8087 支持程序库参考手册

手册号 121725—001

第二十五册

8087 支持程序库参考手册

手册号 121725—001

第二十五册

翻译 韩苏南

校对 叶 宏

航空工业部第五七四厂

支持库手册的使用

这是一本8087支持库的程序员指南手册。8087程序库是一个软件工具包，8087或全8087仿真程序利用这个程序库可为ASM—86和PL/M—86的程序员提供完备的浮点运算功能。

本手册主要作为编写ASM—86和PL/M—86程序的参考指南。我们在介绍每一个过程和函数时，根据实际情况尽可能使描述自明，以期避免过多的查找。

阅读第一章和第二章以及第三、四、五章的第一部分，可使用户对程序库有一个基本的概念。如果用户关心同IEEE标准的一致性，可参阅第六章。熟悉附录中所给的内容也不无益处。

关于各个过程和函数的细节，请查阅第三、第四和第五章后面的具体说明。

值得注意的是，上述说明中包括了许多讨论非正常值的内容，如非规格化数，未规格化数、伪零等。请记住这些值是很不常见的，在大多数应用过程中根本就不会出现。从而用户可以忽略讨论它们的各段落。

当你已成为一个有经验的8087支持库用户时，为了提高查找速度，可参看附录D和附录E。

有关的出版资料

阅读本手册时，以下手册都是非常有用的参考材料。特别重要的是有关8087本身的说明，它们可在汇编语言参考手册和8086用户手册数字增补手册中找到。这些手册都描述了系列III版本（基于8086）的软件产品。

- iAPX86、88系列服务程序用户指南，手册号121616
- PL/M—86用户指南，手册号121636
- 8086/8087/8088宏汇编语言参考手册，手册号121627
- 8086/8087/8088宏汇编操作说明，手册号121628
- The 8086 Family User's Manual Numeric Supplement，手册号121586

8086系列的专用术语

本手册中使用了三种命名规则，以刻画8086系列微处理机：

1. iAPX规则。它表示的是含有一个或多个该系列部件的系统。这个系统必须配备一个8086或8088微处理机。下面的表中给出了最通常的iAPX系统所使用的部件。
2. 8086/8087/8088/8089规则。它表示的是iAPX系统中的某个特定的部件。
3. NDP/NPX规则。它表示了一个数字功能部件，该部件可由一个以上的硬件配

置实现。

·NPX（数字处理机扩展部件Numeric Processor Extension）指的是8087部件或全8087仿真程序软件。

·NDP（数字数据处理机Numeric Data Processor）指的是任何配有一个NPX的iAPX系统。

系统名	8086	8087	8088	8089
iAPX86/10	1			
iAPX86/11	1			1
iAPX86/12	1			2
iAPX86/20	1	1		
iAPX86/21	1	1		1
iAPX86/22	1	1		2
iAPX88/10			1	
iAPX88/11			1	1
iAPX88/12			1	2
iAPX88/20		1	1	
iAPX88/21		1	1	1
iAPX88/22		1	1	2

符号和程序码

本手册中所给出的程序例子，都是一些可原封不动地加入到PL/M—86或ASM—86程序中的代码段，它们不受一般语法程式的限制。

LINK86和LOC86的调用中沿用以下Intel手册常用的约定：

.下横线划出的命令表示用户对操作系统的输入。

·<cr>表示一回车符

目　录

第一章　绪论

第二章　全8087仿真程序和接口程序库

第三章　十进制转换程序库

试读结束：需要全本请在线购买：　www.ertongbook.com

第四章　常用基本函数库

第五章　出错处理程序模块

第六章　IEEE标准的实现

第一章 緒 论

8087支持库为在ASM—86和PL/M—86中运用浮点运算提供了极大的方便。它使以上两种语言增添了许多应用程序设计语言，诸如Pascal — 86和FORTRAN—86等所具备的功能。

全8087仿真程序E8087，以及接口程序库E8087.LIB、8087.LIB和87NULL.LIB，为含浮点代码的模块提供了运行环境的选择余地。仿真程序在功能上以及在程序设计上同8087是等价的，它是由16K字节的8086软件实现的。

十进制转换模块DCON87.LIB执行人可认别的十进制数与NPX支持的所有形式的二进制数之间的转换。它还能执行二进制数之间的转换。

通用基本函数库，CEL87.LIB，提供了一些非常有用的超越函数、舍入函数以及其它同浮点数有关的通用函数。

出错处理程序模块，EH87LIB，可以方便用户编写从浮点错误状态恢复的中断过程。

为什么需要支持库

出于三方面的考虑，许多应用软件需要支持软件来克服NDP在功能上的局限性。

第一，NDP是一台只能进行二进制运算的机器。尽管NDP能够处理很多种数据类型，但它不能直接对浮点十进制数进行运算。必须调用软件，来把十进制输入转换成二进制形式，将二进制的结果转换成十进制的输出。

第二，8087的指令集仅局限于已为它实现了的一些基本函数。例如，虽有正切函数，却无正弦函数，而且正切的输入仅限于0到π/4之间。一个具有完整定义域的完备基本函数集有待于软件来实现。

第三，非屏蔽的异常情况的处理会使应用软件复杂化。虽然出错检测和恢复的屏蔽方式对大多数应用软件足够用了，但有时仍需要特殊要求的出错恢复。这就很难辨认NPX的状态并使NPX回到恢复状态。这件事由软件来做就变得容易了。

8087支持库提供了弥补上述不足所需要的软件。有了支持库，NDP才成为一台完备、精确而又易用的浮点计算机。

使用程序库所需的系统软件

因为支持库是由ASM—86或PL/M—86程序调用的，这就要用到ASM—86汇编程序或者PL/M—86编译程序。

由于支持库的所有过程均为FAR过程，所以应在MEDIUM或LARGE计算模型下编译PL/M—86模块。

还须用LINK86和LOC86，将用户程序生成的目标模块转换成可被装入内存或编程于内存（或编程于ROM）的可执行程序。

所有上述软件均为运行于ISIS—II下8080/8085开发系统上的版本。也可以是借助系列III RUN服务程序运行于8086开发系统上的版本。

为保持一致性，我们给出的所有LINK86调用例子，运行的均为8080/8085版本，并假定LINK86和支持库都在驱动器0上，所有用户提供的文件都在驱动器1上。

支持库无须任何运行时的系统软件。只要你有了最后可执行的程序，此程序便可运行于任何NDP系统。

第二章　全8087仿真程序和接口程序库

概述

全8087仿真程序是一个可在iAPX86/10或iAPX88/10上执行的软件模块，它完整无误地模仿了8087的所有功能。仿真程序可以用来为配备了8087的系统开发软件原型，或者可利用它来完成只需8087浮点功能而不追求其速度的工作。仿真程序被放在文件E80-87中，可用LINK86语句将仿真程序同用户的程序模块连接起来。

与用户程序连接的还有三个接口程序库，E8087.LIB、8087.LIB和87NULL.LIB。可通过在LINK86语句中选择其中的一个程序库来决定用户程序的运行环境：运行中使用仿真程序，或使用8087部件，或不需要NPX的功能。

接口程序库还含有INIT87和INITFP两个过程，它们用于在各种环境情况下对NPX进行初始化。

全8087仿真程序是在同PL/M—86一起提供的非完全8087仿真程序PE8087基础上扩充起来的最新版本。LINK86语句中使用了E8087时不允许再出现PE8087。

本手册中凡提到“仿真程序”的地方，均指全8087仿真程序。

用户程序中怎样使用仿真程序

用户程序调用仿真程序，就象使用8087一样发指令就可以了。使用仿真程序的源程序与使用8087部件的源程序是完全相同的。

此外，在使用仿真程序和使用8087部件的还未连接的目标模块（或程序库）之间也没有什么区别。CEL87.LIB、DCON87.LIB和EH87.LIB都可以用仿真程序或8087部件来运行。

仿真程序和8087部件之间的选择是在调用LINK86的时候决定的。使用仿真程序时，应将文件E8087和E8087.LIB连接到用户的目标模块中；使用8087部件时，只要将8087.LIB一个文件连接进用户的目标模块。

源程序模块和目标在使用仿真程序和使用8087部件两种不同情况下的兼容，是通过一组专用的外部变量来建立的。所有Intel处理8087代码的翻译程序均会自动地生成这些变量。这些变量只有在某些LOC86的列表中才能看到。附录B中给出了这些变量的名。

非屏蔽异常的仿真

当8087检测到一个错误时，它便去查看它的控制字，确定该错误是屏蔽的还是非屏蔽的。若是屏蔽的，则在8087的一个输出引脚上产生一个中断信号。

仿真程序在同样的非屏蔽出错情况下，发出软件中断16（十进制）。这样就将过程INTERRUPT启动并运行；该过程的地址由内存单元00040（十六进制）中的指针

试读结束：需要全本请在线购买：　www.ertongbook.com

给出。

注意，在使用8087部件时，用户应注意配置自己的系统，对中断控制器进行编程，使之在8087检测到非屏蔽错误时产生一个完全相同的中断。因此，源程序代码，甚至包括用户的异常处理程序代码，均可完全保持原样不动。

仿真程序的内存和堆栈需求量

仿真程序理所当然地占据了8086的内存，使用8087部件时，这些内存是不用的。所用的内存分为四个段：

1．仿真程序放在一个叫做aqmCODE的段里，它一般是放在只读存贮器里的。该段占用了大约16K字节。

2．仿真程序使用了一个不可重入的数据块，它只能放在RAM中。该数据块大约有150个字节长，放在称为aqmDATA的段中。

3．仿真程序使用8087的软件中断16（十进制）和中断20至31（十进制）。这些中断的地址是通过INIT87或INITFP存放到内存单元00040（十六进制）至00043（十六进制）以及00050至0007F（十六进制）中的。只要还在使用仿真程序，这些单元的内容就不能被改动。

4．仿真程序使用了调用子程序所需的8086堆栈的大约60个字节。该堆栈须放在一个叫做STACK的段中。

NPX的初始化

当电源加到NDP系统上的时候，NPX必须被设置到一个已知的状态。这是通过调用一个8087初始化过程实现的。如果使用了仿真程序，则在执行浮点指令之前，必须装入中断20至31的地址。

我们在每个接口程序库中都提供了两个8087初始化过程，INIT87和INITFP。8087.LIB程序库对8087部件进行初始化，并设置8087控制字。E8087.LIB程序库在8087的初始化之前，装入中断20至中断31。87NULL.LIB程序库不给8087提供任何外部过程，它只能由不包含NPX指令的软件使用。

INIT87与INITFP之间的区别在于对8087控制字的缺省设置。INIT 87屏蔽所有的异常，INITFP屏蔽除“I”以外的所有异常。尽管以后可以对控制字进行修改，用户还是应该首先调用INIT87或INITFP。

INIT87或INITFP在PL/M—86中的用法

下面是INIT87的说明语句和调用语句，说明语句必须写在初始化模块的最上面，而调用语句则须在所有浮点运算执行之前给出。INITFP的语句也相同，只需将INIT87替换成INITFP。

因为INIT87和INITFP都是FAR过程，用户必须在MEDIUM或LARGE控制下编译PL/M—86模块。

```
INIT87: PROCEDURE  EXTERNAL;
```

```
END;
CALL  INIT87;
```

INIT87或INITFP在ASM—86中的用法

下面是INIT87的说明和调用指令，说明指令必须放在初始化模块的最上面，调用指令必须出现在过程之内。INITFP的指令也相同，只需将INIT87替换成INITF。

```
; The following line must appear outside of all SEGMENT-ENDS
; pairs
    EXTRN  INIT87: FAR

    CALL   INIT87          ; Set up the NPX
```

用户程序与仿真程序和接口程序库的连接

如果目标系统配备了8087部件，应将程序库8087.LIB连接到用户的目标模块中。仿真程序（E8087）和非完全仿真程序（PE8087）都不应在LINK86命令中出现。

如果目标系统没有配备8087部件，应将仿真程序E8087和接口程序库E8087.LIB连接到用户的目标模块中。非完全仿真程序PE8087不应在LINK86命令中出现。

如果用户程序调用了INIT87或INITFP，但并没有含任何其它浮点指令，应将87NULL.LIB连接到用户的目标模块中。

模块E8087可出现在LINK86命令的输入模块序列中的任何地方。接口程序库必须出现在所有含8087代码的模块的后面。

以下是我们所提示的目标模块在LINK86命令中的顺序：

用户目标模块

DCON87·LIB（如果使用了此模块）

CEL87·LIB（如果使用了此模块）

EH·LIB（如果使用了此模块）

8087·LIB(如果使用了8087部件)或者E8087，E8087.LIB（如果使用仿真程序）。

例：假定用户的程序是用ASM—86或PL/M-86生成的，它由两个模块构成：MYMOD1·OBJ和MYMOD2·OBJ。若程序要在一个使用仿真程序的系统上执行，则发命令：

```
-LINK86 :F1:MYMOD1.OBJ, :F1:MYMOD2.OBJ, &<cr>
>>:F0:E8087, :F0:E8087·LIB TO :F1:MYPROG.LNK
```

如果要修改程序使之能在一个配备8087部件的系统上运行，用户不必对源文件进行修改，只须将LINK86命令改成：

```
-LINK86 :F1:MYMOD1.OBJ, :F1:MYMOD2.OBJ, &<cr>
>>:F0:8087.LIB TO :F1:MYPROG.LNK
```

如果程序调用INIT87或INITFP，但并不含其它浮点指令，则可发命令：

```
-LINK86 :F1:MYMOD1.OBJ, :F1:MYMOD2.OBJ, &<cr>
>>:F0:87NULL.LIB TO :F1:MYPROG.LNK
```

第三章　十进制转换程序库

本章描述程序库DCON87.LIB的用法，这是一个由浮点数存贮格式之间的转换过程组成的程序库。

DCON87程序库过程概述

Intel为内部浮点数的表示提供了三种不同的二进制格式(见附录C)。DCON87·LIB能够实现这些二进制格式与ASCII编码的十进制数位串之间的转换。

二进制到十进制的转换过程mqcBIN_DECLOW，可以接收所有三种格式的二进制数。由于浮点数的输出格式太多，mqcBIN_DECLOW并不能提供一个包揽无遗的格式化的正文串。它给用户提供了一种“建立块”，用来建立能够满足规定格式的输出串。

十进制到二进制的转换过程mqcDEC_BIN接收由带可选正负号的十进制数、小数点、和/或以十为底的指数构成的正文串。它将该串转换成调用程序所选择的二进制格式。

还有一个可选用的十进制到二进制的转换过程是mqcDECLOW_BIN，它的输入格式类同于mqcBIN_DECLOW的输出格式。如果调用程序已经将十进制数按其组成部分分开了，mqcDECLOW_BIN还要用到一个低级的输入接口。

过程mqcLONG_TEMP、mqcSHORT_TEMP、mqcTEMP_LONG以及mqcTEMP_SHORT进行最长的二进制格式(TEMP_REAL)与较短的格式之间的浮点数转换。实行这种转换可以使得：输出为NaN当且仅当输入为NaN值。这样可以改善8087在存取其堆栈中较短格式的数据时所使用的转换算法。

所有DCON87过程的名均以三个字母“mqc”起头。我们之所以增加这个前缀，是为了尽量避免DCON87名同其它用户程序名发生冲突。为了使名字阅读时容易分辨，我们在本章中均以小写字母书写前缀“mqc”

DCON87还为Intel的某些翻译程序要使用的过程提供了公用名。这些公用名列于附录F。

ASM—86程序中DCON87过程的说明

在用到DCON87过程的源程序模块中，用户须在调用这些过程之前将其说明成外部过程。

下面是过程的说明语句，它们必须写在ASM—86程序的最前面。用户只需说明自己要用的过程。

所有DOCN87过程都必须是FAR过程。

这些ASM—86程序码中还包括了一些为建立参数方便而设立的说明语句。本章以后所给出的ASM—86例子中，均假定已经作过这些说明了。

为了引用方便，我们在例子中对每一个函数都重复了一遍说明语句。在用户的程序中不必作这样的重复。

```
; ASM—86  declarations for using  the DCON decimal
; conversion  library
; the  following EXTRN  statement  must appear outside of
; all SEGM-ENT—END     pairs

EXTRN mqcBIN_DECLOW:FAR
EXTRN mqcDEC_BIN:FAR
EXTRN mqcDECLOW_BIN:FAR
EXTRN mqcLONG_TEMP:FAR
EXTRN mqcSHORT_TEMP:FAR
EXTRN mqcTEMP_LONG:FAR
EXTRN mqcTEMP_SHORT:FAR

SHORT REAL EQU  0
LONG  REAL EQU  2
TEMP  REAL EQU  3

BIN_DECLOW_BLOCK STRUC
  BIN_PTR           DD    ?
  BIN_TYPE          DB    ?
  DEC_LENGTH        DB    ?
  DEC_PTR           DD    ?
  DEC_EXPONENT      DW    ?
  DEC_SIGN          DB    ?
BIN_DECLOW_BLOCK ENDS
DEC_BIN_BLOCK STRUC
  DD  ? ; The names of these fields are the same as in
  DB  ? ; BIN_DECLOW_BLOCK.
  DB  ?
  DD  ?
DEC_BIN_BLOCK ENDS
DECLCW_BIN_BLOCK STRUC
DD ? ; The names of these fields are the
DB ? ; same as in BIN_DECLOW_BLOCK
DB ?
```

```
    DD ?
    DW ?
    DB ?
    DECLOW_BIN_BLOCK_ENDS
```

PL/M—86程序中DCON87过程的说明

以下说明语句必须写在PL/M—86程序的最前面。用户只需说明自己所需要的过程。

这些PL/M—86程序码中也包括了为建立参数方便而设立的说明语句。本章后文有关DCON87过程的PL/M—86例子中，假定已经作过这些说明了。

为了使引用方便，我们在例子中对每一个函数都重复进行了一遍说明。用户不必在程序中作这样的重复。

使用DCON87.LIB时，因为DCON的过程都是FAR过程，故须用计算模型MEDIUM或LARGE来编译用户的PL/M—86模块。

请参见本章“一种DCON87的出错检测方法”一节，该节讨论了在什么情况下应该将过程说明为BYTE过程（来接收一个出错状态）。

```
    /*  PL/M—86 declarations for using the DCON decimal conversion
        library */

mqcBIN_DECLOW: PROCEDURE (BLOCK_PTR) EXTERNAL;
    DECLARE BLOCK_PTR POINTER;
END mqcBIN_DECLOW;

mqcDEC_BIN: PROCEDURE (BLOCK_PTR) EXTERNAL;
    DECLARE BLOCK_PTR POINTER;
END mqcDEC_BIN;

mqcDECLOW_BIN: PROCEDURE (BLOCK_PTR) EXTERNAL;
    DECLARE BLOCK_PTR POINTER;
END mqcDECLOW_BIN;

mqcLONG_TEMP: PROCEDURE (LONG_REAL_PTR,TEMP_REAL_
    PTR) EXTERNAL;
    DECLARE (LONG_REAL_PTR, TEMP_REAL_PTR) POINTER;
END mqcLONG_TEMP;

mqcSHORT_TEMP: PROCEDURE (SHORT_REAL_PTR,TEMP_
    REAL_PTR) EXTERNAL;
    DECLARE (SHORT_REAL_PTR,TEMP_REAL_PTR) POINTER;
END mqcSHORT_TEMP;
```

```
mqcTEMP_LONG: PROCEDURE (TEMP_REAL_PTR, LONG_REAL_
    PTR) EXTERNAL;
    DECLARE (TEMP_REAL_PTR,LONG_REAL_PTR) POINTER;
END mqcTEMP_LONG;
mqcTEMP_SHORT: PROCEDURE (TEMP_REAL_PTR,SHORT_REAL
    _PTR) EXTERNAL;
    DECLARE (TEMP_REAL_PTR,SHORT_REAL_PTR) POINTER;
END mqcTEMP_SHORT;

DECLARE SHORT_REAL LITERALLY '0';
DECLARE LONG_REAL LITERALLY '2';
DECLARE TEMP_REAL LITERALLY '3';

DECLARE BIN_DECLOW_BLOCK STRUCTURE (
  BIN_PTR POINTER,
  BIN_TYPE BYTE,
  DEC_LENGTH BYTE,
  DEC_PTR POINTER,
  DEC_EXPONENT INTEGER,
  DEC_SIGN BYTE);
DECLARE DEC_BIN_BLOCK STRUCTURE. (
  BIN_PTR POINTER,
  BIN_TYPE BYTE,
  DEC_LENGTH BYTE,
  DEC_PTR POINTER);
```

DCON87过程使用堆栈的方式

DCON87需要占用8087堆栈的176个字节作为它的内部存贮区。DCON 87模块将这些存贮区分配在STACK公共段。

任何DCON87过程都能在进入该过程时保留8087的所有状态，并在退出时恢复这些状态：

十进制转换程序库的精度

DCON87能保证一个18位的十进制数，经过到TEMP_REAL的转换并再转回十进制数，仍保持原样不变。但是DCON87不能保证一个TEMP_REAL数，转换成十进数再转回TEMP_REAL后，还是是原来的形式。这样的精度需要由超过TEMP_REAL精度的运算来实现，这是很慢的。对以上转换，精度的损失最多为三位二进制数。

IEEE标准（在第六章描述）规定了SHORT_REAL型和LONG_REAL型数 的

试读结束：需要全本请在线购买： www.ertongbook.com