RECSIZE);
         END SORT;

END SORTPROC;

                        END SORTMODULE;/*End of Module*/
```

This module is compiled and can then be kept available for use by any program that is linked to it. The main program module follows.

图 1-1 程序实例

该模块经编译后可供任何与之相连接的程序使用。

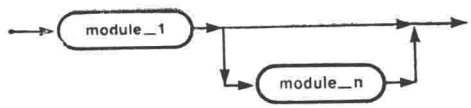
主程序模块如下: 见图 1-1 (续)

主程序将被编译成一个名叫“M”的模块, 它做事不多,但定义了一些数据,然后调用名为 SORTPROC 的过程。这个过程在其它模块中已做过定义, 它将要用 SORTMODULE 这个名进行编译。对于数据定义的概念及过程调用将进行扼要的讨论。

例如一个 PL/M-86 程序可看成一个或多个模块组成的集合体。

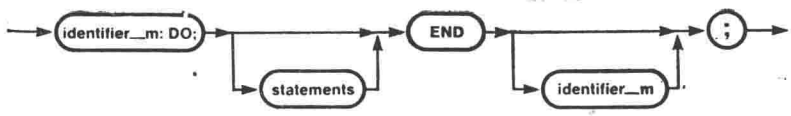
```
                    M: DO; /*Beginning of module*/

        /*Program to sort two sets of records, using SORTPROC*/
SORTPROC: PROCEDURE (PTR,COUNT,RECSIZE, KEYINDEX)EXTERNAL;
                DECLARE PTR POINTER,(COUNT,RECSIZE,KEYINDEX)WORD;
          END SORTPROC;/*End of usage declaration*/

          DECLARE SET1(50) STRUCTURE(ALPHA WORD,
                                     BETA(12) BYTE,
                                     GAMMA INTEGER,
                                        DELTA REAL,
                                     EPSILON BYTE);

        /*Key of Nth record in SET1 is SET1(N).BETA(0), the 3rd byte
                                                        in the record.*/

        DECLARE SET2(500)STRUCTURE(ITEMS(21)INTEGER,
                                           KEY BYTE);

        /*Key of Nth record in SET2 is SET2(N).KEY, the 43rd byte
                                                    in the record.*/

        /*Data is read in to initialize the records.*/

        CALL SORTPROC(@SET1,LENGTH(SET1),SIZE(SET1(1)),2);
        CALL SORTPROC(@SET2,LENGTH(SET2),SIZE(SET2(1)),42);

        /*Data is written out from the records.*/

                        END M; /*End of module*/
```

图 1-1　程序实例(续)



121636-15

一个 PL/M-86 程序必须至少有一个模块，也可以有许多模块。
每个模块的形式为：



121636-16

模块格式