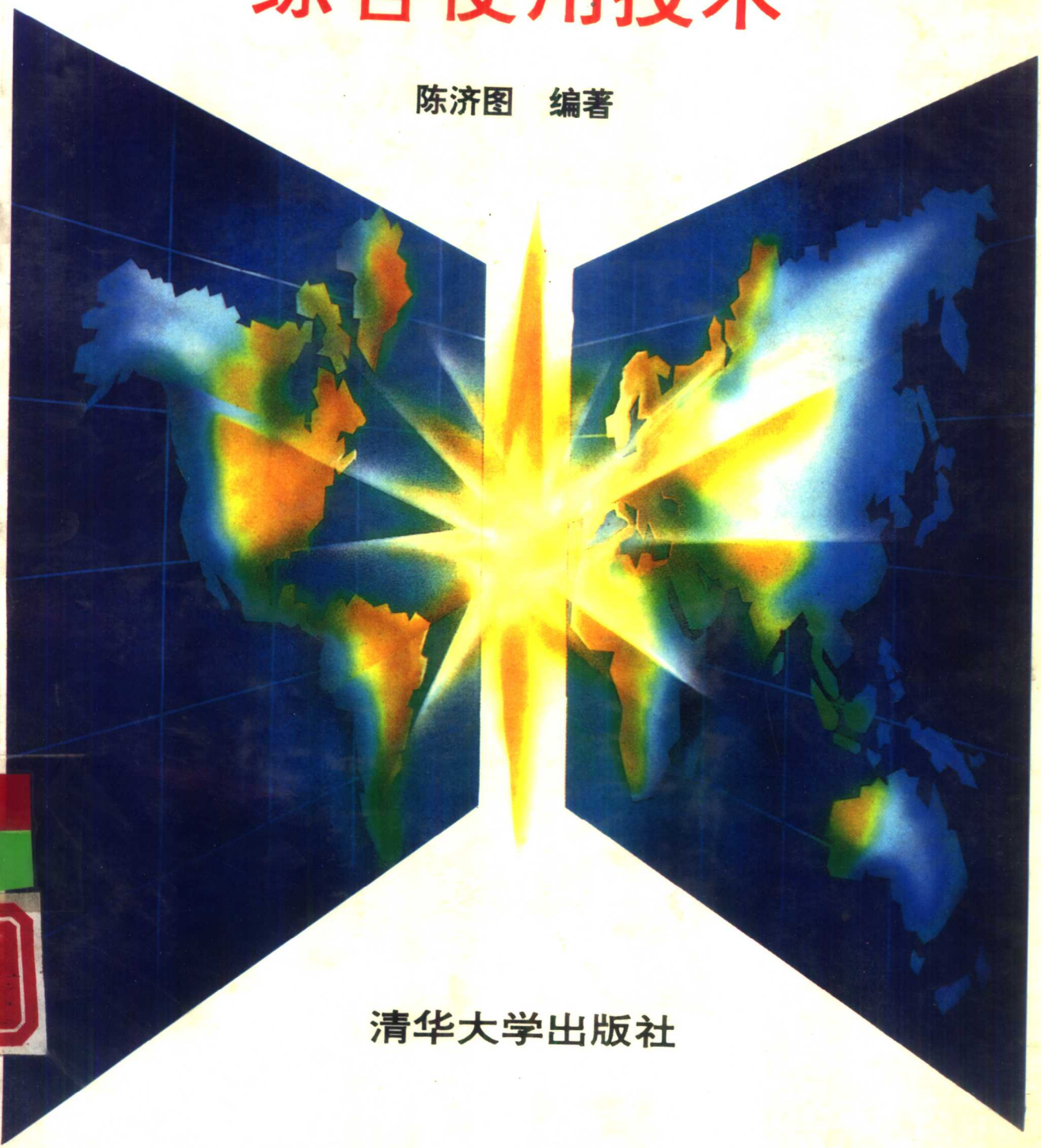
北京科海培训中心系列教材

VAX/VMS 系统资源综合使用技术

陈济图 编著



清华大学出版社

北京科海培训中心系列教材

VAX／VMS系统资源综合使用技术

陈济图　编著

清华大学出版社

内 容 提 要

本书从开发应用系统，尤其是实时多任务在线的应用系统的角度出发，介绍如何利用 VAX / VMS 系统资源解决应用系统中繁杂多样的问题，诸如过程调用，屏幕管理，进程的建立、调度和通讯，I / O 设计，程序模块及数据的共享，多任务应用系统中内存数据库的设计以及扩充应用接口等。书中提供了 90 多个程序例子。本书是从事 VAX 计算机和华南、太极计算机的技术人员的实用工具书，也可供大专院校师生参考。

（京）新登字158号

VAX / VMS 系统资源综合使用技术

陈济图 编著

清华大学出版社出版

（北京 清华园）

门头沟胶印厂印刷

新华书店总店科技发行所发行

开本：787×1092 1 16 印张：10.75 字数：268 千字

1992 年 4 月第 1 版 1992 年 4 月第 1 次印刷

印数：0001—5000

ISBN 7-302-01061-7 / TP · 394

定价：8.00 元

前　言

七十年代以来，VAX 系列计算机已发展到台式机到大型机的各个系列，在世界各地得到广泛的应用。VAX 系列机已成为世界上公认的优秀机型。在我国已安装有各型 VAX 机数千台，且国产的与 VAX 兼容的华南（HN）2000 及 3000 系列机、太极（TJ）2000 系列机已成为我国小型机的主流机型和带头产品，国内 VAX 机及其兼容机的用户不断增加。VAX 系列计算机硬件和软件资源丰富，具有开发各个领域的应用系统所必须的良好开发环境，因此，如何更好地使用它，使其充分发挥作用是应用软件人员面临的一大课题。

开发一个应用系统，尤其是一个多任务实时在线系统，与一个用于科学计算的程序大为不同，单纯用标准的高级语言是远远不够的，需要利用操作系统提供的资源才能解决应用系统中各种繁杂多样的问题。编写本书的目的在于向读者介绍在 VAX／VMS 上开发一个应用系统需要考虑利用的一些系统资源及其基本的使用技术。

VAX／VMS 从台式机到大型机的所有系列都是全面兼容的，即都使用同一操作系统 VMS，并且在 VAX 机的发展中仍保持着兼容性。因此，掌握了利用系统功能的设计方法，在任何 VAX 的机型上都可应用自如。在 VAX／VMS 上既可以开发管理系统，又可以开发过程控制系统，其系统功能十分丰富，用户界面良好，开发效率高。进入 90 年代，VMS 进一步增强了它的开放性。因此，软件人员在 VAX／VMS 上拓展自己的技术业务有着广阔的前景。

VAX／VMS 的系统资源极为丰富，本书所述内容，只是其中一部分。由于灵活利用系统资源解决应用系统中的问题有一定难度，面对大量的 VAX／VMS 技术资料，你可能不知如何下手，通过本书的学习，可以较快较容易地入门。你还可以以本书为线索，进一步查阅有关 VAX 技术资料，就可以更深入地解决应用系统开发中的问题。本书也可以作为一本工具书，书中所列举的 90 余个程序例子，定能帮助你解决若干具体问题。

本书的部分内容曾载于《利用 VAX／VMS 系统功能进行程序设计》小册子中，它在 VAX 协会内部交流。但数量有限，有的大学及计算中心进行了翻印并作为学习班教材。此次修改了原书错误，并扩充了大量新的内容。

编写本书过程中，李景淑工程师为本书编制了部分例题程序并帮助收集资料，书中例子承 VAX 协会山西分会理事阎有朋高级工程师在 VMS 4.4、4.7、5.2 等版本的操作系统上测试通过，在此深致谢意。

编　者

1991.6

目　录

第一章　过程调用

过程包括子程序过程和函数过程，所讨论的过程属于外部过程。VAX-11 高级语言程序中的过程调用，可以是下列情况之一：

(1) 调用同种语言编写的过程；

(2) 调用与调用程序不同的语言编写的过程；

(3) 调用 VAX／VMS 系统过程。

VAX 常用的高级语言，如 FORTRAN、BASIC、PASCAL、COBOL 等，既可单独使用，又可以互相联系和渗透。这需要掌握不同语言间的过程调用方法。

·只要遵守一些共同的约定，程序中不同模块和过程可用不同语言编写，熟悉不同语言的人可以合作。

·不同语言在功能上有各自特长和不足，可以在一个应用中发挥不同语言的特长，从而实现得更完美。

VAX／VMS 系统功能丰富，VAX 高级语言可以调用系统过程，从而可以用高级语言在较低的层次上进行开发，充分利用系统功能。本书 2—6 章所述的问题主要是调用系统过程中要解决的问题。

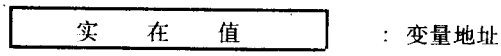
本章主要讨论调用不同语言编写的过程及高级语言调用系统过程，并用实例解释调用方法的技术细节。

1.1　不同语言程序之间过程的相互调用

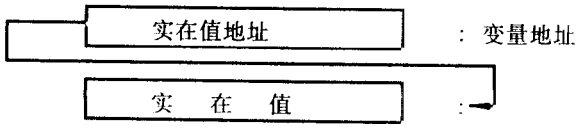
1.1.1　过程的变量传递机构

调用与调用程序不同的语言编写的过程，必需解决变量传递机构的匹配问题。编译程序为过程变元建立变量表，不同语言的编译程序所建立的变量表，形式有所不同，故变量的传递方式有所不同。VAX 变量传递机构有三种方式：

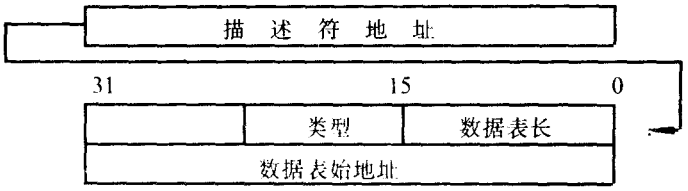
·靠值传递，该变量单元中直接放数据。格式如下：



·靠引用传递，变量单元中是一个地址，该地址为数据单元的地址（或首地址）。格式如下：



• 靠描述符传递，变量单元中是一个地址，该地址指向描述符，描述符是两个长字，其包含数据的有关信息和数据表的首址，格式如下：



高级语言中的描述符和数据表由编译程序建立，MACRO 程序由程序员编程时建立。

不同的语言，对于不同类型的变量，其传递机构有所不同。VAX 几种语言的变量传递机构如表 1.1.1。

表 1.1.1　不同语言中不同类型变量的传递机构

语言类型	不同类型的变量传递机构		
	数字型	字符型	数组型
BASIC	引　用	描 述 符	描 述 符
COBOL	引　用	引　用	…
FORTRAN	引　用	描述符	引　用
PASCAL	引　用	引　用	引　用
MACRO	传递方式由程序决定，要自行建立变量表		

当调用同种语言编写的过程时，无需标注变量的传递机构。当调用不同语言编写的过程时，或当调用程序供给实元的传递机构与被调过程虚元的传递机构不同时，则应按过程语言要求的传递机构，按调用程序的传递机构标识标注出来，编译程序才能用实元按过程要求建立变量表，变量才能正确传递。当两者在各自的语言中的传递机构相同时，则无需标注变量传递机构标识。

几种语言的变量传递机构标识如表 1.1.2。

表 1.1.2　不同语言的变量传递机构标识

传递机构	传 递 机 构 标 志			
	BASIC	COBOL	FORTRAN	PASCAL
靠值传递	BY　VALUE	BY　VALUE	%VAL	%IMED
靠引用传递	BY　REF	BY　REFERENCE	%REF	%REF
靠描述符传递	BY　DESC	BY　DESCRIPTOR	%DESCR	%DESCR
靠串描述符传递	BY　DESC	BY　DESCRIPTOR	%DESCR	%STDESCR

不同的语言，变量传递机构标识的具体标注方法有所不同，下面以调用子程序SUB1,SUB2,SUB3为例说明标注方法。

FORTRAN 程序的标注方法：

- 子程序的语言要求变元 A 靠值传递。

```
CALL    SUB1(%VAL(A))
```

- 子程序的语言要求变元 B 靠引用传递。

```
CALL    SUB2(%REF(B))
```

- 子程序要求变元 C 靠描述符传递。

```
CALL    SUB3(%DESCR(C))
```

BASIC 程序标注方法：

- 子程序的语言要求变元 A 靠值传递。

```
CALL    SUB1(A    BY    VALUE)
```

- 子程序的语言要求变元 B 靠引用传递。

```
CALL    SUB2(B    BY    REF)
```

- 子程序要求变元 C 靠描述符传递。

```
CALL    SUB3(C    BY    DESC)
```

COBOL 程序标注方法：

- 子程序的语言要求变元 A 靠值传递。

```
CALL    "SUB1"    USING    BY    VALUE    A.
```

- 子程序的语言要求变元 B 靠引用传递。

```
CALL    "SUB2"    USING    BY    REFERENCE    B.
```

- 子程序要求变元 C 靠描述符传递。

```
CALL    "SUB3"    USING    BY    DESCRIPTOR    C.
```

PASCAL 语言标注方法：

- 子程序的语言要求变元 A 靠值传递。

```
SUB1(%IMMED    A);
```

- 子程序的语言要求变元 B 靠引用传递。

```
SUB2(%REF    B);
```

- 子程序要求变元 C 靠描述符传递。

```
SUB3(%DESCR    C);
```

如果外部过程是一个函数过程，则将函数值赋给一个符合函数类型要求的变量，或将函数本身放在表达式中。

以调用函数 FUNC(A)为例，变元 A 要求靠值传递，函数值赋给变量 X：

- FORTRAN 程序：

```
X=FUNC(%VAL(A))
```

- BASIC 程序：

```
X=FUNC(A    BY    VALUE)
```

- COBOL 程序

```
CALL    "FUNC" USING    BY    VALUE    A    GIVING    X.
```

- PASCAL 程序：

```
X:=FUNC(%IMMED    A);
```

1.1.2 不同语言程序的具体调用方法

除变元的传递机构外，在调用中要符合不同语言的具体规则，现以例子说明。

下面的例子，列出了四种语言的主程序和子程序以说明它们之间相互调用的方法。主程序将四种不同类型的变量：整形数、字符数据、数组和实型数，通过外部子程序过程的变元传递至程序中并显示出来。注意相互调用的特点：

(1) 不同语言主程序对外部过程的说明不同；

(2) 传递机构的表达方式不同；

(3) 调用时主程序引用子程序的过程名而不是子程序文件名（初学者往往在此搞错）；

(4) 分别用各自的语言编译程序编译各主程序及子程序源文件，生成各自的目标文件，连接时用主程序与欲调的子程序目标文件同时连接。

1. FORTRAN 主程序 FOR__MAIN.FOR 调 PASCAL 子程序 PAS__SUB.PAS

FOR__MAIN.FOR(文件名)

```
C       FOR__MAIN.FOR
        INTEGER TOTAL /255/
        CHARACTER*15 NAME /'CHEN JI TU'/
        INTEGER ARRA(4)
        REAL AVRI
        EXTERNAL PROCP
        DO I=1,4
        ARRA(I)=I
        END DO
        AVRI=3.1415
        CALL PROCP(TOTAL,%REF(NAME),ARRA,AVRI)
        END
```

PAS__SUB.PAS(文件名)

```
{PAS__SUB.PAS}
MODULE PROCP (INPUT,OUTPUT);
TYPE
   TRING__TYPE=PACKED ARRAY[1..20] OF CHAR;
   AR__TYPE=ARRAY[1..4] OF INTEGER;
VAR I: INTEGER;
[GLOBAL] PROCEDURE PROCP(T: INTEGER;
   NA :TRING__TYPE;
   AR:AR__TYPE;AV:REAL);
BEGIN
   WRITE(T,NA,AV);
   FOR I:=1 TO 4 DO
   WRITE (AR[I]);
END.
```

在主程序 FOR__MAIN.FOR 中，用 EXTERNAL PROCP 说明外部子程序过程 PROCP，用 CALL 语句调子程序过程，查表 1.1.1,FORTRAN 与 PASCAL 仅字符变量 NAME 的传递机构不同，故用 PASCAL 的传递机构（靠引用）。查表 1.1.2，用 FORTRAN 的传递机构标识 %REF(NAME) 标注出。

在子程序 PAS__SUB.PAS 中，用 MODULE 说明模块名，在过程说明中用[GLOBAL]属性说明为外部过程，并对过程参数作类型说明，注意与主程序字符变量 NAME 匹配的数据类型为定长压缩字符数组型。

当 FORTRAN 主程序调 BASIC 过程时，除过程名匹配外，仅调过程语句改为

```
CALL   PROCF(TOTAL, NAME, %DESCR (ARRA), AVRI)
```

2. BASIC 主程序 BAS__MAIN.BAS 调 FORTRAN 子程序 FOR__SUB.FOR

BAS__MAIN.BAS(文件名)

```
10!  PROGRAM  BAS__MAIN.BAS
20   DECLARE  INTEGER  TOTAL,ARRA(4)
30   DECLARE  STRING  NAM
40   EXTERNAL  SUB  PROCF(INTEGER,STRING,INTEGER & DIMENSION ( )
       (BY  REF,REAL)
50   FOR  I=0  TO  3
     ARRA(I)=I+1
     NEXT  I
60   TOTAL=255
70   NAM="CHEN  JI__TU"
90   CALL  PROCF(TOTAL,NAM,ARRA(  ),AVRI)
95   PRINT  AVRI
100  END
```

FOR__SUB.FOR

```
C    FOR__SUB.FOR
     SUBROUTINE  PROCF(TO,NA,AR,AV)
     INTEGER  TO
     CHARACTER*50  NA
     INTEGER  AR(4)
     TYPE  *,TO
     TYPE  *,NA
     DO  I=1,4
     TYPE  *,AR(I)
     END  DO
     AV=3.1415
     RETURN
     END
```

与 FORTRAN 主程序不同的是，在 BASIC 主程序中，对外部过程的说明同时说明了变元的类型及传递机构（40 行），在调过程 CALL 语句中，则无需再标注传递机构。也可以将传递机构放在 CALL 语句中，而不放过程说明中。注意仅数组变量的传递机构按 FORTRAN 要求为靠引用传递，用 BASIC 的方式标为 BY REF。可见，FORTRAN 子程序十分简单。

若 BASIC 主程序调 PASCAL 子程序 PAS__SUB.PAS 仅需将 BAS__MAIN.BAS 中外部过程说明语句的字符变元说明改为 STRING BY REF 即可。

3. PASCAL 主程序 PAS__MAIN·PAS 调 FORTRAN 子程序 FOR__SUB.FOR

PAS__MAIN.PAS

```
{PAS__MAIN.PAS}
PROGRAM  PROC(INPUT,OUTPUT);
TYPE
   TRING__TYPE  =  VARYING[50]  OF  CHAR;
```

```
    AR__TYPE = ARRAY[1..4] OF INTEGER;
VAR
    TOTAL: INTEGER;
    NAME: TRING__TYPE;
    ARRA: AR__TYPE;
    AVRI: REAL;
    I: INTEGER;
[EXTERNAL] PROCEDURE PROF(%REF TOTAL : INTEGER;
    %DESCR NAME : TRING__TYPE;
    %DESCR ARRA: AR__TYPE;
    %REF AVRI: REAL); EXTERNAL;
BEGIN
    TOTAL := 255;
    NAME := 'Chen ji-tu';
    FOR I := 1 TO 4 DO
    ARRA[I] := I;
    PROF(TOTAL, NAME, ARRA, AVRI);
    WRITE (AVRI);
END.
```

PASCAL 语言中的过程本书称子程序过程，PASCAL 语言中的函数本书称函数过程。

同 BASIC 主程序类似，PASCAL 主程序 PAS__MAIN.PAS 在外部过程说明中同时对参数(习惯上 PASCAL 过程中变量称参数)的类型及传递机构进行说明。在调过程语句中仅标出实参名。传递机构亦可放在调过程语句中。程序中对每个参数均标出了 FORTRAN 需要的传递机构，若仅标出字符型参数也可。注意主程序字符变量可用变长字符数组型。

PASCAL 调 BASIC 子程序 BAS__SUB.BAS，不同点仅是过程说明语句中传递机构按 BASIC 要求标注，过程名要对应。

BAS__SUB.BAS

```
10 SUB PROF(INTEGER T,STRING NA,INTEGER AR( ),REAL AV)
40 PRINT T,NA,AV
45 FOR I=1 TO 4 \ PRINT AR(I) \ NEXT I
50 END SUB
```

4. COBOL 主程序 COB__MAIN.COB 调 FORTRAN 子程序 FOR__SUB1 · FOR

COB__MAIN.COB

```
IDENTIFICATION DIVISION.
PROGRAM-ID. COB__MAIN
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
    01 NAME   PIC X(11).
    01 TOTAL PIC S9(4) COMP VALUE 255.
    01 AVRI  COMP-1 VALUE 3.1415.
PROCEDURE DIVISION.
BEGIN.
        MOVE "CHEN JI__TU" TO NAME.
        CALL "PROCF1" USING TOTAL,
        BY DESCRIPTOR NAME,
```

```
        BY REFERENCE AVRI.
THE-END.
        STOP RUN.

                        FOR_SUB1.FOR
C       FOR_SUB1.FOR
        SUBROUTINE PROCF1 (TO,NA,AV)
        INTEGER TO
        CHARACTER*50 NA
        TYPE *,TO,NA,AV
        RETURN
        END
```

COBOL 主程序无需对过程说明。在工作存储字节中对变元说明时整数用 COMP 型，实数用 COMP_1 型，在调过程的 CALL 语句中，注意变元 NAME 及 AVRI 的传递机构标注。实型数变元 AVRI 的 COBOL 缺省传递机构亦是引用，但此处不能省略，因它前面的变元 NAME 标注为描述符，若此处省略时，AVRI 传递机构与 NAME 相同，则会出错。在 COBOL 程序中数组变量的传递较复杂，未列出。

1.1.3 变量传递的其它方式

对不同语言的过程调用时，除了通过过程变元（参数）进行变量传递外，还可以通过多种方式进行传递，它们是：

- 公共区
- PI COMMON
- DCL 符号
- 逻辑名
- RMS 文件

详见第三章进程间通讯及第六章数据共享部分。

1.2 调用 VAX / VMS 系统过程

可以在高级语言程序中调用的系统过程有下列几类：

- 运行时库 (RTL)
- 系统服务 (SYSTEM SERVICES)
- 记录管理服务 (RMS)
- 命令语言解释器 (CLI)

关于调用的详细使用说明请参考下列资料：

《VAX / VMS Run_Time_Library Reference Mannul》

《VAX / VMS System Servies Reference Manual》

《VAX Record Management Services Reference Manual》

《VAX Utility Reference Maunal》

本书主要涉及运行时库及系统服务。对这两类过程，还可以用 DCL 命令 $ HELP 查阅每个过程的有关信息，具体命令如下：

```
$ HELP    RTL                查运行时库过程
$ HELP    SYSTEM             查系统服务过程
```

下面简述对系统过程调用的一般性问题。

1.2.1 调用系统过程的一般方法

1.2.1.1 调用方式

可以按子程序方式或函数过程方式调用。按函数过程调用时，调用过程成功与否的状态码及错误条件作为整形长字回送给过程名（少数过程仅返回状态码），此时，函数名及接受函数值的变量均应定义为无符号长字或整型长字。予计过程能成功执行，则可以作为子程序调用，一般情况下推荐作为函数过程调用，以便在调试程序及运行中追踪出错原因及进行错误处理。

1.2.1.2 传递机构标识的标注

在 RTL 过程及系统服务过程中，变量传递机构亦有三种方法：靠值、靠引用、靠描述符。每个过程的每个变元要求何种传递机构，均在该过程的参考手册中说明，或由命令 $ HELP 查询。传递机构标识的标注方法同 1.1 节中的说明。

1.2.1.3 可选变元

过程中的变元有两类，必要的变元及可选变元。后者用方括号[ ]括起来。可选变元均有缺省值。当省略某可选变元时，该变元采用缺省值，多数情况下，系统服务的数字型变元的缺省值为零。有的变元作为存放调过程回送的值，若此值不需要，该项变元可省略。为保证各变元的正确位置（顺序），在 FORTRAN、BASIC、PASCAL 等语言中，省略的变元用一个逗号“,”占位，在 COBOL 语言中用零占位。调用 RTL 过程时，若从某一项开始以后的变元均省略，则“,”也可省略，但调系统服务及 RMS 过程则不可省“,”。

1.2.1.4 过程名及系统符号

对系统过程名及系统符号的说明放于系统缺省库 SYS $ SHARE:IMAGELIB · OLB 及 SYS $ SHARE:STARLET.OLB 中，当连接(LINK)一个程序时，连接程序(LINKER)会自动搜索上述文件以进行对系统过程的调用。将过程作为外部函数过程调用时，程序需定义过程名作为一个整形长字。不同的语言，为使所调过程适合该种语言的需要及方便，还有不同的处理，详见过程调用举例。

系统符号是一些常数和位设置码，包括调过程回送的条件值及代入过程变元中的各种功能码、修饰码等，它们是调系统过程的重要参数。有关系统符号及其含义在前述手册中可以查到，部分系统符号列表于本书内附表中。系统符号本身有值，在程序中定义其类型，并作为外部符号引用，其值在 LINK 时，由 LINKER 搜索系统缺省库解决。为方便和快速地引用系统符号值，有的语言如 FORTRAN、PASCAL 等，还有专用的系统符号库文件，引用方法见程序例题。有一些系统符号仅包含在 MACRO 库 SYS $ SHARE:STARLET.MLB 中，LINKER 在缺省库中找不到时，则该符号得不到正确值，

会使程序出错。为获得任一系统服务的符号值有如下方法:

·用一个 MACRO 模块自动获得值

编写如下两行 MACRO 程序 A.MAR

```
$ XXDEF          〈GLOBAL〉
.END
```

其中 XX 为符号名中 $ 前的字母。

然后对其汇编（$ MAC A）得文件 A·OBJ，用其参与主程序的目标程序连接即可获得对有关系统符号的定义:

```
$ LINK      用户程序名，A
```

则程序中引用的系统符号 XX $ _均自动获得值。

·用一个 MACRO 程序显示系统符号值，将所查到的系统符号值在写程序时代入系统符号的位置。

程序名为 B.MAR

```
·TITLE     B
$ XXDEF          〈GLOBOL〉
·END
```

对其汇编:　　$ MAC　　B

连接:　　$ LINK / NOEXE / MAP　　B

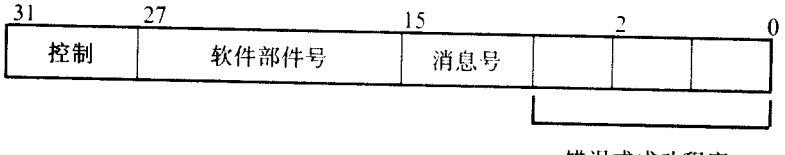
则生成文件 B·MAP，用命令 $ TYPE　　B.MAP 则得所需的符号值（即 XX $ 类的符号），该值以 16 进制表示，化为 10 进制或直接以 16 进制形式数代入相应的系统符号位置。

1.2.1.5　报告调过程的成功和失败

当调系统过程时，可能会出现各种错误，例如调 RTL 过程 MTH $ SQRT 进行开平方运算时，出现了用负数开平方的错误。调过程时出错的原因和条件因具体过程而异。因此，报告调过程的成功或失败以便让程序员决定处理办法。

大多数系统服务、RTL、RMS 过程报告成功和失败的方法是:

用高级语言调用时，必须将过程作为函数过程调用，定义函数名为整形长字，调过程成功与否的状态及错误条件作为函数值返回，检查函数值，则可确定成功与否并依条件值不同决定处理方案。函数值即是一个状态长字。一个完整的状态长字格式如下:



在状态字的 0—2 位表示调过程的成功或严重程度，其编码及含义如表 1.1.3。

3—15 位为调过程发生错误类型的标识号，16—27 位表使用何种软件产生的错误。整个长字的值称为条件值。

错误处理的方式为: