Microsoft Macro 5.0版

混合语言程序设计指南

与

软件开发工具

H

中科先希望微机电话技术公司

■北京市新闻出版局
准印证号： 831138
■订购：北京海淀 8721 信箱
■电话： 2562329
■地址：北京海淀影剧院北侧
■交通：乘 320、332路汽车到海淀黄庄下车

IBM PC 混合语言程序设计指南

前 言

混合语言程序设计是由两个或多个源语言建立程序的过程。这种组合能使你组合microsoft的BASIC、C、FORTRAN、Pascal和宏汇编各种独特的功能。这些语言的任一种（它们的最新版本）都可调用任何其它语言。实际上，出自所有扩展语言程序库的例行程序都可以用于混合语言程序。

例如，混合语言程序设计帮助你有效使用汇编语言。你可以用microsoft C或Quick BASIC快速开发你的程序主要部分，然后针对那些要执行多次和必须以最快速度运行的少量例行程序调用汇编。

混合语言程序设计也便于从一种语言转移到其它。你可能有一个大的FORTRAN程序要转换为C。你可以用相应的C函数逐一替换你的FORTRAN子例行程序。在一旦写好每一函数时，C生成的代码就可投行出现。

最后，如果你正准备推出自己的程序库，混合语言程序设计特别有价值。用这里提供的技术，你可以对上述的任何一种语言产生程序库（往往带有少量的修改）。

本书把注意力集中在写混合语言程序所需的概念、语法和程序设计方法上。假定你对要组合的语言已有基本的了解，并已经知道如何用这些语言书写、编译和连接多个模块的程序。这里不准备介绍任何一种特定语言的程序设计基础。

目　　录

IBMPC混合语言程序设计指南

第一部分　混合语言接口

混合语言程序设计的这一部分解释如何在用 MicrosoftBASIC、C、FORTRAN Pascal和宏汇编程序编写的模块之间建立接口。这里广泛使用例子；但是，这些例子的特点是只对整型参数，它们是相对容易传送的。共享其它类型的数据（象字符串和数组）提出了特殊的问题，将在第二部分介绍。

第一章　混合语言程序设计基础

1.1　进行混合语言调用

混合语言程序设计总是涉及调用；特别是，它涉及函数，过程或子例行程序调用。例如，BASIC主模块可能需要执行你想独立编程的特殊任务。但是，代替调用 BASIC 子程序，你决定调用C函数。

混合语言调用必然涉及多个模块（至少对Microsoft语言如此）。代替用同一编译程序编译所有源模块，你使用不同的编译程序。在上述的例子中，你将用BASIC 编译程序编译主模块源文件，而用C编译程序编译其它源文件（用C语言书写的），然后，把两个目标文件连接在一起。

图1.1解释混合语言调用的语法如何生效（使用上述的例子）

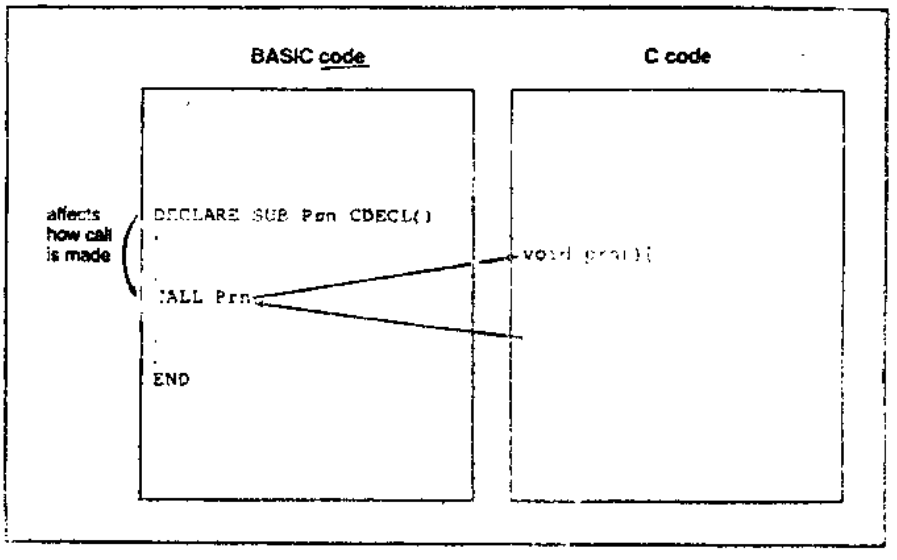


图1.1　混合语言调用

上面的解释中，BASIC调用C是CALL Prnl（类似于调用BASIC子程序）。但是

混合语言调用和两个BASIC模块间的调用二者之间有两点不同：1） 子程序Prn实际在C执行(使用标准的C语法)；2）BASIC调用的实现受DECLARE语句影响，它使用CDECL关键字以便建立与C的兼容。DECLARE语句（在第二章详细讨论）是混合语言“接口”语句的一个例子。每种语言提供它自己的接口形式。

不管语法的差异如何，函数、过程和FORTRAN子例行程序全都是类似的。主要差别是某些类型的例行程序送回值，而其它则不送。你可以交换具有送回值的例行程序，也可以交换不具有送回值的例行程序。（请注意，本书中的“例行程序”指可以由其它模块调用的任何函数、过程或子例行程序。）

表1.1示出不同语言的例行程序调用之间的对应。

表1.1 例行程序调用的语言等价物

语言	送回值	不送回值
BASIC	函数	子程序
	过程	
C	函数	(空) 函数
FORTRAN	函数	子例行程序
Pascal	函数	过程
宏汇编程序	过程	过程

例如，BASIC模块可以对FORTRAN子例行程序执行子程序调用。但为了调用FORTRAN函数，BASIC应进行FUNCTION调用，不然的话，虽然可以进行调用，但将丢失送回值。

注：BASIC DEF FN函数和GOSUB子例行程序不能被其它语言调用。

1.2 命名约定要求

术语“命名约定”指编译程序在把例行程序的名字放入目标文件之前，改变名字的方法。

当你进行混合语言调用时，采用兼容的命名约定很重要。如果被调用例行程序的名字在每一目标文件上不同，那末，连接程序将不能找到匹配，从而，它将报告不可能的外部引用。

Microsoft编译程序把机器码放在目标文件；但它们也把所有需要共同访问的例行程序和变量的名字放在那里。用这个办法，连接程序可以将一个模块中调用的例行程序和另一模块中定义的例行程序名字作比较，并识别匹配。名字是以ASCII（American Standard Code for Information Interchange）格式存放的。如果你使用DEBUG实用程序转储一个目标文件的字节，你可以精确地看到它们怎样被存放。

BASIC、FORTRAN和Pascal大致上使用同样的命名约定。它们翻译每一字母为大写。采用BASIC类型说明字符（%，&、！、#、$）。

但是每一语言识别不同数目的字符，FORTRAN识别任一名字的前6个字符，Pascal是前8个，而BASIC是前40个。如果名字比语言将识别的长，附加的字符简单地

不放入目标文件中。C使用十分不同的约定；C编译程序不翻译任何字母为大写，但是入引导的下划线（_）在每一例行程序的名字之前。C识别名字的前8个字符。

命名约定的差异由混合语言的关键字自动地为你处理。只要你遵循两个规则：

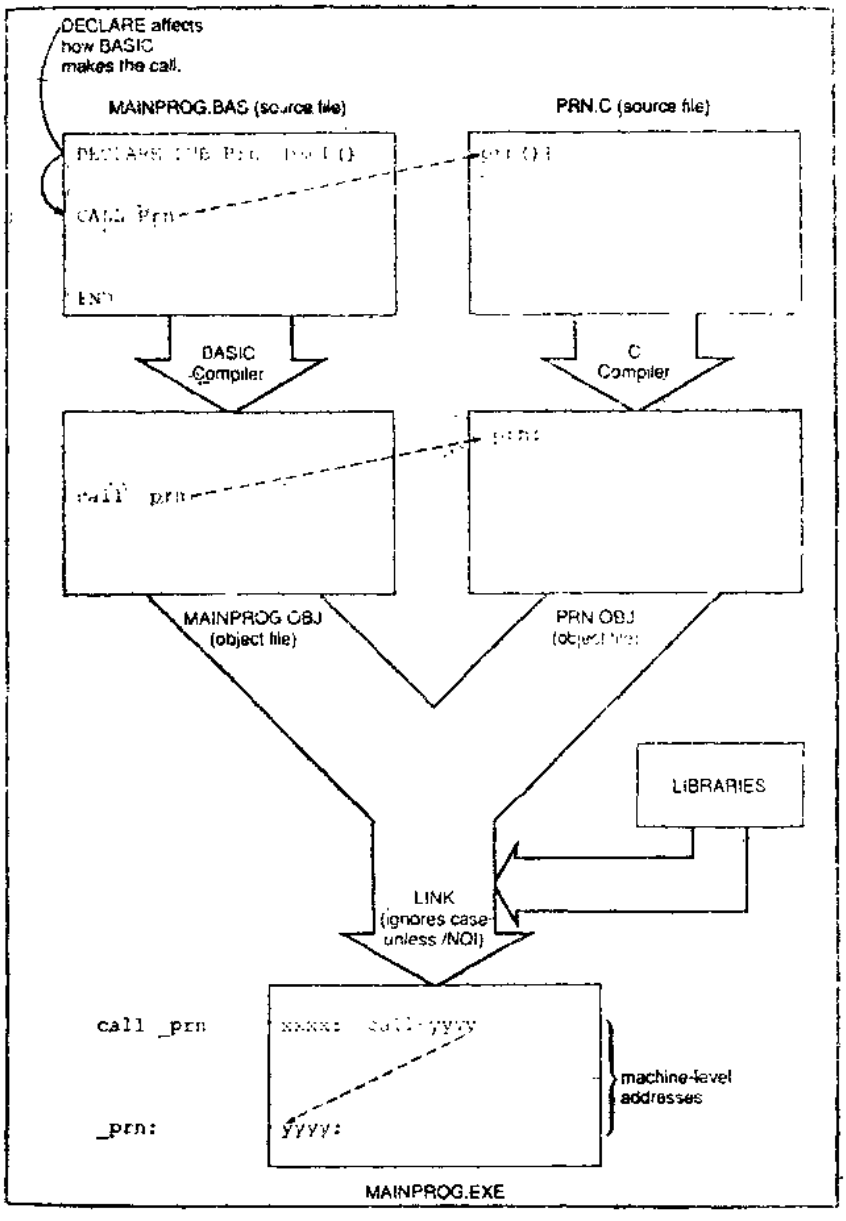
1、如果你正使用某些FORTRAN例行程序，所有名字长度应为6个字符或更少。

2、不要使用/NOIGNORE连接程序任选项（它使连接程序区分开prn和PRN）

对于C模块，这意味着你应小心不要依懒大小写字母的差别。

图1.2解释完整的混合语言开发例子，并示出该过程怎样引用命名约定。

图1.2　命名约定



在上面的例子中，注意BASIC编译程序把名字放入目标文件时，在Prn之前插入了引导的下划线。因为CDECL关键字指示BASIC编译程序使用C的命名约定。当使用这个关键字时，BASIC也将把所有字母转换为小写。（严格地说，转换字母为小写不是C的命名约定的一部分；但是，它是和大多数C程序的程序设计风格一致的。）

1.3 调用约定要求

术语“调用约定”指语言实现调用的方法。调用约定的选择影响编译程序为执行（返回自）函数、过程或子例行程序调用而生成的实际机器指令。

调用约定是低层的协议。两个涉及的例行程序（执行调用的例行程序和被调用的例行程序）认别同一协议是紧要的。否则，处理器可能接受不一致的指令，子是引起系统破坏。

调用约定的使用在两方面影响程序设计：1、调用例行程序使用调用约定以确定用什么次序“传送”自变量（参数）到其它例行程序。这个约定通常可以在混合语言接口说明。

2、被调用例行程序使用调用约定以确定用什么次序“接受”传送给它的参数。在大多数语言中，这个约定可以在例行程序的标题中说明。但是，BASIC总是使用它自己的约定以接受参数。

换句话说，每一对例行程序的“调用”使用某些调用约定，而每一例行程序“标题”说明或假设某些调用约定。这两个约定必须兼容。对于除BASIC以外的每一种语言，改变各自的调用约定是可能的。但是，通常最简单是采用被调用例行程序的约定。例如，C函数将使用它自己的约定去调用其它C函数，而使用Pascal的约定去调用Pascal。

BASIC、FORTRAN和Pascal使用同样的标准调用约定。但是，C使用十分不同的约定。

注：下面的几节讨论一些调用约定的细节。高级语言程序员理解这些细节是不困难的；程序员只需知道不同的约定彼此互不兼容。

Microsoft BASIC FORTRAN和Pascal的调用约定各自以参数出现在源码中的次序把参数下推入堆栈。例如，BASIC语句CALL calc（A、B）在下推B之前把自变量A下推在堆栈。这些约定还说明正好在返回控制到调用程序之前，由被调用例行程序恢复堆栈。（通过移去参数恢复堆栈。）

C的调用约定以参数在源码中出现的逆序把参数下推在堆栈。例如，C函数调用calc（a，b）；在它下推a之前把b下推在堆栈。和其它高级语言相反，C的调用约定指出调用例行程序总是在被调用例行程序返回控制之前直接恢复堆栈。

BASIC、FORTRAN和Pascal的约定产生较少的目标码。但是，C约定能以可变数目的参数进行调用。（因为第一个参数总是下推的最末一个，它总是在堆栈的顶；因此，不管实际上传送了多少参数，它有相对于结构指针的同一地址。）

1.4 传送参数要求

1.3节讨论两个例行程序用以彼此通信的协议（调用约定）；本节涉及各片数据（参数）实际上是为你如何发送的。

如果你的例行程序在参数如何转发送问题上不一致，那末，被调用例行程序将接受不正确的数据。程序还可能引起系统破坏。

Microsoft编译程序支持传送参数的三种方法：

方法	描述
按近引用	传送变量的近（偏移）地址。这个方法使被调用例行程序直接存取变量自身。任何例行程序对参数的改变将反映到调用例行程序中。
按远引用	传送变量的远（段）地址。这个方法除了传送较长的地址之外，类似按近引用传送。这个方法比按近引用传送慢，但当你传送缺省数据段以外的数据时，这是必要的。（这在BASIC或Pascal不成问题，除非你特殊请求远存储。）
按值	只传送变量的值，而不是地址。用这个方法，被调用例行程序知道参数的值，但不访问原来的变量。一旦例行程序终止时，值参数的改变不影响调用例行程序中的参数值。

有不同的参数传送方法的事实给混合语言程序设计带来两点麻烦。

首先，你需要确保被调用例行程序和调用例行程序使用同样的方法传送每一参数（自变量）。在大多数情况下，你需要检验每一语言所用的参数传送缺省，并作出可能的调整。每一语言都有允许你改变参数传送方法关键字或语言特色。其次，你可能需要使用特定的参数传送方法而不是使用任一语言的缺省。（事实上，因为程序逻辑，第二至五章的例子特殊地要求一种特定的方法或其它。）

表1.2 概述每一语言的参数传送缺省值

语言	近引用	远引用	按值
BASIC	全部		
C	近数组	远数组	非数组
FORTRAN		全部	全部
Pascal	VAR,CONST	VAR,CONSTS	其它参数

当PASCAL或C属性应用于FORTRAN例行程序时。按值传送变成缺省值。

每一语言都有重设这些缺省的方法，它们在第七章列出。

1.5 送译和连接

在你已书写源文件并解决在1.2至1.4节所产生的问题以后，你将准备编译各个模块并随后把它们连接在一起。

1.5.1 用适当的存储模型编译

对于Microsoft BASIC FORTRAN和Pascal不要求特殊的任选项以编译作为部分混合语言程序的源文件。但是，对于Microsoft C，你需要注意不是所有存储模型都是和别的语言兼容的。BASIC、FORTRAN和Pascal只用远（段）代码编址。因此，你必须总是以中的、大的或巨大的模型编译C模块，因为这些模型也用远代码编址。当调用C或由C调用时，编译小的或紧致模型将引起混合语言程序破坏。如果你应用far关键字到C函数定义，以说明该函数使用远调用或返回，这个问题可以避免。）

上节涉及代码地址的大小。数据地址大小的差异可以通过编译任选项或在源码中解决。存储模型的选择影响C和FORTRAN中缺省的数据指针大小，虽然这个缺省可以用near和far重设。对于C和FORTRAN，存储模型的选择，还影响数据实体是否定位在缺省数据段，如果数据实体没有定位在缺省数据段，那末，它不能按近引用直接传送。

1.5.2和语言程序库连接

在大多数情况下，连接用不同语言编译的模块可以很容易完成。下列任一措施将确保所有要求的程序库以正确的次序连接。

- 把所有语言程序库象源文件一样放在同一目录中。
- 在LIB环境变量中列出包含所有必需程序库的目录。
- 让连接程序针对程序库向你提出。

在上述的每一种情况中，连接程序按要求的次序找出程序库。如果你在命令行键入程序库，那末，它们必须按特定的次序键入。

但是，如果你正使用FORTRAN的4.0版本产生你的模块之一，那末，你将需要用/NOD（没有缺省程序库）连接，并直接在连接命令行上指定你所需要的全部程序库。你也可以用一个自动应答文件（或批文件）指定这些程序库，但你不能用缺省程序库搜索

对于FORTRAN 4.0的C兼容程序库。要确保在你指定C程序库之前指定FORTRAN程序库，除非你的C版本比FORTRAN版本更新。在那种情况下，先指定C程序库。此外，你指定的任何其它程序库可按任意次序进入。

如果你正在LINK命令行上列出BASIC程序库，那末，BASIC程序库必须在所有其它之前。

例子

```
LINK/NOD mod1 mod2,,,CRAFX+LLIBFORE+LLIBC
```

上面的例子连接两个目标模块以及FORTRAN4.0和C的大模型程序库。此外，还连接一个额外的程序库GRAFX。

第二章 BASIC调用高级语言

Microsoft BASIC支持对用Microsoft C、FORTRAN和Pascal写的例行程序的调用。本章描述调用这些其它语言的必要语法，并随后给出每一种BASIC和其它语言组合的例子。在这些例子中只用整数作为参数。

本章以列出使用C标准程序库函数的限制的一节为结束。如果你正使用某些存储分配或系统程序库函数，可查阅这一节。

2.1 BASIC与其它语言的接口

BASIC DECLARE语句提供一种对其它语言的灵活而方便的接口。它可用于Microsoft Quick BASIC4.0及其后版本。不提供DECLARE语句的版本也不提供和其它语言兼容的程序库，这些早期的版本限制使用混合语言程序库，因为它们不能成功地调用能利用程序库的C、FORTRAN或Pascal的例行程序。

下面概述DECLARE语句的语法。

2.1.1 DECLARE语句

当你调用函数时，DECLARE语句语法如下：

DECLARE FUNCTION 名字〔CDECL〕〔ALIAS“别名”〕〔（参数表）〕

当你调用子程序时，语句语法如下：

DECLARE SUB名字〔CDECL〕〔ALIAS“别名”〕〔（参数表）〕

“名字”域是你希望调用的函数或子程序过程的名字（就象它出现在BASIC源文件中一样）。这是使用DECLARE语句调用其它语言的推荐步骤：

1、对于每一个你计划调用的独特的中间语言例行程序，在调用例行程序之前，把一个DECLARE语句放入BASIC源文件。例如，你的程序可能五次不同地调用子程序Maxparam（每次使用不同的自变量）。但是，你只需说明Maxparam一次。理想地，DECLARE语句应放在靠近源文件的开端。

2、如果你正调用C模块的例行程序，在DECLARE语句使用CDECL（除非C例行程序用Pascal或fortran关键字说明）。CDECL指示BASIC在后来每次对“名字”的调用中使用C的命名和调用约定。对Pascal或FORTRAN不提供类似的关键字，因为它们每一个都使用和BASIC一样的调用约定。

3、如果你正调用带有比6个字符长的名字的FORTRAN例行程序，或带有比8个字符长的名字的C或Pascal例行程序。那末，使用ALIAS特色，ALIAS的使用在下一节解释。

4、使用参数表以确定每一参数是如何传送的。参数表的使用在下面（紧接在有关ALIAS信息后的一节）解释。

5、一旦例行程序被适当地说明。调用它就如同你调用BASIC子程序或函数一样。

下面的讨论解释其它字段。

2.1.2 使用ALIAS

如上所注，ALIAS的使用可能是必需的，因为FORTRAN只把名字的前6个字符放入目标文件，而C和Pascal各自放前8个，但BASIC却把多至40个名字的字符放入目标文件。

注：你不必用ALIAS特色除去类型说明符（%、&、!、#、$）。当BASIC生成目标码时，它自动除去这些字符。于是，BASIC的Fact%匹配Pascal的Fact。

ALIAS关键字指示BASIC把“别名”放入目标文件，而不是“名字”。BASIC源文件仍然包含对“名字”的调用。但是，这些调用解释为就象它们是对“别名”的调用一样。

例子

```
DECLARE FUNCTION Quadratic%ALIAS "QUADRA" (a, b, c)
```

在上例中，QUADRA（“别名”）包含Quadratic%（“名字”）的前6个字符。这使得BASIC把QUADRA放入目标码，因此，模仿了FORTRAN的动作。

2.1.3 使用参数表

下面显示参数表语法，接着是每一字段的解释。请注意，你能使用BYVAL或SEG，但不能同时都用。

语法

```
〔BYVAL | SEG〕 变量〔AS类型〕…,
```

使用BYVAL关键字以说明值参数。在每一后来的调用中，相应的自变量将按值传送（C和Pascal模块的缺省方法）。

注：BASIC提供两种“按值传送”的方法。按值传送的一般方法是使用额外的一组括号。如：

```
CALL HOlm ((A))
```

这个方法实际上建立一个临时值，传送它的地址。BYVAL提供按值传送的真正方法，因为它传送值本身，而不是地址。只有通过使用BYVAL，BASIC程序才和期待值参数的非BASIC例行程序兼容。

使用SEG关键字以说明远引用参数。在每一后来的调用中，将传送相应自变量的远（段）地址（FORTRAN模块的缺省方法）。

你可以为“变量”选择任何一种合法的名字；只要求和名字相关的类型对BASIC有一定意义。和对其它变量一样，类型可以用类型说明符（%、&、!、#、$）或由隐含说明指定。

你可以使用AS“类型”重设“变量”的类型说明。“类型”域可以是INTEGER、LONG、SINGLE、DOUBLE、STRING、用户定义类型或ANY（它指示BASIC允许传送任何类型的数据作为自变量）。

例子