# OpenVMS/Hanyu
# 使用者手冊

產品號碼： AA-PXHCB-TE

2005 年 8 月

本手冊描述中文 OpenVMS 軟體的概念、特色，以及命令。

**增訂 / 更新資料：** 本手冊取代 OpenVMS/Hanyu AXP 1.5 版 的使用者手冊。

**軟體版本：** OpenVMS/Hanyu Alpha 8.2 版

Hewlett-Packard
Palo Alto, California

Printed in Singapore

# 目 錄

# 目錄

## 第 9 章 HTPU/HEVE 公用程式



## 第 10 章 HDUMP 公用程式

## 第 11 章 片語輸入公用程式

## 第 12 章 中文日期與時間支援



## 第 13 章 中文終端機歸化設施



## 第 14 章 偵錯程式公用程式



## 附錄 A 程式語言

# 目錄



## 附錄 B 中文程式特色



## 附錄 C 中文注音碼表



## 圖



## 表

# 序 言

## 適用對象

本手冊適用於所有 OpenVMS/Hanyu 系統使用者。

## 本書之結構

本手冊共十三章及三個附錄。

- 第一章簡介本軟體。
- 第二章詳細說明此軟體所用的 DEC 中文字元集 (DEC Hanyu Character Set)。
- 第三、四、六、七、八、九、十和十一章提供啓動該軟體的資訊。包括許多利用 DCL 命令與公用程式的範例。
- 第五章著重於 HANYUGEN 公用程式。
- 第十二章提供使用中文於國際標準日期與時間的資訊。
- 第十三章提供使用中文終端機歸化設施 (Terminal Fallback Facility) 支援 BIG5 和 MITAC TELEX CODE 終端機及印表機的字元轉換。
- 第十四章提供偵錯器公用程式的中文支援使用者資訊。

對於更高階的讀者，在本手冊後有三個附錄，可依照興趣找到有關的主題。

- 附錄 A 列出 7 個在程式語言中使用中文的範例。
- 附錄 B 包含系統中中文運轉作業庫的資訊，中文運轉作業庫包括 HSYSHR、HSMGSHR、HTPU、SORT 和 MERGE 。另外還列出 HTPU、SORT 和 MERGE 可呼叫常式及其範例程式。
- 附錄 C 列出應用於 SORT 和 MERGE 的中文注音碼表。

## 常規

| 常規 | 意義 |
|---|---|
| OpenVMS、VMS | OpenVMS 及 VMS 均表示 OpenVMS 作業系統。 |
| OpenVMS/Hanyu | OpenVMS/Hanyu 表示台灣專用的中文 OpenVMS Alpha 作業系統。 |
| OpenVMS/Hanyu Alpha | OpenVMS/Hanyu Alpha 表示台灣專用的中文 OpenVMS Alpha 作業系統。 |
| Hanyu | 在這個世界上，不同的地區使用不同的中文字元集。台灣與香港使用傳統中文字，即繁體字。中國大陸與新加坡使用簡體字。日本使用 Kanji，及 韓國使用 Hanja。爲了避免在如此複雜的情況下產生混淆，本手冊使用的中文 (Hanyu) 專指中華民國台灣於 1986 年訂定的通用中文中文標準交換碼 (SICGCC) 所指的傳統中文字。 |

| | |
|---|---|
| <RETURN> | 回頭鍵 (Carriage Return)。<br>在範例與格式中，除另行規定，一般而言，使用者輸入的每一列結尾都應按下 <RETURN>。 |
| 字鍵符號 | 範例中，字鍵與鍵序均以符號形式出現，例如 <PF2> 與 CTRL/Z 。 |
| <CTRL> | CTRL/x 表示必須在按標示 CTRL 的字鍵時，同時按另一字鍵。例如，CTRL/C 、CTRL/Y 及 CTRL/O。 |
| .<br>.<br>. | 垂直省略號表示系統回覆特定命令的某些資料，應由使用者輸入的某些資料，因爲與討論之主題無直接關連，因而被省略不顯示。 |
| ... | 水平省略號表示可外加一些參數、數值或資訊。 |
| () | 格式說明：括弧表示所要的選項必須在小括弧中。 |
| [] | 格式說明：中括弧表示括弧中的內容是可以不輸入的；您可以選擇不選取、選取一個或甚至選取所有的選項。( 然而，在檔案規格中檔目名稱的語法，或是賦值述句中子串規格語法的括弧仍是必須要輸入的。) |
| {} | 格式說明：大括弧內所有可能的選項中；您必須選擇其中之一項。 |

## 相關文件

- 《OpenVMS 文件集》
- 《字元管理者 (CMGR) 使用指南》
- 《HEVE 使用者手冊》
- 《HTPU 和 HEVE 參考手冊》
- 《片語輸入公用程式使用者手冊》
- 《OpenVMS/Hanyu RTL Chinese Processing (HSY$) Manual》
- 《OpenVMS/Hanyu RTL Chinese Screen Management (SMG$) Manual》

# 第 1 章
# 簡介

除了現有的 OpenVMS 軟體特色，OpenVMS/Hanyu 具有處理中文的能力。在各種 OpenVMS/Hanyu 的功能之下，使用者可以使 ASCII 碼和 DEC 中文碼並存。

OpenVMS/Hanyu 提供的 DEC 中文資料 （中文字）爲二位元組 (byte)、八位元 (bit) 的格式，採用中華民國於 1986 年在台灣訂定的通用中文標準交換碼 （Standard Interchange Code for Generally-used Chinese Character, 簡稱 SICGCC 1986)。此外，並提供包含 8,836 個字元位置，以八位元的四位元組表示的擴充區。請參照第二章有關 DEC 中文字元字集的詳細介紹。

OpenVMS/Hanyu 之下，大部分的 OpenVMS/Hanyu DCL 命令都可以使用中文。第 3、4、6、7、8、9、10、11、12、13 和 14 章將會詳細描述這些命令。同時您還可以再透過各種程式語言，把中文寫在程式中。請參閱附錄 A 的範例。

OpenVMS/Hanyu 提供許多非常有用的公用程式，它們都可以處理中文，如：

- 字元管理者 (CMGR) 用於建立和管理使用者自定字元。
- ANYUGEN 公用程式用於設定和顯示中文終端機類型。並提供使用者選擇其所喜好語言之公用程式說明文字及系統訊息。
- HDUMP 公用程式可以十進位、十六進位或八進位顯示檔案內容、磁碟卷或是磁帶卷等。另外也可作 ASCII 和中文字元的轉換。
- HMAIL 即郵件公用程式。可支援中文卷名及主題。
- SORT 和 MERGE 可將中文字及使用者自定字元，以各種對併順序加以排序或合併。
- HTPU 是中文的文字處理公用程式。HEVE 是文字處理和畫線的公用程式。
- 片語輸入公用程式可讓使用者將一組片語載入終端機，因此使用者可以使用片語輸入。也允許使用者修改和運用片語。
- 終端機歸化公用程式可支援 BIG5 和 MITAC TELEX CODE 終端機及印表機上的字元轉換。

# 第 2 章
# DEC 中文字元集

本章描述 DEC 中文字元集的結構。

## 2.1 簡介

DEC 中文字元集包含四個字面；由 CNS 11643-1986 ，亦即通用中文交換碼 (SICGCC) 和迪吉多台灣補充字元集 (DTSCS) 組成。每一字面皆含 8,836 個字位，分別安排於 94 列和 94 行。

DEC 中文字元集可再細分爲兩個部分：

- DEC SICGCC 字元集 (2 個字面 ) 以及
- DTSCS 字元集 (2 個字面 )。

圖 2-1 爲 DEC 中文字元集結構。

**圖 2-1 DEC 中文字元集**

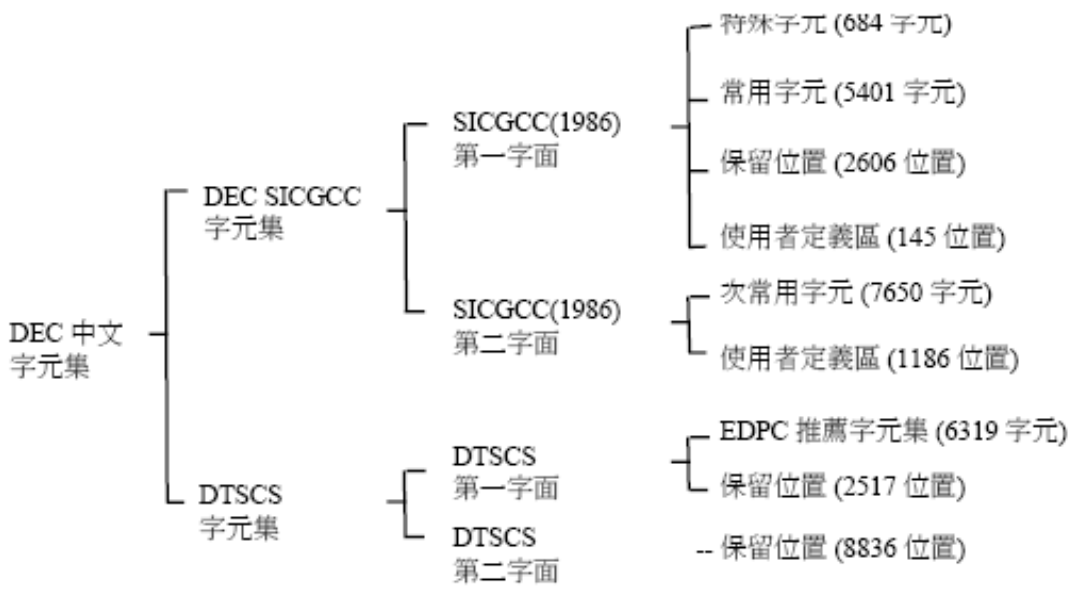



表 2-1 和 2-2 定義 DEC 中文字元集的各個區域。

表 2-1 DEC SICGCC 字元集

| 區域 | 說明 |
|---|---|
| DEC SICGCC | 此字元集是由中文和非中文組成的基本字元集， 並且完全依照 CNS 11643-1986 (SICGCC) 的標準。它涵蓋了 CNS 11643-1986 (SICGCC) 中文字集的第一和第二字面；其所界定的 13,735 個字元。 |
| 使用者自定 | 有 1,331 個字位可供使用者自定字元的區域。 |

表 2-2 DTSCS 字元集

| 區域 | 說明 |
|---|---|
| DTSCS | 第一字面提供 8,836 個字位。有 6,319 個字位指定爲 EDPC 推薦字元集之區域，另外有 2,517 個字位爲保留區域。<br>第二字面提供 8,836 個字位，可保留且供以後使用者自定。 |

## 2.2 DEC 中文碼

DEC 中文碼在 DEC 中文字元集中的圖形字元，是用二位元組和四位元組來表示：

- 二位元組的代碼代表 DEC SICGCC 字元集
- 四位元組的代碼代表 DTSCS 字元集

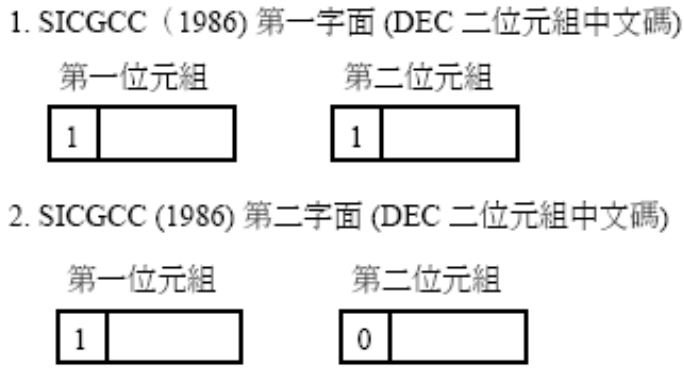
### 2.2.1 代碼結構

#### 2.2.1.1 DEC 二位元組中文碼

DEC SICGCC 字元集的每個字元皆遵循 SICGCC (1986) 的標準，以二位元組代碼表示。第一位元組的最有效位元永遠設定爲 1，而第二位元組的最有效字元可能是 1 ( 即 SICGCC (1986) 第一字面 ) 或 0 ( 即 SICGCC (1986) 第二字面 )。

圖 2-2 爲 DEC SICGCC 字元集中某個字元的表示法。

圖 2-2 DEC 中文之二位元組表示法




就 DEC SICGCC 字元集中每個字元表示法而言，DEC 二位元組中文碼及 SICGCC (l98G) 的轉換方法如下：

1. 第一字面之內碼 = SICGCC(1986) + 8080hex
2. 第二字面之內碼 = SICGCC(1986) + 8000hex

以下兩個範例說明，如何將 DEC SICGCC 第一字面和 DEC SICGCC 第二字面中字元的 DEC 二位元組中文碼轉換成內碼。

1. DEC SICGCC 第一字面

   SICGCC(1986) ：2121 (in hex)

   Internal-code-for-1st-plane = 2121 (in hex) + 8080 (in hex)

   = A1A1 (in hex)

2. DEC SICGCC 第二字面

   SICGCC(1986) ：2121 (in hex)

   Internal-code-for-2nd-plane = 2121 (in hex) + 8000 (in hex)

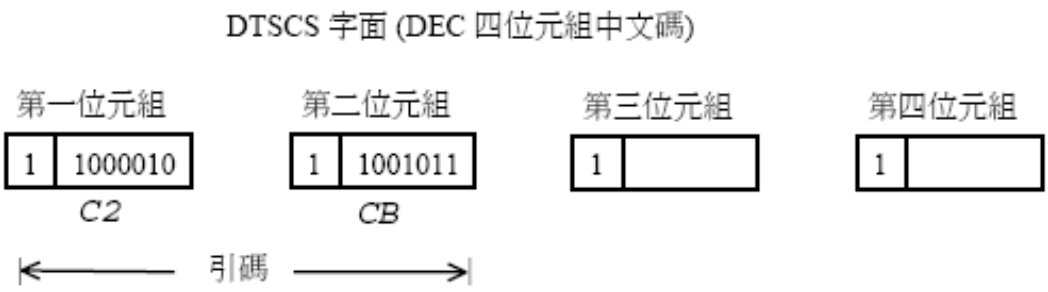
   = A121 (in hex)

### 2.2.1.2 DEC 四位元組中文碼

DTSCS 字元集中每個字元，係採用 EDPC 推薦字元集規定的四位元組代碼。前兩個位元組爲固定的引碼，即 *C2CBhex*，可作爲 DTSCS 字面指引之用。第三位元組的最有效位元爲 1。而第四位元組的最有效位元可能是 1 或 0，結構與 DEC 二位元組中文碼相同。

圖 2-3 DTSCS 字元集的字元表示法。

圖 2-3 DEC 中文字元之四位元組表示法




DEC 四位元組中文碼和 DTSCS 之間的轉換方法，類似於 DEC 二位元組中文碼。位移被加入最後兩個位元組，方法如下：

1. Internal-code-for-1st-plane = EDPC + *C2CB8080hex*
2. Internal-code-for-2nd-plane = Reserved character + *C2CB8000hex*

以下兩個範例說明，如何將 DTSCS 第一字面和 DTSCS 第二字面中字元的 DEC 四位元組中文碼轉換成內碼。

1. DTSCS 第一字面

   EDPC ：2121 (in hex)

   Internal-code-for-1st-plane= 2121(in hex) + C2CB8080(in hex)

   = C2CBA1A1 (in hex)

2. DTSCS 第二字面

   Reserved character ：2121 (in hex)

   Internal-code-for-2nd-plane= 2121(in hex) + C2CB8000 in hex)

   = C2CBA121 (in hex)

## 2.2.2 DEC 中文碼表

圖 2-4 爲二位元組代碼空間的分配和 DEC SICGCC 字元集的位置。

圖 2-5 爲四位元組代碼空間的截分和 DTSCS 字元集的位置。

圖 2-4 二位元組代碼空間的 DEC SICGCC 字元集

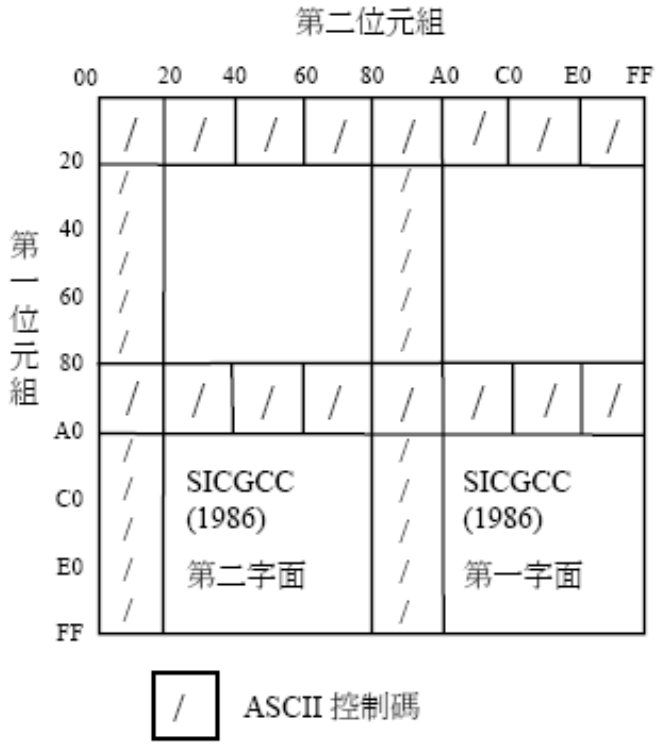
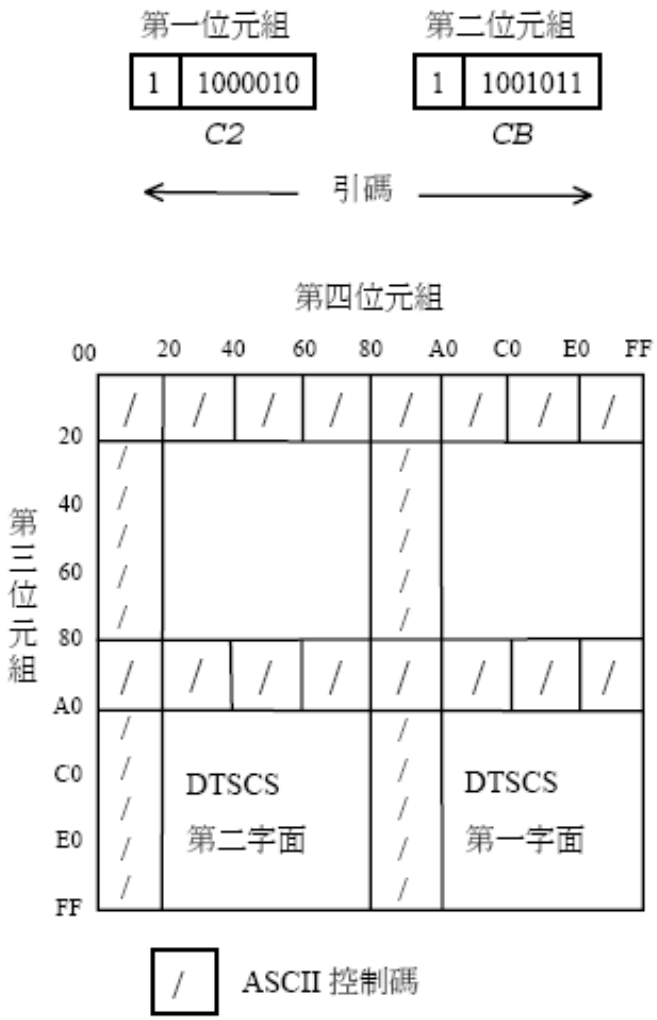


**圖 2-5 四位元組代碼空間的 DTSCS 字元集**




**附註**

陰影部份表示 ASCII 控制碼之指定區。DEC 中文碼爲了要與這些 ASCII 碼相容，因此不在這些陰影部位指定中文碼。

### 2.2.2.1 DEC 中文碼表的配置

圖 2-6 到圖 2-8 爲 DEC 中文碼表的詳細配置。

圖 2-6 DEC SICGCC 第一字面碼表的配置

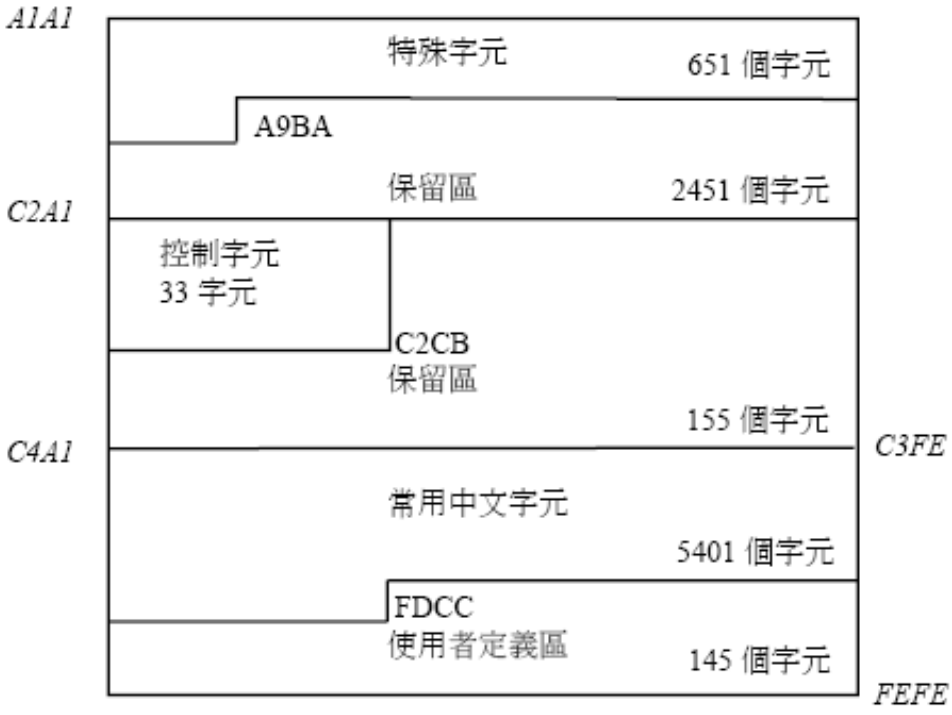


圖 2-7 DEC SICGCC 第二字面碼表的配置

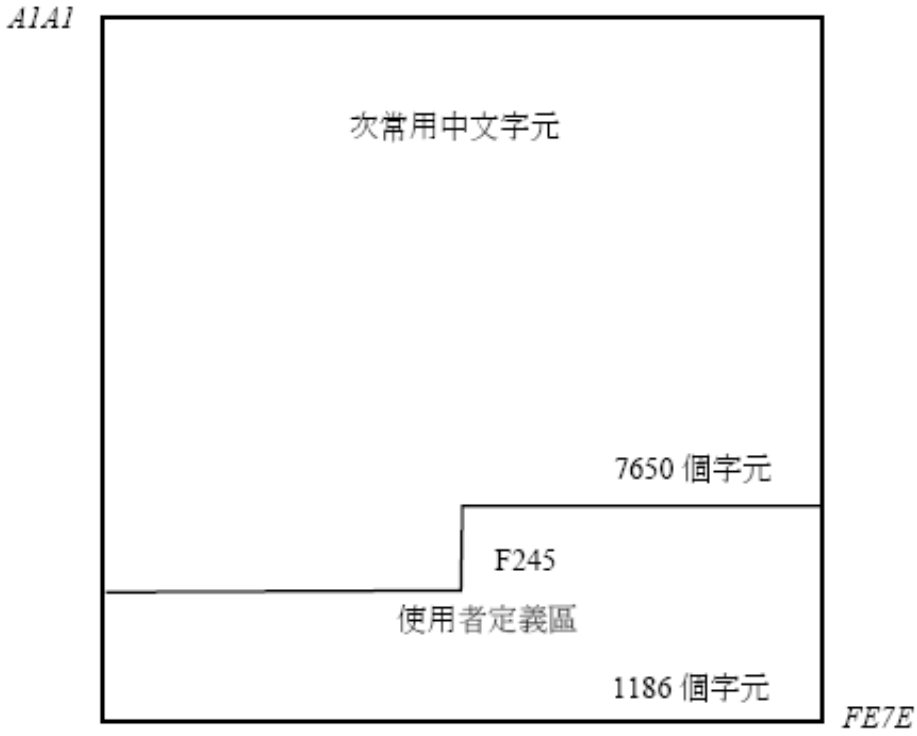
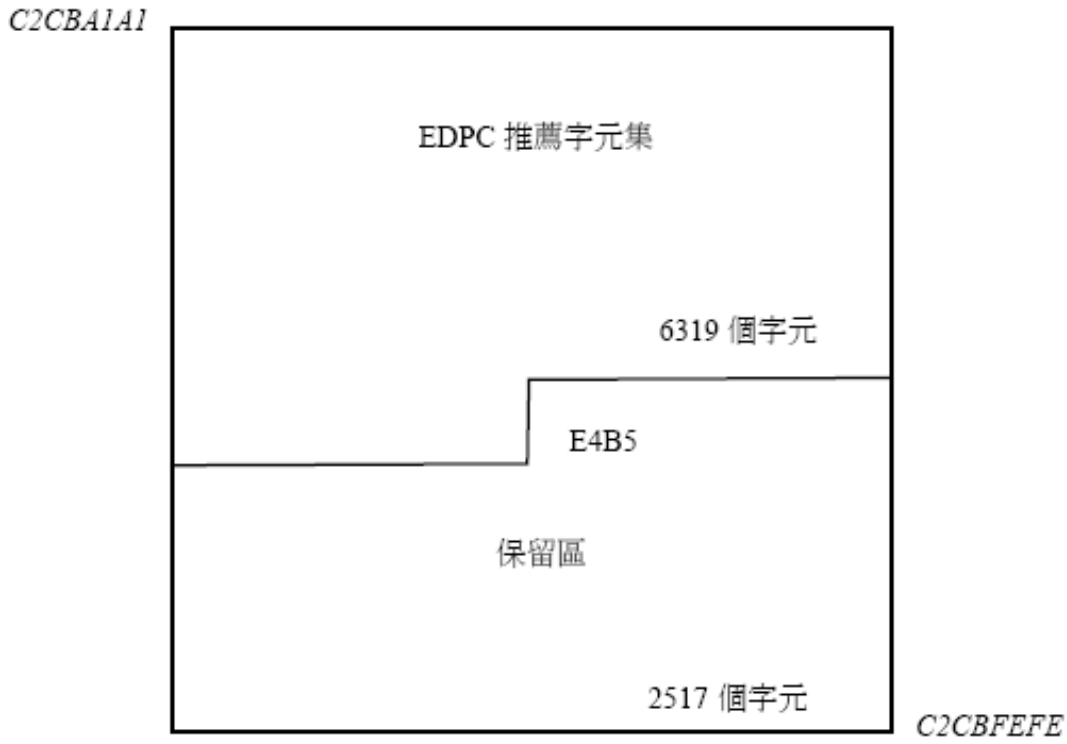


圖 2-8 DTSCS 字面碼表的配置



# 第 3 章
# 中文字終端機輸出 / 入支援

OpenVMS/Hanyu 支援下列中文字輸入和輸出的終端機特色：

- 中英文字混合輸入的編輯
- 中英文輸入的驗證
- 顯示使用者自定字元

## 3.1 編輯輸入字元

OpenVMS/Hanyu 支援所有由 OpenVMS 支援的編輯功能，如游標移動、刪 除、插入、重打和列尾的轉列。參閱 OpenVMS 文件說明以獲得所支援特徵的完整列表。

中文字顯示在終端機螢幕上時佔據一欄以上。因此，在 OpenVMS/Hanyu 編輯特色中，提供兩種作業型態：

1. ASCII 字元模態
   在此模態中，刪除鍵一次刪除終端機上的一欄。游標移動鍵一次移動一欄。
   此也是由 OpenVMS 提供的唯一模態。
2. 中文字模態
   在此模態中，刪除鍵一次刪除一個字元。游標移動鍵一次移動一個中文字。刪除字鍵 (CTRL/J) 刪除一個英文字或一串中文字。
   如果輸入列並未完全填滿，且輸入字元太寬而無法顯示 ( 例如，在該列尾僅有單一的空白顯示儲存格，但是所輸入的是中文字元 )，所輸入的中文字元將被緩衝到下一個讀取請求。

HANYUGEN 公用程式被用來在此兩個編輯模態間作切換。關於切換模態的命令，請參閱第五章《Hanyugen 公用程式》。

## 3.2 讀取檢驗

此特色根據應用程式所指定的規則來驗證輸入資料。

應用程式可以利用記號指定輸入欄；這些記號可將輸入欄分爲多個以上的輸入區。如果 DEC SICGCC 字元集中的中文字元，落在記號定義的界線上 ( 也就是說，它必須分割成兩個部份，以便插入不同的區域 )，則會被視爲無效字元。

此特色可以使用 HANYUGEN。指示將其關閉。當應用程式以一次一個位元組讀取輸入時，此特色也會被自動關閉。

## 3.3 使用者自定的字元支援

OpenVMS/Hanyu 可提供 CMGR 公用程式 ( 請參閱第六章 《字元管理者 (CMGR) 公用程式》) 給使用者建立**使用者自定字元 (User Defined Characters, UDC)**。這些自定字元可作爲 OpenVMS/Hanyu 所支援的標準字元之補充字元。

若要在終端機或印表機上顯示這些字元，OpenVMS/Hanyu 提供**終端機輸出 / 入的需求載入 (ODL)** 協定。具備 ODL 協定的終端機和印表機便可以自動地載入 UDC ( 使用者自定字元 ) 字型資料，以顯示 UDC。

## 3.3.1 硬體需求

ODL 的硬體需求如下：

1. 支援 ODL 協定的中文終端機或中文印表機控制器。其中包含：
   - 終端機
     VT382-D
   - 印表機控制器
     CP382
2. 備有支援 ODL 協定的 LAT 軟體的終端機伺服器。

## 3.3.2 LAT/Master

在 LAT 環境中 需要 LAT/Master 軟體支援 ODL 。它包括在 OpenVMS/Hanyu 和終端機伺服器上執行的軟體。

若要使 ODL 在 LAT 網路上操作，下列 DECserver 軟體的版本必須下載到相對的伺服器單元：

- DECserver 200 軟體 3.0 版 ( 或以上 )
- DECserver 300 軟體 1.0 版 ( 或以上 )
- DECserver 500 軟體 2.1 版 ( 或以上 )
- DECserver 700 軟體 1.0 版 ( 或以上 )
- DECserver 90L 軟體 1.0 版 ( 或以上 )
- DECserver 90TL 軟體 1.0 版 ( 或以上 )

若要從 LAT 功能上獲利，必需確定所有載入的主機已安裝或升級到正確的 DECserver 軟體，並且系統正在執行 OpenVMS。

請參閱有關的文件，以便獲取在載入的主機上安裝 DECserver 軟體並將伺服器影像下載到伺服器的資料。

## 3.3.3 ODL 支援的系統設定

ODL 驅動需要系統設定一虛擬裝置 FHA0: 和 FONT_HANDLER。

必須使用下列的命令來配置系統：

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN> CONNECT FHA0: /NOADAPTER /MAXUNIT=l
SYSGEN> EXIT
$ RUN /DETACH -
      /AST_LIMIT=lOO -
      /PRIORITY=8 -
      /PRIVILEGES=(CMKRNL,PHY_IO) -
      /PROCESS_NAME="FONT_HANDLER" -
      /RESOURCE_WAIT -
      /NOSWAPPING -
```

```
        /UIC=[1,20] -
        HSY$SYSTEM:FONTHANDL.EXE

$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> IO CONNECT FHAO: /NOADAPTER /DRIVER=SYS$FHDRIVER
SYSMAN> EXIT
$ RUN /DETACH -
        /AST_LIMIT=lOO -
        /PRIORITY=8 -
        /PRIVILEGES=(CMKRNL,PHY_IO) -
        /PROCESS_NAME="FONT_HANDLER" -
        /RESOURCE_WAIT -
        /NOSWAPPING -
        /UIC=[1,20] -
        HSY$SYSTEM:FONTHANDL.EXE
```

OpenVMS/Hanyu VAX 的 SYSGEN 命令或 OpenVMS/Hanyu AXP 的 SYSMAN 命令載入一虛擬裝置驅動程式，並且建立一虛擬裝置 FHA0: 以供字型資料的載入。RUN 命令建立一系統處理 FONT HANDLER，以服務來自終端機和印表機之請求。

注意：這些命令已經存在於 OpenVMS/Hanyu 系統啓動命令程序 SYS$STARTUP:HSY$STARTUP.COM，因此 OpenVMS/Hanyu 系統通常可正確地配置 ODL。

### 3.3.4 每個裝置的設定

如果裝置與終端機伺服器連接，終端機伺服器的埠必須適當地配置。

對於連接到直接線路或終端機伺服器的裝置，系統中每個終端機的埠必須個別地配置 ODL。通常，終端機埠的既定值是停用 ODL。HANYUGEN 公用程式可用於個人終端機上啓用 ODL ( 請參閱 《使用 HANYUGEN 設定終端機的埠》)。終端機或印表機也需要做實體的設定，才能啓用終端機或印表機的 ODL 功能。

#### 3.3.4.1 終端機伺服器埠的設定

在 DECserver 軟體中，埠的特色，即 ON-DEMAND [LOADING]，已在 ODL 支援中介紹了。這將允許使用者來啓用或停用每個伺服器埠上的 ODL 功能，伺服器埠爲連接到中文終端機或中文印表機。啓用或停用特定埠的 ODL 功能，可在局部態下發出以下的命令：

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

例如，永久啓用埠 3 的 ODL 功能 ( 也就是說，將設定儲存在永久的資料庫中 )，請發出以下的命令：

Local> DEFINE PORT 3 ON-DEMAND LOADING ENABLED

**附註**

若要在 DECserver 500 中設定永久的特色，必須於載入主機中執行終端機伺服器配置者 (TSC)。

若要檢驗 ON-DEMAND [LOADING] 特色已被啓用，可發出局部態的 LIST PORT CHARACTFRISTICS：

```
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, LossNotification,
Message Codes, Output Flow Control, On-Demand Loading, Verification

Local>
```

### 3.3.4.2 使用 HANYUGEN 設定終端機的埠

在 OpenVMS/Hanyu 中，使用 HANYUGEN 在終端機的埠上啓用 ODL。請參閱第五章 《Hanyugen 公用程式》，以便獲取 HANYUGEN 指令上有關啓用和停用 ODL 的細節。"/FONT" 和 "/NOFONT" 限定詞用來在 HANYUGEN 中啓用或停用需求載入。需求載入因既定值而停用。下列的命令將啓用現用終端機對話期的需求載入：

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

若要永久地啓用 ODL ( 也就是說，直到下一個系統重新開機都有效 )，請使用下列的命令：

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

### 3.3.4.3 VT382 終端機的設定

啓用中文終端機上的 ODL 必需遵循下列步驟：

- 設定 ODL 的裝置。
  按 F3 鍵，以便輸入 " 設定名錄 "。然後選取 " 終端機 " 設定，並且在 " 終端機設定 " 功能表中啓用 ODL。
- 如果終端機連接伺服器的埠，在 "Local>" 提示號的現用埠上啓用 ON-DEMAND [LOADING] 特色。參閱終端機伺服器埠的設定部份以獲知細節。
  Local> SET PORT ON-DEMAND LOADING ENABLED
- 登入 OpenVMS/Hanyu，請使用 HANYUGEN 來啓用終端機埠上的 ODL。參閱使用 HANYUGEN 設定終端機埠的部份，以獲知細節。

目前，只有 VT382 中文終端機具有處理需求載入的功能。

### 3.3.4.4 印表機的設定

在先前對話期中列出的中文印表機上，啓用 ODL 必需遵循下列步驟：

- 依照相對印表機使用手冊中所敘述的需求載入來設定印表機。
- 如果印表機連接伺服器的埠，則在 "Local>" 提示號的現用埠上啓用 ON-DEMAND [LOADING] 特色。參閱設定終端機伺服器埠的部份，以獲知細節。
- 登入 OpenVMS/Hanyu，請使用 HANYUGEN 來啓用終端機埠上的 ODL。參閱使用 HANYUGEN 設定終端機埠，以獲知細節。
  HANYUGEN>SET/DEVICE_TYPE=CP382/FONT/PERMANENT LTA100:
- 如往常一般設定印表機的埠和佇列。如果印表機連接終端機伺服器的埠，則遠程列印佇列的暫停值 (LAT$SYMTIMEOUT 邏輯名稱 ) 應在起始設定列印佇列之前被調整。使用者應該在 SYS$MANAGER:LAT$SYSTART.COM 中，定義 LAT$SYMTIMEOUT 此邏輯。
  LAT$SYMTIMEOUT 的既定值是十分鐘。如果暫停字元出現在列印清單中，則會建議更大的 LAT$SYMTIMEOUT 值，以避免暫停。下列的命令可將 LAT$SYMTIMEOUT 定義爲 15 分鐘。
  $ DEFINE/SYSTEM/EXECUTIVE LAT$SYMTIMEOUT "00:15:00"

## 3.3.5 需求載入的限制

應注意 ODL 只處理 NODMA 和 NOPASTHRU 模態中的終端機。使用者可以使用下列的命令，來察看終端機是否處於必要的模態：

```
$ SHOW TERMINAL
```

終端機若不在要求的模態下，則可以使用下列的命令來設定：

- 將現用終端機設定爲 NODMA 模態
  $ SET TERMINAL/NODMA
- 將現用終端機設定爲 NOPASTHRU 模態
  $ SET TERMINAL/NOPASTHRU

# 第 4 章
# DCL 命令和公用程式

本章列出中文終端機和印表機所需要的特性，並且說明中文字元在 DCL 命令和公用程式中的用法。

## 4.1 啓動

### 4.1.1 設定中文終端機

大多數 OpenVMS/Hanyu 公用程式需要用 HANYUGEN 公用程式將中文終端機設定爲適當的裝置型式。

HANYUGEN> SET/DEVICE_TYPE=VT382

### 4.1.2 設定中文印表機

使用 HANYUGEN 公用程式，來起始設定中文印表機所需之特性：

HANYUGEN> SET/DEVICE_TYPE=CP382/FONT/PERMANENT LTA100:

## 4.2 DCL 命令

本節描述中文字在 DCL 命令下的用法。

### 4.2.1 命令程序之引數

可在引數 (argument) 之字元串 (character strings) 中加入中文字。例如：

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

### 4.2.2 SHOW 中使用中文

您可以使用下列 SHOW 命令，以便顯示中文的系統資訊：

- $ SHOW USER
- $ SHOW QUEUE
- $ SHOW SYSTEM

- $ SHOW MEMORY
- $ SHOW DEVICE
- $ SHOW PROCESS
- $ SHOW LOGICAL
- $ SHOW TRANSLATION

如同 OpenVMS 的 " 說明 " 訊息，除了 SHOW LOGICAL 和 SHOW TRANSLATION 之外，您可以使用 HANYUGEN 公用程式，來選取英文或中文訊息。

### 4.2.3 APPEND，BACKUP，CONVERT，COPY，CREATE 和 TYPE 中使用中文

檔案 (file) 可以包含中文資料。中文字也可以用在 APPEND，BACKUP，CONVERT，COPY，CREATE 和 TYPE 等命令。例如：

$ CREATE HANYU.DAT

獨覽梅花掃臘雪

Exit

$ TYPE HANYU.DAT

獨覽梅花掃臘雪

$

### 4.2.4 ASSIGN，DEASSIGN 和 DEFINE 中使用中文

中文字可用在邏輯名稱 (logical name) 和同等的字元串。亦可用於 ASSIGN，DEASSIGN 和 DEFINE 等命令之中。

您可以使用 SHOW LOGICAL 和 SHOW TRANSLATION 命令，來顯示使用 ASSIGN 及 DEFINE 所定義之中文邏輯名稱，而且詞彙功能 F$LOGICAL 可以轉換中文。例如：

$ DEFINE 邏輯名 等效名

$ SHOW LOGICAL 邏輯名

" 邏輯名 " = " 等效名 " (LNM$PROCESS_TABLE)

$ _ 變數 =F$LOGICAL(" 邏輯名 ")

$ SHOW SYMBOL _ 變數

_ 變數 = " 等效名 "

$

### 4.2.5 DIRECTORY 中使用中文

在 DIRECTORY 命令中可支援雙語系統訊息。您可以使用 HANYUGEN 公用程式，來選取英文或中文訊息。

### 4.2.6 MESSAGE 中使用中文

訊息檔 (message file) 內的文字和簡要欄 (facility field) 可以用中文字，但是識別欄 (ident field) 則不可。

### 4.2.7 OpenVMS HELP 中使用中文

您可以使用 HELP 命令，來引動 OpenVMS " 說明設施 "，並經由 HANYUGEN 公用程式選取中文或英文，來顯示關於 OpenVMS 命令或主題之資訊。

請注意，即使輸出裝置設定爲 HANYU_MSG，您仍可以使用以下所述的命令來直接顯示英文說明。

$HELP @HELPLIB

此外，使用以下的命令，可顯示 OpenVMS/Hanyu 特定之說明主題。

$HELP @HSYHLP

### 4.2.8 READ 和 WRITE 中使用中文

包含中文字元的資料，可以組成輸入和輸出文字的部分。例如：

$ TYPE HANYU.DAT

中文寫讀測試 1

$ OPEN/READ INPUT HANYU.DAT

$ OPEN/WRITE OUTPUT HANYU1.DAT

$ READ INPUT REC

$ WRITE OUTPUT REC

$ WRITE OUTPUT " 中文寫讀測試 2"

$ CLOSE INPUT

$ CLOSE OUTPUT

$ TYPE HANYU1.DAT

中文寫讀測試 1

中文寫讀測試 2

$

### 4.2.9 REPLY 中使用中文

在訊息文字中使用中文，例如：

$ REPLY/TERMINAL=TT "REPLY 中混合中文使用 "

Reply received on HANDVF from user JOHN at _HANDVF$LTA5236: 15:19:48

REPLY 中混合中文使用

$

### 4.2.10 SET 中使用中文

您可在命令檔中指定中文的提示號或動詞，但不可在限定詞或標誌中使用中文。

### 4.2.11 SEARCH 中使用中文

您可以使用中文來搜尋一字元字串。例如：

$ TYPE HANYU.DAT

山高月小

古往今來

$

$ SEARCH/HIGHLIGHT=UNDERLINE HANYU.DAT 小

山高月小

$

### 4.2.12 中文符號

符號 (Symbol) 名稱之第一個字元必須爲英文字母，包括底線 (_) 和錢號 ($)。第二個字元以後才能用中文。中文也可放在指定之字串內。例 如：

$ _ 變數 =" 中文數據 "

$ SHOW SYMBOL _ 變數

_ 變數 = " 中文數據 "

$

### 4.2.13 在命令程序中使用之標籤

您可以使用中文作爲命令程序中之標籤名稱。您可以 GOTO 到中文標籤。

## 4.3 通用 PostScript 列印子系統

通用 PostScript 列印子系統 (WWPPS) 提供對平台上平常文字檔案的高質量列印。它能夠列印同時包含單位元組和多位元組亞洲字元的平常文字檔案。

要調用這個公用程式，您可以在 DCL 提示下使用以下命令：

$ RUN SYS$SYSTEM:WWPPS

這個公用程式將響應爲以下提示：

$ WWPPS>

另外，您可以通過定義以下外部命令縮寫這個調用：

$ WWPPS := = $SYS$SYSTEM:WWPPS

這裏是一個列印平常文字檔案的範例：

$ RUN SYS$SYSTEM:WWPPS $WWPPS>

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

或者縮寫爲：

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

對於 WWPPS 特性的詳細說明，請參閱 《OpenVMS 使用者手冊》。

# 第 5 章
# HANYUGEN 公用程式

本章描述 HANYUGEN 公用程式。

## 5.1 簡介

HANYUGEN 可以設定和顯示中文終端機的型式，並且可以啓用或停用中文終端機和中文印表機的 " 需求載入 " (On-Demand Loading) 。它並且讓使用者選擇其所喜好的語言以顯示公用程式說明文字或系統訊息。

HANYUGEN 之 SET 命令與 SHOW 命令與 DCL 非常相似，只是前者可以操作中文終端機有關的屬性。

## 5.2 啓動順序

以下命令可以啓動 HANYUGEN：

$ RUN HSY$SYSTEM:HANYUGEN

HANYUGEN>

## 5.3 命令摘要

| | |
|---|---|
| SET | 設定中文終端機特性，並且啓用或停用中文終端的 " 需求載入 "。 |
| SHOW | 顯示中文終端機特性。 |
| HELP | 列出 HANYUGEN 的命令。 |
| EXIT | 結束 HANYUGEN 並折回到 DCL 命令層次。 |

### 5.3.1 SET 命令

SET 命令能設定中文終端機型式，並且啓用 " 需求載入 "。中文終端機的型式可以爲 VT382。如果想設定中文終端機爲 ASCII 特性，則指定爲 VT100 或 VT300 系列爲其裝置型式。如果一個公用程式同時提供中文和英文的說明 / 示錯訊息，您可以使用 SET 命令的 /OUTPUT 選項來選擇輸出的語言。

#### 5.3.1.1 命令格式

SET 命令的格式爲：

```
SET [device-name[:]]    [/DEVICE_TYPE=device-type]
                        [/INPUT=input-type]
                        [/OUTPUT=output-type]
                        [/[NO]FONT]
```

[/PERMANENT]

[/SYSTEM]

其中

| | | |
|---|---|---|
| device-name | 指定連結到中文終端機之實際裝置名稱。 | |
| /DEVICE_TYPE=terminal-type | 指定 HANYUGEN 所支援的有效裝置型式： | |
| | VT382 | 指定 VT382-D 爲中文終端機。若未指定任何其他裝置型式，此爲既定值。 |
| | VT100 | 指定 VT100 爲終端機裝置。中文裝置定爲 ASCII 特性。 |
| | VT300 系列 | 指定 VT300 系列爲終端機裝置。這個限定詞可以把 VT382 重設爲 VT300 系列終端機特性。 |
| | CT282 | 指定 CT282 中文終端機的裝置。 |
| | CP382 | 指定 CP382 中文印表機控制器的裝置。 |
| /INPUT=input-type | 設定終端機爲中文或 ASCII 終端機。有效的輸入型態爲： | |
| | HANYU | 此終端機是中文終端機，能輸入中文。 |
| | ASCII | 此終端機是標準 ASCII 終端機。 |
| | 根據既定值，VT382 的輸入型式爲 HANYU。 | |
| /OUTPUT=output-type | 設定終端機能顯示中文或 ASCII 訊息。有效的輸出型態爲： | |
| | HANYU_MSG | 顯示中文訊息。 |
| | ASCII_MSG | 顯示 ASCII 訊息。 |
| | 根據既定值，VT382 的輸入型式爲 HANYU_MSG。 | |
| /[NO]FONT | 可啓用從系統字型資料庫中檢索使用者自定字型的需求載入，並且將其轉移到 VT382 中文終端機。此限定詞只可以與 VT382 裝置型式一起使用。該既定值爲 /NOFONT。 | |
| /PERMANENT | 將特性的改變指定爲永久。除了您自己的裝置外，如果想要設定裝置的特色，您就需要此限定詞。/PERMANENT 不能和 /OUTPUT 搭配使用。此限定詞需要 PHY_IO 或 LOG_IO 權限。 | |
| /SYSTEM | 將特性的改變指定爲系統既定值。/SYSTEM 不能與 /DEVICE_TYPE 及 /OUTPUT 搭配使用。此限定詞需要 PHY_IO 權限。 | |

#### 5.3.1.2 SET 命令之限制

關於 /PERMANENT 和 /SYSTEM 限定詞的限制，下表將摘要命令限定詞的清單。

**表 5-1 /SYSTEM 和 /PERMANENT 限定詞之限制摘要**

| | /SYSTEM | /PERMANENT |
|---|---|---|
| /DEVICE_TYPE | NO | YES |
| /INPUT | YES | YES |
| /OUTPUT | NO | NO |
| /[NO]FONT | YES | YES |

### 5.3.2 SHOW 命令

SHOW 命令可顯示中文終端機的特性，針對一項裝置顯示一列有關資訊，包括下列幾欄：

- 現用 SYS$INPUT 裝置的名稱
- 裝置型式
- 是否啓用 " 需求載入 "
- 輸入型式 (HANYU 或 ASCII)
- 輸出型式 (HANYU_MSG 或 ASCII_MSG)

HANYUGEN 之內，VT382 的裝置型式會以 HANYU_VDU 顯示。

此外，CP382 裝置型式則會顯示爲 HANYU_PRT。

不過，在 DCL SHOW 命令中，經 HANYUGEN 設定爲 VT382 的終端機，將以 VT300 系列作爲裝置名稱。

#### 5.3.2.1 命令格式

SHOW 的命令格式爲：

SHOW [device-name[:]][/ALL][/PERMANENT][/SYSTEM]

其中

| | |
|---|---|
| device-name | 指定由 HANYUGEN 顯示的實際裝置名稱。 |
| /ALL | 顯示所有終端機裝置的特性。此限定詞需要 PHY_IO 和 SHARE 權限。對於不是現用的裝置，輸出裝置將以「未知」顯示。( 請參閱下面的例子 )。 |
| /PERMANENT | 顯示終端機的永久特性。此限定詞需要 PHY_IO 或 LOG_IO 權限。 |
| /SYSTEM | 顯示系統既定值的特性。此限定詞需要 PHY_IO 權限。 |

# 5.4 命令範例

1. 顯示所有設定爲中文終端機之終端機裝置的特性。

```
HANYUGEN>SHOW
Device NameType    On Demand      Input          Output
_RTA2:HANYU_VDUDISABLE          ASCII          HANYU_MSG
HANYUGEN>SHOW/ALL
Device NameType    On Demand      Input          Output
_OPA0:LA36         DISABLE        ASCII          Unknown
_RTAl:VT300_SeriesDISABLE        HANYU          Unknown
_RTA2:HANYU_VDUDISABLE           ASCII          HANYU_MSG
_LTA5006:VT200_SeriesDISABLE     HANYU          Unknown
_LTA5007:HANYU_VDUDISABLE        ASCII          Unknown
_LTA5008:HANYU_VDUDISABLE        ASCII          Unknown
Total ：6 Terminals
```

2. 設定以中文訊息顯示的 VT382 終端機。

```
$ RUN HSY$SYSTEM:HANYUGEN
HANYUGEN> SHOW
Device name          Type          On Demand     Input      Output
_TTA0:               VT200_series  DISABLE       ASCII      ASCII_MSG
HANYUGEN> SET /DEVICE_TYPE=VT382
HANYUGEN> SHOW
Device name          Type          On Demand     Input      Output
_TTA0:               HANYU_VDU     DISABLE       HANYU      HANYU_MSG
```

3. 設定及顯示 CP382 中文印表機控制器，具備啓用的需求載入特性。

```
HANYUGEN> SET TXA1: /DEVICE TYPE=CP382 /PERMANENT /FONT
HANYUGEN> SHOW TXA1:
Device name          Type          On Demand     Input      Output
_TXAl:               HANYU_PRT     ENABLE        HANYU      Unknown
```

4. 將終端機重設成 ASCII 特性並顯示它。

```
HANYUGEN> SHOW TXA2:
Device name          Type          On Demand     Input      Output
_TTA2:               HANYU_VDU     DISABLE       HANYU      ASCII_MSG
HANYUGEN> SET TXA2: /DEVICE_TYPE=VT100 /PERMANENT
HANYUGEN> SHOW TXA2:
Device name          Type          On Demand     Input      Output
_TXA2:               T100          DISABLE       ASCII      ASCII_MSG
```

## 5.5 命令長度超過一列的寫法

HANYUGEN 命令層次之下，使用者也能像 DCL 命令一樣，輸入長度超過一列的命令。只要在列尾加上一個連字符號 「-」，就可以在次列繼續輸入命令了。例如：

HANYUGEN> SET TTA0: /DEVICE_TYPE=VT382 -

_HANYUGEN> /PERMANENT

## 5.6 註解列支援

像 DCL 命令，您可以在 HANYUGEN 命令層次輸入註解列，在註解的開始處置一"！"，該列爲註解。例如，

HANYUGEN> ! set tta0 as VT382 as Chinese terminal

HANYUGEN> SET TTA0: /DEVICE_TYPE=VT382 /PERMANENT

# 第 6 章
# 字元管理者 (CMGR) 公用程式

本章描述字元管理者公用程式的特色。

## 6.1 簡介

字元管理者 (CMGR) 是一公用程式，使用者可藉其建立及管理中文字元，尤指使用者自定字元 (UDC)。

若需要關於如何使用 CMGR 命令的詳細資訊，請參閱 *《字元管理者 (CMGR) 使用指南》*。

## 6.2 CMGR 的特色

CMGR 提供以下的特色：

- 使用三種不同字型的大小，即 24 x 24、32 x 32 和 40 x 40 像素，建立及修訂 UDC 異體字圖樣。像素 ( 圖像元素 ) 是指螢幕上最小且可顯示的單元。
- 定義 UDC 的對併特性，以便在 OpenVMS/Hanyu 的 SORT 及 MERGE 中提供 UDC 支援。代碼的對併值，儲存於對併資料檔或對併資料庫中。
- 顯示位元映射圖格式的異體字圖樣。
- 顯示暫存於系統資料庫中，或包含在使用者資料檔中的字元代碼表。
- 將指定字元代碼的可載入順序及 ( 或 ) 對併特性，抽取到指定的預先載入檔案及 ( 或 ) 對併資料檔案中。
- 抽取文字檔中指定的 UDC 可載入之順序，並且在預先載入檔中將其輸出。
- 支援 " 多重資料庫 " 系統，以便使用者能建立及使用不同字體之字型。
- 倘若使用者有 SYSPRV 權限，則可管理系統資料庫。
- 重新將四位元組字元指定爲二位元組使用者自定字元，以便使用者可使用二位元組之字元顯示四位元組字元。此命令需要使用 SYSPRV 權限來執行。

## 6.3 使用 CMGR

使用 DCL 命令來引動 CMGR 公用程式，

$ CHARACTER_MANAGER

CMGR>

在此提示號時，使用者可以輸入 CMGR 命令來執行可變函數。

在 DCL 層次也可引動 CMGR 命令，使用 CHARACTER_MANAGER DCL 命令引領 CMGR 命令即可。例如，

$ CHARACTER_MANAGER CMGR_command

當完成命令之後，將折回到 DCL 控制。

# 第 7 章
# SORT 和 MERGE 公用程式

本章描述 OpenVMS/Hanyu SORT 和 MERGE 公用程式中所提供之中文處理功能。

## 7.1 簡介

OpenVMS/Hanyu 的 SORT 和 MERGE 公用程式支援包含中文字元的檔案。其中包含的項目如下：

- 使用各種中文對併順序來處理中文字元。
- 使用者自定字元 (UDC) 的支援。UDC 的對併值，可以使用 CMGR 公用程式中的命令加以界定。如果需要關於界定 UDC 對併值的詳細資訊，請參閱 *《字元管理者 (CMGR) 使用指南》*。
- 在可呼叫的 SORT 及 MERGE 常式檔案和記錄介面中，中文對併順序的支援。
- 與 OpenVMS SORT 和 MERGE 公用程式完整相容。

## 7.2 中文對併順序

SORT 和 MERGE 依您指定的索引，從輸入檔安排記錄，並產生一個排序後的輸出檔。SORT 和 MERGE 能依下列對併順序安排含有中文字的記錄：

| | |
|---|---|
| RADICAL | 依照部首的順序而對特定的中文欄位作排序和合併。中文字的分類和安排是以 SICGCC 標準所示的部首順序爲準。 |
| STROKE | 依照筆畫的多寡爲順序，對特定的中文欄位作排序和合併。 |
| PHONETIC CODE | 中文字的分類和安排是以本手冊附錄 C 所示之注音碼順序爲準。 |
| INTERNAL CODE | 依照內碼順序，將指定的中文欄位加以排序和合併。 |

請注意：如果排序關鍵碼導入的資料既不是有效的 DEC 中文字元也不是 UDC，則系統會將該資料的對併值指定爲零。

## 7.3 命令的格式

SORT 命令的格式爲：

$ SORT[qualifiers] input-file-specification[qualifier] -

_$ output-file-specification[qualifiers]

MERGE 命令的格式爲：

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

_$ output-file-specification[qualifers]

MERGE 命令所用的輸入檔 (input file) 在合併前，先確定檔案內容的順序須與排序時所用的對併順序相同 ( 即是以部首、筆畫、注音碼或內碼的升序或降序爲準 )。在 MERGE 命令中指定的對併順序須與輸入檔的內容順序相符。

## 7.3.1 /KEY 限定詞

/KEY 限定詞用來界定 SORT 鍵，在一命令列中，至多可出現 255 次。/KEY 的子限定詞須用括弧括住。

除了在 OpenVMS SORT 和 MERGE 中提供的那些限定詞之外，OpenVMS/Hanyu 的 SORT 和 MERGE 並支援以下的子限定詞。

| | |
|---|---|
| CSIZE:n | SORT/MERGE 的 CSIZE 子限定詞，只能和區域對併順序搭配使用。此處的 n 是指 SORT 或 MERGE 鍵的長度，以字元數計算。請注意：CSIZE 和 SIZE 子限定詞，不能同時以同一鍵規格被指定。 |
| RADICAL | RADICAL 子限定詞是指排序或合併記錄時，以部首對併順序爲依據。 |
| STROKE | STROKE 子限定詞是指排序或合併記錄時，以筆畫的多寡對併順序爲依據。 |
| PHONETIC_CODE | PHONETIC_CODE 子限定詞是指排序或合併記錄時，以注音碼的對併順序爲依據。 |
| INTERNAL_CODE | INTERNAL_CODE 子限定詞是指排序或合併記錄時，依內碼對併順序爲依據。 |

/KEY 限定詞的範例如下：

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

上面的範例中，RADICAL 是主要的對併順序，而 STROKE 是次要的對併順序。請注意：可以同時指定多重中文對併順序。

## 7.3.2 /SPECIFICATION 限定詞

OpenVMS/Hanyu 的 SORT 和 MERGE，提供命令限定詞，透過該命令限定詞，可通過輸入規格檔案。規格檔案的既定檔型是 .SRT，而指定檔案的命令格式如下：

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

規格檔案的格式和功能，在 《VMS User's Manual》中有詳細的描述。規格檔案中的 /FIELD 限定詞，是用來定義 SORT 或 MERGE 鍵欄位，和 /KEY 命令限定詞非常類似。除了 OpenVMS SORT 和 MERGE 公用程式中 FIELD 限定詞的所有有效子限定詞外，OpenVMS/Hanyu 的 SORT 和 MERGE 也接受以下有效的子限定詞：

- RADICAL
- STROKE
- PHONETIC_CODE
- INTERNAL_CODE

這些代表其相對應之對併順序的子限定詞，相當於 /KEY 命令限定詞的相同子限定詞。規格檔案中 FIELD 限定詞的範例如下：

/FIELD=(NAME:Asian-field, POSITION:1, SIZE:4, PHONETIC)

請注意：多重對併順序，在一 /FIELD 限定詞中是不能被指定的。若要指定同一鍵的數個對併順序，則具有不同對併順序子限定詞之相同鍵的多重 /FIELD 限定詞，在規格檔案中應加以指定。

## 7.4 SORT 範例

1. 用部首排序

```
$ TYPE POET.DAT
李商隱   !    " 李 " 字之部首碼 75 ,    筆劃 07 ,    注音 ㄌㄧˇ ,   內碼 CAD7
白居易   !    " 白 " 字之部首碼 106,    筆劃 05 ,    注音 ㄅㄞˊ ,   內碼 C6F5
王維     !    " 王 " 字之部首碼 96 ,    筆劃 04 ,    注音 ㄨㄤˊ ,   內碼 C5DE
李白     !    " 李 " 字之部首碼 75 ,    筆劃 07 ,    注音 ㄌㄧˊ ,   內碼 CAD7
孟浩然   !    " 孟 " 字之部首碼 39 ,    筆劃 08 ,    注音 ㄇㄥˋ ,   內碼 CCF5
$
$ SORT/KEY=(POSITION:1,CSIZE:1,RADICAL) POET.DAT RADICAL.DAT
$ TYPE RADICAL.DAT
孟浩然   !    " 孟 " 字之部首碼 39 ,    筆劃 08 ,    注音 ㄇㄥˋ ,   內碼 CCF5
李白     !    " 李 " 字之部首碼 75 ,    筆劃 07 ,    注音 ㄌㄧˊ ,   內碼 CAD7
李商隱   !    " 李 " 字之部首碼 75 ,    筆劃 07 ,    注音 ㄌㄧˇ ,   內碼 CAD7
王維     !    " 王 " 字之部首碼 96 ,    筆劃 04 ,    注音 ㄨㄤˊ ,   內碼 C5DE
白居易   !    " 白 " 字之部首碼 106,    筆劃 05 ,    注音 ㄅㄞˊ ,   內碼 C6F5
```

2. 按筆畫數目排序。中文鍵的大小爲 2。

```
$ SORT/KEY=(POSITION:1,CSIZE:2,STROKE) POET.DAT STROKE.DAT
$ TYPE STROKE.DAT
王維     !    " 王 " 字之部首碼 96 ,    筆劃 04 ,    注音 ㄨㄤˊ ,   內碼 C5DE
白居易   !    " 白 " 字之部首碼 106,    筆劃 05 ,    注音 ㄅㄞˊ ,   內碼 C6F5
李白     !    " 李 " 字之部首碼 75 ,    筆劃 07 ,    注音 ㄌㄧˊ ,   內碼 CAD7
李商隱   !    " 李 " 字之部首碼 75 ,    筆劃 07 ,    注音 ㄌㄧˇ ,   內碼 CAD7
孟浩然   !    " 孟 " 字之部首碼 39 ,    筆劃 08 ,    注音 ㄇㄥˋ ,   內碼 CCF5
$
```

3. 按多重對併順序排序。在排序中對相同鍵可指定使用多重對併順序。

```
$ TYPE POET1.DAT
李商隱   !    " 李 " 字之部首碼 75 ,    筆劃 07 ,    注音 ㄌㄧˇ ,   內碼 CAD7
白居易   !    " 白 " 字之部首碼 106,    筆劃 05 ,    注音 ㄅㄞˊ ,   內碼 C6F5
王維     !    " 王 " 字之部首碼 96 ,    筆劃 04 ,    注音 ㄨㄤˊ ,   內碼 C5DE
```

```
李白      !     " 李 " 字之部首碼 75 ,    筆劃 07 ,       注音 ㄌㄧˇ ,   內碼 CAD7
孟浩然    !     " 孟 " 字之部首碼 39 ,    筆劃 08 ,       注音 ㄇㄥˋ ,   內碼 CCF5
杜甫      !     " 杜 " 字之部首碼 75 ,    筆劃 07 ,       注音 ㄉㄨˋ ,   內碼 CADB
$
$ SORT/KEY=(POSITION:1,CSIZE:1,STROKE,INTERNAL_CODE)POET1-
_$.DAT STRINTERNAL.DAT
$ TYPE STRINTERNAL.DAT
王維      !     " 王 " 字之部首碼 96 ,    筆劃 04 ,       注音 ㄨㄤˊ ,   內碼 C5DE
白居易    !     " 白 " 字之部首碼 106,   筆劃 05 ,       注音 ㄅㄞˊ ,   內碼 C6F5
李白      !     " 李 " 字之部首碼 75 ,    筆劃 07 ,       注音 ㄌㄧˇ ,   內碼 CAD7
李商隱    !     " 李 " 字之部首碼 75 ,    筆劃 07 ,       注音 ㄌㄧˇ ,   內碼 CAD7
杜甫      !     " 杜 " 字之部首碼 75 ,    筆劃 07 ,       注音 ㄉㄨˋ ,   內碼 CADB
孟浩然    !     " 孟 " 字之部首碼 39 ,    筆劃 08 ,       注音 ㄇㄥˋ ,   內碼 CCF5
$
```

## 7.5 MERGE 範例

以下的範例是將二個檔案按降序合併。

```
$ SORT/KEY=(POSITION:1,CSIZE:1,STROKE,DESCENDING)POET2.DAT-
_$ POET2.SOR
$ TYPE POET2.SOR
孟浩然    !     " 孟 " 字之部首碼 39 ,    筆劃 08 ,   注音 ㄇㄥˋ ,   內碼 CCF5
李白      !     " 李 " 字之部首碼 75 ,    筆劃 07 ,   注音 ㄌㄧˇ ,   內碼 CAD7
白居易    !     " 白 " 字之部首碼 106,   筆劃 05 ,   注音 ㄅㄞˊ ,   內碼 C6F5
$ SORT/KEY=(POSITION:1,CSIZE:1,STROKE,DESCENDING)POET3.DAT-
_$ POET3.SOR
$ TYPE POET3.SOR
溫庭筠    !     " 溫 " 字之部首碼 85 ,    筆劃 13 ,   注音 ㄨㄣㄧ ,   內碼 E4EB
李商隱    !     " 李 " 字之部首碼 75 ,    筆劃 07 ,   注音 ㄌㄧˇ ,   內碼 CAD7
王維      !     " 王 " 字之部首碼 96 ,    筆劃 04 ,   注音 ㄨㄤˊ ,   內碼 C5DE
$ MERGE/KEY=(POSITION:1,CSIZE:1,STROKE,DESCENDING)-
_$ POET2.SOR POET3.SOR POET.MER
$ TYPE POET.MER
溫庭筠    !     " 溫 " 字之部首碼 85 ,    筆劃 13 ,   注音 ㄨㄣㄧ ,   內碼 E4EB
孟浩然    !     " 孟 " 字之部首碼 39 ,    筆劃 08 ,   注音 ㄇㄥˋ ,   內碼 CCF5
```

```
李白        !       " 李 " 字之部首碼 75 ,   筆劃 07 ,  注音 ㄌㄧˇ ,   內碼 CAD7
李商隱      !       " 李 " 字之部首碼 75 ,   筆劃 07 ,  注音 ㄌㄧˇ ,   內碼 CAD7
白居易      !       " 白 " 字之部首碼 106,  筆劃 05 ,  注音 ㄅㄞˊ ,   內碼 C6F5
王維        !       " 王 " 字之部首碼 96 ,   筆劃 04 ,  注音 ㄨㄟˊ ,   內碼 C5DE
$
```

## 7.6 /SPECIFICATION 範例

以下的範例顯示 /SPECIFICATION 限定詞的用法。

```
$ TYPE POET.DAT
李商隱      !     " 李 " 字之部首碼 75 ,     筆劃 07 ,          注音 ㄌㄧˇ ,     內碼 CAD7
白居易      !     " 白 " 字之部首碼 106,    筆劃 05 ,          注音 ㄅㄞˊ ,     內碼 C6F5
王維        !     " 王 " 字之部首碼 96 ,     筆劃 04 ,          注音 ㄨㄟˊ ,     內碼 C5DE
李白        !     " 李 " 字之部首碼 75 ,     筆劃 07 ,          注音 ㄌㄧˇ ,     內碼 CAD7
孟浩然      !     " 孟 " 字之部首碼 39 ,     筆劃 08 ,          注音 ㄇㄥˋ ,     內碼 CCF5
$ 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 " 字之部首碼 39 ,     筆劃 08 ,          注音 ㄇㄥˋ ,     內碼 CCF5
李白        !     " 李 " 字之部首碼 75 ,     筆劃 07 ,          注音 ㄌㄧˇ ,     內碼 CAD7
李商隱      !     " 李 " 字之部首碼 75 ,     筆劃 07 ,          注音 ㄌㄧˇ ,     內碼 CAD7
王維        !     " 王 " 字之部首碼 96 ,     筆劃 04 ,          注音 ㄨㄟˊ ,     內碼 C5DE
白居易      !     " 白 " 字之部首碼 106,    筆劃 05 ,          注音 ㄅㄞˊ ,     內碼 C6F5
$
```

## 7.7 能處理中文之 SORT 和 MERGE 常式

您可將排序和合併的操作納入應用程式；使其能利用 SORT 和 MERGE 的常式 (routine)，處理中文資料錄。這些常式因能支援 " 程序呼叫 " 和 " 條件處置標準 "，所以可被任何一種程式語言呼叫。

有關 SORT 和 MERGE 常式的細節，呼叫它們的方式，以及一些程式範例，請參閱本手冊附錄 B。

# 第 8 章
# HMAIL 公用程式

本章描述 HMAIL 公用程式的特色。

## 8.1 簡介

HMAIL 是 MAIL 公用程式的功能提升，可以

- 使用中文字於個人名稱。
- 使用中文字於主題。
- 使用中文字於卷名。
- 使用中文字於字串搜尋。
- 依據中文字對併順序顯示卷名。
- 引動 HMAIL 的文字編輯程式。

## 8.2 啓動 HMAIL 對話期

使用下列 DCL 命令引動 HMAIL：

$ HMAIL

HMAIL>

## 8.3 HMAIL 的特色

OpenVMS MAIL 公用程式上 HMAIL 的附加特色，將在下一節中一一討論。有關 OpenVMS MAIL 的特色，請參閱 *《VMS User's Manual》*。

### 8.3.1 中文個人名稱

使用 HMAIL，可於個人名稱使用中文字。例如，若使用者 TENG 希望以中文名字發送 FILE.TXT 檔案給使用者 HANYU::LI，則可輸入下列命令：

HMAIL> set personal_name " 鄧立人 , 電腦工程部 "

HMAIL> exit

$ HMAIL/SUBJECT="Meeting agenda" FILE.TXT HANYU::LI

或者 TENG 要設定只適用於現用訊息的個人名稱，則可輸入下列命令：

HMAIL> send/personal_name=" 鄧立人 , 電腦工程部 " file.txt

致 : hanyu::li

主題 : Meeting agenda

於 HANYU 節點上的使用者 "LI" 將收到下列訊息：

新信件在 HANYU 節點來自 TENG " 鄧立人 , 電腦工程部 " (15:51:02)

### 8.3.2 中文主題

HMAIL 允許使用者於主題使用中文，下例說明使用方法：

```
HMAIL> send file.txt
致 : hanyu::li
主題 : 中文信件
```

### 8.3.3 中文卷名

卷名可使用中文，下例說明如何製作卷名爲中文的卷宗。

```
HMAIL> select mail
%MAIL-I-SELECTED, 選取 5 個訊息
HMAIL> move/all 會議報告
會議報告 卷宗不存在
是否要建立 (Y/N, 既定是 N)? y
%MAIL-I-NEWFOLDER, 建立 會議報告 卷宗
```

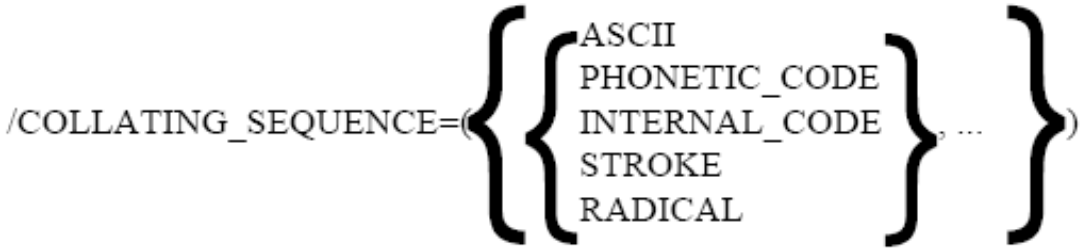
### 8.3.4 中文字串比對

在 HMAIL 中，可於命令中用中文搜尋字串。如 *SEARCH* 命令及 *SELECT*，*SET FOLDER*，*DIRECTORY* 和 *READ* 命令中的 */SUBJECT_SUBSTRING* 限定詞。例如：

```
HMAIL> select 會議報告
%MAIL-I-SELECTED, 選取 5 個訊息
HMAIL> directory

                                                    會議報告

  # 來自              日期              主旨
  1 HANYU::LI         8-JUL-1992        中文資料庫研討會
  2 LIU               17-AUG-1992       中文資訊
  3 SYSTEM            12-OCT-1992       系統資料庫更新
  4 CHAN              14-OCT-1992       緊急會議
  5 SYSTEM            20-OCT-1992       系統資料庫
HMAIL> directory/subject_substring= 資料庫

                                                    會議報告

  # 來自              日期              主旨
  1 HANYU::LI         8-JUL-1992        中文資料庫研討會
  2 SYSTEM            12-OCT-1992       系統資料庫更新
  3 SYSTEM            20-OCT-1992       系統資料庫
```

### 8.3.5 以中文對併順序顯示卷名

標準 MAIL 中，卷名只能以 ASCII 順序顯示。而使用 HMAIL，則可依據指定的中文對併順序顯示卷名。新的限定詞 */COLLATING_SEQUENCE* 以下列語法加入 *DIRECTORY/FOLDER* 命令中：




下列爲適用於 /COLLATION_SEQUENCE 限定詞的規則 :

- 本限定詞只有用在 DIRECTORY/FOLDER 命令時才有效。
- 關鍵字 ASCII 只能單獨存在，不可與其他關鍵字混用。
- 除了 ASCII 外，可以指定一個以上的關鍵字，提出的第一個關鍵字是主要對併順序，第二個是次要對併順序，以此類推。
- 重複的關鍵字會略去。
- 只有在指定一個以上的關鍵字時才須要括弧。
- 若沒有指定此限定詞，卷名將以 ASCII 順序顯示。

下面的範例說明新限定詞的使用方法。

```
HMAIL> select 會議報告
%MAIL-I-SELECTED, 選取 5 個訊息

HMAIL> directory/folder
DISK$USER:[USER]MAIL.MAI;1 中的卷宗清單
                    按 CTRL/C 取消清單
CFMS                                    CRDB
CVMS                                    MAIL
中文資料庫                              每月報告
私人郵遞                                系統資訊
會議報告                                會議程序
HMAIL 公用程式      HMAIL> directory/folder/collating=(stroke,radical)
DISK$USER:[USER]MAIL.MAI;1 中的卷宗清單
                    按 CTRL/C 取消清單
CFMS                                    CRDB
CVMS                                    MAIL
```

```
中文資料庫                                    私人郵遞
每月報告                                      系統資訊
會議報告                                      會議程序
HMAIL> directory/folder/collating=radical
DISK$USER:[USER]MAIL.MAI;1 中的卷宗清單
                    按 CTRL/C 取消清單
CFMS                                          CRDB
CVMS                                          MAIL
中文資料庫                                    會議報告
會議程序                                      每月報告
私人郵遞                                      系統資訊
HMAIL> directory/folder/collating=ascii
DISK$USER:[USER]MAIL.MAI;1 中的卷宗清單
                    按 CTRL/C 取消清單
CFMS                                          CRDB
CVMS                                          MAIL
中文資料庫                                    每月報告
私人郵遞                                      系統資訊
會議報告                                      會議程序
```

除了卷名顯示的順序，提供給 */START* 限定詞的內容，將與根據顯示對併順序選定的卷名做比較。因此若指定了 */COLLAING=PHONETIC_CODE*，則比較卷名的注音碼對併數值以決定那些該顯示。下面的範例說明如何執行。

```
HMAIL> directory/folder/collating=stroke/start= 私
DISK$USER:[USER]MAIL.MAI;1 中的卷宗清單
      按 CTRL/C 取消清單
私人郵遞                                      每月報告
系統資訊                                      會議報告
會議程序
HMAIL> directory/folder/collating=radical/start= 私
DISK$USER:[USER]MAIL.MAI;1 中的卷宗清單
                    按 CTRL/C 取消清單
私人郵遞                                      系統資訊
HMAIL> directory/folder/collating=ascii/start= 私
DISK$USER:[USER]MAIL.MAI;1 中的卷宗清單
                    按 CTRL/C 取消清單
```

| | |
|---|---|
| 私人郵遞 | 系統資訊 |
| 會議報告 | 會議程序 |

### 8.3.6 HMAIL 中的文字編輯程式

在 HMAIL 中，將引動 HTPU 爲既定的編輯程式。不過，您可以使用 SET EDITOR 命令來設定所選擇的編輯程式，以便置換既定值。例如，您可以使用以下的命令，將既定的編輯程式設定爲另一個編輯程式。

HHAIL> set editor [other editor][1]

## 8.4 命令摘要

下表爲與處理中文字有關的 HMAIL 命令摘要。

| 命令 / 限定詞 | 說明 |
|---|---|
| SET PERSONAL_NAME, /PERSONAL_NAME | 可使用中文個人名稱 |
| /SUBJECT | 可使用中文主題 |
| COPY,DIRECTORY,FILE, MOVE,READ,SELECT,SET /SHOW FOLDER,SET WASTE-BASKET | 可指定中文卷名 |
| SEARCH,/ SUBJECT_SUBSTRING | 中文字可放入搜尋字串以找出中文字串 |
| DIRECTORY/FOLDER/ COLLATING_SEQUENCE | 指定以中文對併順序顯示卷名，對併順序如 RADICAL，STROKE，PHONETIC_CODE 與 INTERNAL_CODE |
| DIRECTORY/FOLDER/ START | 依據現在選取的對併順序，指定顯示第一個卷名 |
| HELP | 若在 HANYUGEN 中終端機輸出值是 HANYU_MSG，則顯示中文說明 |

1. [other editor] 表示在 OpenVMS 下所執行的其他文字編輯程式 ，例如 TPU

# 第 9 章
# HTPU/HEVE 公用程式

本章描述 HTPU/HEVE 公用程式的特色。

## 9.1 簡介

HTPU/HEVE 是 DECTPU/EVE 的提升，以便支援中文文字編輯和 HEDT 功能。可支援字元儲存格終端機 (CCT) 和 DECwindows Motif/Hanyu 使用者介面。

根據 DECTPU/EVE，HTPU/HEVE 是由兩個配備，即 HTPU (Hnayu Text Processing Utility) 和 HEVE (Hanyu Extensible Versatile Editor) 所組成。HEVE 透過 HTPU 所提供的內建程序，來執行中文編輯功能。

## 9.2 使用 HTPU/HEVE

若要使用 CCT 介面引動包含 HEVE 的 HTPU 作爲既定的編輯程式，可使

用下列的命令：

$ EDIT/HTPU TEXT.TXT

同時，您可以定義如下的外部命令來縮寫引動：

$ HEVE :== $EDIT/HTPU

若要使用 DECwindows Motif/Hanyu 介面引動包含 HEVE 的 HTPU 作爲既定的編輯程式，可使用下列的命令：

$ SET DISPLAY/CREATE/NODE=< 您的工作站的節點名稱 >

$ HEVE/DISPLAY=MOTIF TEXT.TXT

TEXT.TXT 是您要編輯的檔案。

一旦啓動 HEVE，您可使用鍵盤鍵鍵入文字。您也可以使用 HEVE 命令和其預定的功能鍵，來執行文字編輯。表 9-1 列出預定的功能鍵。

**表 9-1 HEVE 預定的編輯鍵和其功能**

| 鍵 | 功能 | 鍵 | 功能 |
|---|---|---|---|
| <CTRL/A> | 變更模態 ( 插入 / 重打 ) | <Find> | 尋找特定字串 |
| <CTRL/B> | 召回索引前一個 HEVE 命令 | <Insert_Here> | 在目前的游標位置插入在選取範圍中的文字 |
| <CTRL/E> | 將游標移到列尾 | <Remove> | 移開選定範圍的文字 |
| <CTRL/H> | 將游標移到列首 | <Select> | 選取文字範圍 |
| <CTRL/J> | 插入頁分斷 | <Next_Screen> | 下一個螢幕 |
| <CTRL/U> | 消除列首 | <F10> | 退出 HEVE |

表 9-1 ( 續前頁 )HEVE 預定的編輯鍵和其功能

| 鍵 | 功能 | 鍵 | 功能 |
|---|---|---|---|
| <CTRL/W> | 復新螢幕 | <F11> | 變更方向 ( 正向或反向 ) |
| <CTRL/Z> | 退出 HEVE | <F12> | 逐列移動游標 |
| <DELETE> | 刪除先前的字元 | <F13> | 消除現用的字 |
| ↑ | 往上移 | <F14> | 變更模態 ( 插入或重打 ) |
| ↓ | 往下移 | | |
| → | 往右移 | | |
| ← | 往左移 | | |

若需更多有關 HEVE 和其命令的相關資訊，請參閱 *《HEVE 使用者手冊》*或鍵入 <Do> 鍵然後在 HEVE 中輸入 "HELP" 命令。

## 9.3 HTPU 特色

HTPU 提供一組內建程序和可呼叫的介面，以便發展中文編輯程式。您可以全用 HTPU 語言和其內建程序來編輯程式，就像 HTPU 既定的編輯程式 HEVE；或者您可於應用程式中經由可呼叫的介面，呼叫 HTPU。

DECTPU 的 HTPU 提升包含下列各項：

- 新的內建程序能提供字元分類
- 修訂過之內建程序能支援中文字處理
- 在 DECwindows Motif/Hanyu 視窗環境中的中文字輸入

關於新的和修訂過的內建程序之完整資料，請參閱 *《HTPU 和 HEVE 參考手冊》*。若需要有關 DECTPU 內建程序，請參閱 *《DEC Text Processing Utility Reference Manual》*有關的細節。*《Guide to the DEC Text Processing Utility》*描述 HTPU 語言。

有關 HTPU 可呼叫介面的特殊資訊，請參閱這本手冊的附錄 B.4。若需要關於 DECTPU 的可呼叫介面，請參閱 *《VMS Utility Routines Manual》*第十四章。

## 9.4 HEVE 特色

由於 HEVE 是利用 HTPU 語言和內建程序編輯而成的，它可以編輯中文字和 ASCII 字 元。它支援所有 EVE[1] 功能，並且具有下列擴充的功能：

- 符號格式化及畫線和畫框的功能
- 全形字和半形字的轉換
- HEDT 功能模擬
- DECwindows Motif/Hanyu 視窗環境的本地語言之懸垂和即現功能表

有關 HEVE 的特定命令，可參閱 *《HTPU 和 HEVE 參考手冊》*和 *《HEVE 使用者手冊》*。關於 EVE 命令，請參閱 *《Extensible Versatile Editor Reference Manual》*。

1. 除了不支援 MCS 字元編輯之外

# 第 10 章
# HDUMP 公用程式

HDUMP 爲 DUMP 的中文版本。它加強 DUMP 公用程式的功能，包括可顯示和列印一個帶有中文之檔案內容。

二者主要的差別在於 HDUMP 能適時顯示二位元組和四位元組中文字，DUMP 則否。例如：列尾只剩一位元組空間時，對於即將來臨的二位元組中文字，DUMP 不會保留該字之第一位元組至次列與第二位元組一併顯示；因此，導至錯誤的結果。而在 HDUMP 內，不論上述情形是否重複發生，中文字的第一位元組一定會和第二位元組一併顯示於次傾印列。並且顯示 和列印顯示欄左界上的一 ASCII 位置。

當這些字元跨越傾印欄轉列界時，相同的情況也適用於四位元組的中文字。

所有不在中文範圍內之二位元組碼，都以句點 (.) 顯示或列印。

HDUMP 的命令格式爲：

$ HDUMP file-name

所有的命令限定詞皆與 DUMP 相同。

以下爲 HDUMP 的範例。

```
$ TYPE HANYU.DAT
美國ＤＥＣ計算機有限公司電話：3-7315211
美國 DEC 計算機有限公司電話：3-7315211
$
$ HDUMP/RECORD HANYU.DAT
Hdump of file WRKD$:[HANYU.SRC]HANYU.DAT;1 on 02-DEC-1992 12:12:30.24
File ID (17993,21,0) End of file block 1 / Allocated 3
Record number 1 (00000001), 39 (0027) bytes
 A2F1ABEA D3D3C3A4 C5A4C4A4 CFD9A1D3      美國ＤＥＣ計算機 000000
 2D33A8A1 E3E6D9E7 B3C6FCC4 EED3B4C8      有限公司電話：3- 000010
                     313132 35313337      7315211..... 000020
Record number 2 (00000002), 36 (0024) bytes
 D3B4C8A2 F1ABEAD3 D3434544 CFD9A1D3      美國 DEC 計算機有 000000
 3133372D 33A8A1E3 E6D9E7B3 C6FCC4EE     限公司電話：3-731 000010
                           31313235       5211..... 000020
```

# 第 11 章
# 片語輸入公用程式

片語輸入公用程式是 OpenVMS/Hanyu 中的一公用程式，可支援 VT382-D 中文終端機的片語輸入方式，同時也讓使用者修改和運用片語。它將允許使用者建立個人片語資料庫的片語，並可將其歸類。此公用程式可幫助使用者將片語類別載入終端機。載入類別之後，使用者可使用終端機的片語輸入法使用該片語。

所有片語和類別的資訊儲存在兩個檔案中，即 HSY$PHRASE.DAT 和 HSY$PHRASE_CLASS.DAT。系統可擁有整個系統的資料庫；而此資料庫可 藉由系統管理者或有權限的使用者加以修訂；非權限的使用者，允許讀取整個系統的資料庫，但不能作修訂。每個使用者可有其個人的資料

庫。

使用以下的 DCL 命令，來引動片語輸入公用程式：

$ RUN HSY$SYSTEM:PHRASE

片語輸入公用程式是一功能表驅動的全螢幕公用程式。引動之後，將有訊息透過功能表來指導使用者。

片語輸入公用程式，也可藉由定義 DCL 的外部命令來引動：

$ PHRASE :== $HSY$SYSTEM:PHRASE

因此，使用者可鍵入下列的命令，將系統或使用者資料庫的某類片語個別載入終端機中：

$ PHRASE/SYSTEM CLASS_NAME

$ PHRASE/USER CLASS_NAME

若需要關於片語輸入公用程式的細節，請參閱 *《片語輸入公用程式使用者手冊》*。

# 第 12 章
# 中文日期與時間支援

本章說明以中文表示日期與時間的方法。

## 12.1 簡介

輸入與輸出格式上可使用中文的日期與時間。在 OpenVMS/Hanyu 內有中文 語言表以及預定的日期和時間輸出格式。於系統與使用者處理共同設定時，中文日期與時間輸出格式可用於 DCL 命令 DIR，以及 DCL 公用程式 MAIL 與 HAMIL。此外，使用者可利用中文語言表，界定自己的使用者自定日期 與時間輸入 / 輸出格式。這個日期與時間輸入 / 輸出格式可供 OpenVMS RTL 中的日期 / 時間處理常式使用。有關日期 / 時間處理常式的詳細敘述和用法，請參照 *《VMS RTL Library (LIB$) Manual》*。

## 12.2 啓動

使用中文日期與時間之前，系統管理員 ( 或任何擁有 CMEXEC、SYSNAM 與 SYSPRV 特權的使用者 ) 必須執行命令程序 SYS$MANGER:LIB$DT_STARTUP.COM，界定數個日期與時間輸出格式 ( 包含預先界定的中文輸出格式 )。此外也要 界定中文日期與時間元件。命令如下：

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

理論上，關於 OpenVMS/Hanyu 應從 SYSTARTUP_VMS.COM 執行上述的命令。

## 12.3 預定中文輸出格式

下表表示預先界定中文日期與時間輸出格式。有關助憶名稱使用的敘述，請參照 *《VMS RTL Library (LIB$) Manual》*。

**表 12-1 預定輸出日期格式**

| 日期格式邏輯 | 格式 | 範例 |
|---|---|---|
| `LIB$DATE_FORMAT_044` | !Y4 年 !MNB 月 !DB 日 !WAU | 1992 年 11 月 10 日 ( 四 ) |
| `LIB$DATE_FORMAT_045` | !Y4 年 !MNB 月 !DB 日 !WAU | 1992 年 11 月 10 日 星期四 |

表 12-2 預定輸出時間格式

| 時間格式邏輯 | 格式 | 範例 |
|---|---|---|
| **LIB$TIME_FORMAT_022** | !MIU!HB2 時 !MB 分 !SB 秒 | 下午 2 時 16 分 26 秒 |

## 12.4 預定中文語言表

中文語言表界定數個邏輯。這些邏輯可使用於使用者自定輸出格式，如 12.6 節所述。表列於後以供參考：

表 12-3 中文語言表

| 邏輯 | 輸出助記符 | 對等名稱 |
|---|---|---|
| LIB$WEEKDAYS_[UlLlC] | W[UlLlC] | " 星期一 "," 星期二 "," 星期三 ", " 星期四 ", " 星期五 "," 星期六 ", " 星期日 " |
| "LIB$WEEKDAY_ABBREVIATIONS_[UlLlC] | WA[UlLlC] | "( 一 )","( 二 )","( 三 )","( 四 )","( 五 )","( 六 )","( 日 )" |
| LIB$MONTHS_[UlLlC]<br>LIB$MONTHS_ABBREVIATIONS_[UlLlC] | MAA[UlLlC] | " 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_[UlLlC] | MI[UlLlC] | " 上午 "," 下午 " |

## 12.5 執行時選取中文輸出格式

如果爲日期、時間，或兩者選取特殊的中文格式，使用者必須使用在 12.3 節界定的邏輯以界定 LIB$DT_FORMAT 邏輯名稱。例如：

$ DEFINE SYS$LANGUAGE HANYU

$ DEFINE LIB$DT_FORMAT LIB$DATE_FORMAT_044, LIB$TIME_FORMAT_022

上列兩個邏輯名稱的次序將決定其輸出次序。上面的定義使日期以指定的格式輸出，後接空格以及指定格式的時間，如下所示。

1992 年 11 月 10 日 ( 二 ) 下午 2 時 16 分 26 秒

## 12.6 使用者自定的中文輸出格式

使用者可以界定自已的執行模式邏輯，以界定自己的中文日期及時間輸出格式。例如：

$ DEFINE/EXEC/TABLE=LNM$DT_FORMAT_TABLE LIB$DATE_FORMAT_601-
_$ " 系統時間 : !Y4 年 !MNB 月 !DB 日 "

$ DEFINE/EXEC/TABLE=LNM$DT_FORMAT_TABLE LIB$TIME_FORMAT_601-
_$ "!MIU!HB2 時 !MB 分 !SB 秒 !WU "

界定所需的格式後，使用者可於下列命令中使用這些格式：

```
$ DEFINE SYS$LANGUAGE HANYU
$ DEFINE LIB$DT_FORMAT LIB$DATE_FORMAT_601, LIB$TIME_FORMAT_601
```

輸出的中文日期與時間如下：

系統時間 : 1992 年 11 月 10 日 下午 2 時 16 分 26 秒 星期二

## 12.7 執行時選取中文輸入格式

如要爲應用程式的日期與時間輸入選取特殊中文格式，使用者必須界定 LIB$DT_INPUT_FORMAT 邏輯名稱。LIB$DT_INPUT_FORMAT 的對等名稱可包括取自 12.4 節所述之預先界定的中文語言表的助憶符號。

## 12.8 範例

### 12.8.1 HMAIL

```
$ DEFINE SYS$LANGUAGE HANYU
$ DEFINE LIB$DT_FORMAT LIB$DATE_FORMAT_045,LIB$TIME_FORMAT_022
$ HMAIL
                                                                    MAIL
# 來自                    日期                          主旨
1 NODEA::SYSTEM           1992 年 11 月 16 日 星期三    工作報告
HMAIL> 1
#1                        1992 年 11 月 16 日 星期三 上午 11 時 20 分 53 秒   MAIL
來自 : NODEA::SYSTEM " 工業部 : 李國基 "
致 : CHAN
副本 :
主題 : 工作報告
工業部長 :
以下是我的工作報告 :
                                  :
                                  :
HMAIL>exit
```

### 12.8.2 DIR 命令

```
$ DEFINE SYS$LANGUAGE HANYU
$ DEFINE LIB$DT_FORMAT LIB$DATE_FORMAT_045,LIB$TIME_FORMAT_022
$
$ DIR/FULL TEST.DAT
```

```
USER$:[LEE] 檔目
TEST.DAT;1                          檔案：        (2229,2,0)
尺寸：                1/3           擁有者：      [SYSTEM]
建立日期：  1992 年 11 月 17 日 星期日  下午 6 時 35 分 25 秒
修改日期：  1992 年 11 月 17 日 星期日  下午 6 時 35 分 28 秒 (1)
終止日期：  <未指定>
備製日期：  <未曾備製>
檔案結構：            順序
檔案屬性：            配置：3, 擴充：0 ,  總體緩衝區計數：          0 , 無版本限制
紀錄格式：            可變長度, 最大 74 位元組
紀錄屬性：            Carriage return carriage control
RMS 屬性：            無
日誌功能已啓用：      無
檔案保用：            System:RWED, Owner:RWED, Group:RE, World:
存取控制表：          無
合計 1 個檔案, 1/3 個磁區.
$ DIRECTORY/DATE TEST*.DAT
USER$:[LEE] 檔目
TEST.DAT;1            1992 年 11 月 17 日 星期日 下午 6 時 35 分 25 秒
TEST1.DAT;1           1992 年 11 月 17 日 星期日 下午 6 時 36 分 19 秒
TEST2.DAT;1           1992 年 11 月 17 日 星期日 下午 6 時 36 分 22 秒
TEST3.DAT;1           1992 年 11 月 17 日 星期日 下午 6 時 36 分 26 秒
TEST4.DAT;1           1992 年 11 月 17 日 星期日 下午 6 時 36 分 31 秒
合計 5 個檔案.
```

### 12.8.3 RTL 日期 / 時間處理常式

```
$ TYPE DATETIME.C
/* DECLARATIONS */
#include 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_045,LIB$TIME_FORMAT_022
$ DEFINE SYS$LANGUAGE HANYU
$
$ DEFINE LIB$DT_INPUT_FORMAT "!Y4 年 !MNB 月 !DB 日 !MIU!HB2 時 !MB 分 !SB.!C2 秒 "
$ RUN DATETIME
Input date time:1992 年 10 月 11 日 下午 2 時 16 分 26.02 秒
Output date time: 1992 年 10 月 11 日 星期二 下午 2 時 16 分 26 秒
$
$ DEFINE LIB$DT_INPUT_FORMAT "!Y4 年 !MAU 月 !DB 日 !HB4 時 !MB 分 !SB.!C2 秒 "
$ RUN DATETIME
Input date time:1992 年十月 11 日 17 時 12 分 22.02 秒
Output date time: 1992 年 10 月 11 日 星期二 下午 5 時 12 分 22 秒
$
```

# 第 13 章
# 中文終端機歸化設施

本章描述如何使用終端機歸化設施公用程式 (Terminal Fallback FacilityUtility) 來設定系統，使其能使用 TFF 二位元組字元轉換表。

## 13.1 終端機歸化設施

OpenVMS/Hanyu 的終端機歸化設施 ( 簡稱 TFF)，已經擴展爲可以支援 MITAC TELEX CODE 和 BIG5 的終端機和印表機的二位元組字元轉換。本程式允許 MITAC TELEX CODE 和 BIG5 的終端機和印表機輸入和輸出 DEC SICGCC 字元集的中文字，其方法是先行轉換出入終端機和印表機的字元。這種字元轉換是透過字元表轉換到應用軟體執行。因此，MITAC TELEX CODE 和 BIG5 的終端機和印表機的使用者可以使用 DEC SICGCC 字元集發展出來的 OpenVMS/Hanyu 應用軟體。

有關 TFF 的細節，請參閱 *《VMS Terminal Fallback Utility Manual》*。

### 13.1.1 設置 TFF 環境

您在使用 OpenVMS/Hanyu 的 TFF 之前，必須執行下列步驟。關於如何設定和維護 TFF 環境，請參閱 *《VMS Terminal Fallback Utility Manual》*。

#### 13.1.1.1 在系統上安裝 TFF

在系統上安裝 OpenVMS/Hanyu 之後，您必須啓用 TFF。請在系統磁碟 SYS$MANAGER 的檔目內引動 VAX 的命令程序 TFF$STARTUP.COM，或是 AXP 的命令程序 TFF$SYSTARTUP.COM。當重新啓動系統之後，若要啓動 TFF，請修改 OpenVMS/Hanyu 特定位置的啓動命令程序 SYS$STARTUP:SYSTARTUP_VMS.COM，使其包含以下的命令：

```
$ @SYS$MANAGER:TFF$SYSTARTUP.COM
```

#### 13.1.1.2 TFF 表

爲了使 MITAC TELEX CODE 終端機能支援 DEC SICGCC 字元集，OpenVMS/Hanyu VAX 和 OpenVMS/Hanyu AXP 提供 TFF 歸化表 HANYU_TELEX。

在 OpenVMS/Hanyu 中提供 HANYU_BIG5 和 BIG5_HANYU 的 TFF 歸化表，分別爲 DOS 4.1 版支援 PATHWORKS/Hanyu 及爲 BIG5 終端機和印表機支援 DEC SICGCC 字元集。

在系統上啓用 TFF 之後，所需的歸化表必須載入非分頁動態記憶體的系統實際記憶體中，使用者才能加以存取。要完成這項工作，您必須把包含該表的程式庫導入 " 終端機歸化設施 (TFU)"；並且把該表載入系統的實際記憶體中。載入 HANYU_TELEX 的命令如下：

$ RUN SYS$SYSTEM:TFU

VAX/VMS Terminal Fallback Facility (TFF)

TFU> SET LIBRARY SYS$SYSTEM:TFF$MASTER.DAT

TFU> LOAD TABLE HANYU_TELEX

將 HANYU_BIG5 和 BIG5_HANYU 載入 / 顯示到非分頁動態記憶體儲存區的命令如下：

$ RUN SYS$SYSTEM:TFU

```
VAX/VMS Terminal Fallback Facility (TFF)
TFU> SET LIBRARY SYS$SYSTEM:TFF$MASTER.DAT
TFU> LOAD TABLE HANYU_BIG5
TFU> LOAD TABLE BIG5_HANYU
TFU> SHOW TABLES
The following TFF tables are currently loaded
Name                    Type    Base    Crefc    Trefc
----                    ----    ----    -----    ----
ASCII                   Fbk     MCS     *        0 0
LATIN_1                 Cmp     MCS     *        0 0
HANYU_BIG5              Fbk     CNS              0 0
BIG5_HANYU              Fbk     BIG5[1]          0 0
%TFF-W-NOMORETAB, No more tables in wildcard scan
TFU>
```

### 13.1.1.3 設定系統既定表

首次啓用 TFF 時，ASCII 被定義爲既定的歸化表。這個表被永久地載入非分頁動態記憶體中。您將 HANYU_TELEX 表載入非分頁動態記憶體之後，可以用下面的 TFU 命令重新建立既定表。

```
TFU> SET DEFAULT_TABLE HANYU_TELEX
```

在 OpenVMS/Hanyu 設定 HANYU_BIG5 作爲系統既定歸化表的程序與上述的範例相似。

### 13.1.1.4 啓用 TFF 表

您在爲終端機啓用 HANYU_TELEX 或 HANYU_BIG5 轉換表之前，必須啓用八位元字元的輸入功能。要不然 TFF 會清除輸入字元的第八個位元。

啓用八位元字元的命令是：

```
TFU> SET TERMINAL/FALLBACK=ACCEPT
```

## 13.1.2 實際記憶體需求

因爲 TFF 利用昂貴的系統資源 ( 非分頁動態記憶體 )，您必須小心地從系統的層次準備環境。若要載入 HANYU_TELEX 歸化表，則至少需要 120 KBytes 的非分頁動態記憶體。每次在 OpenVMS/Hanyu 系統中載入 HANYU-BIG5 和 BIG5-HANYU 歸化表，至少需要 140 KBytes 的非分頁動態記憶體。

1. BIG5_HANYU 歸化表的基底是 BIG5 字元集。

# 第 14 章
# 偵錯程式公用程式

OpenVMS 偵錯程式是一個符號偵錯程式。它是除錯使用者模式代碼的偏好偵錯程式。它支援字元型使用者界面和圖形 (DECwindows Motif) 使用者界面的所有 OpenVMS 程式語言的符號除錯。這兩種使用者界面現在都支援 DEC 中文字元集的輸入和顯示。在程式變數、源碼注解或 OpenVMS 偵錯程式錯誤訊息中的中文字元都能夠正確顯示。

**附註**

OpenVMS 偵錯程式使用 XPG4 國際化模型實現對 DEC 中文字元集的支援。它要求一個 DEC 中文字元集的 XPG4 現場資料庫。您可以在隨同 OpenVMS 一起發行的 OpenVMS 118n 儲存集中找到這個現場資料庫。

## 14.1 使用者環境設定

本段描述如何設定 OpenVMS 偵錯程式的字元型使用者界面和 DECwindows Motif 使用者界面來使用中文字元。

### 14.1.1 字元型使用者界面

OpenVMS 偵錯程式的字元型使用者界面一起使用中文 SMG 和 DEC C 的 XPG4 本地化公用程式來支援 DEC 中文字元集。要支援它，必須定義以下邏輯名 :

- DBG$SMGSHR
  這個邏輯名定義 OpenVMS 偵錯程式用於支援畫面模態的 SMG 共享映像的名稱。要允許支援 DEC 中文字元集，定義這個邏輯名如下 :
  $ DEFINE/JOB DBG$SMGSHR HSMGSHR
- SMG$DEFAULT_CHARACTER_SET
  這個邏輯名定義中文 SMG 支援的既定字元集。要允許支援 DEC 中文字元集，定義這個邏輯名如下 :
  $ DEFINE/JOB SMG$DEFAULT_CHARACTER_SET HANYU
- LANG
  這個邏輯名爲 DEC C 的 XPG4 本地化公用程式和運轉時用程式庫的所有現場定義既定的現場設定。要允許在 OpenVMS 偵錯程式中支援 DEC 中文字元集，定義這個邏輯名如下 :
  $ DEFINE/JOB LANG ZH_TW_DECHANYU

### 14.1.2 DECwindows Motif 使用者界面

OpenVMS 偵錯程式的 DECwindows Motif 使用者界面可以顯示 DEC 中文字元集中的字元。在檔目 DECW$SYSTEM_DEFAULT 或 DECW$USER_DEFAULTS 中的偵錯程式資源檔案 VMSDEBUG.DAT 可以爲每個偵錯程式視窗明確指定字體。

在 VMSDEBUG.DAT 中 , DebugDefault.font 資源爲所有偵錯程式視窗指定一種既定字體。每個視窗也可以使用與既定字元的不同字體。要知詳情，請參閱 VMSDEBUG.DAT 檔案。

要允許在 OpenVMS 偵錯程式視窗中支援 DEC 中文字元集，該視窗必須使用 DEC 中文字元集字體。可通過以下方法之一指定該字體：

1. 通過偵錯程式視窗的資源明確指定；
2. 當沒有按上述方法明確指定字體時，通過偵錯程式的既定字體資源 DebugDefault.font 來指定；
3. 當既沒有用偵錯程式既定字體資源，也沒有用偵錯程式視窗的字體資源來指定字體時，通過系統既定字體資源來指定。

## 14.2 偵錯程式命令

本段描述在偵錯程式命令中使用中文字元。

### 14.2.1 EXAMINE 命令

EXAMINE 命令支援一個新限定詞 /WCHAR_T:n。這個命令把每個被檢查實體解釋和顯示爲 n 個長字元 (n 個字元 ) 的多位元組檔案代碼序列。n 的既定值是 1。

當轉換被檢查的字元串時，OpenVMS 偵錯程式使用它運行處理過程的現場資料庫。既定是 C 現場。

### 14.2.2 DEPOSIT 命令

DEPOSIT 命令支援一個新限定詞 /WCHAR_T:n。這個命令把被轉換的多位元組檔案代碼序列中的 n 個長字元存放在指定位置。n 的既定值是 1。

當轉換這個 DEPOSIT 命令指定的字元串時，OpenVMS 偵錯程式使用它運行處理過程的現場資料庫。既定是 C 現場。

# 附錄 A

# 程式語言

OpenVMS/Hanyu 容許使用者在註解、字元及文字字串使用中文，並於任何程式語言中處理包含中文的輸入與輸出資料，但有下列限制：

- 使用 DEC SICGCC 第二字面時有所限制。
- 在 COBOL 上，若中文字元的第二個位元組在字串文字宣告中包含 ASCII 字元 ""，亦十六進位的 22；COBOL 編譯程式將視其爲字串引號，結果字串宣告產生編譯錯誤。解決的方式可在中文字元後輸入第二個 ""，或者在 COBOL 程式中使用 ' 爲字串引號。
- 至於註解，只有 C、PASCAL 與 PL/I 程式語言於開放式註解分界號，與關閉式註解分界號間包含中文字元時，才會產生問題。

一般程式語言使用中文的範例程式列示於後。這些範例也放在 OpenVMS/Hanyu 系統，HSY$EXAMPLE 檔目之下。

範例 ---- 中文應用於 MACRO-32

## A.1 範例 ---- 中文應用於 MACRO-32

```
$ TYPE SAMPLE.MAR
                VAX_MACRO = 1       ; If you want to compile this example
                                    ; with MACRO-32, then the VAX_MARCO 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 與中文共用 /
DATA2:          STRING <VAX MACRO 中 MACRO 使用中文 >
inp:            string < 數據輸入 :>
oup:            .long b3-b1 中文
                .address b1
buff:           .long b3-b2
                .address b2
b1:             .ascii / 數據輸出 : /
b2:             .blkb 40
b3:
```

## A.1 範例 ---- 中文應用於 MACRO-32

```
.PSECT CODE
.IF DF VAX_MACRO
        .ENTRY SAMPLE, ^m<> ; 在注釋中使用中文 .
.IFF
SAMPLE::
        .CALL_ENTRY PRESERVE=<R2,R3,R4,R5,R6,R7,R8,R9,R10,-
                                R11,R12,R13,R14,R15> ; 在注釋中使用中文 .
.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$GET_INPUT
        BLBC R0,EOF
PUSHAQ OUP
        CALLS #1,G^LIB$PUT_OUTPUT
        BRB LOOP
EOF:    $EXIT_S
        .END SAMPLE
$
$ MACRO SAMPLE
$ LINK SAMPLE
$ RUN SAMPLE
$ RUN SAMPLE_MAR.EXE
VAX MACRO 與中文共用
VAX MACRO 中 MACRO 使用中文
數據輸入 : 金木水火土
數據輸出 : 金木水火土
數據輸入 :Exit
$
```

## A.2 範例 ---- 中文應用於 DEC FORTRAN

FORTRAN

```
$ TYPE SAMPLE.FOR
                        program sample
                        character*40 buff
!                       在數據中使用中文
                        type *,'VAX FORTRAN 與中文共用 '
                        do while (.true.)
                                write (*,'(a)') '$ 數據輸入 : '
                                read (*,'(a)',end=100) buff
                                write (*,'(a)') '$ 數據輸出 : '//buff
                        end do
100                     continue
                        end
$
$ FORTRAN SAMPLE
$ LINK SAMPLE
$ RUN SAMPLE
VAX FORTRAN 與中文共用
數據輸入 : 金木水火土
數據輸出 : 金木水火土
數據輸入 : Exit
$
```

## A.3 範例 ---- 中文應用於 BASIC

```
$ TYPE SAMPLE.BAS
100 rem 中文注釋
110 print "VAX BASIC 與中文共用 "
120 on error go to 180
130 while 1
140                     print " 數據輸入 :";
150                     input buff$
160                     print " 數據輸出 : ";buff$
170 next
```

```
180 resume 190
190 end
$
$ BASIC SAMPLE
$ LINK SAMPLE
$ RUN SAMPLE
VAX BASIC 與中文共用
數據輸入 : 金木水火土
數據輸出 : 金木水火土
數據輸入 : Exit
$
```

## A.4 範例 ---- 中文應用於 PASCAL

```
$ TYPE SAMPLE.PAS
program sample(input,output);
var buff:packed array [1..40] of char;
begin (* 在注釋中使用中文 *)
                writeln ('VAX PASCAL 與中文共用 ');
                write (' 數據輸入 :');
                while not eof do begin
                                readln (buff);
                                writeln(' 數據輸出 :',buff);
                                write (' 數據輸入 :')
                end;
end.
$
$ PASCAL SAMPLE
$ LINK SAMPLE
$ RUN SAMPLE
VAX PASCAL 與中文共用
數據輸入 : 金木水火土
數據輸出 : 金木水火土
數據輸入 : Exit
$
```

# A.5 範例 ---- 中文應用於 COBOL

```
$ TYPE SAMPLE.COB
* 在注釋中使用中文
identification division.
program-id. sample.
author. 劉劍明 .
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 與中文共用 ".
loop.
                display " 數據輸入 :" with no advancing.
                accept buff at end go to eof.
                display " 數據輸出 :",buff.
                go to loop.
eof.
                stop run.
$
$ COBOL SAMPLE
$ LINK SAMPLE
$ RUN SAMPLE
VAX COBOL 與中文共用
數據輸入 : 金木水火土
數據輸出 : 金木水火土
數據輸入 : Exit
$
```

## A.6 範例 ---- 中文應用於 PL/I

```
$ TYPE SAMPLE.PLI
sample:proc options (main); /* 在注釋中使用中文 */
dcl buff char(40);
                    put list ('VAX PL/I 與中文共用 ');
                    on endfile (sysin) stop;
                    do while ('1'b);
                    put skip edit (' 數據輸入 : ') (a);
                                get edit (buff) (a(40));
                                put edit (' 數據輸出 : ',buff) (a,a);
                                end;
                    end;
$
$ PLI SAMPLE
$ LINK SAMPLE
$ RUN SAMPLE
VAX PL/I 與中文共用
數據輸入 : 金木水火土
數據輸出 : 金木水火土
數據輸入 : Exit
$
```

## A.7 範例 ---- 中文應用於 DEC C

```
$ TYPE SAMPLE.C
#include <stdio.h>
main () /* 在注釋中使用中文 */
{
            char buff[40];
            printf("\nVAX C 與中文共用 ");
            while (printf("\n 數據輸入 :"), gets(buff) != NULL)
                        printf(" 數據輸出 : %s",buff);
}
$
$ CC SAMPLE
```

```
$ LINK SAMPLE
$ RUN SAMPLE
VAX C 與中文共用
數據輸入 : 金木水火土
數據輸出 : 金木水火土
數據輸入 : Exit
$
```

**注意事項**

用 #define 指令界定的內容不可包含中文，以及用作參考的引數 內也不可包含中文。

# 附錄 B

# 中文程式特色

## B.1 HSYSHR

HSYSHR 爲在 OpenVMS/Hanyu 系統的中文運轉作業庫。提供基本中文處理功能常式的一般用途程式庫。

系統中有此程式庫的線上說明。要獲得有關常式功能，呼叫格式，傳達引數的方法，與送回值的資訊，請輸入：

$HELP @HSYHLP HSYSHR

程式庫包含於可共用影像庫 HSYIMGLIB.OLB 內。可以在任何 OpenVMS/Hanyu 支援的程式語言呼叫程式庫功能。如要將程式與執行時程式庫連結，請輸 入：

$LINK PROGRAM，SYS$LIBRARY:HSYIMGLIB/LIB

詳細說明，請參閱 *《OpenVMS/Hanyu RTL Chinese Processing (HSY$) Manual》*。

### B.1.1 應用可呼叫 HSYSHR 常式的範例

```
$ 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
        ormats 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 trailing 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 checks for a valid command line, open all
        necessary files and calls 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);
}
```

## B.2 HSMGSHR

HSMGSHR，中文螢幕管理作業庫，包含在 OpenVMS/Hanyu 系統中。它是一 組協助設計、組合和在影像螢幕上追蹤映像的常式。

螢幕管理設施提供兩項重要的服務：終端機的獨立性和組合的便利。

可由 OpenVMS/Hanyu 支援的任何程式語言呼叫作業庫功能。若要將作業庫與程式連結，請輸入：

$ LINK PROGRAM, SYS$INPUT/OPTION

SYS$LIBRARY:HSMGSHR.EXE/SHARE

若需要詳細的說明，請參閱 *《VMS RTL Chinese Screen Management (SMG$) Manual》*。

## B.3 SORT 和 MERGE 可呼叫常式

這些常式能讓使用者將排序 (sort) 或合併 (merge) 的運作，整合入應用 程式；使之能處理包含中文資料的記錄，對它們作排序或合併，然後再處理之。

下列爲使用排序合併可用之可呼叫常式。若需有關細節，請參閱 *《VMS Utility Routine Manual》*。

SOR$BEGIN_MERGE 設定關鍵引數並且執行合併。這是 MERGE 所特有的唯一常式。

SOR$BEGIN_SORT 藉著傳遞關鍵資訊和排序選項來起始設定排序運作。這個是 SORT 所特有的唯一常式。

| | |
|---|---|
| SOR$END_SORT | 執行清除功能，如關閉檔案和鬆開記憶體。 |
| SOR$PASS_FILES | 將輸入和輸出檔案的名稱傳遞到 SORT 或 MERGE；每個輸入檔案必須重複。 |
| SOR$RELEASE_REC | 將一份輸入記錄傳遞到 SORT 或 MERGE；每一份記錄必須呼叫一次。 |

| | |
|---|---|
| SOR$RETURN_REC | 將一份排序或合併的記錄折回到一個程式；每一份記錄必須呼叫一次。 |
| SOR$SORT_MERGE | 將記錄排序。 |
| SOR$SPEC_FILE | 傳遞規格檔案或規格文字。必須在所有其他 SOR 常式呼叫之前，呼叫此常式。 |
| SOR$STAT | 折回關於排序或合併運作的統計。 |

這些常式可以從支援 VMS 程序呼叫和 " 條件處置標準 " 的任何一個語言來呼叫。若要使用 OpenVMS/Hanyu 的 SORT 和 MERGE 可呼叫常式，則應輸入 以下的命令，來連結應用程式與作業庫：

$ LINK PROGRAM

呼叫之後，此常式將執行其功能，並且將控制折回呼叫程式。

所有屬於字串型的參數，以 descriptor 方式傳輸。所有屬於純量的參數以 references 方式傳輸。這些參數有次序性，您可以將選擇性的參數以 value 方式傳輸零值。主程式也可以省略最後面選擇性的參數，直接以較短的參數列帶入，有關細節，請參閱 *《OpenVMS Utility Routine Reference Manual》*。

您可以利用下列假設的資料類型，指定中文的對併順序；對使用者而言，可將它們視爲一般性的符號。

| | |
|---|---|
| SOR$K_SEQ_PHONETIC_CODE | 中文十六位元的文字注音碼對併順序 |
| SOR$K_SEQ_INTERNAL_CODE | 中文十六位元的文字內碼對併順序 |
| SOR$K_SEQ_RADICAL | 中文十六位元的文字部首對併順序 |
| SOR$K_SEQ_STROKE | 中文十六位元的文字筆畫對併順序 |
| SOR$K_SEQ_CHAR_PHONETIC_CODE | 中文十六位元的文字注音對併順序，並且以中文字爲單元 |
| SOR$K_SEQ_CHAR_lNTERNAL_CODE | 中文十六位元的文字內碼對併順序，並且以中文字爲單元 |
| SOR$K_SEQ_CHAR_RADICAL | 中文十六位元的文字部首對併順序，並且以中文字爲單元 |
| SOR$K_SEQ_CHAR_STROKE | 中文十六位元的文字筆畫對併順序，並且使用中文字爲單元 |

此虛擬的資料型式應分類爲資料型式的關鍵字。

### B.3.1 狀態碼

上述的常式於處理中文資料所回復的狀態碼 (status code) 與處理 ASCII 資料所回復的狀態碼，是相同的。若需詳細說明，請參閱 *《VMS Utility Routines Manual》*。

## B.3.2 可呼叫常式的介面

您可將資料以一完整的檔案或單一的紀錄，送給上述的常式作處理。當您的程式送出一個或若干檔案以產生一個排序或合併後的輸出檔案時，您就已經使用檔案的介面了。當您的程式在同時間送出多筆記錄，並且於同一時間接收這些經過排序的記錄時，您就已經使用記錄的介面了。

您可將檔案介面與記錄介面組合應用，亦即將資料以檔案方式傳入，再陸續接收這些經過排序後的記錄；或將資料以記錄方式一一傳入，再將這些經過排序後的記錄，寫到一檔案中。此二種介面的組合，提供了更多的彈性。

有關細節，請參閱 *《OpenVMS Utility Routine Reference Manual》*。

## B.3.3 使用可呼叫 SORT/MERGE 常式的範例

### B.3.3.1 使用注音碼對併順序呼叫 SORT 常式的程式

```
$ TYPE HSY$EXAMPLE:SAMPLE_SORT1.FOR
C
C       The following FORTRAN program sorts the records in the file
C       POET.DAT and creates an output file named EXAM1OUT.DAT
C       All the records in the input file are sorted in Phonetic code
C       collating sequence based on the first 12 bytes, that is，
C       the first 6 Chinese characters in each record.
C
C
        IMPLICIT        INTEGER*4(A-Z)
        INTEGER*2       KEYBUFF(5)
        CHARACTER*9     IN_FILE
        CHARACTER*12    OUT_FILE
        EXTERNAL SOR$K_SEQ_PHONETIC_CODE
        DATA IN_FILE, OUT_FILE /'POET.DAT','EXAM1OUT.DAT'/
        KEYBUFF(1)                                      !Number of KEY
        KEYBUFF(2)=%LOC(SOR$K.SEQ_PHONETIC_CODE)        !Phonetic code
                                                        !dictionary sequence
        KEYBUFF(3)=0                                    !Ascending sort
        KEYBUFF(4)=0                                    !Key offset
        KEYBUFF(5)=2                                    !Size in bytes
        STATUS=SOR$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,SYS$LIBRARY:HSYIMGLIB.OLB/LIB
$ RUN SAMPLE_SORT1
$
```

### B.3.3.2 使用多種對併順序呼叫 SORT 常式的程式

```
$ TYPE HSY$EXAMPLE:SAMPLE_SORT2.FOR
C This is a FORTRAN language example calling srot routines using
C multiple collating sequences.
C
C This program sorts data with file interface on input and record
C interface on output.Two collating sequences are specified here,
C the first being the number of storke count with ascending sort
C order and the second being the radical sequence with descending
C sort order.
C
C
C
C Define external functions and data and initialize
        IMPLICIT        INTEGER*4       (A-Z)
        CHARACTER*80    RECBUF
        CHARACTER*9     INPUTNAME               !Input file name
        INTEGER*2       KEYBUF(9)               !key definition buffer
        EXTERNAL        SS$_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'/
      KEYBUFF(1)=2
      KEYBUFF(2)=%LOC(SOR$K_SEQ_STROKE)          !Primary key STROKE
      KEYBUFF(3)=0                               !Ascending
      KEYBUFF(4)=0                               !Offset 0
      KEYBUFF(5)=2                               !Size 2
      KEYBUFF(6)=%LOC(SOR$K_SEQ_RADICAL)         !Secondary key RADICAL
      KEYBUFF(7)=1                               !Descending
      KEYBUFF(8)=0                               !Offset 0
      KEYBUFF(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$_ENDOFFILE) GOTO 7
      GOTO 10
C     Clean up the work areas and files.
7     ISTAUTS=SOR$END_SORT( )
      IF (.NOT.ISTATUS)GOTO 10
      STOP 'SORT SUCCESSFUL'
10    STOP 'SORT UNSUCCESSFUL'
      END
$     FOR SAMPLE_SORT2
```

```
$       LINK SAMPLE_SORT2.SYS$LIBRARY:HSYIMGLIB.OLB/LIB
$       RUN SAMPLE_SORT2
$
```

### B.3.3.3 呼叫 MERGE 常式

```
$ TYPE HSY$EXAMPLE.SAMPLE_MERGE.C
/*      This is a C language example.
        This program merges two input files into one output file. The
        key is  starting  from the  first  byte  of  the   record
        (offset zero byte) of size two bytes(i.e. 1 Hanyu character).
        It also checks if the input files sequence are in the right
        order.
*/
#include stsdef
struct DESCRIPTOR     { int leng ;
                        char *ptr ;
};
globalvalue           SOR$K_SEQ_PHONETIC.CODE，
                      SOR$M_SEQ_CHECK;
main()
{
        char          infile1[ ]="EXAM31.DAT" ,
                      infile2[ ]="EXAM32.DAT" ,
                      outfile[ ]="EXAM30OUT.DAT" ;
        struct DESCRIPTOR
                      inptr1={strlen(infile1) ,infile1},
                      inptr2={strlen(infile2) ,infile2},
                      outptr={strlen(outfile) ,outfile};
        short lrl,key[5] ;
        int status,option;
/* initialize the key */
        key[0]=1;                                   /*key no. */
        key[1]=SOR$K_SEQ_PHONETIC_CODE/*phonetic code */
        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 可呼叫的 HTPU 常式

可呼叫的 HTPU 常式可使 HTPU 在任何的程式語言和應用程式中皆可存取。HTPU 可以從任何一種程式語言寫成的程式呼叫出來；而此程式使用 OpenVMS 程序呼叫和條件處置標準來產生呼叫。亦可以從 OpenVMS 公用程式呼叫 HTPU，例如 HMAIL。可呼叫的 HTPU 可在程式之中執行中

文文 字處理功能。

HTPU 與 DECTPU 支援相同的可呼叫介面，如 " 簡化的可呼叫介面 " 和 " 完整的可呼叫介面 "。若需有關 DECTPU 可呼叫介面的細節，請參閱 *《VMS Utility Routines Manual》*第十四章。

簡言之，HTPU 可呼叫介面和 DECTPU 可呼叫介面之間的差異如下：

1. TPU$CLIPARSE 接受 "HTPU" 命令動詞而不是 "TPU" 命令動詞。
2. 應該與 SYS$SHARE:HTPUSHR.EXE 連結，而不是與 SYS$SHARE:TPUSHR.EXE 連結。
3. TPU$EXECUTE_COMMAND 常式可以執行 HTPU 特定的內建程式以及 DECTPU 內建程式。若需要關於 HTPU 特定的內建程式，請參閱 *《HTPU 和 HEVE 參考手冊》*；而 *《DEC Text Processing Utility Reference Manual》*則描述 DECTPU 內建程式。

### B.4.1 HTPU 可呼叫常式

下列可呼叫常式可在 HTPU 運作中使用。

1. 簡化的可呼叫介面

| | |
|---|---|
| TPU$TPU | 引動 HTPU 同等於 DCL 的命令 HTPU。 |
| TPU$EDIT | 引動 HTPU 來讀取和編輯輸入檔案，並且把修訂的緩衝區寫到輸出檔案。 |

2. 完整的可呼叫介面

| | |
|---|---|
| TPU$CLEANUP | 清除 HTPU 的內部資料結構，並且準備額外的引用。 |
| TPU$CLIPARSE | 剖析命令列，並且建造 TPU$INITIALIZE 的項目列。 |
| TPU$CLOSE_TERMINAL | 呼叫使用者常式中所呼叫的此常式和呼叫使用者常式的折回間之歷時，關閉 HTPU 終端機的通道。 |
| TPU$CONTROL | 呼叫 TPU$INITIALIZE 之後，將控制傳遞到 HTPU，直到使用者退出或離開 HTPU。 |
| TPU$EXECUTE | TPU$INITIALIZE 之後，執行 HTPU_COMMAND 敘述。 |
| TPU$EXECUTE_INIFILE | TPU$INITIALIZE 之後，立刻執行初設檔。 |
| TPU$FILEIO | 既定檔案輸出 / 入常式。 |
| TPU$HANDLER | 既定條件處置程式。 |
| TPU$INITIALIZE | 建立條件處置程式之後，起始設定 HTPU。 |
| TPU$MESSAGE | 使用內建程序 MESSAGE 寫入錯誤訊息和字串。 |
| TPU$PARSEINFO | 使用 CLI$DCL_PARSE 剖析命令之後，建造 TPU$INITIALIZE 的項目列。 |

## B.4.2 程式範例

以下是使用完整的可呼叫介面，並以 BLISS 寫入來引動 HTPU 的程式範例。

```
!++
!
! 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$LIBRARY:STARLET';
LITERAL
            TEST_USER_ARG = %X'FO';! an arbitrary pattern
!
! Table of Content
!
FORWARD ROUTINE
            test_main,              ! Module entry
            test_callback,          ! Callback for tpu$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 [12];
                        $test_bpv [0] = proc;
                        $test_opv [l] = 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 tpus$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 tpusmessage
!
!--
        BEGIN
        EXTERNAL ROUTINE
                lib$sgetl_dd,
                lib$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$W_LENGTH) = 80;
!
! Construct return string
! We need to use lib$sgetl_dd to allocate the memory
! for the return string since TPU uses lib$sfreel_dd
! to free our string when it needs to.
!
str_out_len = %CHARCOUNT(%STRING('Success'));
status = lib$sgetl_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

# 附錄 C

# 中文注音碼表

中文注音符號包括三個部份，即聲母、韻母和聲調。注音符號又分爲二十一個聲母、三十八個韻母和五個聲調。以下三張表分別對每個聲母、韻母，以及聲調指定對併值；對遞增順序的排序或合併，首先以聲母值的遞增順序對併。若聲母值相同時，再按遞增順序對併相同韻母值的字元。同理，若聲母、韻母相同時，再按遞增順序對併相同聲調值的字元。遞減順序的排序或合併，則以較大數值的字元首先對併。

## C.1 聲母表

| 聲母 | 代碼 |
|---|---|
| ㄅ | 1 |
| ㄆ | 2 |
| ㄇ | 3 |
| ㄈ | 4 |
| ㄉ | 5 |
| ㄊ | 6 |
| ㄋ | 7 |
| ㄌ | 8 |
| ㄍ | 9 |
| ㄎ | 10 |
| ㄏ | 11 |
| ㄐ | 12 |
| ㄑ | 13 |
| ㄒ | 14 |
| ㄓ | 15 |
| ㄔ | 16 |
| ㄕ | 17 |
| ㄖ | 18 |
| ㄗ | 19 |

| 聲母 | 代碼 |
| --- | --- |
| ㄘ | 20 |
| ㄙ | 21 |

## C.2 韻母表

| 韻母 | 代碼 |
| --- | --- |
| ㄚ | 1 |
| ㄛ | 2 |
| ㄜ | 3 |
| ㄝ | 4 |
| ㄞ | 5 |
| ㄟ | 6 |
| ㄠ | 7 |
| ㄡ | 8 |
| ㄢ | 9 |
| ㄣ | 10 |
| ㄤ | 11 |
| ㄥ | 12 |
| ㄦ | 13 |
| ㄧ | 14 |
| ㄧㄚ | 15 |
| ㄧㄛ | 16 |
| ㄧㄝ | 17 |
| ㄧㄞ | 18 |
| ㄧㄠ | 19 |
| ㄧㄡ | 20 |
| ㄧㄢ | 21 |
| ㄧㄣ | 22 |

| 韻母 | 代碼 |
|---|---|
| ㄧㄠ | 23 |
| ㄧㄡ | 24 |
| ㄨ | 25 |
| ㄨㄚ | 26 |
| ㄨㄛ | 27 |
| ㄨㄞ | 28 |
| ㄨㄟ | 29 |
| ㄨㄢ | 30 |
| ㄨㄣ | 31 |
| ㄨㄤ | 32 |
| ㄨㄥ | 33 |
| ㄩ | 34 |
| ㄩㄝ | 35 |
| ㄩㄢ | 36 |
| ㄩㄣ | 37 |
| ㄩㄥ | 38 |

## C.3 聲調表

| 聲調 | 代碼 |
|---|---|
| | 1 |
| ˊ | 2 |
| ˇ | 3 |
| ˋ | 4 |
| • | 5 |