OpenVMS/Hanzi
用户手册

BA322-90016

2005 年 5 月

本用户手册描述 OpenVMS/Hanzi 软件的基本概念、特性及命令。

修订 / 更新：	本用户手册取代 OpenVMS/Hanzi V1.5 AXP 的 OpenVMS/Hanzi 用户手册。
软件版本：	OpenVMS/Hanzi I64 Version 8.2 OpenVMS/Hanzi Alpha Version 7.3-2

Hewlett-Packard Company

Palo Alto, California

Printed in Singapore。

目录

目录

目录

图

表

序言

对象

本用户手册的对象为所有 OpenVMS/Hanzi 系统的用户。

手册结构

本手册包含 11 章及 2 个附录。

- 第 1 章介绍本软件。
- 第 2 章详述本软件所用的 DEC 汉字字符集。
- 3, 4, 6, 7, 8, 9 和 10 章提供资料使读者由浅入深使用本软件，各章备有众多例子向读者逐一说明各个 DCL 命令及公用程序。
- 第 5 章重点叙述 HANZIGEN 公用程序。
- 第 11 章提供有关国际日期和时间的汉字支持用户资料。
- 第 12 章提供有关调试器公用程序的汉字支持用户资料。

手册后面有 2 个附录供对使用软件有经验的用户继续探讨。

- 附录 A 列出 7 个例子，介绍如何在编程语言中使用汉字字符。
- 附录 B 包含有关本系统上汉字处理运行时间库的资料，其中有 HSYSHR、HSMGSHR、HTPU、SORT 和 MERGE, 同时它也列出了 SORT 和 MERGE 的可调用程序，以及样本程序 。

常规

本手册使用以下常规：

OpenVMS, VMS	术语 OpenVMS 和 VMS 意指 OpenVMS 操作系统。
OpenVMS/ Hanzi	术语 OpenVMS/Hanzi 意指汉字 OpenVMS Alpha 和汉字 OpenVMS I64 操作系统，用于中华人民共和国。
OpenVMS/ Hanzi I64	术语 OpenVMS/Hanzi I64 意指汉字 OpenVMS I64 操作系统，用于中华人民共和国。
OpenVMS/ Hanzi Alpha	术语 OpenVMS/Hanzi Alpha 意指汉字 OpenVMS Alpha 操作系统，用于中华人民共和国。
Hanzi	世界不同的地方正使用着不同的汉字字符集：中华人民共和国及新加坡使用简体汉字，香港及台湾使用繁体汉字，日本使用 Kanji, 而韩国则使用 Hanja。为针对这复杂的情况并避免混淆，本手册采用了 "Hanzi" 一字来说明所指的简体汉字是由中华人民共和国国家标准信息交换用汉字编码字符基本集 (GB2312-80) 所定义的简体汉字。
<RETURN>	回车。 在例子或格式中，除非另有声明，否则用户输入的每一行的末尾都出现一个隐含的回车。用户必须在输。

键符号	在例子中，键及键序列都会以符号表示， 如〈PF2〉、CTRL/Z 等。
〈CTRL〉	CTRL/x 表示用户必须在按住标号为〈CTRL〉键的同时按另一个键。例如 ,CTRL/C、CTRL/Y 及 CTRL/O。
. . .	纵向省略号表示不展示系统在应答某一命令时要显或者不展示用户输入的所有数据。
...	横向省略号表示可输入附加参数、值或资料。
( )	在格式描述中， 如果选择多个任选项， 则必须将其括在格式描述中。
[ ]	在格式描述中， 凡是括以中括号的项目都可从中选择任何一项、所有项或不予选择。(然而，在赋值语句的子串说明的语法中，括号不是任选的。
{ }	在格式描述时， 大括号括住所需的选择项； 您必须从任选项中选择一项。

有关文献

- OpenVMS 文献集
- 《字符管理程序 (CMGR) 用户手册》
- 《HEVE 用户手册》
- 《HTPU 和 HEVE 用户 参考手册》
- 《OpenVMS/Hanzi RTL Chinese Processing (HSY$) Manual》
- 《OpenVMS/Hanzi RTL Chinese Screen Management (SMG$) Manual》

第 1 章
简介

OpenVMS/Hanzi 在现有的 OpenVMS 软件特性上添加汉字字符处理能力。现在，用户可在各种 OpenVMS/Hanzi 功能下处理 ASCII 及 DEC 汉字码。

OpenVMS/Hanzi 支持以双字节八字位格式表示的中文数据，符合中华人民共和国国家标准信息交换用汉字编码字符基本集 (GB-2312-80)。有关 DEC 汉字字符集的详情，请参阅本手册第 2 章。

在 OpenVMS/Hanzi 上，大部分的 OpenVMS DCL 命令都支持汉字字符。有关这些命令的详细介绍，请参阅第 3, 4, 6, 7, 8, 9，10，11 和 12 章。用户也可于多种编程语言写成的程序使用汉字。附录 A 列有多种样本程序供参考。

OpenVMS/Hanzi 提供了许多能够处理汉字字符的公用程序，例如：

- 字符管理程序 (CMGR)，用来建立及管理用户定义的汉字字符。

HANZIGEN 公用程序可设置并显示汉字终端类型。它也可提供机制，供用户为公用程序选择语言。

- HDUMP 公用程序可用十进制、十六进制或八进制格式，以及 ASCII 字符和汉字转换来显示文件、磁盘卷或磁带卷的内容。
- HMAIL 是汉字邮递公用程序，可支持汉字文件夹名及主题名。
- SORT 和 MERGE 是汉字排序和合并公用程序，用来把汉字字符及 用户定义字符以不同的整理序列排序和合并。
- HTPU 是汉字字符的文本处理公用程序。HEVE 是字处理及绘制图表的公用程序。

第 2 章
DEC 汉字字符集

本章叙述 DEC 汉字字符集的结构。

2.1 综述

DEC 汉字字符集是包含汉字和非汉字字符的基本字符集，与 GB-2312-80[1] 兼容。

DEC 汉字字符集分为两部分 -- DEC GB-2312 字符集和扩充 GB 字符集，共有 7,445 个字符及 8,836 个字符位。

扩充 GB 字符集是一个扩充区域，有 8,836 个字符位供用户定义自己的字形，供以后扩充字符集之用。

图 2-1 展示 DEC 汉字字符集的结构。

图 2-1 DEC 汉字字符集

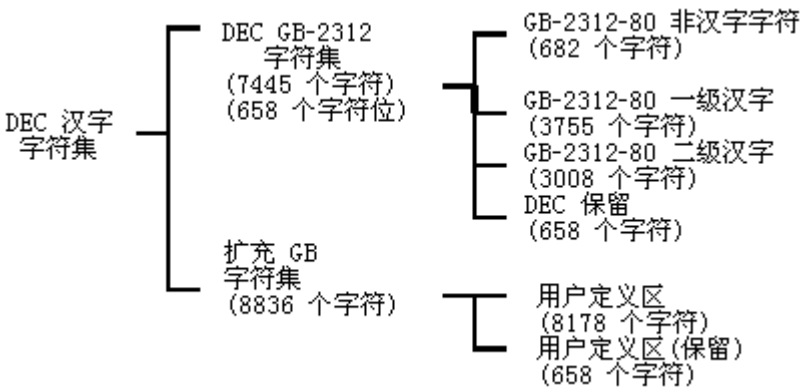


表 2-1 和 2-2 列出 DEC 汉字字符集内各区域的定义。

表 2-1 DEC GB-2312 字符集

区域	描述
GB-2312	此处的字符位是按照 GB-2312-80 标准永久地定义的，用户不能变换。
DEC 保留字符位	此处的字符位专供 DEC 使用而保留。

1. GB-2312-80 标准是用于汉字信息交换的图形字符基本集的国家标准。由中华人民共和国国家标准局于 1981 年 5 月公布。

表 2-2 扩充 GB 字符集

区域	描述
用户定义	用户可在此处的字符位上用 CMGR 公用程序定义自己的字符。
用户定义（保留）	用户可在此处的字符位上用 CMGR 以及 /RESERVED 限定词定义自己的字符。

注意

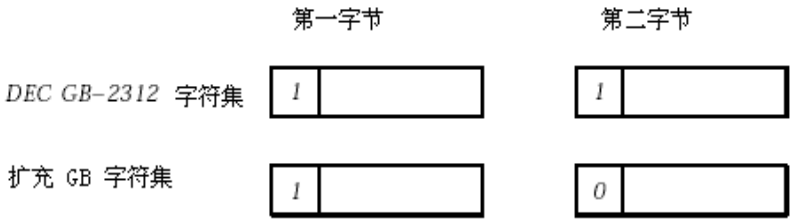
在扩充 GB 字符集中定义的字符可按需装入汉字打印机。

2.2 DEC 汉字码

DEC 汉字码用双字节表示一个图形字符。为区别汉字码和一般的 ASCII 码，一个字符的第一字节的最高有效位总是设置为 1，而第二字节的最高有效位则可以是 1（即字符属于 DEC GB-2312 字符集）或 0（即字符属于扩充 GB 字符集）。

图 2-2 展示一个汉字在 DEC GB-2312 字符集及扩充 GB 字符集中的表示方法。

图 2-2 汉字字符的双字节表示法



2.2.1 代码结构

DEC 汉字码符合 GB-2312-80 标准。DEC GB-2312 字符集和扩充 GB 字符集的代码表都划分为 94 个区（节），编号自 1 至 94。每个区又分为 94 个位，编号也自 1 至 94。

DEC 汉字码的第一字节决定该码的区号，而第二字节则决定该码的位号。此外，第二字节的最高有效位决定该码是属于 DEC GB-2312 字符集还是扩充 GB 字符集。

下面两个例子说明在 DEC GB-2312 字符集及扩充 GB 字符集中一个汉字字符的双字节汉字码和区位号的相互关系。通常，区号及位号是十进制的，而双字节汉字码则以十六进制表示。

例子：

1. 在 DEC GB-2312 字符集中，

区号（十进制）= 第一字节（十六进制）- A0（十六进制）和

位号（十进制）= 第二字节（十六进制）- A0（十六进制）

所以，当 16 = B0 — A0 和

01 = A1 — A0

区号 16、位号 1 就等于 DEC 汉字码 B0A1。

2. 在扩充 GB 字符集中，

区号（十进制）= 第一字节（十六进制）— A0（十六进制）和

位号（十进制）= 第二字节（十六进制）— 20（十六进制）

所以，当 16 = B0 — A0 和

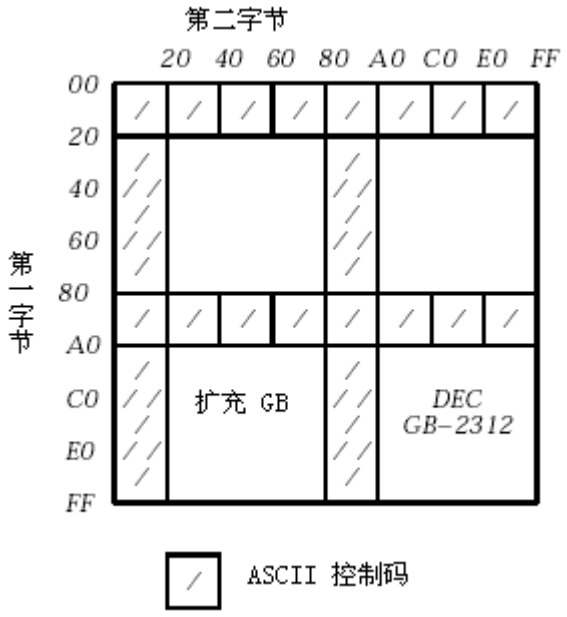
01 = 21 — 20

区号 16、位号 1 就等于 DEC 汉字码 B021。

2.2.2 汉字码表

图 2-3 展示双字节码空间的划分及 DEC 汉字字符集的位置。

图 2-3 汉字码表



注意

斜线区域为 ASCII 控制码的分布位置。DEC 汉字码未分配在斜线区域，目的是使 DEC 汉字码与 ASCII 控制码兼容。

图 2-4 DEC GB-2312 汉字字符集

图 2-5 扩充 GB 字符集

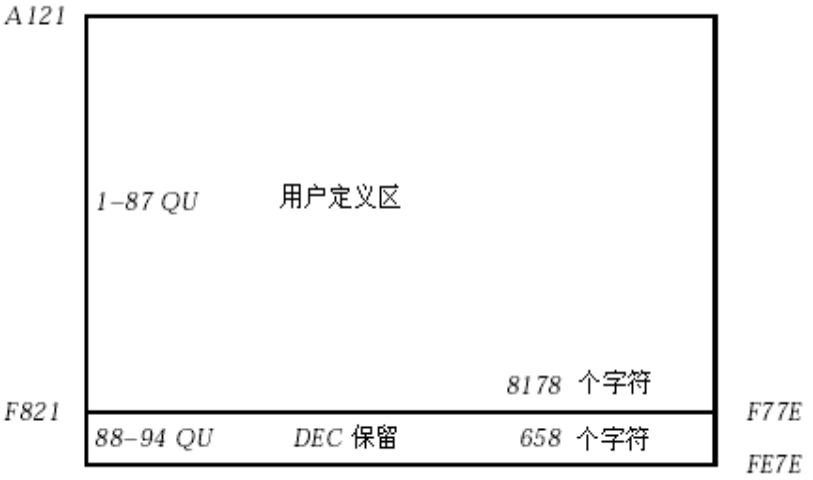


图 2-4 及图 2-5 展示 DEC GB-2312 字符集及扩充 GB 字符集如何进一步划分为区。

第 3 章
汉字字符终端输入输出支持

OpenVMS/Hanzi 支持下列汉字字符的终端输入和输出特性：

- 汉字和英文字符混合输入时可进行编辑
- 进行汉字和英文字符输入校验
- 显示用户定义字符

3.1 编辑输入字符

OpenVMS/Hanzi 可支持所有由 OpenVMS 支持的行编辑功能，诸如光标移动、删除、插入、重键以及行绕回。有关完整的支持特性，请参阅 OpenVMS 文献。

终端屏幕上显示的汉字字符占据的位置多于一列。因此， OpenVMS/Hanzi 提供两种操作态进行编辑：

1. ASCII 字符态
   在此态中，在汉字终端上删除键一次删除一列。光标移动键一次移过一列。
   这就是 OpenVMS 提供的唯一编辑态。
2. 汉字字符态
   在此态中，删除键一次删除一个字符。光标移动键一次移过一个汉字字符。而删除字键 (CTRL/J) 可删除一个英文字或一串汉字字符。
   如果输入行没有完全填满和输入字符太宽无法显示（例如， 行末尾只有一个显示单元 空着，而该输入是一个汉字字符）则这个输入的汉字会被暂存起来为下一次读入要求的数据输入。

HANZIGEN 公用程序用来在这两种编辑态之间进行转换。有关转换态的命令， 请参阅第 5 章《HANZIGEN 公用程序》。

3.2 读校验功能

此特性根据应用程序指定的规则校验输入数据。

应用程序可指定一个用标记分成一个以上的输入区的输入字段。如果 DEC GB-2312 字符集中的汉字字符处于标记定义的界上（也就是说， 它必须分成两半以 插入不同的区域），它就被视为无效字符。

此特性可用 HANZIGEN 关断。当该应用程序一次读一个字节输入时，

它也可自动关断。

3.3 用户定义字符支持

OpenVMS/Hanzi 可为用户提供 CMGR 公用程序（请参阅第 6 章 《字符管理程序 (CMGR) 公用程序》）建立用户定义字符 (UDC)。这些字符可以用来补充由 OpenVMS/Hanzi 支持的汉字标准定义字符。

为了要在终端或打印机上显示这些字符，OpenVMS/Hanzi 支持用于终端输入输出的按需装入 (ODL) 协议。支持 ODL 协议的终端和打印机可自动卸下 UDC 字形数据来显示 UDC。

3.3.1 硬件要求

ODL 的硬件要求如下：

1. 支持 ODL 协议的汉字终端或汉字打印机包括：
   •终端：
   VT382-C
   •打印机：
   LA88-C
   LA380-C
2. 带支持 ODL 协议的 LAT 软件的终端服务器。

3.3.2 LAT/Master

LAT/Master 是在 LAT 环境中支持 ODL 所需要的软件，它包括 OpenVMS/Hanzi 上运行的软件和终端服务器上运行的软件。

为了使 ODL 在 LAT 网络上操作，必须把以下版本的 DECserver 软件下行装入对应的服务器单元：

- DECserver 200 软件 3.0 版本（或更高）
- DECserver 300 软件 1.0 版本（或更高）
- DECserver 500 软件 2.1 版本（或更高）
- DECserver 700 软件 1.0 版本（或更高）
- DECserver 90L 软件 1.0 版本（或更高）
- DECserver 90TL 软件 1.0 版本（或更高）

要利用 LAT 能力，须确保所有装入主机安装了正确的 DECserver 软件版本以及该系统运行 OpenVMS。

要了解如何在装入主机上安装 DECserver 软件和把服务器映象下行装入服务器，请参阅随 DECserver 提供的有关文献。

3.3.3 支持 ODL 所需要的系统设置

ODL 机制要求系统设置伪设备 FHA0: 和 FONT_HANDLER。

系统必须用下列命令配置：

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN> CONNECT FHA0: /NOADAPTER /MAXUNIT=1
SYSGEN> EXIT
$ RUN /DETACH -
    /AST_LIMIT=100 -
    /PRIORITY=8 -
    /PRIVILEGES=(CMKRNL,PHY_IO) -
    /PROCESS_NAME="FONT_HANDLER" -
    /RESOURCE_WAIT -
    /NOSWAPPING -
    /UIC=[1,20] -
    HSY$SYSTEM:FONTHANDL.EXE
```

OpenVMS/Hanzi 上的 SYSGEN 命令为字形数据下行装入而装入一个伪设备驱动器并建立一个伪设备 FHA0:。RUN 命令建立一个系统进程 FONT_HANDLER 为终端及打印机的按需装入请求服务。

注意，这些命令已编辑在 OpenVMS/Hanzi 系统启动命令过 SYS$STARTUP:HSY$STARTUP.COM 中，因此正常情况下， OpenVMS/Hanzi 系统对 ODL 配置是正确的。

3.3.4 每一设备设置

如果设备与终端服务器相连接，则必须恰当地配置终端服务器端口。

对于直接连接或与一个终端服务器相连接的设备，系统中的每个终端端口必须对 ODL 进行个别配置。通常，终端端口对 ODL 的默认值为撤销 ODL。HANZIGEN 公用程序用来对个别终端（参阅用 HANZIGEN 设置终端端口一节）允许 ODL。终端或打印机也要求设置，以便允许终端或打印机的 ODL 能力。

3.3.4.1 终端服务器端口设置

DECserver 软件中，为 ODL 支持引入新端口特征 ON-DEMAND [LOADING]。这就使得用户可在每个与汉字终端或汉字打印机相连接的服务器端允许或撤销 ODL 能力。要对一个具体端口允许或撤销 ODL，则在 Local 态发出下列命令：

```
Local> {SET|DEFINE} PORT [n] ON-DEMAND [LOADING]
       {ENABLED|DISABLED}
```

例如，要对端口 3 永久地允许 ODL 能力（即把这些设置存储到永久数据库中），则发出：

```
Local> DEFINE PORT 3 ON-DEMAND LOADING ENABLED
```

注意

要在 DECserver 500 中设置永久的特征，则需要在装入主机中执行终端服务器配置程序 (TSC)。

要校验是否已允许 ON-DEMAND [LOADING] 特征，则在 Local 态发出 LIST PORT CHARACTERISTICS 命令：

```
Local> LIST PORT CHARACTERISTICS
Port 3:                    Server: HGD111
Character Size:        8   Input Speed:       9600
Flow Control:        XON   Output Speed:      9600
Parity:             None   Modem Control:  Disabled
Access:            Local   Local Switch:      None
Backwards Switch:   None   Name:
Break:              Local  Session Limit:        4
Forwards Switch:     None  Type:              Soft
Preferred Service: None
Authorized Groups:    0
```

```
Enabled Characteristics:
Autobaud, Autoprompt, Broadcast, Input Flow
Control, Loss Notification,Massage Codes, Output
Flow Control, On-Demand Loading, Verification
Local>
```

3.3.4.2 用 HANZIGEN 设置终端端口

在 OpenVMS/Hanzi 中，使用 HANZIGEN 在终端端口上允许 ODL。有关使用 HANZIGEN 命令来允许和撤销 ODL 的详情，请参阅第 5 章《HANZIGEN 公用程序》一章。限定词 "/FONT" 和 "/NOFONT" 在 HANZIGEN 中用来允许或撤销按需装入。按照默认是撤销按需装入。下列命令将对当前终端

```
$ RUN HSY$SYSTEM:HANZIGEN
HANZIGEN> SET /FONT
```

要永久地允许 ODL（即在下一次系统再引导前一直有效），使用下列命令：

```
$ RUN HSY$SYSTEM:HANZIGEN
HANZIGEN> SET /FONT /PERMANENT
```

3.3.4.3 设置 VT382 终端

在汉字终端上允许 ODL，需要进行下列步骤：

- 对 ODL 设置设备。
  按 <F3> 键以进入 "设置目录"，然后选择 "终端" 设置并且在 "终端设置" 项目单内允许 ODL。
- 如果终端与一个服务器端口连接，则在 "Local>" 提示处允许当前端口上的 ON-DEMAND [LOADING] 特性。有关详情，请参阅终端服务器端口设置一节。
  Local> SET PORT ON-DEMAND LOADING ENABLED
- 注册进入 OpenVMS/Hanzi，使用 HANZIGEN 来允许在终端端口上的 ODL。有关详情，请参阅用 HANZIGEN 设置终端端口一节。

目前，只有 VT382 汉字终端具有处理按需装入的能力。

3.3.4.4 设置打印机

在上一个对话期中列出的汉字打印机上允许 ODL，需要进行下列步骤：

- 按相应的打印机手册所述，按需装入设置打印机。
- 如果打印机与一个服务器端口相连接，则在 "Local>" 提示处允许该端口上的 ON-DEMAND [LOADING] 特征。有关详情，请参阅终端服务器端口设置一节。
- 注册进入 OpenVMS/Hanzi，使用 HANZIGEN 允许在终端端口上的 ODL。有关详情，请参阅用 HANZIGEN 设置终端端口一节。
  HANZIGEN> SET/DEVICE_TYPE=LA380/FONT/PERMANENT LTA100:
- 如常设置打印机端口和队列。如果打印机与一个终端服务器端口连接，则在初始化打印队列前应调整远程打印队列的超时值（LAT$SYMTIMEOUT 逻辑名）。用户应在 SYS$MANAGER:LAT$SYSTARTUP.COM 中定义此 LAT$SYMTIMEOUT 逻辑名。LAT$SYMTIMEOUT 的默认值是 10 分钟。如果超时字符出现於打印表，建议用一个较 大的 LAT$SYMTIMEOUT 值以免超时。下列命令把 LAT$SYMTIMEOUT 定义为 15 分钟。
  $ DEFINE/SYSTEM/EXECUTIVE LAT$SYMTIMEOUT "00:15:00"

3.3.5 按需装入的限制

应该注意，ODL 只在终端处于 NODMA 和 NOPASTHRU 态时才起作用。用户可用下列命令来检验终端是否处于所要求的态：

```
$ SHOW TERMINAL
```

如果终端不是处于所要求的态，则可使用下列命令来设置：

- 把当前终端设置为 NODMA 态
  ```
  $ SET TERMINAL/NODMA
  ```
- 把当前终端设置为 NOPASTHRU 态
  ```
  $ SET TERMINAL/NOPASTHRU
  ```

第 4 章
DCL 命令和公用程序

本章列出汉字终端及汉字打印机所需的设备特征，并叙述在 DCL 命令及公用程序中如何应用汉字字符。

4.1 操作准备

4.1.1 设置汉字终端

在 OpenVMS/Hanzi 的大部分公用程序中，您可用 HANZIGEN 公用程序把汉字终端初始化成恰当的设置类型：

```
HANZIGEN> SET /DEVICE_TYPE=VT382
```

4.1.2 设置汉字打印机

用户应使用 HANZIGEN 公用程序来初始化汉字打印机，以便具有所需的特征：

```
HANZIGEN> SET/DEVICE_TYPE=LA380/FONT/PERMANENT LAT100:
```

4.2 DCL 命令

本节说明在 DCL 命令中如何使用汉字字符。

4.2.1 命令过程中的变元

您可在变元中使用有汉字字符的字符串。例如：

```
$ TYPE SAMPLE.COM
$ SHOW SYMBOL P1
$ SHOW SYMBOL P2
$ SHOW SYMBOL P3
$ EXIT
$
$ @SAMPLE 参数 1 参数 2 参数 3
P1="参数 1"
P2="参数 2"
P3="参数 3"
$
```

4.2.2 SHOW 中使用汉字

您可以使用下列 SHOW 命令，以汉字显示系统资料：

- $ SHOW USER
- $ SHOW QUEUE
- $ SHOW SYSTEM
- $ SHOW MEMORY
- $ SHOW DEVICE
- $ SHOW PROCESS
- $ SHOW LOGICAL
- $ SHOW TRANSLATION

与 OpenVMS 求助信息相似，用户可以应用 HANZIGEN 公用程序来选择英文或汉字信息，唯 SHOW LOGICAL 及 SHOW TRANSLATION 例外。

4.2.3 APPEND、BACKUP、CONVERT、COPY、CREATE 和 TYPE 中使用汉字

用户可在文件中使用汉字数据，并可在 APPEND、BACKUP、CONVERT、COPY、CREATE 和 TYPE 命令中包括汉字。例如：

```
$ CREATE HANZI.DAT
独览梅花扫腊雪
$
$ TYPE HANZI.DAT
独览梅花扫腊雪
$
```

4.2.4 ASSIGN、DEASSIGN 和 DEFINE 中使用汉字

您可在逻辑名及其等效字符串中使用汉字，也可于 ASSIGN、DEASSIGN、DEFINE 命令中使用汉字。

您可用 SHOW LOGICAL 和 SHOW TRANSLATION 命令显示由 ASSIGN 和 DEFINE 以汉字字符定义的逻辑名，而词法功能 F$LOGICAL 则可转换汉字。例如：

```
$ DEFINE 逻辑名 等效名
$ SHOW LOGICAL 逻辑名
  逻辑名 =" 等效名 " (PROCESS)
$ _变数 =F$LOGICAL(" 逻辑名 ")
$ SHOW SYMBOL _变数
  _变数 = " 等效名 "
$
```

4.2.5 DIRECTORY 中使用汉字

在 DIRECTORY 命令中支持两种语言的系统信息。可用 HANZIGEN 公用程序选择英文或汉字信息。

4.2.6 MESSAGE 中使用汉字

您可在信息文件中的设施字段及文本中使用汉字，但汉字不适用于 ident 字段。

4.2.7 OpenVMS HELP 中使用汉字

您可使用 HELP 命令和 HANZIGEN 公用程序来调用 OpenVMS HELP 设施，以汉字或英文显示有关 OpenVMS 命令或题目的资料。

注意，您可使用下列命令直接显示英文求助（即使您的输出设备设置成 HANZI_MSG):

```
$ HELP @HELPLIB
```

此外，使用下列命令可显示 OpenVMS/Hanzi 特定求助题目：

```
$ HELP @HSYHLP
```

4.2.8 READ 和 WRITE 中使用汉字

包含汉字字符的数据可以构成输入和输出文本的一部分。例如：

```
$ TYPE HANZI.DAT
汉字写读测试 1
$ OPEN/READ INPUT HANZI.DAT
$ OPEN/WRITE OUTPUT HANZI1.DAT
$ READ INPUT REC
$ WRITE OUTPUT REC
$ WRITE OUTPUT "汉字写读测试 2"
$ CLOSE INPUT
$ CLOSE OUTPUT
$ TYPE HANZI1.DAT
汉字写读测试 1
汉字写读测试 2
$
```

4.2.9 REPLY 中使用汉字

您可在信息文本中使用汉字。例如：

```
$ REPLY/TERMINAL=TT "REPLY 中混合汉字使用"
Reply received from user SYSTEM at _RTA3:12:34:56.78
REPLY 中混合汉字使用
$
```

4.2.10 SET 中使用汉字

您可使用汉字指定提示符或在命令描述文件的动词中使用汉字，但汉字不适用于限定词或标号。

4.2.11 SEARCH 中使用汉字

您可在被检索的字符串中使用汉字。例如：

```
$ TYPE HANZI.DAT
```

```
山高月小
古往今来
$
$ SEARCH/HIGHLIGHT=UNDERLINE HANZI.DAT 小
山高月小
$
```

4.2.12 汉字符号

符号名应以一个英文字母（包括 _ 及 $）开始。第二个以及接着的字符可用汉字。要赋值的字符串中亦可使用汉字。例如

```
$ _变数 =" 汉字数据 "
$ SHOW SYMBOL _变数
 _变数 =" 汉字数据 "
```

4.2.13 命令过程中的标号

您可在命令过程中用汉字作为标号名，您可用 GOTO 命令转移到一个汉字标号。

4.3 通用 PostScript 打印子系统

通用 PostScript 打印子系统 (WWPPS) 提供对 Alpha 和 I64 平台上纯文本文件的高质量打印。它能够同时打印包含单字节和多字节亚洲字符的纯文本文件。

要调用这个公用程序，您可以在 DCL 提示下使用以下命令：

```
$ RUN SYS$SYSTEM:WWPPS
```

这个公用程序将响应为以下提示：

```
$ WWPPS>
```

另外，您可以通过定义以下外部命令简化这个调用：

```
$ WWPPS := = $SYS$SYSTEM:WWPPS
```

这里是一个打印纯文本文件的例子：

```
$ RUN SYS$SYSTEM:WWPPS
WWPPS>PRINT/QUEUE=hp$printer/LOCALE=zh_cn_dechanzi hanzi-text_file.txt
```

或者简化为：

```
$ WWPPS PRINT/QUEUE=hp$printer/LOCALE=zh_cn_dechanzi hanzi-text_file.txt
```

对于 WWPPS 特性的详细描述，请参阅 《OpenVMS 用户手册》。

第 5 章
HANZIGEN 公用程序

本章描述 HANZIGEN 公用程序的功能。

5.1 综述

HANZIGEN 公用程序设置及展示汉字终端类型、允许及撤销汉字终端及打印机的按需装入。

HANZIGEN 的 SET 和 SHOW 命令与 DCL 的相似，但 HANZIGEN 的 SET 和 SHOW 命令操纵与汉字终端有关的属性。

5.2 启动序列

键入以下命令来启动 HANZIGEN:

```
$ RUN HSY$SYSTEM:HANZIGEN
HANZIGEN>
```

5.3 命令概要

SET	设置汉字终端特征，并允许或撤销汉字终端及打印机的按需装入。
SHOW	展示汉字终端属性。
HELP	列出 HANZIGEN 命令。
EXIT	终止 HANZIGEN 并返回 DCL 命令级。

5.3.1 SET 命令

SET 命令设置汉字终端类型及允许按需装入。汉字终端类型可以是 VT82、VT382、LA84、LA86、LA88、LA280 或 LA380。要设置汉字终端为 ASCII 特征，您可指定 VT100 或 VT300_series 为设备类型。对于提供英文和汉字求助/出错信息的公用程序，您可使用 SET 命令通过 /OUTPUT 任选项选择输出语言。

5.3.1.1 命令格式

SET 的命令格式为：

```
SET [ 设备名 [ : ]] [/DEVICE_TYPE= 设备类型 ]
                    [/INPUT= 输入类型
                    [/OUTPUT= 输出类型 ]
```

[/[NO]FONT]
[/PERMANENT]
[/SYSTEM]

其中

设备名　　指定连接汉字终端的设备的物理设备名。

/DEVICE_TYPE=设备类型　　指定 HANZIGEN 支持的有效设备类型：

VT82　　标识为 VT82 汉字终端设备。

VT100　　标识为 VT100 终端设备。此限定词可用来以 VT100 终端特征重置 VT82。

VT382　　标识为 VT382-CB 汉字终端设备。

VT300_series　　标识为 VT300 系列终端设备。此限定词可用来以 VT300 系列终端特征重置 VT382。

LA84　　标识为 LA84-C 汉字打印设备。

LA86　　标识为 LA86-C 汉字打印设备。

LA88　　标识为 LA88-C 汉字打印设备。

LA280　　标识为 LA280-CB 汉字打印设备。

LA380　　标识为 LA380-CB 汉字打印设备。

/INPUT=输入类型　　设置终端为汉字或 ASCII 终端。 输入类型的有效值为：

HANZI　　终端能输入汉字，并有汉字终端的操作特性。

ASCII　　终端为标准的 ASCII 终端。

VT82、VT382、LA84、LA86、LA88、LA280 及 LA380 的默认输入类型为 HANZI。

/OUTPUT=输出类型　　设置终端以显示汉字或 ASCII 信息。输出类型的有效类型值为：

HANZI_MSG　　显示汉字信息。

ASCII_MSG　　显示 ASCII 信息。

LA84、LA86、LA88、LA280、LA380、VT82 及 VT382 的默认输出类型为 HANZI_MSG。

/[NO]FONT　　允许按需装入。按需装入从系统字形数据库检索用户定义字形，并把这些字形传送到 LA84-C、LA86-C、LA88-C、LA280-CB 或 LA380-CB 汉字打印机或 VT382 汉字终端。/[NO]FONT 限定词只与 LA84、LA86、LA88、LA280、LA380 及 VT382 设备类型一同使用。默认是 /NOFONT。

/PERMANENT　　指定特征更改为永久性。如果您要设置特征的设备不是自己的设备，便需要使用这个限定词。/PERMANENT 不能与 /OUTPUT 一起使用。这个限定词要求用户有 PHY_IO 或 LOG_IO 特权。

/SYSTEM	指定特征更改为系统默认。/SYSTEM 只不能与 /DEVICE_TYPE 和 /OUTPUT 一起使用。这个限定词要求用户有 PHY_IO 特权。

5.3.1.2 SET 命令限制

下表概述一系列对 /PERMANENT 和 /SYSTEM 限定词有限制的命令限定词。

表 5-1 /SYSTEM 和 /PERMANENT 限定词的限制概要

	/SYSTEM	/PERMANENT
/DEVICE_TYPE	不可并用	可并用
/INPUT	可并用	可并用
/OUTPUT	不可并用	不可并用
/[NO]FONT	可并用	可并用

5.3.2 SHOW 命令

SHOW 命令展示汉字终端及打印机特征，显示格式是每种设备占一行，并且具如下字段：

- 当前 SYS$INPUT 设备的名称
- 设备类型
- 允许或不允许按需装入
- 输入类型 (HANZI 或 ASCII)
- 输出类型 (HANZI_MSG 或 ASCII_MSG)

在 HANZIGEN 命令级上，VT82 和 VT382

的设备类型将展示为 HANZI_VDU, 而 LA84-C、LA86-C、LA88-C、LA280-CB 和 LA380-CB 的设备类型则为 HANZI_PRT。

但是，在 DCL SHOW 命令上，已由 HANZIGEN 设置为 VT82 或 VT382 终端的设备名是 VT80 或 VT300, 而由 HANZIGEN 设置为 LA84、LA86、LA88、LA280 或 LA380 终端的设备名则是 LA84。

5.3.2.1 命令格式

SHOW 的命令格式为：

SHOW [设备名 [:]][/ALL][/PERMANENT][/SYSTEM]

其中

设备名	指定 HANZIGEN 将显示的物理设备名。
/ALL	显示所有终端设备的特征。这个限定词要求用户有 PHY_IO 和 SHARE 特权。对于非当前设备的那些设备，输出类型显示则是 "Unknown"（见下面的例子）。
/PERMANENT	显示终端的永久特征。这个限定词要求用户有 PHY_IO 或 LOG_IO 特权。

/SYSTEM　　显示系统默认的特征。这个限定词要求用户有 PHY_IO 特权。

注意，在 DCL SHOW 命令中，由 HANZIGEN 设置为汉字终端的终端设备名为 VT80。

5.4 命令例子

1. 展示设置为汉字终端的所有终端设备的特征。

```
HANZIGEN>SHOW
Device Name Type         On Demand  Input  Output
_RTA2:      HANZI_VDU    DISABLE    ASCII  HANZI_MSG
HANZIGEN>SHOW/ALL
Device Name Type         On Demand  Input  Output
_OPA0:      LA36         DISABLE    ASCII  Unknown
_RTA1:      VT300_Series DISABLE    HANZI  Unknown
_RTA2:      HANZI_VDU    DISABLE    ASCII  HANZI_MSG
_LTA5006:   VT200_Series DISABLE    HANZI  Unknown
_LTA5007:   HANZI_VDU    DISABLE    ASCII  Unknown
_LTA5008:   HANZI_VDU    DISABLE    ASCII  Unknown

   Total : 6 Terminals
```

2. 设置显示汉字信息的 VT382 终端。

```
$ RUN HSY$SYSTEM:HANZIGEN
HANZIGEN> SHOW
Device name Type       On Demand Input  Output
_TTA0:  VT200_series   DISABLE   ASCII  ASCII_MSG
HANZIGEN> SET /DEVICE_TYPE=VT382
HANZIGEN> SHOW
Device name Type       On Demand Input  Output
_TTA0:      HANZI_VDU  DISABLE   HANZI  HANZI_MSG
```

3. 设置并展示已允许按需装入的 LA380-C 汉字打印机。

```
HANZIGEN> SET TXA1: /DEVICE_TYPE=LA380 /PERMANENT /FONT
HANZIGEN> SHOW TXA1:
Device name Type       On Demand Input  Output
_TXA1:      HANZI_PTR  ENABLE    HANZI  Unknown
```

4. 重置终端为 ASCII 特征并展示之。

```
HANZIGEN> SHOW TXA2:
Device name Type      On Demand Input  Output
_TAX2:      HANZI_VDU DISABLE   HANZI  Unknown
HANZIGEN> SET TXA2: /DEVICE_TYPE=VT100 /PERMANENT
HANZIGEN> SHOW TXA2:
Device name Type     On Demand Input  Output
_TXA2:      VT100    DISABLE   ASCII  Unknown
```

5.5 超越一行继续命令

如 DCL 命令一样，您可把一个命令串分几行输入。做法是在 HANZIGEN 命令级把连字符 "-" 放在一行之尾。例如：

```
HANZIGEN> SET TTA0: /DEVICE_TYPE=VT382 -
_HANZIGEN> /PERMANENT
```

5.6 注释行支持

如 DCL 命令一样，您可在 HANZIGEN 命令级输入注释行，做法是在注释前加入感叹号 "!"。例如：

```
HANZIGEN> ! set tta0 as VT382 as Hanzi terminal
HANZIGEN> SET TTA0:/DEVICE_TYPE=VT382 /PERMANENT
```

第 6 章
字符管理程序 (CMGR) 公用程序

本章描述字符管理程序 (CMGR) 公用程序的特性。

6.1 综述

字符管理程序 (CMGR) 是一个公用程序，您可用它来建立及管理汉字字符，特别是用户定义字符 (UDC)。

有关如何使用 CMGR 命令的详情，请参阅 *《字符管理程序 (CMGR) 用户手册》*。

6.2 CMGR 的一般特性

CMGR 提供下列特性：

- 建立和修改三种不同字形大小的 UDC 字块模式，这三种字形大小是 24x24、32x32 和 40x40 象素。一个象素（图象单元）是屏幕上最小的可显示单元。
- 定义 UDC 的整理特征，从而在 OpenVMS/Hanzi 的 SORT/MERGE 公用程序中提供 UDC 支持。代码的整理值存储在整理数据文件或整理数据库中。
- 显示位映象格式中的字块模式。
- 显示在系统数据库中登记的或包含在用户数据文件中的字符代码表。
- 抽取指定字符代码的可装入序列和 / 或整理特征，放入指定的预装入文件或整理数据文件。
- 抽取指定文本文件中找到的 UDC 可装入序列，并将这些序列输出到在一个预装入文件中。
- 支持多数据库系统，从而用户可以建立及使用不同字体的字形。
- 具有 SYSPRV 特权的用户可管理系统数据库。

6.3 使用 CMGR

CMGR 公用程序由一个 DCL 命令调用：

```
$ CHARACTER_MANAGER
CMGR>
```

在这个提示处，用户可以输入 CMGR 命令以执行不同功能。用户也可以从 DCL 级以 "CHARACTER_MANAGER" DCL 命令开头，调用 CMGR 命令。例如：

```
$ CHARACTER_MANAGER CMGR_command
```

命令完成后，控制送回 DCL。

第 7 章
SORT 和 MERGE 公用程序

本章描述 OpenVMS/Hanzi SORT 和 MERGE 公用程序中提供的汉字处理能力。

7.1 综述

OpenVMS/Hanzi SORT 和 MERGE 公用程序支持包含汉字字符的文件。其功能如下：

- 用各种汉字整理序列处理汉字字符。
- 支持用户定义字符 (UDC)。UDC 的整理值可用 CMGR 公用程序命令定义。有关 如何定义 UDC 的整理值详情，请参阅 *《字符管理程序(CMGR)用户手册》*。
- SORT 和 MERGE 可调用例行程序可以支持汉字整理序列的文件及记录接口。
- 与 OpenVMS SORT 和 MERGE 公用程序完全兼容 。

7.2 汉字整理序列

OpenVMS/Hanzi 的 SORT 和 MERGE 公用程序根据用户所指定的关键字来排列输入文件记录，然后产生一个已排序的输出文件。SORT 和 MERGE 可按照下列整理序列来排列汉字数据记录：

RADICAL	把指定的汉字字段以部首序列排序和合并。汉字字符根据 GB-2312-80 标准的部首索引分类和排序。
STROKE	把指定的汉字字段以笔画数序列排序和合并。
PINYIN	把指定的汉字字段以拼音序列排序和合并。汉字字符根据 GB-2312-80 标准的拼音索引分类及排序。
QUWEI	把指定的汉字字段以区位码序列排序和合并。(结果与使用内相同。)

注意： 如果排序键包括的数据既不是有效的 DEC 汉字字符，又不是 UDC,则系统会给此数据分配一个为零的整理值。

7.3 命令格式

SORT 的命令格式是：

```
$ SORT[qualifiers]
input-file-specification[qualifiers] -
_$ output-file-specification[qualifiers]
```

MERGE 的命令格式是：

```
$ MERGE[qualifiers]
input-file-specifications[qualifiers] -
```

_$ output-file-specification[qualifiers]

在合并前，MERGE 命令里的输入文件必须先以同一种整理序列及同一种次序（升序或降序）预先排序。MERGE 命令所指定的排序次序及整理序列必须与输入文件的相同。

7.3.1 /KEY 限定词

/KEY 限定词定义 SORT 的关键字，在一命令行中最多可出现 255 次。用圆括号把 /KEY 子限定词括起来。

除了在 OpenVMS SORT 和 MERGE 中提供的之外，OpenVMS/Hanzi 的 SORT 和 MERGE 还支持下列子限定词。

CSIZE:n	SORT 或 MERGE 的 CSIZE 子限定词只能用于汉字整理序列。n 是 SORT 或 MERGE 关键字的字符数长度。注意，不能在一个关键字说明中同时指定 CSIZE 和 SIZE。
RADICAL	RADICAL 子限定词指定用部首整理序列来把记录排序或合并。
STROKE	STROKE 子限定词指定用笔画整理序列来把记录排序或合并。
PINYIN	PINYIN 子限定词指定用拼音整理序列来把记录排序或合并。
QUWEI	QUWEI 子限定词指定用区位码整理序列来把记录排序或合并。

/KEY 限定词的例子如下：

/KEY=(POSITION:1, CSIZE:5, RADICAL, STROKE)

上例中，RADICAL 是主整理序列；STROKE 是次整理序列。注意，同时可指定多重汉字整理序列。

7.3.2 /SPECIFICATION 限定词

OpenVMS/Hanzi 的 SORT 和 MERGE 提供一个命令限定词，通过该限定词传送一个输入文件说明。说明文件的默认文件类型是 .SRT。指定该文件的命令格式如下：

$ SORT/SPECIFICATION=input-specification-file ...

在 *《OpenVMS User's Manual》* 中，说明文件的格式和功能有详述。

说明文件中的 /FIELD 限定词用来定义 SORT 或 MERGE 关键字字段，与 /KEY 命令限定词很类似。除了 OpenVMS SORT 或 MERGE 公用程序中 /FIELD 限定词的所有有效的子限定词外，OpenVMS/Hanzi SORT 和 MERGE 还把下列当作有效子限定词：

- RADICAL
- STROKE
- PINYIN
- QUWEI

这些表示其相应整理序列的子限定词等于 /KEY 命令限定词中相同子限定词。下面是说明文件中

/FIELD 限定词的一个例子：

/FIELD=(NAME:Hanzi-field, POSITION:1, SIZE:4, PINYIN)

注意，在一个 /FIELD 限定词内不能指定多个整理序列。如果要对一个关键字指定几种整理序列，则在说明文件中对同一关键字指定不同整理序列的多个 /FIELD 限定词。

7.4 SORT 例子

1. 以部首整理序列排序。

```
$ TYPE POET.DAT

李商隐      !"李"字之部首码 72 , 笔画 07 , 拼音 LI  , 区位 3278
白居易      !"白"字之部首码 119, 笔画 05 , 拼音 BAI , 区位 1655
王维        !"王"字之部首码 70 , 笔画 04 , 拼音 WANG, 区位 4585
李白        !"李"字之部首码 72 , 笔画 07 , 拼音 LI  , 区位 3278
孟浩然      !"孟"字之部首码 65 , 笔画 08 , 拼音 MENG, 区位 3547
$

$ SORT/KEY=(POSITION:1,CSIZE:1,RADICAL)POET.DAT RADICAL.DAT

$ TYPE RADICAL.DAT

孟浩然      !"孟"字之部首码 65 , 笔画 08 , 拼音 MENG, 区位 3547
王维        !"王"字之部首码 70 , 笔画 04 , 拼音 WANG, 区位 4585
李白        !"李"字之部首码 72 , 笔画 07 , 拼音 LI  , 区位 3278
李商隐      !"李"字之部首码 72 , 笔画 07 , 拼音 LI  , 区位 3278
白居易      !"白"字之部首码 119, 笔画 05 , 拼音 BAI , 区位 1655
$
```

2. 以笔画数整理序列排序，汉字关键字的长度为 2。

```
$ SORT/KEY=(POSITION:1,CSIZE:2,STROKE)POET.DAT STROKE.DAT

$ TYPE STROKE.DAT

王维        !"王"字之部首码 70 , 笔画 04 , 拼音 WANG, 区位 4585
白居易      !"白"字之部首码 119, 笔画 05 , 拼音 BAI , 区位 1655
李白        !"李"字之部首码 72 , 笔画 07 , 拼音 LI  , 区位 3278
李商隐      !"李"字之部首码 72 , 笔画 07 , 拼音 LI  , 区位 3278
孟浩然      !"孟"字之部首码 65 , 笔画 08 , 拼音 MENG, 区位 3547
$
```

3. 以多种整理序列排序。在一个排序操作中，对同一个关键字，用户可指定多种整理序列。

```
$ TYPE POET1.DAT

李商隐      !"李"字之部首码 72 , 笔画 07 , 拼音 LI  , 区位 3278
白居易      !"白"字之部首码 119, 笔画 05 , 拼音 BAI , 区位 1655
王维        !"王"字之部首码 70 , 笔画 04 , 拼音 WANG, 区位 4585
李白        !"李"字之部首码 72 , 笔画 07 , 拼音 LI  , 区位 3278
孟浩然      !"孟"字之部首码 65 , 笔画 08 , 拼音 MENG, 区位 3547
杜甫        !"杜"字之部首码 72 , 笔画 07 , 拼音 DU  , 区位 2237
$

$ SORT/KEY=(POSITION:1,CSIZE:2,STROKE,QUWEI) -_$ POET1.DAT STRQUWEI.DAT

$ TYPE STRQUWEI.DAT

王维        !"王"字之部首码 70 , 笔画 04 , 拼音 WANG,  区位 4585
白居易      !"白"字之部首码 119, 笔画 05 , 拼音 BAI ,  区位 1655
杜甫        !"杜"字之部首码 72 , 笔画 07 , 拼音 DU  ,  区位 2237
李白        !"李"字之部首码 72 , 笔画 07 , 拼音 LI  ,  区位 3278
李商隐      !"李"字之部首码 72 , 笔画 07 , 拼音 LI  ,  区位 3278
```

```
孟浩然          !"孟"字之部首码 65 , 笔画 08 , 拼音 MENG,  区位 3547
$
```

7.5 MERGE 例子

以下例子显示两个文件以降序合并起来。

```
$ SORT/KEY=(POSITION:1,CSIZE:1,STROKE,DESCENDING) POET2.DAT POET2.SOR
$ TYPE POET2.SOR
孟浩然              !  "孟" 字之部首码 65 , 笔画 08 , 拼音 MENG, 区位 3547
李白                !  "李" 字之部首码 72 , 笔画 07 , 拼音 LI  , 区位 3278
白居易              !  "白" 字之部首码 119, 笔画 05 , 拼音 BAI , 区位 1655
$ SORT/KEY=(POSITION:1,CSIZE:1,STROKE,DESCENDING) POET3.DAT POET3.SOR
$ TYPE POET3.SOR
温庭筠              !  "温" 字之部首码 55 , 笔画 12 , 拼音 WEN , 区位 4634
李商隐              !  "李" 字之部首码 72 , 笔画 07 , 拼音 LI  , 区位 3278
王维                !  "王" 字之部首码 70 , 笔画 04 , 拼音 WANG, 区位 4585
$ MERGE/KEY=(POSITION:1,CSIZE:1,STROKE,DESCENDING) POET2.SOR, POET3.SOR -
_$ POET.MER
$ TYPE POET.MER
温庭筠              !  "温" 字之部首码 55 , 笔画 12 , 拼音 WEN , 区位 4634
孟浩然              !  "孟" 字之部首码 65 , 笔画 08 , 拼音 MENG, 区位 3547
李白                !  "李" 字之部首码 72 , 笔画 07 , 拼音 LI  , 区位 3278
李商隐              !  "李" 字之部首码 72 , 笔画 07 , 拼音 LI  , 区位 3278
白居易              !  "白" 字之部首码 119, 笔画 05 , 拼音 BAI , 区位 1655
王维                !  "王" 字之部首码 70 , 笔画 04 , 拼音 WANG, 区位 4585
$
```

7.6 /SPECIFICATION 例子

以下例子显示 /SPECIFICATION 限定词的用法：

```
$ TYPE POET.DAT
李商隐              !  "李" 字之部首码 72 , 笔画 07 , 拼音 LI  , 区位 3278
白居易              !  "白" 字之部首码 119, 笔画 05 , 拼音 BAI , 区位 1655
王维                !  "王" 字之部首码 70 , 笔画 04 , 拼音 WANG, 区位 4585
李白                !  "李" 字之部首码 72 , 笔画 07 , 拼音 LI  , 区位 3278
孟浩然              !  "孟" 字之部首码 65 , 笔画 08 , 拼音 MENG, 区位 3547
$ TYPE SPECIFICATION.DAT
/FIELD=(NAME=POET_NAME,POSITION:1,SIZE:2,RADICAL)
/KEY=POET_NAME
$ SORT/SPECIFICATION=SPECIFICATION.DAT POET.DAT RADICAL.DAT
$ TYPE RADICAL.DAT
孟浩然              !  "孟" 字之部首码 65 , 笔画 08 , 拼音 MENG, 区位 3547
王维                !  "王" 字之部首码 70 , 笔画 04 , 拼音 WANG, 区位 4585
李白                !  "李" 字之部首码 72 , 笔画 07 , 拼音 LI  , 区位 3278
李商隐              !  "李" 字之部首码 72 , 笔画 07 , 拼音 LI  , 区位 3278
白居易              !  "白" 字之部首码 119, 笔画 05 , 拼音 BAI , 区位 1655
$
```

7.7 可调用 SORT 和 MERGE 例行程序中的汉字支持

您使用可调用 SORT 和 MERGE 例行程序把排序或合并操作加入应用程序中去处理汉字数据记录。这些例行程序 OpenVMS 过程调用和条件处理标准的编程语言调用。

有关可调用 SORT 和 MERGE 例行程序的详情，以及这些例行程序如何调用和样本程序，请参阅附录 B，在 SORT 和 MERGE 可调用例行程序中的汉字支持。

第 8 章
HMAIL 公用程序

本章描述 HMAIL 公用程序的特性。

8.1 综述

HMAIL 在 MAIL 公用程序上增加了下述特性：

- 支持汉字个人名
- 支持汉字主题名
- 支持汉字文件夹名
- 支持汉字字符串检索
- 根据汉字整理序列显示文件夹名
- HMAIL 中的文本编辑程序

8.2 开始 HMAIL 对话期

HMAIL 可用 DCL 命令调用：

```
$ HMAIL
HMAIL>
```

8.3 HMAIL 特性

HMAIL 在 OpenVMS MAIL 公用程序上增加的特性，将会在以下的各节中逐一描述。有关 OpenVMS MAIL 公用程序的特性，请参阅 *《OpenVMS User's Manual》*。

8.3.1 汉字个人名

HMAIL 让用户可以在个人名中包含汉字字符。例如，用户 TENG 要以汉字个人名发送文件 FILE.TXT 至用户 HANZI::LEE, 可以输入以下命令：

```
HMAIL> send/personal_name="邓立人，计算机工程部" file.txt
送往：hanzi::lee
主题：紧急会议
```

在节点 HANZI 上的用户 LEE 会收到以下信息：

```
新邮件到达 HANZI 节点来自 TENG "邓立人，计算机工程部" (15:51:02)
```

8.3.2 汉字主题名

HMAIL 让用户可以在主题名中包含汉字。以下例子说明如何进行。

```
HMAIL> send file.txt
送往： hanzi::lee
主题： 紧急会议
```

8.3.3 汉字文件夹名

文件夹名可以包含汉字。以下例子说明如何产生一个汉字名称的文件夹。

```
HMAIL> select mail
%MAIL-I-SELECTED, 选择 5 个信息

HMAIL> move/all 会议报告
文件夹 会议报告 不存在
您想建立吗 (Y/N, 默认是 N)? y
%MAIL-I-NEWFOLDER, 文件夹 会议报告 已建立
```

8.3.4 汉字字符串匹配

在 HMAIL 中，*SEARCH* 命令的检索字符串和 *SELECT*、*SET FOLDER*、*DIRECTORY* 以及 *READ* 命令的 */SUBJECT_SUBSTRING* 限定词均可使用汉字。例如：

```
HMAIL> select 会议报告
%MAIL-I-SELECTED, 选择 5 个信息

HMAIL> directory
                                                    会议报告
    # 来自                日期          主题

    1 HANZI::LEE          8-JUL-1991    中文邮递
    2 LIU                 17-JUL-1991   数据库资料
    3 SYSTEM              12-OCT-1991   数据库更新需知
    4 CHAN                14-OCT-1991   系统数据库资料
    5 SYSTEM              20-OCT-1991   紧急会议

HMAIL> directory/subject_substring=数据库
                                                    会议报告
    # 来自                日期          主题

    1 LIU                 17-JUL-1991   数据库资料
    2 SYSTEM              12-OCT-1991   数据库更新需知
    3 CHAN                14-OCT-1991   系统数据库资料
```

8.3.5 汉字整理序列的文件夹名显示

在 OpenVMS MAIL 中，文件夹名只可以按 ASCII 整理序列显示。在 HMAIL 中，用户可以指定根据汉字整理序列显示文件夹名。*DIRECTORY/FOLDER* 命令增加了一个新的限定词 */COLLATING_SEQUENCE*，语法如下：

/COLLATING_SEQUENCE = ({ ASCII | { QUWEI | PINYIN | STROKE | RADICAL } , ... })

以下规则适用于 */COLLATING_SEQUENCE* 这个新的限定词：

- 此限定词要在 *DIRECTORY/FOLDER* 命令中指定才有效。
- 关键字 ASCII 只可单独指定，不可与其他关键字混合。
- 除 ASCII 外的其他关键字，都可指定多关键字；提供的第一个关键字是主整理序列，第二个是次整理序列，依此类推。
- 重复的关键字不予理会。
- 指定一个以上的关键字时才需要括号。
- 如不指定此限定词，文件夹名就会按 ASCII 序列显示。

以下例子说明这个新限定词的用法：

```
HMAIL> select 会议报告
%MAIL-I-SELECTED, 选择 5 个信息
HMAIL> directory/folder
在 DISK$USER:[USER]MAIL.MAI;1 中的文件夹列表
     要取消列表，可按 CTRL/C
CFMS                    CRDB
CVMS                    MAIL
会议报告                会议程序
数据库资料              私人邮件
系统资料                中文邮件

HMAIL>directory/folder/collating=(stroke,radical)
在 DISK$USER:[USER]MAIL.MAI;1 中的文件夹列表
     要取消列表，可按 CTRL/C
CFMS                    CRDB
CVMS                    MAIL
中文邮件                会议报告
会议程序                私人邮件
系统资料                数据库资料

HMAIL> directory/folder/collating=radical
在 DISK$USER:[USER]MAIL.MAI;1 中的文件夹列表
     要取消列表，可按 CTRL/C
CFMS                    CRDB
CVMS                    MAIL
中文邮件                系统资料
会议报告                会议程序
数据库资料              私人邮件

HMAIL> directory/folder/collating=ascii
在 DISK$USER:[USER]MAIL.MAI;1 中的文件夹列表
     要取消列表，可按 CTRL/C
CFMS                    CRDB
CVMS                    MAIL
```

会议报告	会议程序
数据库资料	私人邮件
系统资料	中文邮件

除了文件夹名显示的序列外，在 */START* 限定词中提供的值还会按选择的显示整理序列与文件夹名比较。因此，您如果指定
/COLLATING=STROKE, 文件夹名的笔画整理值就会进行比较，以决定哪些文件夹名要显示出来。以下例子说明如何进行。

```
HMAIL> directory/folder/collating=stroke/start= 记
在 DISK$USER:[USER]MAIL.MAI;1 中的文件夹列表
     要取消列表，可按 CTRL/C
会议报告                        会议程序
系统资料                        私人邮件
数据库资料

HMAIL>directory/folder/collating=radical/start= 记
在 DISK$USER:[USER]MAIL.MAI;1 中的文件夹列表
     要取消列表，可按 CTRL/C
数据库资料                      私人邮件

HMAIL> directory/folder/collating=ascii/start= 记
在 DISK$USER:[USER]MAIL.MAI;1 中的文件夹列表
     要取消列表，可按 CTRL/C
数据库资料                      私人邮件
系统资料                        中文邮件
```

8.3.6 调用 HTPU 作为默认编辑程序

在 HMAIL 中，会调用 HTPU 作为默认编辑程序。您可以使用 SET EDITOR 命令设置选择的编辑程序来取代此默认。例如，您可以用以下命令把默认编辑程序设置成其他的编辑程序：

```
HMAIL> set editor [其他编辑程序]
```
[1]

1. [其他编辑程序]指在 OpenVMS 之下运行的其他文本编辑程序， 例如 TPU。

8.4 命令概要

下表概述与汉字处理有关的 HMAIL 命令。

命令 / 限定词	描述
SET PERSONAL_NAME, /PERSONAL_NAME	用户可使用汉字个人名字。
/SUBJECT	用户可使用汉字主题名。
COPY, DIRECTORY, FILE, MOVE, READ, SELECT, SET/SHOW FOLDER, SET WASTEBASKET	可指定汉字文件夹名。
SEARCH, /SUBJECT_SUBSTRING	检索字符串中可包含汉字，以找出汉 字字符串。
DIRECTORY/FOLDER /COLLATING_SEQUENCE	指定文件夹名按汉字整理序列如 QUWEI, RADICAL, STROKE 及 PINYIN 显示。
DIRECTORY/FOLDER/START	指定根据当前选择的整理序列要显示的第一个文件夹名。
HELP	如果 HANZIGEN 的终端输出设置是 HANZI_MSG, 便显示汉字求助文本。

第 9 章
HTPU 和 HEVE 公用程序

本章描述 HTPU 和 HEVE 公用程序的特性。

9.1 综述

HTPU 和 HEVE 增强 DECTPU 和 EVE 的功能，可支持汉字文本编辑和 HEDT 功能。这种公用程序既支持字符单元终端 (CCT) 用户接口，也支持 DECwindows Motif/Hanzi 用户接口。

HTPU/HEVE 以 DECTPU/EVE 为基础，由两层组成，即 HTPU (Hanzi Text Processing Utility) 及 HEVE (Hanzi Extensible Versatile Editor)。HEVE 通过 HTPU 内设过程调用汉字文本处理的功能。

9.2 使用 HTPU 和 HEVE

要使用 CCT 接口调用 HTPU, 并以 HEVE 作为默认编辑程序，您可使用以下的命令：

```
$ EDIT/HTPU TEXT.TXT
```

您也可以如下定义一个外来命令，从而缩短调用命令：

```
$ HEVE:== $EDIT/HTPU
```

要使用 DECwindows Motif/Hanzi 接口调用 HTPU, 并以 HEVE 作为默认编辑程序，您可使用以下的命令：

```
$ SET DISPLAY/CREATE/NODE=< 您的工作站的节点名 >
$ HEVE/DISPLAY=MOTIF TEXT.TXT
```

TEXT.TXT 是要编辑的文件。

一旦启动 HEVE, 您可按键盘键输入文本。您也可以使用 HEVE 命令和预定义功能键对文本进行编辑。 表 9-1 列出那些预定义功能键。

表 9-1 HEVE 预定义编辑键及其功能

键	功能	键	功能
CTRL/A	转换态（插入 / 重写）	<FIND>	寻找指定串
CTRL/B	再调用先前的一个 HEVE 命令	<INSERT_HERE>	把选定范围的文本插入当前光标位置
CTRL/E	把光标移至行尾	<REMOVE>	移去选定范围
CTRL/H	把光标移至行首	<SELECT>	选定文本范围

表 9-1 HEVE 预定义编辑键及其功能

CTRL/J	擦除当前字	<PREV_SCREEN>	上一个屏幕
CTRL/L	插入分页	<NEXT_SCREEN>	下一个屏幕
CTRL/U	擦除行首	<F10>	退出 HEVE
CTRL/W	刷新屏幕	<F11>	更改方向（正向或反向）
CTRL/Z	退出 HEVE	<F12>	按行移动光标
<DELETE>	删除前一个字符	<F13>	擦除当前字
上箭头	上移	<F14>	转换态（插入 / 重写）
下箭头	下移		
右箭头	右移		
左箭头	左移		

有关 HEVE 及其命令的详情，可参阅 *《HEVE 用户手册》*，或在 HEVE 内 按下 <DO> 键，输入 HELP 命令。

9.3 HTPU 特性

HTPU 提供一套内设过程及可调用的接口，供用户编制汉字编辑程序。您可以仿照 HTPU 默认编辑程序 HEVE, 只运用 HTPU 语言及其内设过程编写编辑程序，或透过可调用接口在应用程序中调用 HTPU。

较之 DECTPU, HTPU 增强的功能包括：

- 提供字符分类功能的新内设过程
- 支持汉字字符处理的经修改内设过程
- 在 DECwindows Motif/Hanzi 窗口环境中使用的汉字输入法

有关一整套新的或经修改的内设过程的详情，可参阅 *《HTPU 和 HEVE 用户参考手册》*。有关 DECTPU 内设过程的详情，可参阅 *《DEC Text Processing Utility Reference Manual》*。*《Guide to the DEC Text Processing Utility》* 描述 HTPU 语言。

HTPU 可调用接口的详情，可参阅本手册附录 B.4。有关 DECTPU 可调用接口的详情，请参阅 *《OpenVMS Utility Routines Manual》* 第 14 章。

9.4 HEVE 特性

HEVE 以 HTPU 语言及内设过程为基础，可以编辑汉字字符及 ASCII 字符。HEVE 支持 EVE 的所有功能[1]，且有下列扩充功能：

- 形成符号及画线画框功能
- 半形 / 全形字符互换
- 模拟 HEDT 的功能

1. 唯不支持 MCS 字符编辑

- DECwindows Motif/Hanzi 窗口环境的汉字下拉及弹出项目单

有关 HEVE 专有的命令，可参阅 *《HTPU 和 HEVE 用户参考手册》*。有关 EVE 的命令，可参阅 *《Extensible Versatile Editor Reference Manual》*。

第 10 章
HDUMP 公用程序

HDUMP 是 DUMP 的汉字版本，可以用于显示和打印 DEC 汉字字符集的文件内容，增强了 DUMP 公用程序。

HDUMP 和 DUMP 的主要区别在于 HDUMP 允许正确显示双字节汉字字符。当一行上只剩下一个字节而接着来的是一个双字 DUMP 便不能进行正确处理，DUMP 只会把该汉字的第一字节放在行末，把第二字节卸出到下一行上，所以造成错误结果。然而，HDUMP 可把该汉字的第一、第二字节一同移到下一卸出行上，在显示字段的左边线前显示或打印一个 ASCII 字符位。

所有在认可汉字范围外的双字节代码均以句点（.）显示或打印。

HDUMP 的命令格式是：

$ HDUMP 文件名

所有的命令限定词均与 DUMP 的相同。

以下是 HDUMP 的一个例子。

```
$ TYPE HANZI.DAT

美国ＤＥＣ计算机有限公司电话： 3-7315211
美国 DEC 计算机有限公司电话： 3-7315211

$ HDUMP/RECORD HANZI.DAT

Hdump of file WRKD$:[HANZI.SRC]HANZI.DAT;2 on 4-MAR-1991 15:16:48.94
File ID (3210,43,0)    End of file block 1 / Allocated 3
Record number 1 (00000001), 39 (0027) bytes
ABBE3CB C6BCC3A3 C5A3C4A3 FAB9C0C3  美国ＤＥＣ计算机 000000
2D33BAA3 B0BBE7B5 BECBABB9 DECFD0D3 有限公司电话:3- 000010
                    313132 35313337 7315211.........000020
Record number 2 (00000002), 36 (0024) bytes
CFD0D3FA BBE3CBC6 BC434544 FAB9C0C3 美国 DEC 计算机有 000000
3133372D 33BAA3B0 BBE7B5BE CBABB9DE 限公司电话:3-731 000010
                          31313255 5211.........000020
```

第 11 章
汉字日期和时间支持

本章描述汉字日期和时间的支持。

11.1 综述

输入输出格式中的汉字日期和时间均受支持。另备有汉字语言表及预定义日期和时间输出格式。系统及用户进程如相应地设汉字日期和时间输出格式可在 DCL 命令 DIR 以及 DCL 公用程序 MAIL 和 HMAIL 中使用。此外，用户在定义自己的用户定义日期和时间输入及输出格式时，可以使用汉字语言表。这些日期和时间输入及输出格式可以由 OpenVMS RTL 的日期和时间操纵例行程序使用。有关日期和时间操纵例行程序及其用法的详述请参阅 *《VMS RTL Library (LIB$) Manual》*。

11.2 操作准备

用汉字日期时间前，系统经理（或有 CMEXEC、SYSNAM 和 SYSPRV 特权的用户）必须执行命令过程 SYS$MANAGER:LIB$DT_STARTUP.COM，以便为日期时间定义多种输出格式（包括预定义汉字输出格式）。此外，也可定义汉字日期时间单元。命令如下：

```
$ ! Choose Hanzi as language
$ DEFINE SYS$LANGUAGES HANZI
$ ! Define Hanzi output formats and date and time elements
$ @SYS$MANAGER:LIB$DT_STARTUP.COM
```

最理想是从 OpenVMS/Hanzi 的 SYSTARTUP_VMS.COM 执行以上命令。

11.3 预定义汉字输出格式

下表展示预定义汉字日期时间输出格式。有关使用的助忆符的描述，请参阅 *《VMS RTL Library (LIB$) Manual》*。

表 11-1 预定义输出日期格式

日期格式逻辑名	格式	例子
LIB$DATE_FORMAT_042	!Y4 年 !MNB 月 !DB 日 !WAU	1991 年 11 月 12 日（二）
LIB$DATE_FORMAT_043	!Y4 年 !MNB 月 !DB 日 !WU	1991 年 11 月 12 日星期二

表 11-2 预定义输出时间格式

时间格式逻辑名	格式	例子
LIB$TIME_FORMAT_021	!MIU!HB2 时 !MB 分 !SB 秒	下午 2 时 16 分 26 秒

11.4 预定义汉字语言表

汉字语言表可定义多个逻辑名。这些逻辑名可按 11.6 节的描述在用户定义输出格式中使用。下表列出这些逻辑名以作参考：

表 11-3 汉字语言表

逻辑名	输出助忆符	等效名
LIB$WEEKDAYS_ [U|L|C]	W[U|L|C]	″星期一″, ″星期二″,″星期三″,″星期四″, ″星期五″,″星期六″, ″星期日″
LIB$WEEKDAY_ ABBREVIATIONS_ [U|L|C]	WA[U|L|C]	″(一)″, ″(二)″, ″(三)″,″(四)″, ″(五)″, ″(六)″, ″(日)″
LIB$MONTHS_ [U\|L\|C], LIB$MONTHS_ ABBREVIATIONS_ [U|L|C]	MA[U|L|C], MAA[U|L|C]	″一月″, ″二月″,″三月″, ″四月″,″五月″, ″六月″, ″七月″, ″八月″, ″九月″, ″十月″,″十一月″, ″十二月″
LIB$MI_ [U|L|C]	MI[U|L|C]	″上午″, ″下午″

11.5 选择运行时间汉字输出格式

要为日期或时间，或日期和时间两者选择一个特定的汉字格式，用户必须使用 11.3 节定义的逻辑名来定义 LIB$DT_FORMAT 逻辑名。例如：

```
$ DEFINE SYS$LANGUAGES HANZI
$ DEFINE LIB$DT_FORMAT LIB$DATE_FORMAT_042,LIB$TIME_FORMAT_021
```

上述两个逻辑名的次序决定了其输出的次序。上述定义使日期以指定的格式输出日期后是一个空格，之后是以指定格式表示的时间，如下所示：

1991 年 11 月 12 日 （二） 下午 2 时 16 分 26 秒

11.6 用户定义汉字输出格式

用户可使用自己的执行态逻辑名以汉字来定义其个人的输出格式。例如：

```
$ DEFINE/EXEC/TABLE=LNM$DT_FORMAT_TABLELIB$DATE_FORMAT_601 -
_$  "系统时间： !Y4 年 !MNB 月 !DB 日"
$ DEFINE/EXEC/TABLE=LNM$DT_FORMAT_TABLE LIB$TIME_FORMAT_601 -
_$  "!MIU!HB2 时 !MB 分 !SB 秒 !WU"
```

定义所要的格式后，用户可以在下列命令中使用这些格式：

```
$ DEFINE SYS$LANGUAGES HANZI
$ DEFINE LIB$DT_FORMAT LIB$DATE_FORMAT_601,LIB$TIME_FORMAT_601
```

产生的汉字日期时间输出如下：

系统时间： 1991 年 11 月 12 日 下午 2 时 16 分 26 秒 星期二

11.7 选择运行时间汉字输入格式

要为日期时间输入选择一个特定的汉字格式以供应用，用户必须定义 LIB$DT_INPUT_FORMAT 逻辑名。LIB$DT_INPUT_FORMAT 的等效名可以包含 11.4 节所描述的预定义汉字语言表中的助忆符。

11.8 例子

11.8.1 HMAIL

```
$ DEFINE SYS$LANGUAGE HANZI
$ DEFINE LIB$DT_FORMAT LIB$DATE_FORMAT_043,LIB$TIME_FORMAT_021
$ HMAIL
                                                              MAIL
   # 来自            日期                          主题
   1 NODEA:SYSTEM    1991 年 11 月 16 日 星期六   作报告

HMAIL> 1
#1       1991 年 11 月 16 日 星期六 上午 11 时 20 分 53 秒        MAIL
来自： NODEA::LEE "工程部： 李国基"
送往： CHAN
抄送：
主题： 工作报告

工业部长：

以下是我的工作报告：

                :
```

:

```
HMAIL> exit
```

11.8.2 DIR 命令

```
$ DEFINE SYS$LANGUAGE HANZI
$ DEFINE LIB$DT_FORMAT LIB$DATE_FORMAT_043,LIB$TIME_FORMAT_021
$
$ DIR/FULL TEST.DAT

目录 USER$:[LEE]
TEST.DAT;1                          文件 ID:(2229,2,0)
大小：        1/3       拥有者：           [SYSTEM]
建立于：  1991 年 11 月 17 日 星期日    下午 6 月 35 分 25 秒
修订于：  1991 年 11 月 17 日 星期日    下午 6 月 35 分 28 秒 (1)
满期：    〈未指定〉
备份：    〈没有备份被记录〉
文件组织：          顺序的
文件属性 ：          分配： 3,    扩展： 0, 全局缓冲计数： 0, 没有版本极限
记录格式：          可变长度， 最大 74 个字节
记录属性：          Carriage return carriage control
RMS 属性：          无
记日志已启动：      无
文件保护：          System:RWED, Owner:RWED,Group:RE, World:
存取控制表： 无

共 1 个文件， 1/3 块。

$ DIR/DATE TEST*.DAT

目录 USER$:[LEE]



TEST.DAT;1    1991 年 11 月 17 日 星期日       下午 6 时 35 分 25 秒
TEST1.DAT;1   1991 年 11 月 17 日 星期日       下午 6 时 36 分 19 秒
TEST2.DAT;1   1991 年 11 月 17 日 星期日       下午 6 时 36 分 22 秒
TEST3.DAT;1   1991 年 11 月 17 日 星期日       下午 6 时 36 分 26 秒
TEST4.DAT;1   1991 年 11 月 17 日 星期日       下午 6 时 36 分 31 秒
```

共 5 个文件。

11.8.3 RTL 日期 / 时间操纵例行程序

```
$ TYPE DATETIME.C
/* DECLARATIONS */
#inciude stdio.h
#include descrip.h
int lib$convert_date_string();
int lib$format_date_time();
int lib$get_input();
main()
{
unsigned int quadtime[2];  /*quadword to store time */
unsigned char instr[40];   /*input date time buffer */
unsigned char outstr[40];  /*output date time buffer*/
struct dsc$descriptor_s indate =   /*input descriptor */
{ 40, DSC$K_DTYPE_T, DSC$K_CLASS_S, instr };
struct dsc$descriptor_s outdate =  /*output descriptor */
{ 40, DSC$K_DTYPE_T, DSC$K_CLASS_S, outstr };
    indate.dsc$a_pointer  = instr;
    outdate.dsc$a_pointer = outstr;
    printf("Input date time:");
    lib$get_input(&indate);
    /* convert date time from input format to quadword storage*/
    lib$convert_date_string(&indate,quadtime);
    /* convert date time from quadword to output format */
    lib$format_date_time(&outdate,quadtime);
    printf("Output date time: %40.40s \n",outstr);
}
$ DEFINE LIB$DT_FORMAT LIB$DATE_FORMAT_043, LIB$TIME_FORMAT_021
$ DEFINE SYS$LANGUAGE HANZI
$
$ DEFINE LIB$DT_INPUT_FORMAT -
_$ "!Y4 年 !MNB 月 !DB 日 !MIU!HB2 时 !MB 分 !SB.!C2 秒"
$ RUN DATETIME
Input date time:1991 年 10 月 11 日 下午 2 时 16 分 26.02 秒
```

11.8 例子

```
Output date time:1991 年 10 月 11 日 星期五 下午 2 时 16 分 26 秒
$
$ DEFINE LIB$DT_INPUT_FORMAT "!Y4 年 !MAU 月 !DB 日 !HB4 时 !MB 分 !SB.!C2 秒"
$ RUN DATETIME
Input date time:  1991 年十月 11 日 17 时 12 分 22.02 秒
Output date time: 1991 年 10 月 11 日 星期五 下午 5 时 12 分 22 秒
$
```

第 12 章
调试器公用程序

OpenVMS 调试器是一个符号调试器。它是调试用户模式代码的首选调试器。它支持字符单元用户界面和图形 (DECwindows Motif) 用户界面的所有 OpenVMS 编程语言的符号调试。这两种用户界面现在都支持 DEC 汉字字符集的输入和显示。在程序变量、源代码注释或 OpenVMS 调试器出错信息中的汉字字符都能够正确显示。

注意

OpenVMS 调试器使用 XPG4 国际化模型实现对 DEC 汉字字符集的支持。它要求一个 DEC 汉字字符集的 XPG4 现场数据库。您可以在随同 OpenVMS 一起发行的 OpenVMS 118n 保存集中找到这个现场数据库。

12.1 用户环境设置

本节描述如何设置 OpenVMS 调试器的字符单元用户界面和 DECwindows Motif 用户界面来使用汉字字符。

12.1.1 字符单元用户界面

OpenVMS 调试器的字符单元用户界面一起使用汉字 SMG 和 DEC C 的 XPG4 本地化公用程序来支持 DEC 汉字字符集。要支持它，必须定义以下逻辑名：

- DBG$SMGSHR
  这个逻辑名定义 OpenVMS 调试器用于支持屏幕模式的 SMG 共享映像的名称。要允许支持 DEC 汉字字符集，定义这个逻辑名如下：
  $ DEFINE/JOB DBG$SMGSHR HSMGSHR
- SMG$DEFAULT_CHARACTER_SET
  这个逻辑名定义汉字 SMG 支持的默认字符集。要允许支持 DEC 汉字字符集，定义这个逻辑名如下：
  $ DEFINE/JOB SMG$DEFAULT_CHARACTER_SET HANZI
- LANG
  这个逻辑名为 DEC C 的 XPG4 本地化公用程序和运行时库的所有现场定义默认的现场设置。要允许在 OpenVMS 调试器中支持 DEC 汉字字符集，定义这个逻辑名如下：
  $ DEFINE/JOB LANG ZH_CN_DECHANZI

12.1.2 DECwindows Motif 用户界面

OpenVMS 调试器的 DECwindows Motif 用户界面可以显示 DEC 汉字字符集中的字符。在目录 DECW$SYSTEM_DEFAULT 或 DECW$USER_DEFAULTS 中的调试器资源文件 VMSDEBUG.DAT 可以为每个调试器窗口明确指定字体。

在 VMSDEBUG.DAT 中，DebugDefault.font 资源为所有调试器窗口指定一种默认字体。每个窗口也可以使用与默认字符的不同字体。要知详情，请参阅 VMSDEBUG.DAT 文件。

要允许在 OpenVMS 调试器窗口中支持 DEC 汉字字符集，该窗口必须使用 DEC 汉字字符集字体。可通过以下方法之一指定该字体：

1. 通过调试器窗口的资源明确指定；
2. 当没有按上述方法明确指定字体时，通过调试器的默认字体资源地 DebugDefault.font 来指定；
3. 当既没有用调试器默认字体资源，也没有用调试器窗口的字体资源来指定字体时，通过系统默认字体资源来指定。

12.2 调试器命令

本节描述在调试器命令中使用汉字字符。

12.2.1 EXAMINE 命令

EXAMINE 命令支持一个新限定词 /WCHAR_T:n。这个命令把每个被检查实体解释和显示为 n 个长字（n 个字符）的多字节文件代码序列。n 的默认值是 1。

当转换被检查的字符串时，OpenVMS 调试器使用它运行进程的现场数据库。默认是 C 现场。

12.2.2 DEPOSIT 命令

DEPOSIT 命令支持一个新限定词 /WCHAR_T:n。这个命令把被转换的多字节文件代码序列中的 n 个长字存放在指定位置。n 的默认值是 1。

当转换这个 DEPOSIT 命令指定的字符串时，OpenVMS 调试器使用它运行进程的现场数据库。默认是 C 现场。

附录 A
编程语言

OpenVMS/Hanzi 让用户在注释、字符和字符串文字中使用汉字，并在编程语言中能处理包含汉字的输入及输出数据。

有如下的限制：

- 使用 DEC GB2312 字符集的扩充字符时有一些限制。
- 以 COBOL 为例，一个字符串文字说明内如有一个汉字字符的第二字节包含 ASCII 字符 '"'（是十六进制的 22），COBOL 编译程序会把该 ASCII 字符作为一个字符串引号处理，结果字符串说明会生成一个编译错误。要解决这个问题，可以在该汉字字符后输入第二个 '"' 字符，或就 COBOL 程序来说使用 '"' 作为字符串引号。
- 对于注释，只有在开的注释定界符及关的注释定界符中包含汉字字符时 C、PASCAL 及 PL/I 编程语言才会出现问题。

下面列出一些如何在一般编程语言中使用汉字的样本程序。这些样本程序也包括 OpenVMS/Hanzi 系统的 HSY$EXAMPLE 目录下面。

A.1 在 MACRO-32 中使用汉字的例子

```
 $ TYPE SAMPLE MAR
  VAX_MACRO = 1 ;If you want to compile this  example
                ;with MACRO-32, then the VAX-MACRO symbol
                ;must be set to 0. Otherwise, you may
                ;encounter a problem during compilation.
  .TITLE SAMPLE
  .MACRO STRING STR
  .ASCID /STR/
  .ENDM STRING
.PSECT  DATA
DATA1:  .ASCID /VAX MACRO 与汉字共用/
DATA2:  STRING <VAX MACRO 中 MACRO 使用汉字>
inp:    string <数据输入:>
oup:    .long b3-b1
        .address b1
buff:   .long b3-b2
        .address b2
b1:     .ascii /数据输出：/
```

A.1 在 MACRO-32 中使用汉字的例子

```
b2:  .blkb 40
b3:

.PSECT CODE
.IF DF VAX_MACRO
        .ENTRY SAMPLE, ^m<>    ;  在注释中使用汉字.
.IFF
SAMPLE::
        .CALL_ENTRY PRESERVE=<R2,R3,R4,R5,R6,R7,R8,R9,R10, -
                                R11,R12,R13,R14,R15>
          ;在注释中使用汉字。
.ENDC
        PUSHAQ DATA1
        CALLS #1, G^LIB$PUT_OUTPUT
        PUSHAQ DATA2
        CALLS #1, G^LIB$PUT_OUTPUT
LOOP:
        PUSHAQ INP
        PUSHAQ BUFF
        CALLS #2, G^LIB$GET_ INPUT
        B1BC R0, EOF
        pushaq oup
        CALLS #1, G^LIB$GET_ ONPUT
        BRB    LOOP
EOF:    $EXIT_S
        .END SAMPLE
$
$ MACRO SAMPLE
$ LINK SAMPLE
$ RUN SAMPLE
VAX MACRO 与汉字共用
VAX MACRO 中 ＭＡＣＲＯ 使用汉字
数据输入： 金木水火土
数据输出： 金木水火土
```

```
数据输入：^Z
$
```

A.2 在 DEC FORTRAN 中使用汉字的例子

```
$ TYPE SAMPLE.FOR
        program sample
        character*40 buff
!       在注释中使用汉字
        type *,'VAX FORTRAN 与汉字共用   '

        do while (.true.)
           write (*,'(a)') '$数据输入：'
           read (*,'(a)',end=100) buff
           write (*,'(a)') '$数据输出：'//buff
        end do
100     continue
        end
$
$ FORTRAN SAMPLE
$ LINK SAMPLE
$ RUN SAMPLE

VAX FORTRAN 与汉字共用
数据输入：金木水火土
数据输出：金木水火土
数据输入：^Z
$
```

A.3 在 BASIC 中使用汉字的例子

```
$ TYPE SAMPLE.BAS

100 rem 汉字注释
110 print "VAX BASIC 与汉字共用"
120 on error go to 180
130 while 1
```

```
140         print ″数据输入:″;
150         input buff$
160         print ″数据输出 : ″;buff$
170 next
180 resume 190
190 end
$
$ BASIC SAMPLE
$ LINK SAMPLE
$ RUN SAMPLE

VAX BASIC 与汉字共用
数据输入:? 金木水火土
数据输出 : 金木水火土
数据输入:? ^Z
$
```

A.4 在 PASCAL 中使用汉字的例子

```
$ TYPE SAMPLE.PAS

program sample(input,output);
var   buff:packed array [1..40] of char;
begin (* 在注释中使用汉字 *)
        writeln ('VAX PASCAL 与汉字共用');
        write (' 数据输入:');
        while not eof do begin
                readln (buff);
                writeln (' 数据输出:',buff);
                write (' 数据输入:')
        end;
end.

$
$ PASCAL SAMPLE
```

```
$ LINK SAMPLE
$ RUN SAMPLE

VAX PASCAL 与汉字共用
数据输入： 金木水火土
数据输出： 金木水火土
数据输入： ^Z
$
```

A.5 在 COBOL 中使用汉字的例子

```
$ TYPE SAMPLE.COB

* 在注释中使用汉字
identification division.
program-id. sample.
author. 刘剑明.
environment division.
configuration section.
source-computer. vax.
object-computer. vax.
data division.
working-storage section.
77 buff pic x(40).
procedure division.
start-sample.
           display "VAX COBOL  与汉字共用   ".
loop.
           display "数据输入:" with no advancing.
           accept buff at end go to eof.
           display "数据输出:",buff.
           go to loop.
eof.
           stop run.
$
```

```
$ COBOL SAMPLE
$ LINK SAMPLE
$ RUN SAMPLE

VAX COBOL 与汉字共用
数据输入： 金木水火土
数据输出： 金木水火土
数据输入： ^Z
$
```

A.6 在 PL/I 中使用汉字的例子

```
$ TYPE SAMPLE.PLI

sample:proc options (main); /* 在注释中使用汉字 */
dcl buff char(40);
        put list ('VAX PL/I 与汉字共用');
        on endfile (sysin) stop;
        do while ('1'b);
                put skip edit (' 数据输入： ') (a);
                get edit (buff) (a(40));
                put edit (' 数据输出： ',buff) (a,a);
                end;
        end;
$
$ PLI SAMPLE
$ LINK SAMPLE
$ RUN SAMPLE

VAX PL/I 与汉字共用
数据输入： 金木水火土
数据输出： 金木水火土
数据输入： ^Z
$
```

A.7 在 DEC C 中使用汉字的例子

```
$ TYPE SAMPLE.C

#include <stdio.h>
main () /* 在注释中使用汉字 */
{
   char buff[40];
   printf("/nVAX C 与汉字共用 ");
   while (printf("/n 数据输入 :"), gets(buff)!=NULL)
      printf(" 数据输出 : %s",buff);
}

$
$ CC SAMPLE
$ LINK SAMPLE
$ RUN SAMPLE

VAX C 与汉字共用
数据输入 : 金木水火土
数据输出 : 金木水火土
数据输入 : ^Z
$
```

注意

汉字不能用 #define 包括在定义中，也不能包括在变元中供引用。

附录 B
处理汉字的编程特性

B.1 HSYSHR

HSYSHR 包括在 OpenVMS/Hanzi 系统的汉字处理运行时间库中，这是个通用运行时间库，载有提供基本汉字处理功能的例行程序。

在此系统备有关于这个库的联机求助。要知道这些例行程序的功能、调用格式、输入：

```
$ HELP @HSYHLP HSYSHR
```

运行时间库包括在可共享映象库 HSYIMGLIB.OLB 中，您可用 OpenVMS/Hanzi 所支持的任何一种编程语言来调用库的功能。要把程序与运行时间库连接，输入：

```
$ LINK PROGRAM, SYS$LIBRARY:HSYIMGLIB/LIB
```

有关详情，请参阅 *《OpenVMS/Hanzi RTL Chinese Processing (HSY$) Manual》*。

B.1.1 使用可调用 HSYSHR 例行程序的例子

```
$ TYPE HSY$EXAMPLE:SAMPLE_HSYSHR.C
/* SAMPLE_HSYSHR.C
   ---------------
   This is a C language example to demostrate the
   ability of the     HSY$ facility of the OpenVMS
   Run Time Library, HSYSHR. The program
   accepts an input file from the command line and
   formats the text within it in the following ways:
      - Left margin = 0, Right margin = 40
      - Convert all half form ASCII characters to their full
      - forms.
      - Remove leading and trailing blanks of each
      - input line.
      - Remove all embedded controls or space characters.
*/
#include <stdlib.h>
#include <stdio.h>
typedef unsigned char mstr; /*Local language string type */
typedef unsigned char *mstr_p;/* Pointer type to local  */
```

```
                                /* language string */
typedef unsigned long mchar; /*Multi-byte character type*/
#define INPUT_STR_LEN  128   /* Max length of input line*/
#define OUTPUT_STR_LEN 128*2 /* Max length of output line */
#define MARGIN         40    /* Width of output text */
#define EMPTY          0     /* Zero */
#define SUCCESS        1     /* Successful call */
#define FAILURE        0     /* Unsuccessful call */
#define SPACE          0x00000020  /* Space character in
                                      HSYSHR  format */
extern  HSY$IS_VALID(),    /* External HSY$ routines to */
        HSY$CH_RNEXT(),HSY$CH_WNEXT(),/* used */
        HSY$CH_SIZE(), HSY$CH_NCHAR(), HSY$CH_NBYTE(),
        HSY$CH_TRIM(), HSY$SKPC(),     HSY$TRA_ROM_FULL();
FILE    *fdin,             /* Input file descriptor  */
        *fdout;            /* Output file descriptor */
int     line_length;       /* Remaining number of bytes */
                           /* of the current output line*/
/* WRITE_TEXT
   ----------
   This routine will accept an input buffer and its length
   in number of characters, and write the string to the
   output file at 40 bytes per line. A blank line is printed
   if the input buffer is null.
*/
int write_text(buffer, buf_len)
mstr_p  buffer;              /* Output buffer  */
int   buf_len;               /* Buffer length in number  */
                             /* of characters            */
{
   int  char_size,          /* Character size in bytes  */
        offset,             /* Offset of input buffer   */
        i, j;               /* Loop index               */
   mchar curr_char;         /* Current multi-byte char  */
```

```
    mstr_p  buf_ptr;        /* Pointer to buffer        */
    if (buffer == NULL){    /* Empty line, next paragraph*/
        fprintf(fdout, "\n");
        if (line_length != EMPTY) fprintf(fdout,"\n");
        line_length = MARGIN;
        return SUCCESS;
    }
    offset=EMPTY;buf_ptr=buffer;/*Initialize for the loop */
    for (i= EMPTY; i < buf_len; i++) {
        curr_char = HSY$CH_RNEXT(&buf_ptr);
                                /*Read next character  and  */
                               /* advance string pointer.  */
        char_size = HSY$CH_SIZE(curr_char);
                               /* Calculate the size of the */
        if (line_length < char_size) {
                               /* character in bytes.          */
            line_length = MARGIN;
            fprintf(fdout,"\n");
        }
        for (j=0; j < char_size; j++){
                                /* Put a multi-byte character*/
             fputc(buffer[offset++], fdout);
                               /* to the output file       */
        }
        line_length -= char_size;
     }
     return SUCCESS;
}
/* TEXT_FORMAT
   -----------
   This routine accepts an input string, removes its
   railing and leading blanks, converts all half form ASCII
   to their full forms and places the
   converted string into a buffer which will be written to
```

```
   the output file.
*/
int text_format(in_str)
mstr in_str[];               /* Zero terminated input */
                             /* string                */
{
  mstr buffer[OUTPUT_STR_LEN];/* Output buffer         */
  mstr_p str_ptr,           /* Pointer to source string */
         dst_ptr;           /* Pointer to dest. string */
  mchar curr_char;          /* Current multi-byte char */
  int status,               /* Return status           */
      pos,                 /* Position of trimmed string*/
      charsofar,           /* Character processed so far*/
      noofchar,           /* No. of characters and     */
      noofbytes;          /* bytes in a string.        */
  if (!(str_ptr = HSY$SKPC(SPACE, in_str, strlen(in_str)-1)))
     return write_text(NULL,EMPTY);/* Skip leading blanks */
  pos = HSY$TRIM(str_ptr, strlen(str_ptr)-1);
                                  /* Trim trailing blanks */
  if (!(HSY$TRA_ROM_FULL(str_ptr, pos, buffer, OUTPUT_STR_LEN, &noofbytes)))
    return FAILURE;            /* Convert to full form */
  noofchar = HSY$CH_NCHAR(buffer,noofbytes);
                                  /* Calculate the length of */
                                 /* string in characters. */
  charsofar = EMPTY;
  str_ptr = dst_ptr = buffer;/* Initialize for the loop */
  while (noofchar--) {
    curr_char = HSY$CH_RNEXT(&str_ptr);
                               /* Read the next character */
    if (HSY$IS_VALID (curr_char)) {
                                  /* Test for valid multi-byte */
      HSY$CH_WNEXT(curr_char, &dst_ptr);
                               /* character and write to the*/
      charsofar ++ ;          /* destination.              */
```

```
    }
  }
  write_text(buffer, charsofar);
                                /* Write the text out to file*/
  return SUCCESS;
}
/* MAIN
  ----
  This main program check for a valid command line, open all
  necessary files and call the appropriate routine to
  process the input data.
*/
main(argc,argv)
int argc;
unsigned char *argv[];
{
  mstr  in_str[INPUT_STR_LEN]; /* Input line             */
  int   status;
  if (argc != 3) {
    printf("Usage: $ FORMAT <input_file> <output_file>\n");
    exit(EXIT_FAILURE);
  }
  if ((fdin = fopen(argv[1], "r")) == NULL) {
    printf("Error: Cannot open input file %s.\n", argv[1]);
    exit(EXIT_FAILURE);
  }
  if ((fdout = fopen(argv[2],"w")) == NULL) {
    printf("Error: Cannot open output file %s.\n", argv[2]);
    exit(EXIT_FAILURE);
  }
  line_length = MARGIN;
  while (fgets(in_str, INPUT_STR_LEN, fdin) != NULL)
    if (!(text_format(in_str))) {
        printf("Information: Processing error.\n");
```

```
        break;
    }
  printf("Information: Finish Processing.\n");
  fclose (fdin);
  fclose (fdout);
  exit(EXIT_SUCCESS);
}

$ CC SAMPLE_HSYSHR.C
$ LINK SAMPLE_HSYSHR, SYS$LIBRARY:HSYIMGLIB/LIB
$ FORMAT :== $SAMPLE_HSYSHR.EXE
$ FORMAT TEXT.IN TEXT.OUT
```

B.2 HSMGSHR

OpenVMS/Hanzi 系统包括汉字屏幕管理运行时间库 HSMGSHR 。它由一组例行程序组成，这些例行程序有助于在影象屏幕上设计、构成和监视屏幕映象。

HSMGSHR 提供两种重要功能：使程序不受终端特性转换的影响； 易于构成屏幕映象。

您可用 OpenVMS/Hanzi 所支持的任何一种编程语言来调用库功能。要把程序与 HSMGSHR 连接， 输入：

```
$ LINK PROGRAM, SYS$INPUT/OPTION
SYS$LIBRARY:HSMGSHR.EXE/SHARE
```

有关详情， 请参阅 *《OpenVMS RTL Chinese Screen Management (SMG$) Manual》*。

B.3 SORT 和 MERGE 可调用例行程序

这些例行程序使您能把排序或合并操作加入到程序应用中去处理汉字数据记录、把这些汉字数据记录排序或合并，然后再处理之。

以下的用户可调用例行程序供排序或合并操作使用。有关详情，

请参阅 *《OpenVMS Utility Routine Reference Manual》*。

SOR$BEGIN_MERGE	设置关键字变元并进行合并操作。这是唯一 MERGE 专有的例行程序。
SOR$BEGIN_SORT	传送关键字资料及排序任选项， 从而 初始化排序操作。这是只有 SORT 才有的例行程序。
SOR$END_SORT	进行清理功能，例如关闭文件及释放存储器。

SOR$PASS_FILES	把输入输出文件的名称传送至 SORT 或 MERGE; 对每一个输入文件，这必须重复一次。
SOR$RELEASE_REC	把一个输入记录传送至 SORT 或 MERGE; 对每一个记录，这必须调用一次。
SOR$RETURN_REC	把一个排序或合并记录送回程序 ; 对每一个记录，这必须调用一次
SOR$SORT_MERGE	把记录排序。
SOR$SPEC_FILE	传送一个说明文件或说明文本。必须先调用这例行程序，然后调用 SOR 的其他例行程序。
SOR$STAT	送回关于排序或合并操作的统计数字。

任何支持 OpenVMS 过程调用及条件处理标准的语言都可调用以上的例行程序。为了使用 OpenVMS/Hanzi 的 SORT 和 MERGE 可调用程序，应用程序必须使用以下命令来与运行时间库连接 :

```
$ LINK PROGRAM
```

例行程序被调用后会执行本身的功能，然后送回控制到调用程序去。

所有的串参数按描述符传送，而所有的标量参数则按地址传送。这些参数都是位置参数，用户可按值传送一个零来指定任选参数。调用程序后传送一个缩短列表可以省略详情见 *《OpenVMS Utility Routine Reference Manual》*。

用户可用下列的伪数据类型来指定汉字整理序列，这些伪数据类型以通用符号 (universal symbols) 的形式出现供用户使用。

SOR$K_SEQ_PINYIN	用于汉字 16 位文本拼音整理序列。
SOR$K_SEQ_RADICAL	用于汉字 16 位文本部首整理序列。
SOR$K_SEQ_STROKE	用于汉字 16 位文本笔画整理序列。
SOR$K_SEQ_QUWEI	用于汉字 16 位文本区位码整理序列。
SOR$K_SEQ_CHAR_PINYIN	用于汉字 16 位文本拼音整理序列，并 以汉字字符为单元。
SOR$K_SEQ_CHAR_RADICAL	用于汉字 16 位文本部首整理序列，并以汉字字符为 16 位单元。
SOR$K_SEQ_CHAR_STROKE	用于汉字 16 位文本笔画整理序列，并以汉字字符为单元。
SOR$K_SEQ_CHAR_QUWEI	用于汉字 16 位文本区位码整理序列，并以汉字字符为单元。

这些伪数据类型应被指定为关键字的数据类型。

B.3.1 状态码

上述可调用例行程序在处理汉字数据时送回的状态码与处理 ASCII 数据时送回的状态码相同。有关详情，请参阅 *《OpenVMS Utility Routine Reference Manual》*。

B.3.2 可调用例行程序的接口

用户可把数据以完整文件或单个记录的形式提交可调用例行程序。当您的程序提您就在使用文件接口。当您的程序一次提交一个记录，然后一次接收一个已排序记录时，

您就在使用记录接口。

用户要结合文件接口与记录接口，可在输入时提交文件，并在输出时接收已排序记录。另一个可行的方法是在输入时释放记录，并在输出时把已排序记录写入一个文件中。将这两个接口连接起来具有更大的灵活性。

有关详情，请参阅 *《OpenVMS Utility Routine Reference Manual》*。

B.3.3 使用可调用 SORT 和 MERGE 例行程序的例子

B.3.3.1 使用拼音整理序列调用 SORT 例行程序的程序

```
$ TYPE HSY$EXAMPLE:SAMPLE_SORT1.FOR
C
C The following FORTRAN program sorts the records in the file
C POET.DAT and creates an output file named EXAM1OUT.DAT.
C All the records in the input file are sorted in Pinyin
C collating sequence based on the first 12 bytes, that is,
C the first 6 Chinese characters in each record.
C
C
  IMPLICIT        INTEGER*4      (A-Z)
  INTEGER*2       KEYBUFF(5)
  CHARACTER*9     IN_FILE
  CHARACTER*12    OUT_FILE
  EXTERNAL SOR$K_SEQ_PINYIN
  DATA  IN_FILE, OUT_FILE /'POET.DAT','EXAM1OUT.DAT'/
  KEYBUFF(1) = 1                ! Number of KEY
  KEYBUFF(2) = %LOC(SOR$K_SEQ_PINYIN)
                                ! Pinyin dictionary sequence
  KEYBUFF(3) = 0                ! Ascending sort
  KEYBUFF(4) = 0                ! Key offset
  KEYBUFF(5) = 2                ! Size in bytes
  STATUS = SOR$PASS_FILES(IN_FILE, OUT_FILE)
  IF (.NOT. STATUS) GOTO 10
```

```
  STATUS = SOR$BEGIN_SORT(KEYBUFF)
  IF (.NOT. STATUS) GOTO 10
  STATUS = SOR$SORT_MERGE( )
  IF (.NOT. STATUS) GOTO 10
  STATUS = SOR$END_SORT( )
  IF (.NOT. STATUS) GOTO 10
  STOP 'SUCCESS'
10 CONTINUE
   STOP 'FAILURE'
   END
$ FOR SAMPLE_SORT1
$ LINK SAMPLE_SORT1
$ RUN SAMPLE_SORT1
$
```

B.3.3.2 使用多种整理序列调用 SORT 例行程序的程序

```
$ TYPE HSY$EXAMPLE:SAMPLE_SORT2.FOR
C This is a FORTRAN language example calling sort routines using
C multiple collating sequences.
C
C This program sorts data with file interface on input and record
C interface on output.Two collating sequences are specified here,
C the first being the number of stroke count with ascending sort
C order and the second being the radical sequence with descending
C sort order.
C
C
C
C Define external functions and data and initialize
  IMPLICIT INTEGER*4 (A-Z)
  CHARACTER*80 RECBUF
  CHARACTER*9  INPUTNAME ! Input file name
  INTEGER*2    KEYBUF(9) ! Key definition buffer
  EXTERNAL     SS$_ENDOFFILE
  EXTERNAL     SOR$GK_RECORD
```

```
      EXTERNAL     SOR$K_SEQ_RADICAL
                                    ! Radical collating sequence
      EXTERNAL     SOR$K_SEQ_STROKE ! Stroke collating sequence
      DATA INPUTNAME /' POET.DAT' /
       KEYBUF(1) = 2
       KEYBUF(2) = %LOC(SOR$K_SEQ_STROKE) ! Primary key STROKE
       KEYBUF(3) = 0                    ! Ascending
       KEYBUF(4) = 0                    ! Offset 0
       KEYBUF(5) = 2                    ! Size 2
       KEYBUF(6) = %LOC(SOR$K_SEQ_RADICAL)
                                    ! Secondary key RADICAL
       KEYBUF(7) = 1                    ! Descending
       KEYBUF(8) = 0                    ! Offset 0
       KEYBUF(9) = 2                    ! Size 2
       SRTTYPE = %LOC(SOR$GK_RECORD)
C  Pass SORT the file names
       ISTATUS = SOR$PASS_FILES(INPUTNAME)
       IF (.NOT. ISTATUS) GOTO 10
C  Initialize the work areas and keys
       ISTATUS = SOR$BEGIN_SORT(KEYBUF,,,,,,SRTTYPE,%REF(3))
       IF (.NOT. ISTATUS) GOTO 10
C  Sort the records
       ISTATUS = SOR$SORT_MERGE( )
       IF (.NOT. ISTATUS) GOTO 10
C  Now retrieve the individual records and display them.
5  ISTATUS = SOR$RETURN_REC(RECBUF)
       IF (.NOT. ISTATUS) GOTO 6
       ISTATUS = LIB$PUT_OUTPUT(RECBUF)
       GOTO 5
6  IF (ISTATUS .EQ. %LOC(SS$_ENDOFFILE)) GOTO 7
GOTO 10
C  Clean up the work areas and files.
7  ISTATUS = SOR$END_SORT( )
       IF (.NOT. ISTATUS) GOTO 10
```

```
   STOP ' SORT SUCCESSFUL'
10 STOP ' SORT UNSUCCESSFUL'
   END
$ FOR SAMPLE_SORT2
$ LINK SAMPLE_SORT2
$ RUN SAMPLE_SORT2
$
```

B.3.3.3 调用 MERGE 例行程序的程序

```
$ TYPE HSY$EXAMPLE:SAMPLE_MERGE.C
/* This is a C language example.
   This program merges two input files into one output file.
   The key is starting from the first byte of the record
   (offset zero byte) of size two bytes(i.e. 1 Chinese
    character).It also checks if the input files sequence
    are in the right order.
*/
#include stsdef
struct DESCRIPTOR { int leng ;
                    char *ptr ;
};
globalvalue SOR$K_SEQ_PINYIN,
            SOR$M_SEQ_CHECK;
main()
{
  char infile1[] = "EXAM31.DAT" ,
       infile2[] = "EXAM32.DAT" ,
       outfile[] = "EXAM3OUT.DAT" ;
  struct DESCRIPTOR
       inptr1 = {strlen(infile1) ,infile1},
       inptr2 = {strlen(infile2) ,infile2},
       outptr = {strlen(outfile) ,outfile};
  short lrl, key[5] ;
  int status,option;
  /* initialize the key */
```

```
    key[0] = 1 ;                    /* key no. */
    key[1] = SOR$K_SEQ_PINYIN ;     /* pinyin  */
    key[2] = 0 ;                    /* ascend  */
    key[3] = 0 ;                    /* offset  */
    key[4] = 2 ;                    /* size    */
    lrl = 131;                      /* LRL     */
    option = SOR$M_SEQ_CHECK;       /* check input sequence */
    /* merge the files */
    if(!((status=sor$pass_files(&inptr1,&outptr))
       & STS$M_SUCCESS))
        lib$stop(status);
    if(!((status=sor$pass_files(&inptr2))
       & STS$M_SUCCESS))
        lib$stop(status);
    if(!((status=sor$begin_merge(key,&lrl,&option))
       & STS$M_SUCCESS))
        lib$stop(status);
    if(!((status=sor$end_sort())
       & STS$M_SUCCESS))
        lib$stop(status);
}
$ CC SAMPLE_MERGE.C
$ LINK SAMPLE_MERGE
$ RUN SAMPLE_MERGE
$
```

B.4 可调用 HTPU 例行程序

由于可调用 HTPU 例行程序关系，您可从任何程序语言及应用程序中调用 HTPU。您可从以任何使用 OpenVMS 过程调用及条件处理标准生成调用的编程语言编写的程序中调用 HTPU。 您也可以从 OpenVMS 公用程序如 HMAIL 中调用 HTPU。 可调用 HTPU 例行程序容许您在程序中使用汉字文本处理功能。

HTPU 支持 DECTPU 所支持的同一可调用接口，即简化可调用接口和全面可调用接口。有关 DECTPU 可调用接口的详情，请参阅 *《VMS Utility Routines Manual》* 第 14 章。

简而言之，HTPU 可调用接口和 DECTPU 可调用接口的分别如下：

1. TPU$CLIPARSE 接受 ″ HTPU ″ 命令动词，而不接受 "TPU" 命令动词。
2. 您应连接 SYS$SHARE:HTPUSHR.EXE, 而不是 SYS$SHARE:TPUSHR.EXE。

3. TPU$EXECUTE_COMMAND 例行程序可执行 HTPU 专用内设过程及 DECTPU 的内设过程。有关 HTPU 专用内设过程，请参阅《HTPU 和 HEVE 用户参考手册》。《DEC Text Processing Utility Reference Manual 》描述 DECTPU 的内设过程。

B.4.1 HTPU 可调用例行程序

下列可调用例行程序可供于 HTPU 操作中使用：

1. 简化可调用接口

TPU$TPU	调用 HTPU, 相当于 DCL 命令 EDIT/HTPU。
TPU$EDIT	调用 HTPU 来读和编辑输入文件并把经修改的缓冲区写至输出文件。

2. 全面可调用接口

TPU$CLEANUP	清理 HTPU 内部数据结构，并为附加的调用作好准备。
TPU$CLIPARSE	分析命令行并为 TPU$INITIALIZE 建立项目单。
TPU$CLOSE_TERMINAL	在用户调用的例行程序中调用此例行程序和返回用户调用的例行程序期间关闭 HTPU 的终端通道。
TPU$CONTROL	调用 TPU$INITIALIZE 后，把控制传给 HTPU, 直至用户从 HTPU 退出或放弃。
TPU$EXECUTE_COMMAND	调用 TPU$INITIALIZE 后，执行 HTPU 语句。
TPU$EXECITE_INIFILE	调用 TPU$INITIALIZE 后，随即执行初始化文件。
TPU$FILEIO	默认文件输入输出例行程序。
TPU$HANDLER	默认条件处理程序。
TPU$INITIALIZE	建立异常处理程序后，初始化 HTPU。
TPU$MESSAGE	使用内设过程 MESSAGE 写出错信息和串。
TPU$PARSEINFO	使用 CLI$DCL_PARSE 分析命令后，为 TPU$INITIALIZE 建立项目单。

B.4.2 样本程序

以下是用 BLISS 编写的样本程序，以使用全面可调用接口调用 HTPU:

```
!++
!
! EXAMPLE.B32
!
!  Example program of using HTPU full callable interface.
```

```
!  To run this example:
!     1. $ BLISS EXAMPLE
!     2. $ LINK EXAMPLE,SYS$INPUT/OPTION
!         SYS$SHARE:HTPUSHR/SHARE
!         <EXIT>
!     3. $ DEFINE HTPU$TESTING <your directory>
!     4. $ RUN EXAMPLE
!
!--
MODULE test (IDENT = '001',
          MAIN = test_main,
          ADDRESSING_MODE (EXTERNAL = GENERAL)) =
BEGIN

!++
!
! FUNCTIONALITY:
!
!  To test the callable interface of HTPUSHR
!
!--
REQUIRE
   'SYS$LIBRARY:STARLET';

LITERAL
   TEST_USER_ARG = %X'F0';        !an arbitrary pattern
!
! Table of Content
!
FORWARD ROUTINE
   test_main,         ! Module entry
   test_callback,     ! Callback for tpu$initialize
   test_call_user,    ! Call user routine
   test_edit;         ! Edit using HTPU
```

```
!
! HTPU callable routines
!
EXTERNAL ROUTINE
   tpu$fileio,        ! HTPU file routine
   tpu$message,       ! Displays HTPU message
   tpu$cliparse,      ! HTPU CLI parser
   tpu$handler,       ! HTPU handler
   tpu$initialize,    ! Initialize HTPU
   tpu$execute_inifile, ! Execute initial commands
   tpu$execute_command, ! Execute HTPU statements
   tpu$control,       ! HTPU main control loop
   tpu$cleanup;       ! Clean up HTPU
!
! HTPU Literals for cleanup
!
EXTERNAL LITERAL
   tpu$m_last_time,
   tpu$m_delete_journal,
   tpu$m_delete_exith,
   tpu$m_delete_buffers,
   tpu$m_delete_windows,
   tpu$m_execute_file,
   tpu$m_execute_proc,
   tpu$m_delete_context,
   tpu$m_reset_terminal,
   tpu$m_kill_processes,
   tpu$m_close_section;
MACRO
   $test_set_bpv (bpv, proc) =
      (BEGIN
       BIND
          $test_bpv = bpv : VECTOR [2];
       $test_bpv [0] = proc;
```

```
        $test_bpv [1] = 0;
        END) %;

ROUTINE test_main =
!++
! FUNCTIONALITY:
!
!  Main routine to call HTPU
!
!--

   BEGIN
   LOCAL
      callback_bpv : BLOCK [8, BYTE],
      user_arg,
      cleanup_flag,
      status;
   ENABLE
      tpu$handler;
   !
   ! Setup the callback procedure
   !
   $test_set_bpv (callback_bpv, test_callback);
   !
  !Set the user argument for test_callback,but ignored there
   !
   user_arg = TEST_USER_ARG;
   !
   !Initialize HTPU. Initialization options are specified in
   ! test_callback
   !
   status = tpu$initialize (callback_bpv, .user_arg);
   IF NOT .status
   THEN
```

```
      RETURN .status;
   !
   ! Set up the default section file
   !
   status = tpu$execute_inifile();
   IF NOT .status
   THEN
      RETURN .status;
   !
   ! Perform editing
   !
   status = test_edit();
   IF NOT .status
   THEN
      RETURN .status;
   !
   ! Finally, we clean up the HTPU and then return
   !
   cleanup_flag = tpu$m_delete_context;
   RETURN tpu$cleanup (cleanup_flag);
   END;

ROUTINE test_callback (user_arg) =
!++
! FUNCTIONALITY:
!
! Sets up the item list for tpu$initialize
!
!--
   BEGIN
   LOCAL
      fileio_bpv : BLOCK [8, BYTE],
      call_user_bpv : BLOCK [8, BYTE],
      status;
```

```
    !
    ! Construct fileio bpv
    !
    $test_set_bpv (fileio_bpv, tpu$fileio);
    !
    ! Construct call_user bpv
    !
    $test_set_bpv (call_user_bpv, test_call_user);
    !
    ! Calls tpu$cliparse to construct item list for
    ! tpu$intialize
    ! passing it tpu$fileio as the fileio routine and
    ! test_call_user as the call_user routine
    !
    RETURN tpu$cliparse (
        $DESCRIPTOR ('HTPU htpu$testing:test.txt'),
        fileio_bpv,
        call_user_bpv);
    END;

ROUTINE test_call_user (int_param,
                        str_param : REF BLOCK[,BYTE],
                        str_out : REF BLOCK [,BYTE]) =
!++
! FUNCTIONALITY
!
! It simply prints out the integer parameter int_param
! and the string parameter str_param by means of tpu$message
!
!--
    BEGIN
    EXTERNAL ROUTINE
        lib$sget1_dd,
        lib$get_vm;
```

```
LOCAL
   buffer : VECTOR [80, BYTE],
   buffer_desc : BLOCK [8, BYTE],
   buffer_len,
   str_out_len : WORD,
   status;
!
! Form a fixed-length string descriptor
!
buffer_desc [DSC$W_LENGTH] = 80;
buffer_desc [DSC$B_DTYPE] = DSC$K_DTYPE_T;
buffer_desc [DSC$B_CLASS] = DSC$K_CLASS_S;
buffer_desc [DSC$A_POINTER] = buffer;
!
! Construct the ini_param message and print it out
!
status = $FAO ($DESCRIPTOR('Integer: !UL'),buffer_len,
               buffer_desc, ..int_param);
IF NOT .status
THEN
   RETURN .status;
buffer_desc [DSC$W_LENGTH] = .buffer_len;
status = tpu$message (buffer_desc);
buffer_desc [DSC$W_LENGTH] = 80;
!
! Construct the str_param message and print it out
!
status = $FAO ($DESCRIPTOR('String: !AF'), buffer_len,
               buffer_desc, .str_param [DSC$W_LENGTH],
                            .str_param [DSC$A_POINTER]);
IF NOT .status
THEN
   RETURN .status;
buffer_desc [DSC$W_LENGTH] = .buffer_len;
```

```
    status = tpu$message (buffer_desc);
    buffer_desc [DSC$W_LENGTH] = 80;
    !
    ! Construct return string
    ! We need to use lib$sget1_dd to allocate the memory
    ! for the return string since TPU uses lib$sfree1_dd
    ! to free our string when it needs to.
    !
    str_out_len = %CHARCOUNT(%STRING('Success'));
    status = lib$sget1_dd (str_out_len, str_out [0,0,0,0]);
    IF NOT .status
    THEN
       RETURN .status;
    CH$MOVE (.str_out_len,
           UPLIT BYTE ('Success'),
           .str_out [DSC$A_POINTER]);
    RETURN SS$_NORMAL;
    END;

ROUTINE test_edit =
!++
! FUNCTIONALITY:
!
!  Perform editing to the buffer
!
!--
    BEGIN
    MACRO
       $test_execute (command) =
         (BEGIN
          status=tpu$execute_command ($DESCRIPTOR (command));
          IF NOT .status
          THEN
              RETURN .status;
```

```
        END)%;
    LOCAL
       status;
    !
    ! It first copies two lines to the MAIN buffer
    !
    $test_execute ('COPY_TEXT("This is the first statement");
                   SPLIT_LINE');
    $test_execute('COPY_TEXT("This is the second statement");
                   SPLIT_LINE');
    !
    ! It then passes control to HTPU
    !
    RETURN tpu$control (1);
    END;
END
ELUDOM
```