HTPU 和 HEVE 用户参考手册

产品号： BA322-90020

2005 年 5 月

本手册概述汉字文本处理公用程序 (HTPU) 和汉字可扩充通用编辑程序 (HEVE)。

修订/更新资料： 这是取代适用于 VMS/Hanzi AXP 1.5 版本的 《 HTPU/HEVE 用户参考手册》的修订文件

软件版本： OpenVMS/Hanzi I64 8.2 版本
OpenVMS/Hanzi Alpha 7.3−2 版本

Hewlett−Packard Company
Palo Alto, California

Printed in Singapore

目 录

图

表

序 言

手册目的

本手册描述汉字文本处理公用程序 (HTPU) 语言和汉字可扩充通用编辑程序 (HEVE) 用户接口。其重点只集中在 HTPU 和 HEVE 与 OpenVMS DECTPU 和 EVE 不同之处, 因此用户应该参阅《DEC Text Processing Utility Reference Manual》 和 《Extensible Versatile Editor Reference Manual》以了解汉字特性以外的其他特性。本手册只可作参考文件, 而非详细的用户手册。

对象

本手册描述 HTPU 的汉字特性, 阅读本手册的用户对于 DECTPU 的一 般特性应有一些认识。有关 DECTPU 和 EVE 的一般特性, 请参阅《DEC Text Processing Utility Reference Manual》和《Extensible Versatile Editor Reference Manual》 。

手册结构

本手册包括两个部分和两个附录。第一部分有一章, 内容对 HTPU/HEVE 作一个综合描述, 并详细地说明 HTPU/HEVE 的主要特性。第二部分有 两章, 分别对新的和已修改的 HTPU 内设过程以及新的和已修改的 HEVE 命令提供详细的资料。附录 A 讨论 HTPU/HEVE 的限制, 附录 B 则列出新的 HTPU/HEVE 信息。

有关文献

1 《HEVE 用户手册》

2 《Guide to the DEC Text Processing Utility》

3 《Extensible Versatile Editor Reference Manual》

4 《DEC Text Processing Utility Reference Manual》

第 1 章

HTPU/HEVE 综述

汉字文本处理公用程序 (HTPU) 在 OpenVMS/Hanzi 操作系统下运行。HPTU 以 DEC 文本处理公用程序 (DECTPU) 为基础, 在文本处理过程中能够处理多字节字符[1] 和 ASCII 字符。HTPU 通过一整套新的 HTPU 内设过程和已修改的 DECTPU 内设过程提供汉字文本处理功能。

HTPU 提供以下的附加特性:

- 识别和操纵多字节字符能力
- 符号构成和画线能力
- 字符分类
- 全形和半形字符转换
- 多字节字符删除和光标移动
- 包括多字节字符的模式匹配
- DECwindows Motif/Hanzi 窗口环境支持

汉字可扩充通用编辑程序 (HEVE) 旨在为用户和 HTPU 之间提供一个方便的接口。除了与 EVE 相同的功能外[2], HEVE 还具有一些新的和增强的功能, 以处理汉字字符。

---

[1] 多字节字符包括三种类型的字符: a) 中文字符 b) 在DEC GB-2312字符集表中单字节 ASCII 字符的等价双字节字符, 即全形字符 c) 双字节符号

[2] 惟 HEVE 不支持 MCS 字符编辑

1.1 调用 HTPU/HEVE

1.1.1 HTPU 的新 DCL 命令动词

调用带有 HEVE (默认编辑程序) 的 HTPU 的基本 DCL 命令如下:

```
$ EDIT/HTPU
```

HTPU 使用逻辑名 HTPU$SECTION 定位编辑程序用户接口, HTPU$SECTION 的系统默认值为 HEVE$SECTION, 即 HTPU 将会调用 HEVE 节文件作为编辑程序 。

为节省键入数目, 您可以如下定义外部命令:

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

有了这个外部命令, 要启动带有 HEVE 的 HTPU, 只须输入 :

```
$ HEVE
```

要在 DECwindows Motif/Hanzi 环境下调用 HTPU 和使用 HEVE 作为用户接口, 您可以使用以下的 DCL 命令:

```
$ SET DISPLAY/CREATE/NODE=<您的工作站的节点名>
$ EDIT/HTPU/DISPLAY=MOTIF
```

或当 HEVE 作了以上的外部命令定义:

```
$ HEVE/DISPLAY=MOTIF
```

如果您包括一个文件名作为 HTPU 命令的参数, 那么 HTPU 就打开该文件以进行编辑。

HTPU 使用的一套默认命令限定词值与 DECTPU 使用的相似。以下数节说明新的命令限定词、已修改的命令限定词值、逻辑名和文件名。

注意: 在启动 HTPU 之前, 应该使用 HANZIGEN 公用程序来设定适当的终端特征和进程环境。有关详情, 请参阅《OpenVMS/Hanzi 用户手册》。

1.1.2 已更改的命令限定词值

/COMMAND
在 HTPU 中, /COMMAND 限定词的默认值是 HTPU$COMMAND。

/DISPLAY
/DISPLAY 限定词的默认值是 CHARACTER_CELL, 搜索映象 HTPU$CCTSHR.EXE。当指定 MOTIF 时, 则使用 HTPU$MOTIFSHR.EXE。

/INITIALIZATION
HEVE 编辑程序的默认初始化文件是 HEVE$INIT.EVE。如果您想对 HEVE 初始化文件使用一个逻辑名, 可以定义逻辑名 HEVE$INIT 指向初始化文件。

/SECTION
在 HTPU 中, /SECTION 限定词的默认值是 HTPU$SECTION。

/WORK
HTPU 的默认工作文件名是 HTPU$WORK.TPU$WORK。

1.1.3 已更改的逻辑名

虽然 HTPU/HEVE 中的逻辑名与 DECTPU/EVE 的不同, 但使用方法相似。下表总结了 HTPU/HEVE 中逻辑名的更改。

表 **1-1** 逻辑名的更改

DECTPU/EVE	**HTPU/HEVE**	**描述**
TPU$COMMAND	HTPU$COMMAND	命令文件逻辑名
TPU$DEBUG	HTPU$DEBUG	调试程序文件逻辑名
TPU$DISPLAY_ MANAGER	HTPU$DISPLAY_ MANAGER	屏幕更新程序
TPU$JOURNAL	HTPU$JOURNAL	日志文件目录
TPU$SECTION	HTPU$SECTION	节文件逻辑名
TPU$WORK	HTPU$WORK	工作文件逻辑名
EVE$INIT	HEVE$INIT	HEVE 初始化文件
EVE$KEYPAD	HEVE$KEYPAD	HEVE 默认小键盘

OpenVMS/Hanzi 有一个定义为 HEVE$SECTION 的全系统逻辑名 HTPU$SECTION。

1.1.4 已更改的默认文件名

下表总结了 HTPU/HEVE 中默认文件名的更改。

表 1-2 默认文件的更改

DECTPU/EVE	HTPU/HEVE	描述
TPU.DAT	HTPU.DAT	HTPU 默认 DECwindows Motif/Hanzi 资源文件
TPU$COMMAND.TPU	HTPU$COMMAND.TPU	默认命令文件
TPU$DEBUG.TPU	HTPU$DEBUG.TPU	默认调试程序文件
TPU$WORK.TPU $WORK	HTPU$WORK.TPU $WORK	默认工作文件
EVE.DAT	HEVE.DAT	HEVE 默认 DECwindows Motif/Hanzi 资源文件
EVE$SECTION.TPU $SECTION	HEVE$SECTION.TPU $SECTION	默认节文件
EVE$INIT.EVE	HEVE$INIT.EVE	HEVE 的默认初始化文件
EVE$WIDGETS _MOTIF.UID	HEVE$WIDGETS _MOTIF.UID	HEVE 的默认用户接口描述 (UID) 文件

1.2 用于 DEC GB-2312 字符集分类的新 HTPU 关键字定义

HTPU 增加了以下的关键字, 以增强功能:

- ASCII_CHAR
- DISPLAY_CURSOR
- FULL_FORM
- HANZI
- JUSTIFY_BOTH
- JUSTIFY_LEFT
- JUSTIFY_RIGHT
- NON_HANZI
- SIZE_INVERT

ASCII_CHAR 代表 ASCII 字符类别。关键字 ASCII_CHAR、FULL_FORM 和 SIZE_INVERT的使用, 在 CHANGE_CASE 内设过程的描述中有详述。半形到全形转换和全形到半形转换的限制, 在 CHANGE_CASE 内设过程的描述中也有叙述。

JUSTIFY_LEFT、 JUSTIFY_RIGHT 和 JUSTIFY_BOTH 向 JUSTIFY 内设过程指明执行对齐的方向。有关详情, 请参阅关于 JUSTIFY 内设过程的描述。

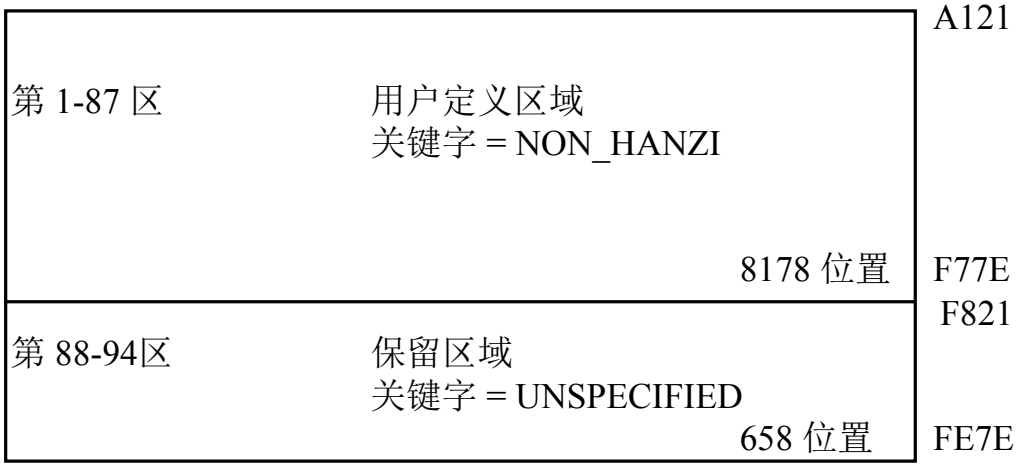
DISPLAY_CURSOR 用于 SET (DISPLAY_CURSOR) 内设过程, 以开关 HTPU 的显示光标态。有关详情, 请参阅关于 SET (DISPLAY_CURSOR) 内设过程的参考页。

FULL_FORM、HANZI 和 NON_HANZI 是用来划分 DEC GB-2312 字符集表的区域的。下图说明作为字符集表分类的关键字定义。

图 1-1 DEC GB-2312 汉字字符集表

第 1-2 区	GB-2312 非汉字 166 字符 关键字 = NON_HANZI	A1A1 A2FE
第 3 区	GB-2312 非汉字 94 字符 关键字 = FULL_FORM	A3A1 A3FE
第 4-9 区	GB-2312 非汉字 422 字符 关键字 = NON_HANZI	A4A1 A9FE
第 10-15 区	DEC 保留区域 关键字 = UNSPECIFIED	
第 16-55 区	GB-2312 一级汉字 3755字符 关键字 = HANZI	B0A1 D7FE
第 56-87 区	GB-2312 二级汉字 3008字符 关键字 = HANZI	D8A1 F7FE
第 88-94 区	DEC 保留区域 关键字 = UNSPECIFIED	F8A1 FEFE

图 1-2 扩充的 GB 字符集表



1.3 在 DECwindows Motif 环境下 HTPU/HEVE 与 DECTPU/EVE 之间的差别

在 DECwindows Motif 环境下, 除了 HTPU/HEVE 支持中文编辑以外, HTPU/HEVE 与 DECTPU/EVE 基本上相同。以下各小节说明它们的差别。

1.3.1 中文下拉和弹出项目单

HEVE 设有中英文下拉和弹出项目单。您可转换 HEVE 的项目单文本语言， 方法与转换 DECwindows Motif/Hanzi 对话期管理程序语言一样。

您只需从对话期管理程序下拉项目单 "任选项" 中选择 "语言... "。在 "语言任选项" 弹出项目单中，挑选 "中文" 以把项目单文本转为中文, 或挑选 "英国英语" 以把项目单文本转为英文, 余此类推。

1.3.2 HTPU 窗口接口名

DECTPU 建立如表 1-3 所示的三个窗口接口。

表 1-3 DECTPU 窗口接口

类别	名称	描述
Tpu	tpu	应用程序外壳
Main	tpu$mainwindow	主窗口
DECterm	DECwindows DECTPU	DECterm 窗口接口

在 HTPU 中, 窗口接口名已作了表 1-4 所示的更改。

表 1-4 HTPU 窗口接口

类别	名称	描述
Htpu	htpu	应用程序外壳
Main	tpu$mainwindow	主窗口
DECterm	DECwindows HTPU	DECterm 窗口接口

通常, 这不影响在 HTPU 与/或 HEVE 上的分层应用程序。然而, 如果您的应用程序设置那些窗口接口的资源, 您必须对 HTPU 建立的窗口接口使用新名。

1.3.3 CSText 窗口接口支持

您可以在 HTPU 内建立 CSText 窗口接口, 然后通过 SET(TEXT) 内设过程操纵 CSText 的文本资源。

1.3.4 中文输入法

当 HTPU 在 DECwindows Motif/Hanzi 下启动时, 您可以在 LK201 键盘上使用 <COMPOSE><SPACE> 键序列, 或者在 LK401 键盘上只使用 <COMPOSE> 键, 来启动中文字符的输入。当中文输入法启动之后, 它们的操作与您终端的一样, 惟您须使用与启动时相同的键序列来结束输入法。

编辑对话期间, DECwindows Motif/Hanzi 的输入法进程可能基于某种原因而终止。当您重新启动输入法进程时, HTPU 不能自动地使用该新的输入法进程, 您需重置 HTPU 的输入法处理。

HEVE 在下拉项目单 "任选项" 中提供一个 "重置输入法" 项。当您的输入法进程被终止并重新启动时, 您可以使用这个命令来重置 HTPU 的输入法处理。

1.4 受支持的终端

通常, HTPU/HEVE 在 OpenVMS/Hanzi 操作系统下运行并支持在 OpenVMS/Hanzi 所支持的映象终端上进行汉字和 ASCII 的面向屏幕编辑。

1.5 双语言求助

HTPU/HEVE 在子系统上提供双语言求助。在调用 HTPU/HEVE 之前, 您可使用 HANZIGEN 公用程序来预先设定所需语言。

1.6 双语言信息

HEVE 允许通知信息以英文或中文显示。在调用 HTPU/HEVE 之前, 您可使用 HANZIGEN 公用程序来预先设定所需语言。

第 2 章

新的和已修改的 HTPU 内设过程

本章描述 HTPU 中新的和已修改的内设过程。有关 DECTPU 内设过程的完整列表, 请参阅《DEC Text Processing Utility Reference Manual》。

下表列出 HTPU 的新内设过程。

表 2-1 新的 HTPU 内设过程

内设过程
ALIGN_CURSOR
CHANGE_SIZE
CHARACTER_CLASS
COMPOSE
DEC_HANZI
IS_CLASS
JUSTIFY
SET(ALIGNMENT_DEFAULT)
SET(DISPLAY_CURSOR)
SET(FILL_NOT_BEGIN)

表 2-1 新的 HTPU 内设过程 (续)

内设过程
SET(FILL_NOT_END)
SET(MARGIN_ALLOWANCE)
SET(UNIT_DEFAULT)

下表列出 HTPU 的已修改内设过程。

表 2-2 已修改的 HTPU 内设过程

内设过程
CHANGE_CASE
CURRENT_OFFSET
CURSOR_HORIZONTAL
CURSOR_VERTICAL
EDIT
FILL
GET_INFO
INDEX

表 **2-2** 已修改的 **HTPU** 内设过程 (续)

内设过程
KEY_NAME
LENGTH
MARK
READ_CHAR
READ_KEY
SCROLL
SEARCH
SELECT
SELECT_RANGE
SPLIT_LINE
SUBSTR

ALIGN_CURSOR

这个内设过程将光标移至字符的第一列, 没有其他副作用。

格式 **ALIGN_CURSOR**

描述 当光标未被定位于多字节字符的第一字节时, ALIGN_CURSOR 则将光标移至该字符的第一列上。

CHANGE_CASE

这个内设过程修改指定的文本单元中所有字母字符的大小写, 并把双字节全形字符更改为半形 ASCII 字符, 或把半形 ASCII 字符更改为双字节全形字符。

格式

[return_value:=] CHANGE_CASE ({ *buffer* / *range* / *string* } **,**
*keyword1***[,**
*keyword2***])**

参数

buffer
您要在其内更改所有字符大小写的缓冲区。注意, 如果您为第一参数指定一个缓冲区, 就不能使用关键字 NOT_IN_PLACE 作为第三参数。

range
您要在其内更改所有字符大小写的范围。注意, 如果您为第一参数指定一个范围, 就不能使用关键字 NOT_IN_PLACE 作为第三参数。

string
这是一个表示字符串常数或字符串表达式的变量名称，用来指定您需要更改的字符串。如果您指定 IN_PLACE 为第三参数, CHANGE_CASE 就会对第一参数中指定的字符串作指定的更改。CHANGE_CASE 对字符串常数则无效。

keyword1
这个关键字指定您要进行的更改类型。

- LOWER - 把指定的字符串中的字母字符 (全形和半形) 更改为小写。

- UPPER - 把指定的字符串中的字母字符 (全形和半形) 更改为 大写。

- INVERT - 把指定的字符串中的字母字符 (全形和半形) 由小写 更改为大写, 或由大写更改为小写。如果字符是大写, 会更改为 小写; 如果是小写, 则更改为大写。

- ASCII_CHAR - 把指定的字符串中属于 DEC GB-2312 字符表 第 3 区的双字节全形字符和第 1 区中的 '$' 和 '～' 更改为 ASCII (半形) 字符。注意 CHANGE_CASE 在 ASCII_CHAR 下对 '￥' 无效。

- FULL_FORM - 把指定的字符串中的 ASCII 字符 (半形) 更改为双字节全形字符。除 '$' 和 '～' 外, 所有其他半形字符都被转换为 DEC GB-2312 字符表第 3 区中的双字节全形字符。'$' 和 '～' 在 DEC GB-2312 字符表第 1 区中有其等同的双字节全形字符。

- SIZE_INVERT - 把指定的字符串中的 ASCII 字符更改为全形字符, 或把全形字符更改为 ASCII 字符。在 CHANGE_CASE 中使用 ASCII_CHAR 和 FULL_FORM 关键字的限制也适用于 SIZE_INVERT。

keyword2

这个关键字指定结果应送回的地方。

- IN_PLACE - 指引 HTPU 在指定的缓冲区、范围或字符串中进行指明的更改。这是默认值。

- NOT_IN_PLACE - 指引 HTPU 将指定的字符串保持不变并送回一个指定更改的结果的字符串。如果第一参数被指定为范围或缓冲区, 就不能使用 NOT_IN_PLACE。如果要使用NOT_IN_PLACE, 必须为 CHANGE_CASE 指定一个送回值。

送回值 指定结果的变量。

- returned_buffer - 如果您为第一参数指定一个缓冲区, 这就是指向包含已修改文本的缓冲区的缓冲区类型变量。变量 "returned_buffer" 指向的缓冲区与第一参数所指向的缓冲区相同。
- returned_range - 如果您为第一参数指定一个范围, 这就是包 含已修改文本的范围。送回的范围与指定为参数的范围一样跨越同一文本, 但是, 它们是不同的范围。如果您其后更改或删除其中一个范围, 对另一范围并无影响。
- return_string - 如果您为第一参数指定一个字符串, 这就是包 含已修改文本的字符串。即使您指定 IN_PLACE, CHANGE_CASE 能送回一字符串。

描述 如果指定送回值的话, CHANGE_CASE 送回一个结果。

出错信号

TPU$_TOOFEW	出错	参数太少。
TPU$_TOOMANY	出错	参数太多。
TPU$_ARGMISMATCH	出错	为 CHANGE_CASE 提供的其中一个参数数据类型错误。
TPU$_INVPARAM	出错	为 CHANGE_CASE 提供的其中一个参数数据类型错误。
TPU$_BADKEY	警告	您给 CHANGE_CASE 的关键字错误。
TPU$_NOTMODIFIABLE	警告	您不能在不可修改缓冲区内更改文本的大小写。
TPU$CONTROLC	出错	您在 CHANGE_CASE 执行期间按下 CTRL/C。

CHANGE_SIZE

这个内设过程修改指定的文本单元中所有字母字符的大小, 即把双字节全形字符更改为半形 ASCII 字符, 或反之亦然。

格式

[return_value :=]CHANGE_SIZE ({ *buffer* / *range* / *string* } **,** *keyword1***[,** *keyword2***])**

参数

buffer
您要在其内更改所有字符大小的缓冲区。注意, 如果您为第一参数指定一个缓冲区, 就不能使用关键字 NOT_IN_PLACE 作为第三参数。

range
您要在其内更改所有字符大小的范围。注意, 如果您为第一参数指定一个范围, 就不能使用关键字 NOT_IN_PLACE 作为第三参数。

string
这是一个表示字符串常数或字符串表达式的变量名称，用来指定您需要更改的字符串。如果您指定 IN_PLACE 为第三参数, CHANGE_SIZE 就会对第一参数中指定的字符串作指定的更改。CHANGE_SIZE 对字符串常数则无效。

keyword1
这个关键字指定您要进行的更改类型。

- ASCII_CHAR - 把指定的字符串中属于 DEC GB-2312 字符表第 3 区 的双字节全形字符和第一区中的 ’$’ 和 ’～’ 更改为 ASCII (半形) 字符。注意, CHANGE_SIZE 在 ASCII_CHAR 情况下对 ’￥’ 无效。

- FULL_FORM - 把指定的字符串中的 ASCII 字符 (半形) 更改为双字节全形字符。除 ’$’ 和 ’～’ 外, 所有其他半形字符都被转换为 DEC GB2312 字符表第 3 区中的双字节全形字符。’$’ 和 ’～’ 在 DEC GB2312 字符表第 1 区中有其双字节全形等同字符。

- SIZE_INVERT - 把指定的字符串中的 ASCII 字符更改为全形字符, 或把全形字符更改为 ASCII 字符。在 CHANGE_SIZE 中使用 ASCII_CHAR 和 FULL_FORM 关键字的限制也适用于 SIZE_INVERT。

  keyword2
  这个关键指定结果应送回的地方。

- IN_PLACE - 指引 HTPU 在指定的缓冲区、范围或字符串中进行指明的更改。这是默认值。

- NOT_IN_PLACE - 指引 HTPU 将指定的字符串保持不变并送回一个指定更改的结果的字符串。如果第一参数被指定为范围或缓冲区, 就不能使用 NOT_IN_PLACE。如果要使用 NOT_IN_PLACE, 必须为 CHANGE_SIZE 指定一个送回值。

送回值　指定结果的变量

- returned_buffer - 如果您为第一参数指定一个缓冲区, 这就是指向包含已修改文本的缓冲区的缓冲区类型变量。变量 "returned_buffer" 指向的缓冲区与第一参数所指向的缓冲区相同。

- returned_range - 如果您为第一参数指定一个范围, 这就是包含已修改文本的范围。送回的范围与指定为参数的范围一样跨越同一文本, 但是, 它们是不同的范围。如果您其后更改或删除其中一个范围, 对另一范围并无影响。

- return_string - 如果您为第一参数指定一个字符串, 这就是包含已修改文本的字符串。即使您指定 IN_PLACE , CHANGE_SIZE 能送回一字符串。

描述　　如果指定送回值的话, CHANGE_SIZE 送回一个结果。

出错信号

TPU$_TOOFEW	出错	参数太少。
TPU$_TOOMANY	出错	参数太多。
TPU$_ARGMISMATCH	出错	为 CHANGE_SIZE 提供的其中一个参数数据类型错误。
TPU$_INVPARAM	出错	为 CHANGE_SIZE 提供的其中一个参数数据类型错误。
TPU$_BADKEY	警告	您给 CHANGE_SIZE 的关键字错误。
TPU$_NOTMODIFIABLE	警告	您不能在不可修改缓冲区内更改文本的大小写。
TPU$CONTROLC	出错	您在 CHANGE_SIZE 执行期间按下 CTRL/C。

CHARACTER_CLASS

这个内设过程识别输入字符串中第一个字符的字符类别 (如汉字、非汉字)。注意, HTPU 把一个多字节的字符作为一个字符来处理。

格式 **keyword := CHARACTER_CLASS** *(string)*

参数 ***string***
一个用引号括起来的字符串、表示一个字符串常数的变量名或指示一个字符串的表达式。如果字符串的长度超过一个字符, 则会送回第一个字符的字符类别。

描述 可表示送回的字符串中第一个字符的字符类别的关键字如下:

- ASCII_CHAR - ASCII 字符 (半形字符)。
- FULL_FORM - 在 DEC GB-2312 字符表第 3 区中的双字节全形字符。
- NON_HANZI - 有关此关键字的描述, 请参阅本手册图 1-1 和图 1-2。
- HANZI - 有关此关键字的描述, 请参阅本手册的图 1-1 和图 1-2。
- UNSPECIFIED - 有关此关键字的描述, 请参阅本手册的图 1-1 和 图 1-2。

出错信号	TPU$_TOOFEW	出错	参数太少。
	TPU$_TOOMANY	出错	参数太多。
	TPU$_INVPARAM	出错	参数类型不正确。
	TPU$_NEEDTOASSIGN	出错	CHARACTER_CLASS 必须在赋值语句的右边。

COMPOSE

这个内设过程把一个 ASCII 字符串变换为一些特别的双字节符号。

格式 **string2 := COMPOSE** *(string1)*

参数 ***string1***
一个用引号括起来的字符串、表示一个字符串常数的变量名或指示一个字符串的表达式。ASCII 字符串会被转换为相应的双字节符号和双字节字符, 如下表所述。

表 2-3 ASCII 序列及相应的双字节符号

ASCII 序列	相应的双字节符号
-	━
0	┃
1	┗
2	┻
3	┛
4	┣
5	╋
6	┫

表 **2-3 ASCII** 序列和相应的双字节符号 **(**续**)**

ASCII 序列	相应的双字节符号
7	┌
8	┬
9	┐
f-	─
f0	│
f1	└
f2	┴
f3	┘
f4	├
f5	┼
f6	┤
f7	┌
f8	┬
f9	┐
t1	└
t2	┴
t3	┘
t4	├
t5	┼
t6	┤
t7	┌
t8	┬
t9	┐
y1	└
y2	┴
y3	┘
y4	├
y5	┼
y6	┤
y7	┌
y8	┬

表 **2-3 ASCII** 序列和相应的双字节符号 **(续)**

ASCII 序列	相应的双字节符号
y9	┐
| |	‖
| ^	↑
| v	↓
([	〔
( )	○
)]	〗
[[	「
[(	〖
[ ]	□
])	〕
]]	」
< <	《
< =	≤
< >	◇
< \	△
< /	≮
< -	←
> >	》
> =	≥
> <	☆
> -	→
: -	÷
+ -	±
x x	×
= /	≠
= |	∵
, .	…

表 **2-3 ASCII** 序列和相应的双字节符号 (续)

ASCII 序列	相应的双字节符号
. .	¨
. ;	∴
. c	℃
‘ ‘	“
’ ’	〃
c /	¢
| -	£
s s	§
o o	∞
o >	♂
o +	♀
K > <	★
k ( )	●
k < >	◆
k [ ]	■
K < \	▲

出错信号	TPU$TOOFEW	出错	参数太少
	TPU$TOOMANY	出错	参数太多。
	TPU$INVPARAM	出错	参数类型不正确。
	TPU$NEEDTOASSIGN	出错	COMPOSE 必须在赋值语句的右边。

例子

```
1     PROCEDURE user_compose
      COPY_TEXT(COMPOSE('k><><'))
      ENDPROCEDURE
```

上述过程把 '★☆' 插入当前编辑位置上。

```
2     PROCEDURE user_part_compose
      COPY_TEXT(COMPOSE('=/aK><'))
      ENDPROCEDURE
```

上述过程把 '≠a★' 插入当前编辑位置上。

CURRENT_OFFSET

这个内设过程送回一个整数, 表示当前字符位置在当前行内的字节或字符偏移。

格式 **integer := CURRENT_OFFSET** ***[(keyword)]***

参数 ***keyword***
可以指定的关键字如下:

- CHARACTERS - 字符偏移。偏移值以字符为单位计算。
- BYTES - 字节偏移。偏移值以字节为单位计算。

如果您不指定关键字, 默认为 CHARACTERS。您也可以用 SET(UNIT_DEFAULT) 内设过程更改默认。有关的完整描述, 请参阅 SET(UNIT_DEFAULT) 一节。

描述 有关 CURRENT_OFFSET 内设过程的基本描述, 请参阅《DEC Text Processing Utility Reference Manual》。

在 HTPU 中, 如果加上关键字参数, 则字符偏移和字节偏移都可以使用。送回的偏移值总是以字符的边界为准。

出错信号	TPU$_NEEDTOASSIGN	出错	CURRENT_OFFSET 必须在赋值语句的右边。
	TPU$_TOOFEW	出错	参数太少。
	TPU$_NOCURRENTBUF	出错	您的位置不在缓冲区内。

例子

```
1     my_char_off:=CURRENT_OFFSET(CHARACTERS)
```

这个赋值语句为变量 my_char_off 赋予一个整数, 该整数代表当前字符的字符偏移。

```
2     my_byte_off:=CURRENT_OFFSET(BYTES)
```

这个赋值语句为变量 my_byte_off 赋予一个整数, 该整数代表当前字符的字节偏移。

CURSOR_HORIZONTAL

这个内设过程把光标水平地移过指定的屏幕列数。

格式 **[integer2 :=]CURSOR_HORIZONTAL** *(integer1 [,keyword])*

参数

integer1
带符号 (+/-) 的整数值, 该整数值指定光标要移过的屏幕列数。

keyword
可以指定的关键字如下:

- CHARACTERS - 如果光标移动后, 光标位置不在一个多字节字符的第一字节上, 则 CURSOR_HORIZONTAL 会令光标移位到第一列。
- BYTES - 移动光标时不理会缓冲区内容, 惟当 DISPLAY_CURSOR 态被启动时则例外。当 DISPLAY_CURSOR 态被启动时, 尽管内部光标位置可能不在第一列, 但是屏幕光标显示在字符的第一列。有关详情, 请参阅关于 SET (DISPLAY_CURSOR) 的描述。

如果您不指定关键字, 默认为 BYTES。您也可以用 SET(ALIGNMENT_DEFAULT) 内设过程更改默认。有关的完整描述, 请参阅 SET(ALIGNMENT_DEFAULT) 一节。

送回值 送回值表示光标移动了的列数。如果 HTPU 不能按 integerl 指定的列数移动光标, 光标会尽可能移动最多的列数。HTPU 可允许送回负数值, 但这种表示法保留给 HTPU 日后的版本。送回值表示光标向左或右移动了的确实列数。 如果光标没有移

动, 则送回值为 0。如果 CURSOR_HORIZONTAL 产生了错误, 则送回值是不确定的。

描述

有关 CURSOR_HORIZONTAL 内设过程的基本描述, 请参阅《DEC Text Processing Utility Reference Manual》。

在 HTPU 中, 通过加上关键字参数, 您可以控制内设过程操作后光标在多字节字符上的位置。

由于一个汉字字符占一列以上, 通过在第二个参数中指定 BYTES 关键字, CURSOR_HORIZONTAL 可以把光标移到多字节字符的非第一列的位置上。

当 DISPLAY_CURSOR 态被启动时, 光标总是显示在字符的第一字节, 而不管它是单字节还是多字节字符。可是, 光标内部存放位置可能在字符的非第一字节的位置。有关更详细的资料, 请参阅 SET (DISPLAY_CURSOR) 的有关页。

出错信号

TPU$_TOOFEW	出错	参数太少。
TPU$_TOOMANY	出错	参数太多。
TPU$_INVPARAM	出错	参数类型不正确。

CURSOR_VERTICAL

这个内部过程按指定的屏幕行数向上或向下移动光标。

格式 **[integer2 :=]CURSOR_VERTICAL** *(integer1 [,keyword])*

参数

integer1
带符号 (+/-) 的整数值, 该整数值指定光标要移动的屏幕行数。

keyword
可以指定的关键字如下:

- CHARACTERS - 如果光标移动后, 光标位置在一个多字节字符的非第一字节上, 则 CURSOR_HORIZONTAL 会令光标移位到第一列。
- BYTES - 移动光标时不理会缓冲区内容, 惟当 DISPLAY_CURSOR 态被启动时则例外。当 DISPLAY_CURSOR 态被启动时, 尽管内部光标位置可能不在第一列, 但是屏幕光标显示在字符的第一列。有关详情, 请参阅关于 SET (DISPLAY_CURSOR) 的描述。

如果您不指定关键字, 默认为 BYTES。您也可以用 SET(ALIGNMENT_DEFAULT) 内设过程更改默认。有关的完整描述, 请参阅 SET(ALIGNMENT_DEFAULT) 一节。

送回值 送回值表示光标向上或向下移动了的行数。如果 HTPU 不能按 integer1 指定的行数移动光标, 光标会尽可能移动最多的行数。如果 CROSS_WINDOW_BOUNDS 设定为 ON, CURSOR_VERTICAL 可以把光标放在另一个窗口。在这种情

况下, 送回值是一个负数。一个负的送回值不表示光标向上移动了。如果光标没有移动, 送回值为 0。如果 CURSOR_VERTICAL 产生了错误, 则送回值是不确定的。

描述

有关 CURSOR_VERTICAL 内设过程的基本描述, 请参阅《DEC Text Processing Utility Reference Manual》。

在 HTPU 中, 通过加上关键字, 您便可以控制内设过程操作后光标在多字节字符上的位置。

这个过程基本上不更改光标的列位置。但是, 如果在关键字字段里指定了 CHARACTERS, 则会把移到一个多字节字符非第一列的光标移位到第一列。如果您不希望移动光标的列位置, 可以在关键字字段里指定 BYTES。

当您在 DISPLAY_CURSOR 态设定为 ON 的情况下使用 BYTES 关键字时, 虽然光标在内部存放的列位置没有更改, 但是屏幕光标总是在多字节字符的字符边界位置。有关更详细的资料, 请参阅 SET(DISPLAY_CURSOR) 的有关页。

出错信号

TPU$_TOOFEW	出错	参数太少。
TPU$_TOOMANY	出错	参数太多。
TPU$_INVPARAM	出错	参数类型不正确。

DEC_HANZI

这个内设过程互换字符和字符代码 (ASCII 代码或 DEC GB-2312 字符代码)。

格式 **integer := DEC_HANZI** *(string1)*

格式 **string := DEC_HANZI** *(integer1)*

参数

integer1
表示您要转换的 ASCII 字符或多字节字符的整数。

决定 ASCII 或 DEC GB-2312 字符集的整数 (十进位数字)。

string1
您要获得其 ASCII 代码或 DEC GB-2312 字符代码的字符。

描述

如果给定了整数参数, DEC_HANZI 内设过程会按指定的数字送回一个相应的单字节或多字节字符。数字与字符的匹配是根据 ASCII 或 DEC GB-2312 字符集的规则来确定的。但是, 给定的整数在 DEC GB-2312 表里是否有一个有效字符, DEC_HANZI 不会提供有关的信息。

如果给定了字符串参数, DEC_HANZI 内设过程会按 ASCII 或 DEC GB-2312 字符表送回字符串中第一个字符的等价整数。

出错信号	TPU$_NEEDTOASSIGN	出错	DEC_HANZI 必须在赋值语句的右边。
	TPU$_TOOFEW	出错	参数太少。
	TPU$_TOOMANY	出错	参数太多。
	TPU$_ARGMISMATCH	出错	参数的数据类型不正确。
	TPU$_NULLSTRING	出错	您传递了一个长度为 0 的字符串给 DEC_HANZI。

例子

以下过程把 DEC GB-2312 字符集的任何一区中的一个表插入当前缓冲区内。例如, 您如果执行 user_hanzi_list(20), 第 20 区的表便会插入编辑点的当前位置上。

```
PROCEDURE user_hanzi_list (section)
LOCAL cnt, col, low_byte;

cnt := section * 256 + 41120;
max := cnt + 94;

COPY_TEXT (FAO ('第 !ZL 区',section));
SPLIT_LINE;

COPY_TEXT ("          0 1 2 3 4 5 6 7 8 9 A B C D E F");
SPLIT_LINE;

COPY_TEXT (user_hex (cnt) + ' ');
col := 1;
LOOP
   EXITIF cnt > max;
   IF col > 16 THEN
      SPLIT_LINE; UPDATE (current_window);
      col := 1;
      COPY_TEXT (user_hex (cnt) + ' ');
   ENDIF;
   low_byte := cnt - ((cnt / 256) * 256);
   IF low_byte <> 160 THEN
      COPY_TEXT (DEC_HANZI (cnt));
```

```
   ELSE
      COPY_TEXT (' ');
   ENDIF;
   cnt := cnt + 1;
   col := col + 1;
ENDLOOP;
SPLIT_LINE;
ENDPROCEDURE

PROCEDURE user_hex (dec_num)
LOCAL res, rmn, temp;

temp := dec_num;
IF temp = 0 THEN res := '0' ELSE res := '' ENDIF;
LOOP
   EXITIF temp <= 0;
   rmn := temp - ((temp / 16) * 16);
   temp := temp / 16;
   IF (0 < rmn) AND (rmn < 16) THEN
      res := SUBSTR ('123456789ABCDEF', rmn, 1) + res;
   ELSE
      res := '0' + res;
   ENDIF;
ENDLOOP;
user_hex := res;
ENDPROCEDURE
```

如果用户发出 user_hanzi_list(3) 命令, 以下各行会在当前编辑点之后显示出来。

```
第 3 区
    0 1 2 3 4 5 6 7 8 9 A B C D E F
03A0 ！＂＃￥％＆＇（）＊＋，－．／
03B0 ０１２３４５６７８９：；＜＝＞？
03C0 ＠ＡＢＣＤＥＦＧＨＩＪＫＬＭＮＯ
03D0 ＰＱＲＳＴＵＶＷＸＹＺ［＼］＾＿
03E0 ＇ａｂｃｄｅｆｇｈｉｊｋｌｍｎｏ
03F0 ｐｑｒｓｔｕｖｗｘｙｚ｛｜｝－
```

EDIT

这个内设过程按指定的关键字修改一个字符串。EDIT 与 DCL 词法功能 F$EDIT 相似, 但不是完全相同。内设过程与词法功能的差异在以下描述中有说明。

格式

[return_value :=]EDIT ({ *buffer* / *range* / *string* } ***,keyword1[*** ***, . . .] [,keyword2][,keyword3])***

参数

buffer
您要在其内编辑文本的缓冲区。注意, 如果您为第一参数指定一个缓冲区, 就不能使用关键字 NOT_IN_PLACE 作为第三参数。

range
您要在其内编辑文本的范围。注意, 如果您为第一参数指定一个范围, 就不能使用关键字 NOT_IN_PLACE 作为第三参数。

string
您要修改的字符串, 如果您指定一个送回值, 该送回值由您在第一参数中指定的字符串组成, 而该字符串已按您在第二或以后的参数中指定的方法作出修改。如果您指定 IN_PLACE 作第三参数, EDIT 就会对第一参数中指定的字符串作指定的更改。EDIT 对字符串常数则无效。

keyword1
您可以从下表中随意选择任何数目的关键字:

- COLLAPSE - 除去所有空格 (包括全形空格)和跨栏标记。
- COMPRESS - 用单个半形空格替换多个空格 (包括全形空格）和跨栏标记。
- TRIM - 除去前导空格 (包括全形空格) 和跨栏标记, 以及尾随空格 (包括全形空格) 和跨栏标记。
- TRIM_LEADING - 除去前导空格 (包括全形空格) 和跨栏标记。
- TRIM_TRAILING - 除去尾随空格 (包括全形空格) 和跨栏标记。
- LOWER - 把所有大写字母 (包括全形字母) 转换为小写。
- UPPER - 把所有小写字母 (包括全形字母) 转换为大写。
- INVERT - 转换指定的字母 (包括全形字母) 当前的大小写; 小写字母转换为大写, 大写字母转换为小写。
- ASCII_CHAR - 把指定字符串中属于 DEC GB-2312 字符表第 3 区中的双字节全形字符和第 1 区的 ’$’ 和 ’～’ 转换为 ASCII (半形) 字符。注意, EDIT 在 ASCII_CHAR 的情况下对 ’￥’ 无效。
- FULL_FORM - 把指定字符串中的 ASCII 字符 (半形) 转换为双字节全形字符。除 ’$’ 和 ’～’ 外, 所有其他半形 ASCII 字符转换为 DEC GB-2312 字符表第 3 区中的双字节全形字符。’$’’ 和 ’～’ 在 DEC GB-2312 字符表的第 1 区中有等价的双字节字符。
- SIZE_INVERT - 把指定字符串中的半形字符转换为全形字符, 或把指定字符串中的全形字符转换为半形字符。有关 EDIT 使用 ASCII_CHAR 和 FULL_FORM 关键字的限制也适用于 SIZE_INVERT。

如果您指定超过一个 TRIM 关键字 (TRIM、TRIM_LEADING、TRIM_TRAILING), 那么所有您指定的TRIM 操作都会执行。如果您指定超过一个字符转换关键字 (UPPER、LOWER、INVERT、ASCII_CHAR、FULL_FORM、SIZE_INVERT), 那么您指定的最后一个 关键字会确定字符串中的字符如何改。

keyword2
您可以指定 ON 或 OFF 作为任选的最后一个参数。

- ON - 指示 HTPU 启动作为 HTPU 引号的单引号或双引号的识别功能。 在这种情况下, 用引号括起来的子串不会更改。
- OFF - 指示 HTPU 关闭作为 HTPU 引号的单引号或双引号的识别功 能。用引号括起来的子串会按指定更改。

keyword3
这个关键字指示 HTPU 在何处作指明的更改。有效关键字及其意义 如下:

- IN_PLACE - 在原位作指明的更改。这是默认值。
- NOT_IN_PLACE - 使指定的字符串保持不变并送回一个指定编辑的结果的字符串。如果第一参数被指定为范围或缓冲区, 就不能使用 NOT_IN_PLACE。如果要使用 NOT_IN_PLACE, 必须为 EDIT 指 定一个送回值。

送回值 指定结果的变量。

- returned_buffer - 如果您为第一参数指定一个缓冲区, 这就是指向包含已修改文本的缓冲区的缓冲区类型变量。变量 "returned_buffer" 指向的缓冲区与第一参数所指向的的缓冲区相同。
- returned_range - 如果您为第一参数指定一个范围, 这就是包含已 修改文本的范围。送回的范围与指定为参数的范围一样跨越同一文本, 但是, 它们是不同的范围。如果您其后更改或删除其中一个范围, 对另一范围并无影响。

- return_string - 如果您为第一参数指定一个字符串, 这就是包含已 修改文本的字符串。即使您指定IN_PLACE, EDIT 能送回一字符串。

描述

HTPU 在原位修改指定的缓冲区、范围和字符串的第一参数。

如果不指定 keyword2, EDIT 不会修改字符串中用引号括起来的文本。例如, 以下命令不会把 WELL 改为小写:

```
EDIT ('HE SANG "WELL"', LOWER)
```

结果, 字符串会变为:

```
he sang "WELL"
```

如果您指定的字符串有一个开引号标记, 但没有关引号标记, 则 TPU$_MISSINGQUOTE 状态会被送回。在这种情况下, 从没有关闭的开引号标记开始至字符串结尾的文本, 会被视为是用引号括起来的字符串的一部分, 不会被修改。

您可以使用关键字 OFF 作为 EDIT 内设过程的最后一个参数, 以便 撤销识别单引号或双引号作为 HTPU 引号标记的功能。因此, 您如果指定了关键字 OFF, 引号会被作为普通文本处理。EDIT 与 DCL 词法功能 F$EDIT 相似。不过, 您应该注意以下的不同:

- EDIT 在原位修改字符串, F$EDIT 则送回结果。
- EDIT 接受关键字作为参数, F$EDIT 则需要由字符串指定编辑命令。 F$EDIT 不能处理汉字字符。

出错信号	TPU$_TOOFEW	出错	参数太少。
	TPU$_TOOMANY	出错	参数太多。
	TPU$_INVPARAM	出错	参数类型不正确。
	TPU$_ARGMISMATCH	出错	给 EDIT 的其中一个参数数据类型错误。
	TPU$_MISSINGQUOTE	出错	字符串缺少关引号标记。
	TPU$_BADKEY	警告	使用了一个错误的关键字。

例子

以下语句把"P R O D U C T N A M E"全形字符串转换为ASCII字符, 并在信息窗口显示编辑后的字符串。

```
pn:='P R O D U C T  N A M E';

EDIT(pn,ASCII_CHAR);
MESSAGE(pn);
```

FILL

这个内设过程把指定的缓冲区或范围内的文本重新格式化, 使各行文本的长度差不多相等。

格式

FILL ({ *buffer* / *range* } ***[,string[,integer1[,integer2[,integer3]]]]***)

参数

buffer
您要把其内的文本重新格式化的缓冲区。

range
您要把其内的文本重新格式化的范围。

string
您要用来作为缓冲区内文本字分隔符的一列表半形 ASCII 字符。ASCII 空格字符一定是字分隔符。这个字符串不能包括任何多字节字符。

integer1
左边线的值。左边线值必须大于 1, 并小于右边线值。此值默认为缓冲区的左边线。

integer2
右边线的值。右边线值必须大于左边线值, 并且不能超过缓冲区的最大记录容量。此值默认为缓冲区的右边线。

integer3
第一行内缩的值。此值修改第一行的左边线。它可以是正或负。但是, 将此值与左边线值相加, 结果必须大于 1, 并且小于右边线值。此值默认为 0。

描述

有关 FILL 内设过程的基本描述, 请参阅《DEC Text Processing Utility Reference Manual》。

在 HTPU 中, FILL 内设过程已经增强至可处理多字节字符串。如果第二个字符位置超出右边线, FILL 会将这两个相邻的多字节字符分开。

增强的 FILL 内设过程也可处理异常字符。用户可通过 SET(FILL_NOT_BEGIN) 和 SET(FILL_NOT_END) 内设过程指定异常字符。FILL 执行时, 所指定的异常字符将获处理。用户可使用 SET(MARGIN_ALLOWANCE) 设定 MARGIN_ALLOWANCE, 从而指定置于右边线外的异常字符数目。

出错信号

TPU$_TOOFEW	出错	参数太少。
TPU$_TOOMANY	出错	参数太多。
TPU$_ARGMISMATCH	出错	给 FILL 的其中一个参数数据类型错误。
TPU$_BADMARGINS	警告	您指定的其中个填补边线不正确。
TPU$_INVPARAM	出错	给 FILL 的其中一个参数数据类型错误。
TPU$_NOTMODIFIABLE	警告	您不能更改不可修改的缓冲区内文本。
TPU$_CONTROLC	出错	您在执行 FILL 时按下 CTRL/C。
TPU$_NOCACHE	出错	没有足够内存来分配一个新的高速缓存器。

GET_INFO

这个内设过程提供有关您的编辑上下文当前状态的资料。表 2-4 和 表 2-5 列出了 GET_INFO 送回的资料和相关的 HTPU 增强功能。有关原来的 GET_INFO 功能, 请参阅《DEC Text Processing Utility Reference Manual》。

格式 **return_value := GET_INFO** *(parameter1,parameter2)*

格式 **return_value := GET_INFO** *(parameter1,parameter2, parameter3)*

参数

parameter1
这是一个 HTPU 数据类型或一个关键字。GET_INFO 提供有关指定为 parameter1 的项目的资料。表 2-4 列出可被指定为 parameter1 的数据类型和相关的 HTPU 增强的特性。表 2-5 列出可被指定为 parameter1 的关键字以及相关的 HTPU 增强的特性。

parameter2
一个用引号括起来的字符串、一个变量名或一个表示表 2-4 或表 2 -5 中一个字符串常数的表达式。作为 parameter2 的字符串指示由 parameter1 指定的项目所需的资料种类。您可以用大写或小写输入字符串。

parameter3
一个用引号括起来的字符串或一个变量名。有关这个参数的详细描述, 请参阅《DEC Text Processing Utility Reference Manual》。

描述 下表总结了 HTPU 的新特性。

表 2-4 用变量作为 GET_INFO 的 parameter1

parameter1	parameter2	送回值	送回值的描述
缓冲区变量	"byte_offset"	整数	偏移是根据行首相对于当前字符位置的字节计算出来的。
	"character_ length"	整数	光标所在的字符中的字节数目。例如, 当光标位于一个双字节字符,就送回值 2。
	"character_ index"	整数	字符的第一列和光标位置之间的距离。当光标位于 ASCII 字符上, 就送回值 0。当光标位于一个双字节字符的第 二列, 就送回值 1。

表 **2-4** 用变量作为 **GET_INFO** 的 **parameter1 (** 续 **)**

parameter1	**parameter2**	送回值	送回值的描述
标记变量	"byte_offset"	整数	偏移是根据行首相对于标记的字节计算出来的。
	"character_ length"	整数	标记所在的字符中的字节数目。例如,当光标位于一双字节字符上, 就送回值 2 。
	"character_ index"	整数	字符的第一列和标记位置之间的距离。当光标位于 ASCII 字符上, 就送回值 0。当光标位于一个双字节字符的第二列上, 就送回值 1。
窗口变量	"character_ index"	整数	HTPU 内部光标位 置与屏幕光标位置之间的距离。仅当显示光标态设定为 ON 时此值才可以是非零值。

表 **2-5** 用关键字作为 **GET_INTO** 的 **parameter 1**

parameter1	parameter2	parameter3	送回值	送回值的描述
SYSTEM	"FILL_NOT_ BEGIN"		字符串	SET(FILL_NOT_ BEGIN) 内设过程指定的字符串。
SYSTEM	"FILL_NOT_ END"		字符串	SET(FILL_NOT_ END)内设过程指定的字符串。
SYSTEM	"MARGIN_ ALLOWANCE"		整数	由 SET(MARGIN_ ALLOWANCE) 内设过程指定的值。
SYSTEM	"ALIGN-" MENT_ DEFAULT" ‘	"CURSOR_ HORIZONTAL," "CURSOR_ VERTICAL", 或 "SCROLL"	关键字	指定内设过程的当前 ALIGNMENT_ DEFAULT, 是关键字 CHARACTERS 或 BYTES。
SYSTEM	"UNIT_ DEFAULT"	"SUBSTR", "INDEX", "LENGTH", 或 "CURRENT_ OFFSET"	关键字	指定内设过程的当前UNIT_DEFAULT, 将是关键字 CHARACTERS 或 BYTES。
SCREEN	"DISPLAY_ CURSOR"		关键字	当显示光标态设定为 ON 时送回 ON, 而当显示光标态设定为 ON 或再次关闭时送回 OFF。

出错信号	TPU$_BADREQUEST	警告	第二参数表示的请求与第一参数的数据类型不相符。
	TPU$_BADKEY	警告	错误的关键字值或不能识别的数据类型作为第一参数。
	TPU$_NOCURRENTBU	警告	没有定义当前缓冲区。
	TPU$_NOKEYMAP	警告	没有定义键映象。
	TPU$_NOKEYMAPLIST	警告	没有定义键映象表。
	TPU$_INVPARAM	出错	参数的数据类型不正确。
	TPU$_NEEDTOASSIGN	出错	GET_INFO 内设过程只能用于赋值语句的右边。
	TPU$_NOBREAKPOINT	警告	这个字符串常数只能用于一个断点之后。
	TPU$_NONAMES	警告	名称与请求名不匹配。
	TPU$_TOOFEW	出错	参数太少。
	TPU$_TOOMANY	出错	参数太多。
	TPU$UNKKEYWORD	出错	用了错误的关键字作为参数。

例子

```
buffer_byte_offset := GET_INFO (CURRENT_BUFFER,
"byte_offset");
```

这个赋值语句把从行首开始到缓冲区当前编辑位置的字节偏移值赋予变量 buffer_byte_offset。

INDEX

这个内设过程在一个字符串中找出一个字符或子串的位置, 并送回字符或子串在该字符串中的位置。

格式 **integer := INDEX** *(string1,string2[,keyword])*

参数

string1
内含您要查找的字符或子串的字符串。

string2
您要在 string1 内找出其最左边字符位置的字符或子串。

keyword
您可以指定以下任何一个关键字:

- CHARACTERS - 送回字符串首至子串位置的字符数目。
- BYTES - 送回字符串首至子串位置的字节数目。

如果您不指定关键字, 默认为 CHARACTERS。您也可用 SET(UNIT_DEFAULT) 内设过程更改默认。有关完整描述, 请参阅 SET (UNIT_DEFAULT) 一节。

描述

有关 INDEX 内设过程的基本描述, 请参《DEC Text Processing Utility Reference Manual》。

在 HTPU 中, 通过加上关键字参数, 您可选择以字节或字符为单位计算来获得字符位置的信息。

出错信号	TPU$_TOOFEW	出错	参数太少。
	TPU$_TOOMANY	出错	参数太多。
	TPU$_INVPARAM	出错	参数类型不正确。
	TPU$_NEEDTOASSIGN	出错	INDEX 只能用在赋值语句的右边。

例子

```
1     char_loc :=INDEX (’1 2 3 4 5 6 7’, ’6 7’, CHARACTERS)
```

这个赋值语句把一个整数值 6 赋予变量 char_loc, 因为子串 ’6 7’ 是在字符串 ’1 2 3 4 5 6 7’ 内的字符位置 6 开始的。

```
2     loc :=INDEX(’1 2 3 4 5 6 7’, ’6 7’, BYTES)
```

这个赋值语句把一个整数值 11 赋予变量 loc, 因为子串 ’6 7’ 是在字符串 ’1 2 3 4 5 6 7’ 内的字节位置 11 开始的。

IS_CLASS

这个内设过程检验您指定的字符串的字符类别是否属于关键字指定的类别。如果满足条件, IS_CLASS 会送回 1, 否则送回 0。

格式 **integer := IS_CLASS** *(string, keyword)*

参数

string
一个用引号括起来的字符串、一个表示字符串常数的变量名或一个表示字符串的表达式。如果字符串的长度超过一个字符, 则只检验第一个字符。

keyword
这个关键字指示字符类别。您可以指定以下任何一个关键字:

- ASCII_CHAR - ASCII 字符 (半形字符)。
- FULL_FORM - 双字节全形字符。
- NON_HANZI - 有关这个关键字的描述, 请参阅本手册的图 1-1 和 图 1-2。
- HANZI - 有关这个关键字的描述, 请参阅本手册的图 1-1 和 图 1-2。

描述 如果您指定的输入字符串中第一个字符的类别属于您在关键字参数中指定的类别, 这个内设过程会送回 1, 否则送回 0。

出错信号	TPU$_TOOFEW	出错	参数太少。
	TPU$_TOOMANY	出错	参数太多。
	TPU$_INVPARAM	出错	参数类型不正 确。
	TPU$_NEEDTOASSIGN	出错	IS_CLASS 只能用在赋值语句的右边。

JUSTIFY

这个内设过程把指定的缓冲区或范围内的文本重新格式化, 使各行文本可以向左、向右或向左右缩排。

格式

JUSTIFY({ *buffer* / *range* } ***[****,keyword* ***[****, integer1* ***[,****integer2* ***[****, integer3****]]]])***

参数

buffer
您要把其内的文本对齐的缓冲区。

range
您要把其内的文本对齐的范围。

keyword
关键字可取以下的值:

- JUSTIFY_LEFT - 除第一行外, 对齐每一行, 每一行在 integer1 或左边线开始。这是默认。
- JUSTIFY_RIGHT - 除最后一行外, 对齐每一行, 每一行在 integer2 或右边线结束。
- JUSTIFY_BOTH - 对齐每一行, 每一行在 integer1 或左边线开始(除了第一行)及在 integer2 或右边线结束(除了最后一行)。

integer1
左边线的值。左边线值必须大于 1, 并且小于右边线值。此值默认为缓冲区的左边线。

integer2
右边线的值。右边线值必须大于左边线值, 并且不能超过缓冲区的最大记录容量。此值默认为缓冲区的右边线。

integer3
第一行内缩的值。此值修改第一行的左边线。它可以是正或负。可是, 此值与左边线相加, 结果必须大于 1, 并且小于右边线值。此值默认为 0。

描述

JUSTIFY 仅识别单字节与多字节字符之间的空格 (%×20) 和接口为字分隔符。它首先调用 FILL 内设过程, 用边线修正值 0 来格式化范围或缓冲区, 以设置在指定范围或缓冲区内每一行的边线值和增补在每一行所包含的字符数目。

JUSTIFY 设置行的左边线以对齐左边线, 除非该行是要对齐缓冲区或范围的第一行。在这种情况下, JUSTIFY 设置行的左边线为对齐左边线加上第一行的缩进。可是, 如果您正在对齐一个范围, 而该范围不在一行的开始, JUSTIFY 就不更改该行的左边线。在这种情况下, 您必须指定第一行缩进到 0。

如果只要求左边对齐, JUSTIFY 内设过程在调用 FILL 之后返回。如果要求右边对齐, 它插入空格到每个记录的开始处。可是, 如果指定一个范围, 且起始偏移不是 0, 空格将插入到该范围第一行的 起始偏移处。

如果要求左和右都对齐, 空格将插入到那些包含空格的位置, 或 插入到 ASCII 和多字节字符之间的位置。如果找不到这样的位置, 就不插入空格且范围或缓冲区将不向右对齐。如果对齐 一个范围, 空格将仅在起始偏移之后插入。还要注意最后一行将不会向右对齐, 并可能延伸到右边线外。

注意, JUSTIFY 内设过程为对齐而插入空格字符到缓冲区, 并且在对齐以前不执行压缩, 这就使得用不同的边线设置连续调用 JUSTIFY 时不重新整理字, 因为以前插入的空格会作为数据处理。这样一来, 如果用户希望更改对齐的边线, 那么在再次调用 JUSTIFY 之前, 应该调用 EDIT 内设过程并用关键字 COMPRESS 来压缩要格式化的范围或缓冲区。

JUSTIFY 在完成对齐之后, 把光标放在已对齐文本的结尾。

出错信息

TPU$_TOOFEW	出错	参数太少。
TPU$_TOOMANY	出错	参数太多。
TPU$_ARGMISMATCH	出错	给 JUSTIFY 的其中一个参数数据类型错误。
TPU$_BADMARGINS	警告	您指定的其中一个填补边线不正确。
TPU$_INVPARAM	出错	给 JUSTIFY 的其中一个参数数据类型错误。
TPU$_NOTMODIFIABLE	警告	您不能更改不可修改缓冲区内文本 。
TPU$_CONTROLC	出错	您在执行 JUSTIFY 时 按下 CTRL/C。
TPU$_NOCACHE	出错	没有足够的内存来分配新的高速缓存器。

例子

```
JUSTIFY (current_buffer, JUSTIFY_BOTH, 1, 60, 10)
```

这个语句对齐当前缓冲区的左边线和右边线。第一行在列 1+10 (=11)开始, 而最后一行将不向右对齐。

KEY_NAME

这个内设过程送回一个键或组合键的 HTPU 关键字。

格式

keyword2 := KEY_NAME *({ integer / key_name / string } , [,keyword][, { FUNCTION / KEYPAD }])*

参数

integer
表示一个键的关键字的整数, 或是 HTPU 解释为 DEC 多国字符集中字符值的 0 和 255 之间的值, 或是 HTPU 解释为 DEC GB-2312 字符代 码的整数。

key_name
表示一个键的 HTPU 名的关键字。

string
表示主键盘中一个键值的字符串 。

keyword
这个关键字修改指定的键且可取以下的值:

- SHIFT_KEY - 指定建立的键名包括 HTPU SHIFT 键, 即 HEVE 的 GOLD 键。多字节字符仅接受这个关键字修改符。当使用于 多字节字符时, 不理会其他的关键字值。
- SHIFT_MODIFIED - 指定建立的键名包括键盘上标记为 SHIFT 的键。当应用到多字节字符时, 不理会这个值。

- ALT_MODIFIED - 指定建立的键名包括 ALT 键。当应用到多字节字符时, 不理会这个值。
- Ctrl_MODIFIED - 指定建立的键名包括 Ctrl 键。当应用到多字节字符时, 不理会这个值。
- HELP_MODIFIED - 指定建立的键名包括 HELP 键。当应用到多字节字符时, 不理会这个值。

FUNCTION
指定结果键名就是功能键名的参数。

KEYPAD
指定结果键名就是小键盘键名的参数。

描述

有关 KEY_NAME 内设过程的基本描述, 请参阅《DEC Text Processing Utility Reference Manual》。

您可为参数 string 指定一个多字节字符。也可用多字节字符定义一个键。

出错信号

TPU$_INCKWDCOM	警告	不一致的关键字组合。
TPU$_MUSTBEONE	警告	字符串必须是一个字符长。
TPU$_NOTDEFINABLE	警告	第二个参数非一个键的有效引用。
TPU$_NEEDTOASSIGN	出错	KEY_NAME 调用必须在赋值语句的右边。
TPU$_ARGMISMATCH	出错	错误类型的数据送到 KEY_NAME 内设过程。

TPU$_BADKEY	出错	KEY_NAME 接受 SHIFT_KEY、FUNCTION 或 KEYPAD 作为一个关键字变元。
TPU$_TOOFEW	出错	传送到 KEY_NAME 内设过程的参数太少。
TPU$_TOOMANY	出错	传送到 KEY_NAME 内设过程的参数太多。

LENGTH

这个内设过程送回一个整数, 该整数指示在一个字符串或范围内字符 或字节的数目。

格式

integer := LENGTH *(*{ *buffer* | *range* | *string* } *[,keyword])*

参数

buffer
您要知道其长度的缓冲区名。注意, 如果您指定一个缓冲区, 则行终止符不会作为字符计算。

range
您要知道其长度的范围名。注意, 如果您指定了一个范围, 则行终止符不会作为字符计算。

string
您要知道其长度的字符串。

keyword
您可以指定以下任何一个关键字:

- CHARACTERS - 送回以字符数目表示的字符串长度。
- BYTES - 送回以字节数目表示的字符串长度。

如果您不指定关键字, 默认为 CHARACTERS。您也可以用 SET(UNIT_ DEFAULT) 内设过程更改默认。有关完整描述, 请参阅 SET(UNIT_DEFAULT) 一节。

描述

有关 LENGTH 内设过程的基本描述, 请参阅《DEC Text Processing Utility Reference Manual》。

在 HTPU 中, 通过加上关键字参数, 可选择以字符或字节为单位计算来获得字符串长度的信息。

出错信号

TPU$_TOOFEW	出错	参数太少。
TPU$_TOOMANY	出错	参数太多。
TPU$_NEEDTOASSIGN	出错	LENGTH 只能用于赋值语句的右边。
TPU$_ARGMISMATCH	出错	参数的数据类型不正确。
TPU$_CONTROLC	出错	您在 LENGTH 执行时按下 CTRL/C。

MARK

这个内设过程送回一个标记, 该标记是为当前缓冲区内的当前字符位置而设定的。您必须指定标记如何在屏幕上显示 (NONE、REVERSE、BOLD、BLINK 或 UNDERLINE)。

格式

marker := MARK (*keyword* ***[,*** { *buffer* / *window* } ***[,****integer* ***[,****integer2* ***]]])***

参数

keyword
您必须使用以下其中一个关键字:

- BLINK - 指定标记以闪烁重现方式显示。
- BOLD - 指定标记以粗体重现方式显示。
- FREE CURSOR - 指定标记是一个自由标记 (即标记不以字符为边界)。指定参数 FREE_CURSOR 不建立一个自由标记, 除非在执行语句 MARK(FREE_CURSOR) 时, 光标位置在一行 行首之前、在一行结束之后、在一个跨栏标记中间, 或在缓冲区底部之下。自由标记没有影象属性。
- NONE - 指定影象属性不应用于标记上。
- REVERSE - 指定标记以反相重现方式显示。
- UNDERLINE - 指定标记加下划线。

buffer
在其内设置标记的缓冲区。按照默认, HTPU 在当前缓冲区内设置标记。

window
在其内设置标记的缓冲区的映射窗口。只当窗口已被映射到一个缓冲区, 才可指定一个窗口变量。按照默认, HTPU 在当前缓冲区内设置标记。

integer1
标记设置处的屏幕列的整数。您可从 1 到 32,767 之间指定一个整数。默认设置标记在对应当前屏幕列的缓冲区偏移处。

integer2
标记被设置的缓冲区内的行的整数。默认是在当前行内设置标记。

描述

有关 MARK 内设过程的基本描述, 请参阅《DEC Text Processing Utility Reference Manual》。

标记设定在字符边界上。因此, 如果光标是在一个多字节字符上, 在设定标记后, 即使当前编辑点是在多字节字符的第二列, 光标也会被设定在字符的第一列。

用影象属性来标记多字节字符, 会令影象重现显示在整个多字节字符上。

出错信号

TPU$_TOOFEW	出错	参数太少。
TPU$_TOOMANY	出错	参数太多。
TPU$_NEEDTOASSIGN	出错	MARK 只能用于赋值语句的右边。
TPU$_NOCURRENTBUF	警告	您必须在缓冲区内设定标记。

TPU$_INVPARAM	出错	一个或多个指定参数的类型错误。
TPU$_BADKEY	出错	关键字必须是 NONE、BOLD、BLINK、REVERSE、UNDERLINE 或 FREE_CURSOR。
TPU$_UNKKEYWORD	出错	您指定了一个未知的关键字。
TPU$_INSVIRMEM	严重出错	没有足够的内存来建立标记。

READ_CHAR

这个内设过程送回一个字符串, 该字符串包含了键盘输入的下一个字符。

格式 **string := READ_CHAR**

送回值 一个字符串类型变量, 该变量包含了一个由键盘输入的字符。

描述 有关 READ_CHAR 内设过程的基本描述, 请参阅《DEC Text Processing Utility Reference Manual》。

HTPU 已经增强, 能够启动 READ_CHAR 内设过程中多字节字符的输入。

出错信号

TPU$_TOOMANY	出错	参数太多。
TPU$_NEEDTOASSIGN	出错	READ_CHAR 只能用于赋值语句右边。

READ_KEY

这个内设过程等待您按一个键, 然后回送该键的键名。READ_KEY 也处理换码序列和控制字符。

格式 **keyword := READ_KEY**

送回值 这是送回值, 包含着刚刚所按键的键名。

描述 有关 READ_KEY 内设过程的基本描述, 请参阅《DEC Text Processing Utility Reference Manual》。

READ_KEY 已经增强, 可启动多字节字符的输入。如果输入的字符为多字节字符, 就送回该字符的关键字。

SCROLL

这个内设过程在屏幕上按您指定的行数向上或向下移动缓冲内的文本行。

格式 **[integer2 :=] SCROLL** *(window [,integer1 [,keyword]])*

参数

window
分配给您要卷动其内文本的缓冲区的窗口。

integer1
带符号 (+/-) 的整数值, 指示您要卷动文本的行数。如果您指定一个正整数, 窗口内的文本会从屏幕的底部向上卷动。如果您指定一个负整数, 窗口内的文本会从屏幕的顶部向下卷动。如果您指定 0 作为整数值, 则不进行卷动。

这个参数可以省略。如果您省略了这个参数, 文本会连续卷动,直至达到缓冲区边界或您按下一个键为止。如果缓冲区的当前方向是正向, 文本会向缓冲区底部卷动。如果缓冲区的当前方向是反向, 文本会向缓冲区顶部卷动。如果您按下一个有命令的键, 卷动会停止, 同时 HTPU 执行该键的命令。

keyword
您可以指定以下的关键字:

- CHARACTERS - 卷动后, 如果光标在一个多字节字符的非第一字节上, CHARACTERS 会把光标移位到第一列。
- BYTES - 卷动后, 光标会停留在一个字符的字节上, 这就是光标的位置, 不作任何移位。但是, 当显示光标态被启动时, 虽然内部光标位置可能在多字节字符的非第一字节上, 但

屏幕光标总是在多字节字符的字符边界上。有关详情, 请参阅关于 SET(DISPLAY_ CURSOR) 的描述。

默认为 BYTES。您也可用 SET(ALIGNMENT_DEFAULT) 内设过程更改默认。有关完整描述, 请参阅 SET(ALIGNMENT_DEFAULT) 一节。

送回值 送回值指示在使用 SCROLL 后实际卷动了的行数。

描述 有关 SCROLL 内设过程的基本描述, 请参阅《DEC Text Processing Utility Reference Manual》。

HTPU 已经增强, 可指定 BYTES 或 CHARACTERS 关键字作为第三个参数。

出错信号

TPU$_TOOFEW	出错	参数太少。
TPU$_TOOMANY	出错	参数太多。
TPU$_INVPARAM	出错	一个或多个指定参数的类型错误。
TPU$_CONTROLC	出错	您按下 CTRL/C 停止卷动。
TPU$_WINDNOTMAPPED	警告	您试图卷动一个没有映象的窗口。

SEARCH

这个内设过程寻找在一个缓冲区或范围内的特别的字符排列, 并送回包含那些字符的一个范围。

格式

[range2 :=] SEARCH *(pattern,* { *FORWARD* / *REVERSE* } *[,*

{ *EXACT* / *NO_EXACT* / *integer* } *[,*{ *buffer* / *range1* } *]])*

参数

pattern
有关详情, 请参阅《DEC Text Processing Utility Reference Manual》。

FORWARD
指示正向检索。

REVERSE
指示反向检索。

EXACT
指示 SEARCH 尝试匹配的字符的字母大小写和字形大小 (全形或半形) 必须与作为 SEARCH 第一参数的 pattern 的相同。

NO_EXACT
指示 SEARCH 尝试匹配的字符不需要大小写相同, 也不需要字形大小相同 (全形和半形字母字符作等同处理)。这是默认值。

integer
假定您希望匹配一个属性而忽略其他的属性, 这个参数指定 SEARCH 如何处理字母大小写和字形大小 (全形/半形) 的信息。 DEC 计算机公司建议您使用可供使用的预定义常数指定这个整数:

- TPU$K_SEARCH_CASE - 等效于整数 1。这指定检索要匹配第一参数的字母大小写但是对第一参数的字形大小 (全形/半形) 不敏感。
- TPU$K_SEARCH_FORM - 等效于整数 4。在 HTPU 中, 这指定检索要匹配第一参数的字形大小 (全形/半形) 但是对第一参数字符的字母大小写不敏感。

要组合关键字的作用, 您只需将预定义整数值相加。例如, 要匹配字形和字母大小写, 可指定 TPU$K_SEARCH_CASE +TPU$K_SEARCH_FORM 的值作为 "integer" 参数。

注意, 由于 HTPU 没有区分发音符敏感性, 如果单独或与 TPU$K_SEARCH_CASE 一起指定, 那么关于 TPU$K_SEARCH_DIACRITICAL 的说明就会被忽略, 而当与 TPU$K_SEARCH_FORM 一起被指定时, 就出现出错信号 TPU$MAXVALUE。

buffer
在其内检索的缓冲区。

range1
在其内检索的范围。

送回值 送回包含匹配指定模式的文本的范围。

描述 有关 SEARCH 内设过程的基本描述, 请参阅《DEC Text Processing Utility Reference Manual》。

HTPU 已增强 SEARCH 内设过程，对被检索模式的字母大小写和字形大小是可选择敏感的。

出错信号	TPU$_STRNOTFOUND	警告	对字符串或模式的检索不成功。
	TPU$_TOOFEW	出错	SEARCH 需要至少两个参数 。
	TPU$_TOOMANY	出错	SEARCH 接受不多于四个的参数。
	TPU$_ARGMISMATCH	出错	给 SEARCH 的一个参数类型错误。
	TPU$_INVPARAM	出错	给 SEARCH 的一个参数类型错误。
	TPU$_BADKEY	警告	对SEARCH指定了一个不正确的关键字。
	TPU$_MINVALUE	警告	给 SEARCH 的整数参数必须大于或等于 -1。
	TPU$_MAXVALUE	警告	给 SEARCH 的整数参数必须小于或等于 5。
	TPU$_NOCURRENTBUF	出错	如果您未指定检索的缓冲区或范围, 那么在检索前您必须定位到一个缓冲区。
	TPU$_CONTROLC	出错	在执行 SEARCH 时按下 Ctrl/C。
	TPU$_ILLPATAS	出错	给 SEARCH 的模式中包括一变量的部分模式赋值, 该变量在当前上下文中没有定义过。

SELECT

这个内设过程在当前缓冲区内的编辑点设定一个标记, 并送回该标记。您必须指定如何在屏幕上显示该标记 (NONE、REVERSE、 BOLD、BLINK 或 UNDERLINE)。

由 SELECT 送回的标记指示选择范围内的第一个字符位置。您为标记指定的影象属性将影响选择范围内的所有字符。有关建立选择范围的资料, 请参阅 SELECT_RANGE 内设过程的描述。

格式

marker := SELECT *(keyword)*

参数

keyword
这个关键字指定送回的标记如何在屏幕上显示。您必须使用以下其中一个关键字:

- BLINK - 指定标记以闪烁重现方式显示。
- BOLD - 指定标记以粗体重现方式显示。
- NONE - 指定没有属性应用于标记上。
- REVERSE - 指定标记以反相重现方式显示。
- UNDERLINE - 指定标记须作下划线。

描述

SELECT 送回一个建立选择范围开始处的特别标记。 该标记位于执行内设过程 SELECT 时编辑点的字符位置上。如果编辑点是在一个多字节字符的非第一列, 该标记会被置于上一个字符和该多字节字符的第一列之间。

出错信号	TPU$_ONESELECT	警告	SELECT 在当前缓冲区内已是活动的。
	TPU$_TOOFEW	出错	参数太少。
	TPU$_TOOMANY	出错	参数太多。
	TPU$_NEEDTOASSIGN	出错	SELECT 只能用于赋值语句的右边。
	TPU$_NOCURRENTBUF	警告	使用 SELECT 前, 您的位置必须在缓冲区内。
	TPU$_INVPARAM	出错	一个或多个指定参数的类型错误。
	TPU$_BADKEY	出错	您给 SELECT 指定了错误的关键字。

SELECT_RANGE

这个内设过程送回一个范围, 该范围包含了由 SELECT 内设过程设定的标记和当前编辑位置之间的所有字符。执行 SELECT_RANGE 时, 选定范围的最后位置不包含当前字符。

格式 **range := SELECT_RANGE**

描述 有关 SELECT_RANGE 内设过程的基本描述, 请参阅《 DEC Text Processing Utility Reference Manual》。

执行 SELECT_RANGE 时, 当前编辑点在一个多字节字符非第一列上, 与它在第一列的作用相同。

出错信号

TPU$_NOSELECT	警告	当前缓冲区中没有活动的选择范围。
TPU$_SELRANGEZERO	警告	选择范围内没有选定的字符。
TPU$_TOOMANY	出错	参数太多。
TPU$_NEEDTOASSIGN	出错	SELECT_RANGE 只能用于赋值语句 的右边。
TPU$_NOCURRENTBUF	警告	没有当前缓冲区。

SET(ALIGNMENT_DEFAULT)

格式

SET *(ALIGNMENT_DEFAULT, string,* { *BYTES* / *CHARACTERS* } *)*

参数

ALIGNMENT_DEFAULT
这是一个关键字, 指示 SET 为特定的内设过程定义光标对齐特征。

string
这是一个字符串, 包含设定 ALIGNMENT_DEFAULT 的内设过程的名称。有效内设过程是 CURSOR_HORIZONTAL、CURSOR_VERTICAL 和 SCROLL。

BYTES
这是一个关键字, 指定光标按字节单位计对齐。

CHARACTERS
这是一个关键字, 指定光标按字符单位计对齐。

描述

SET(ALIGNMENT_DEFAULT) 指定默认值是否把光标对齐到多字节字符的第一列。CURSOR_HORIZONTAL, CURSOR_VERTICAL 和 SCROLL 的 ALIGNMENT_DEFAULT 是 BYTES。那些内设过程不会对齐光标。

可参阅 CURSOR_HORIZONTAL、CURSOR_VERTICAL 和 SCROLL。

出错信号	TPU$_TOOFEW	出错	参数太少。
	TPU$_TOOMAN	出错	参数太多。
	TPU$_INVPARA	出错	参数类型不正确。

SET(DISPLAY_CURSOR)

格式

[keyword :=] SET *(DISPLAY_CURSOR,* { *ON* / *OFF* } *)*

参数

DISPLAY_CURSOR
指示 SET 更改 HTPU 的显示光标态的关键字。

ON
指定启动显示光标态的关键字。

OFF
指定停掉显示光标态的关键字。

送回值 送回值指示应用这个内设过程前 HTPU 的显示光标态。

描述

这个内设过程启动或停掉 HTPU 的显示光标态, 并且可选择地得到显示光标态的当前设置。

在显示光标态是 ON 的情况下, 如果光标当前位于多字节字符上, 您在屏幕上看见的光标 (称为屏幕光标), 与存放在 HTPU 内的那个光标 (称为内部光标) 可能不同。

例如, 当内部光标位于多字节字符的第二字节上时, 在显示光标态是 OFF 的情况下, 屏幕光标将位于该字符的第二列; 而在显示光标态是 ON 的情况下, 屏幕光标将位于该字符的第一列。

显示光标态总是使屏幕光标位于字符的边界。注意, 当显示光标态是 ON 而内部光标位于多字节字符的非第一字节上时, 对 HTPU 的字符插入和重键将假设内部光标是在屏幕光标的位置上, 而后者总是在字符的边界上的。

在 HTPU 启动时显示光标态默认为 OFF, 通过 HEVE 启动过程设置 到 ON。

SET(FILL_NOT_BEGIN)

格式 **SET** *(FILL_NOT_BEGIN, string)*

参数

FILL_NOT_BEGIN
这是一个关键字, 指示 SET 定义异常字符, 后者在执行 FILL 和 JUSTIFY 命令时不能出现在行首。

string
这是一个字符串, 包含着不能出现在行首的异常字符。

描述

SET(FILL_NOT_BEGIN) 为 FILL 和 JUSTIFY 内设过程指定 FILL_ NOT_BEGIN 字符。FILL 和 JUSTIFY 确定行分开的位置, 这样, FILL_NOT_BEGIN 字符就不会出现在行首。

可参阅 FILL、JUSTIFY、SET(FILL_NOT_END) 和 SET(MARGIN_ALLOWANCE)。

出错信号

TPU$_TOOFEW	出错	参数太少。
TPU$_TOOMANY	出错	参数太多。
TPU$_INVPARAM	出错	参数类型不正确。

SET(FILL_NOT_END)

格式 **SET** *(FILL_NOT_END, string)*

参数

FILL_NOT_END
这是一个关键字, 指示 SET 定义异常字符, 后者在执行 FILL 和 JUSTIFY 命令时不能出现在行尾。

string
这是一个字符串, 包含着不能出现在行尾的异常字符。

描述

SET(FILL_NOT_END) 为 FILL 和 JUSTIFY 内设过程指定 FILL_NOT_ END 字符。FILL 和 JUSTIFY 确定行分开的位置, 这样, FILL_NOT_END 字符就不会出现在行尾。

可参阅 FILL、JUSTIFY、SET(FILL_NOT_BEGIN) 和 SET(MARGIN_ALLOWANCE)。

出错信号

TPU$_TOOFEW	出错	参数太少。
TPU$_TOOMANY	出错	参数太多。
TPU$_INVPARAM	出错	参数类型不正确。

SET(MARGIN_ALLOWANCE)

格式 **SET** *(MARGIN_ALLOWANCE, integer)*

参数

MARGIN_ALLOWANCE
这是一个关键字, 指示 SET 定义在执行 FILL 命令时可出现在右边线以外的异常字符的数目。

integer
可出现在右边线以外的异常字符的数目。

描述

SET(MARGIN_ALLOWANCE) 指定可出现在右边线以外的异常字符数目。FILL 内设过程填补估计到异常字符 (FILL_NOT_BEGIN 和 FILL_NOT_END 字符) 的行。

默认值是 0。当值为 0 时, FILL_NOT_BEGIN 字符不能出现在右边线以外。要防止 FILL_NOT_BEGIN 字符出现在一行的开头, 须将前一行末的字符移至当前行首。因此, 可能出现在右边线前没有填满一行的情况。

可参阅 FILL、SET(FILL_NOT_BEGIN) 和 SET(FILL_NOT_END)。

出错信号

TPU$_TOOFEW	出错	参数太少。
TPU$_TOOMAN	出错	参数太多。
TPU$_INVPARAM	出错	参数类型不正确。

SET(UNIT_DEFAULT)

格式

SET *(UNIT_DEFAULT,string,* { *BYTES* / *CHARACTERS* } *)*

参数

UNIT_DEFAULT
这是一个关键字, 指示 SET 定义特定的内设过程的计算单位。

string
这是一个字符串, 包含着设定 UNIT_DEFAULT 的内设过程名。有效的内设过程是 CURRENT_OFFSET、INDEX、LENGTH 和 SUBSTR。

BYTES
这是一个关键字, 指定按字节数的计算单位。

CHARACTERS
这是一个关键字, 指定按字符数的计算单位。

描述

SET(UNIT_DEFAULT) 指定默认计算单位为字节或字符。CURRENT_OFFSET, INDEX, LENGTH 和 SUBSTR 的 UNIT_DEFAULT 是 CHARACTERS。

可参阅 CURRENT_OFFSET、INDEX、LENGTH 和 SUBSTR。

出错信号	TPU$_TOOFEW	出错	变元太少。
	TPU$_TOOMAN	出错	参数太多。
	TPU$_INVPARA	出错	变元类型不正确。

SPLIT_LINE

这个内设过程在当前编辑点的位置把当前行分为两行。

格式 **SPLIT_LINE**

描述 有关 SPLIT_LINE 内设过程的基本描述, 请参阅《DEC Text Processing Utility Reference Manual》。

如果当前编辑位置不在一个多字节字符的第一字节上, SPLIT_LINE 分行的方法与编辑位置在字符的第一字节的方法相同。执行 SPLIT_LINE 后, 编辑位置移到该字符的第一字节。

出错信号

TPU$_NOCURRENTBUF	警告	您的位置不在缓冲区内。
TPU$_TOOMANY	出错	参数太多。
TPU$_NOTMODIFIABLE	警告	您不能修改一个不可修改的缓冲区。
TPU$_NOCACHE	出错	没有足够内存来分配一个新的高速缓存器。

SUBSTR

这个内设过程在一个字符串或范围内建立一个子串, 并送回表示该子串的字符串。

格式

string1 := SUBSTR ({ *buffer* / *range* / *string* } **,***integer1,integer2 [,*
keyword] **)**

参数

buffer
包含该子串的缓冲区。

range
包含该子串的范围。

string
包含该子串的字符串。

integer1
子串开始的字符或字节位置。第一个字符位置为 1。

integer2
包括在子串中的字符或字节数目。

keyword
您可以使用以下的关键字:

- CHARACTERS - 使用第二和第三个参数作为字符数目。
- BYTES - 使用第二和第三个参数作为字节数目。

如果您没有指定一个关键字, 默认为 CHARACTERS。您也可用 SET (UNIT_DEFAULT) 内设过程更改默认。有关完整描述, 请参阅 SET(UNIT_DEFAULT) 一节。

描述

有关 SUBSTR 内设过程的基本描述, 请参阅《DEC Text Processing Utility Reference Manual》。

如果您指定 BYTES 作为关键字, 包含多字节字符的字符串中的子串会根据指定的字节数目来构成。如果要抽取的部分是在一个多字节字符的中间, 子串的不完整多字节字符会由空格替换。

出错信号

TPU$_TOOFEW	出错	参数太少。
TPU$_TOOMANY	出错	参数太多。
TPU$_NEEDTOASSIGN	出错	SUBSTR 只能用于赋值语句的右边。
TPU$_INVPARAM	出错	一个或多个指定参数的类型错误。
TPU$_ARGMISMATCH	出错	SUBSTR 的其中一个参数类型错误。
TPU$_STRTOOLARGE	出错	您指定的长度超过允许的最大字符串长度。

第 3 章

新的和已修改的 HEVE 命令

本章叙述新的或从 EVE (可扩充通用编辑程序) 命令增强的 HEVE 命令的参考资料, 应作为《Extensible Versatile Editor Reference Manual》的补充。

本章中的命令描述按字母顺序排列。本章不包括与汉字字符使用无关的命令。有关详情, 请参阅《Extensible Versatile Editor Reference Manual》。注意, 有些没有包括在本章内的命令也能处理汉字字符。对于这些命令, 带有中文字符的操作与带有 ASCII 字符的操作相同。关于如何使用 HEVE 命令来进行编辑, 可参阅《HEVE 用户手册》。

下表列出本章包括的命令的名称。

表 3-1 新的和已修改的 HEVE 命令

命令
DELETE
DRAW BOX
DRAW LINE
ERASE CHARACTER
ERASE PREVIOUS WORD
ERASE WORD
EXTEND HEVE

表 **3-1** 新的和已修改的 **HEVE** 命令 (续)

命令
EXTEND HTPU
FILL/FILL PARAGRAPH/FILL RANGE
FIND
FULLFORM WORD
HALFFORM WORD
HTPU
LEFT ADJUST
LEFT INDENT
LOWERCASE WORD
MOVE BY WORD
MOVE DOWN/UP
MOVE LEFT/RIGHT
REPLACE
RIGHT ADJUST
RIGHT INDENT
SAVE EXTENDED HEVE

表 3-1 新的和已修改的 HEVE 命令 (续)

命令
SAVE EXTENDED HTPU
SET TABS
SET [NO]DISPLAY CURSOR
SET FIND CASE [NO]EXACT
SET FIND EXACT
SET FIND FORM [NO]EXACT
SET FIND GENERAL
SPECIAL INSERT
SYMBOL
UPPERCASE WORD

有些命令与受支持的终端上的一组键相关联。每一个命令描述的格式一节列出与命令相关联的键。例如, VT300/VT200 一栏指出相应的命令与所有属于 VT300 和 VT200 系列的终端上下列键相关联。VT382 和 VT82 这两个受支持的汉字终端分别归 入 VT300 和 VT100 系列内。

DELETE

格式 **DELETE**

VT300,VT200	VT100
<X>	*DELETE*

描述 删除光标左边的字符。在插入态中, 该行其余部分向左移一个字符来覆盖该空格。在重写态中, 删除的字符由等于该字符的屏幕列宽的空格数替换, 亦即单字节字符为一列, 双字节字符为两列, 等等。在一行行首, 此功能删除回车, 并把当前行移到上一行。

要恢复已删除的内容, 可以使用 RESTORE CHARACTER 命令。

DRAW BOX

格式 **DRAW BOX**

描述 调用 HEVE 画框特性。此命令与使用 DRAW LINE 命令通过按 REMOVE 而进入画框态一样。有关详情, 请参阅 DRAW LINE。

DRAW LINE

格式 **DRAW LINE**

描述 调用 HEVE 画线和画框特性。此特性提供简易及用户方便接口。线条和框子会随光标移动而显示。光标用上、下、左、右箭头键控制。该画线功能是受简便的转换键控制的。按控制键时, 画线状态即行更新。画线和画框的步骤如下:

1 按 DO 键, 得命令提示。

2 在命令提示处, 键入 DRAW LINE。HEVE 会进入画线态。

3 在画线态中, 您可用上、下、左、右箭头键移动光标画线或擦线。表 3-2 列出画线的所有控制键。

- 按 SELECT, 选择画线态或移动光标态。在移动光标态中, 光标可自由移动, 不会影响屏幕上的数据。
- 按 INSERT, 选择粗线、细线或擦除线。擦除线于光标沿线移动时擦除画线符。
- 按 REMOVE 进入画框态。控制键在画框态中的功能与在画线态中的不同。
- 按 RETURN 退出画线态。

4 在画框态中, 您可用上、下、左、右箭头键移动光标来画框。表 3-3 列出所有画线控制键。

- 按 SELECT 退出画框态并返回画线态。
- 按 REMOVE 取消画框态并返回画线态。

- 按 RETURN 退出画线态和画框态。

按 DO 键也能退出画线态和画框态。

画线符总是向屏幕奇数列对齐。每个画线符占据两个屏幕列。

下列表概括画线态和画框态的控制键:

表 3-2 画线态中的控制键

功能	转换键
画线/移动光标	SELECT 或 KP1
使线加粗/使线变细/擦除线	INSERT 或 KP2
进入画框态	REMOVE 或 KP3
退出画线态	RETURN

表 3-3 画框态中的控制

功能	转换键
退出画框态	SELECT 或 KP1
取消画框态	REMOVE 或 KP3
退出画框和画线态	RETURN

有关如何使用 DRAW LINE 和 DRAW BOX 的例子, 请参阅《HEVE 用户手册》第5章。

ERASE CHARACTER

格式 **ERASE CHARACTER**

描述 删除光标所在位置的字符。在插入态中, 行的其余部分左移一个字符来覆盖该空格。在重写态中, 删除的字符由等于该字符的屏幕列宽的空格数替换, 亦即单字节字符由一个空格替换,双字节字符则由两个空格替换, 等等。在一行结尾, 此功能删除回车, 并把下一行上移到当前行。

要恢复已删除的内容, 可以使用 RESTORE CHARACTER 命令。

ERASE PREVIOUS WORD

格式　　ERASE PREVIOUS WORD

描述

擦除前一个字或光标所在位置的字。如果光标在两个字之间或在一个字的开首位置, 此功能会擦除前一个字。如果光标在一个字的中间, 此功能会擦除该字及其尾随的空格。如果光标在一行的行首, 此功能只擦除上一行的回车, 并把当前行向上移。

此功能处理多字节中文数据的方法与单字节数据相同, 只是 "字" 的定义更改了 (一个由中文字符组成的字是一系列的同类字符) (参阅 HTPU 的 CHARACTER_CLASS)。如果当前字符的字符类别与前一个字符不同, 或已到达行尾时, 此功能会假设找到一个字的结尾。

要恢复已擦除的内容, 可以使用 RESTORE 或 RESTORE WORD 功能。

ERASE WORD

格式 **ERASE WORD**

VT300,VT200	**VT100**
CTRL/J	*CTRL/J* 小键盘 *COMMA*

描述

擦除光标所在位置的字。如果光标在两个字之间, 此功能擦除下一个字及其尾随的空格。如果在一行的行末, 此功能只擦除当前行的回车，并把下一行上移。

此功能处理多字节中文数据的方法与单字节数据相同, 只是"字" 的定义更改了 (一个由中文字符组成的字是一系列的同类字符) (参阅 HTPU 的 CHARACTER_CLASS)。如果当前字符的字符类别与前一个字符不同, 或已到达行尾时, 此功能会假设找到一个字的结尾。

要恢复已擦除的内容, 可以使用 RESTORE 或 RESTORE WORD 功能。

EXTEND HEVE

格式

EXTEND HEVE *(procedure-name/*)*

描述

编译一个或多个 HTPU 过程来扩充 HEVE。EXTEND HTPU 是此功能的同义词。此功能替换 EVE 中的 EXTEND EVE 功能。

参数

procedure-name

要编译的 HEVE 过程名。您可以缩写过程名。如果一个以上的名字与请求匹配, HEVE 会列出匹配名, 并再调用 EXTEND HEVE 命令, 使您能够选择所需要的过程。

***

通配符指示 HEVE 编译缓冲区内所有的过程和语句。这个参数产生的结果与 EXTEND ALL 命令产生的相同。

例子

```
Command: EXTEND HEVE USER_PROC
```

上述例子编译当前缓冲区内一个称为 USER_PROC 的过程。

EXTEND HTPU

格式

EXTEND HTPU *(procedure-name/*)*

描述

编译一个或多个 HTPU 过程来扩充 HEVE。EXTEND HEVE 是此功能的同义词。此功能替换 EXE 中的 EXTEND TPU 功能。

参数

procedure-name

要编译的 HEVE 过程名。您可以缩写过程名。如果一个以上的名字与请求匹配, HEVE 会列出匹配名, 并再调用 EXTEND HTPU 命令, 使您能够选择所需要的过程。

*****

通配符指示 HEVE 编译缓冲区内所有的过程和语句。这个参数产生的结果与 EXTEND ALL 命令产生的相同。

例子

```
Command: EXTEND HTPU USER_PROC
```

上述例子编译当前缓冲区内一个称为 USER_PROC 的过程。

FILL/FILL PARAGRAPH/FILL RANGE

格式　　FILL/FILL PARAGRAPH/FILL RANGE

描述

根据当前缓冲区的边线把当前段或选定的范围重新格式化, 这样一行内便可放入最多的字。如使用 FILL RANGE, 必须选择一个范围。要使用这个功能:

- 任选地使用 SELECT 突出显示 FILL 要使用的文本。如果使用FILL RANGE, 必须选择一个范围。
- 使用 FILL。突出显示消除, 且光标移到范围的末尾。

如果您不选择任何文本, FILL 会使用当前段。段由以下任何一项来定界:

- 空白行。
- 缓冲区顶部或底部。
- 分页。
- DIGITAL Standard Runoff 软件命令。

FILL 除去段落或范围开始和结尾位置的跨栏标记和空格, 但不会影响文本内的跨栏标记和空格。

处理多字节字符时, 如果行末只有一个空列而下一个字符是多字节字符, FILL 可能在该行末留下一个空格。FILL 不会把双字节空格字符作为真正的空格, 因此不会用它来分隔字。

FIND

格式 **FIND** *(search-string)*

VT300,VT200	VT100
FIND	*PF1*

描述

在当前缓冲区内检索指定的字符串。如果找到指定的字符串, HEVE 会把文本突出显示, 并把光标放在该字符串的开始位置。

要检索上次使用的字符串, 可以按 FIND 键两次。如果您按 FIND 键一次, 然后没有键入字符串便按 RETURN, HEVE 不会执行 FIND 命令。

按 RETURN 键开始按当前方向进行检索。要按一个特定方向开始检索, 可按一个方向设置键。例如, 如果您按 F11 终止该功能, 就会按缓冲区当前方向相反的方向开始进行检索。如果您按一个定义为 FORWARD 的键, 则无论缓冲区方向是向前或向后, 检索都会开始向前进行。

此功能首先按您指定的方向在缓冲区检索, 如果找不到指定的字符串, FIND 会按相反方向检索。如果找到指定的字符串, 提示会问您是否要到该字符串位置。如果您要到该位置, 按 RETURN, 否则键入 NO 并按 RETURN 以结束检索。

与选定的范围一样, 找到的字符串会被突出显示。您可以使用 REMOVE、STORE TEXT、LOWERCASE WORD, 或其他可以应用于选定范围的功能。离开突出显示的字符串会消除突出显示。

FIND 已经增强能够查找包含多字节字符的字符串。您检索的字符串可以包含中文字符以及单字节 ASCII 字符。此外, 依赖于 SET FIND 命令的使用, 在您为检索字符串适当地指定字母大小写和字形大小的情况下, 在执行 FIND 命令期间, 字母大小写敏感和字形大小敏感可以启 动或撤消, 如下表所示:

表 3-4 FIND 的字母大小写和字形大小敏感

字母大小写准确	字形大小准确	检索字符串	效果
NO	NO	全部小写	字母大小写和字形大小不敏感
Yes	NO	任意	字母大小写敏感, 字形大小不敏感
NO	Yes	全部小写	字母大小写不敏感, 字形大小敏感
Yes	Yes	任意	字母大小写和字形大小皆不敏感

亦可参阅 SET FIND 命令的描述。

参数

search_string
您要查找的文本字符串。如果您要匹配字母的出现而不管大小写和字形大小, 可使用小写 ASCII 字母。要以准确的大小写和字形大小匹配字符串, 需指定一些不是小写 ASCII 的字母, 或使用 SET FIND 命令。有关详情, 请参阅 SET FIND 命令。

FULLFORM WORD

格式 **FULLFORM WORD**

描述 把当前字或突出显示文本中的全部单字节 ASCII 字符转换为全形字符。要 使用这个功能:

- 任选地使用 SELECT 键或 FIND 键来突出显示文本。
- 使用 FULLFORM WORD。突出显示取消, 且下面如果还有字, 光标会移到下一个字的开始位置上。

如果您没有突出显示任何文本, 此功能会把当前字的字母转换为全形字符; 如果光标在两个字之间, 随后的字将受到影响。

HALFFORM WORD

格式 **HALFFORM WORD**

描述 把当前字或突出显示文本中的全部全形字符转换为半形字符 (即是 ASCII 字母)。要使用这个功能:

- 任选地使用 SELECT 键或 FIND 键来突出显示该文本
- 使用 HALFFORM WORD。突出显示取消, 且下面如果还有字, 光标会移到下一个字的开始位置上。

如果您没有突出显示任何文本, 此功能会把当前字的字母转换为半形字符; 如果光标在两个字之间, 随后的字将受到影响。

HTPU

格式　　**HTPU** *(procedure-name/statement)*

描述　　在编辑对话期间执行一个 HTPU 过程或语句。此功能与 EVE 中的 TPU 功能相同。

参数　　***procedure-name/statement***
要执行的 HTPU 过程或语句的名称。

例子

```
Command: HTPU COPY_TEXT(FAO("!%D",0));
```

上述例子执行 COPY_TEXT 内设过程, 以使用 FAO 内设过程输入当前日期和时间。

LEFT ADJUST

格式 **LEFT ADJUST**

描述 移动部分或全部包含在当前选择区的文本行, 通过调整行的前导空格, 将其与当前缓冲区的左边线对准。

这一命令与 HEDT 的 ADJL 命令等效。

LEFT INDENT

格式 **LEFT INDENT**

描述 重新格式化当前选定范围, 使得最大字数装在一行上, 而每一行内的字符会在该范围的开始列开始, 而不超过缓冲区的右边线。在选定范围的每一行, 除第一行外, 在使用 LEFT INDENT 后会获得一个关于该范围 开始列的左边线值。

要使用这个功能:

- 使您要 LEFT INDENT 的文本的范围在您要求的列数开始
- 选定文本的范围
- 应用 LEFT INDENT 命令

有关详情, 请参阅《HEVE 用户手册》 第 3 章。

LOWERCASE WORD

格式 **LOWERCASE WORD**

描述 把当前字或突出显示的文本改变为小写。要使用这个功能:

- 任选地使用 SELECT 键或 FIND 键来突出显示文本。
- 使用 LOWERCASE WORD。突出显示取消, 且下面如果还有字, 光标会移到下一个字的开始位置上。

如果您没有突出显示任何文本, 此功能会把当前字改变为小写; 如果光标在两个字之间, 此功能会把下一个字改变为小写。

此功能不会改变中文字符, 但属于 FULL_FORM 类别的字符则除外。此功能把大写格式的 FULL_FORM 字符转换为小写 FULL_FORM 格式(请参阅 CHARACTER_CLASS 内设过程)。

MOVE BY WORD

格式 **MOVE BY WORD**

描述 把光标按当前方向每次移动一个字。如果以向前方向移动, 光标移到下一个字的开始位置。如果以向后方向移动, 光标移到当前字的开始位置; 如果已在当前字的开始位置, 则移到上一个字的开始位置。注意, 中文的一个字是一系列定义为相同类别的字符 (请参考 CHARACTER_CLASS 内设过程)。因此, 使光标每次移动一个字会把光标移到不同类别的新字符上, 如果行上没有不同类别的新字符, 则移到该行的行尾。

MOVE DOWN/UP

格式 **MOVE DOWN/UP**

描述 把光标向上或向下每次移动一行, 并保持光标在当前字符的第一列上。

当显示光标态设置为 ON 时 (HEVE 默认), 内部光标列位置不会被这个命令更改。但是, 当显示光标态设置为 OFF 时, MOVE 命令会把内部光标移到每个字符的第一列, 当连续使用多个 MOVE 命令时, 光标会逐渐移到左边。有关详情, 请参阅 SET [NO]DISPLAY CURSOR。

MOVE LEFT/RIGHT

格式 **MOVE LEFT/RIGHT**

描述 把光标按命令指定的方向每次移动一个字符。如果光标是自由光标, 您可以在缓冲区内任何地方移动光标, 不论缓冲区内有没有字符。如果光标是约束光标, 且方向是向左, 则光标会从一行的行首移到上一行的行尾。如果光标在一行的行尾, 且光标是约束光标, 而方向如果是向右, 则光标会移到下一行的行首。光标总是移到中文字符的第一个字节上。

REPLACE

格式 **REPLACE** *(old-string new-string/"old-string" "new-string')*

描述 用另一个文本字符串替换一个文本字符串。HEVE 检索旧字符串, 如果找到旧字符串, 便突出显示, 并要求下一个动作:

- YES - 替换找到的字符串, 并查找下一个。
- NO - 跳过找到的字符串, 并查找下一个。
- ALL - 替换所有出现的旧字符串 (不再有提示)。
- LAST - 替换找到的字符串, 并停止。
- QUIT - 跳过找到的字符串, 并停止。

对于 YES 和 ALL, 如果检索缓冲区多于一次, 此功能会问您是否要继续。操作完成后, 此功能会在屏幕底部显示所作替换的数目。

REPLACE 命令首先使用 FIND 来检索要替换的字符串。SET FIND 命令对 REPLACE 的作用与对 FIND 的作用相同。有关详情, 请参阅 FIND 命令。

当检索到目标字符串时, 如果被检索的字符串属于下列的一种类型以及新字符串的字母数字字符全部是小写和半形的, 则替换将根据被检索字符串的大小写和字形大小进行:

- 全部字母数字字符是小写、半形
- 全部字母数字字符是大写、半形

- 全部字母数字字符是小写、全形
- 全部字母数字字符是大写、全形
- 除第一个字符可以是大写和半形外, 其余全部字母数字字符是小写、 半形
- 除第一个字符可以是大写和全形外, 其余全部字母数字字符是小写、 全形

HEVE 把替换字符串放入当前缓冲区中, 并保存被替换字符串的大小写和字形大小, 否则, 替换字符串的大小写和字形大小将完全与您在新字符串参数中提供的相同。请参阅 SET FIND 命令的描述。

参数

old-string
您要删除的文本。该文本可以包含单字节和多字节字符。

new-string
您要用来替换旧字符串的文本。该文本可以包含单字节和多字节字符。

例子

```
1     Command: REPLACE Wrod Word
      Replace? Type yes, no, all, last, or quit: ALL
      Replaced 8 occurrences
```

上述例子用字符串 "Word" 替换所有出现的字符串 "Wrod"。有八个字符串被替换。

要替换短语 (几个字), 可以用引号把字符串括起来, 如下例所示:

```
2     Command: REPLACE "TEXT-EDIT PROCESS" "TEXT PROCESSING"
```

RIGHT ADJUST

格式 **RIGHT ADJUST**

描述 移动部分或全部包含在当前选定范围的文本行, 通过插入前导空格到行中, 将其与右边线对齐。

这个命令与 HEDT 的 ADJR 命令等效。

RIGHT INDENT

格式 **RIGHT INDENT**

描述 重新格式化当前选定范围, 使得最大字数放在一行上, 而每一行内的字符会在该范围的开始列开始, 而不超过缓冲区的右边线。此外, RIGHT INDENT 通过插入空格, 向右对齐每一受影响的行, 惟最后一行除外。在使用 RIGHT INDENT 之后, 选定范围内的每一行获得一个该范围开始列的左边线值, 而行尾在缓冲区的右边线。

这个命令与 HEDT 的 NDTR 命令等效。

SAVE EXTENDED HEVE

格式

SAVE EXTENDED HEVE *(section-filespec)*

描述

把当前键定义及其他扩充保存在节文件内, 以供日后的编辑对话期 使用。SAVE EXTENDED HTPU 是这个命令的同义词。

节文件是累加的。这个命令把一个编辑对话期间所作的任何键定义或扩充保存起来, 其中包括那些已经保存在您使用的节文件内的键定义及扩充。因此建立一个节文件是建立您自己的 HEVE 用户定制版本的一个方法。不过, 节文件通常不保存您的边线设置、光标设置或跨栏标记停止列。您可以用一个初始化文件来为这些设置设定专用默认。

参数

section-filespec
您要建立的节文件。默认文件类型是 TPU$SECTION。

例子

```
Command: SAVE EXTENDED HEVE sys$login:myeve
```

上述例子建立一个称为 MYEVE.TPU$SECTION 的节文件。

SAVE EXTENDED HTPU

格式

SAVE EXTENDED HTPU *(section-filespec)*

描述

把当前键定义及其他扩充保存在节文件内, 以供日后的编辑对话期使用。SAVE EXTENDED HEVE 是这个命令的同义词。

节文件是累加的。这个命令把一个编辑对话期间所作的任何键定义或扩充保存起来, 其中包括那些已经保存在您使用的节文件内的键定义及扩充。因此建立一个节文件是建立您自己的 HEVE 用户定制版本的一个方法。不过, 节文件通常不保存您的边线设置、光标设置或跨栏标记停止列。您可以用一个初始化文件来为这些设置设定专用默认。

参数

section-filespec
您要建立的节文件。默认文件类型是 TPU$SECTION。

例子

```
Command: SAVE EXTENDED HTPU sys$login:myeve
```

上述例子建立一个称为 MYEVE.TPU$SECTION 的节文件。

SET [NO]DISPLAY CURSOR

格式　　SET [NO]DISPLAY CURSOR

描述

这个命令启动 (ON) 或停掉 (OFF) HTPU 的显示光标态。

在显示光标态是 ON 的情况下, 如果光标当前位于多字节字符上, 您在屏幕上看见的光标 (称为屏幕光标), 与存放在 HTPU 内的那 个光标 (称为内部光标) 可能不同。

例如, 当内部光标位于多字节字符的第二字节上时, 在显示光标态是 OFF 的情况下, 屏幕光标将位于该字符的第二列; 而在显示光标态是 ON 的情况下, 屏幕光标将位于该字符的第一列。

显示光标态总是使屏幕光标位于字符的边界。注意, 当显示光标态是 ON 而内部光标位于多字节字符的非第一字节上时, 对 HTPU/HEVE 的字符插入和重键将假设内部光标是在屏幕光标的位置上, 而后者总是在字符的边界上的。

默认是 SET DISPLAY CURSOR。

SET FIND CASE [NO]EXACT

格式 **SET FIND CASE [NO]EXACT**

描述 当要查找或替换的字符串中的全部字母均为字母小写时, 不论它们的字形大小, 这个命令设置 FIND 和 REPLACE 命令的字母大小写敏感。字形大小敏感由 SET FIND FORM [NO]EXACT 命令确定。

SET FIND CASE EXACT 启动字母大小写敏感。

SET FIND CASE NOEXACT 停掉字母大小写敏感。

默认是 SET FIND CASE NOEXACT。

SET FIND FORM [NO]EXACT

格式 **SET FIND FORM [NO]EXACT**

描述 这个命令设置 FIND 和 REPLACE 命令的字形大小敏感 (即它是否识别全形和半形字符)。字母大小写敏感由 SET FIND CASE [NO]EXACT 命令确定。

SET FIND FORM EXACT 启动字形大小敏感。

SET FIND FORM NOEXACT 停掉字形大小敏感。

默认是 SET FIND FORM NOEXACT。

SET FIND GENERAL

格式 **SET FIND GENERAL**

描述 这个命令等于 SET FIND CASE NOEXACT 与 SET FIND FORM NOEXACT 合起来。

请参阅 SET FIND CASE [NO]EXACT 和 SET FIND FORM [NO]EXACT 命令。

SET FIND EXACT

格式 **SET FIND EXACT**

描述 这个命令等于 SET FIND CASE EXACT 与 SET FIND FORM EXACT合起来。

请参阅 SET FIND CASE [NO]EXACT 和 SET FIND FORM [NO]EXACT 命令。

SET TABS (MOVEMENT)

格式 **SET TABS** *(MOVEMENT)*

描述 设定 TAB 键特性, 使在按下 TAB 键时, 光标会移到下一个跨栏标记停止列。如果下一个跨栏标记停止列位于一个多字节字符中, 光标会移到跨栏标记停止列之后的字符的开始位置。这样保证光标在一个字符的开始位置，而不会在多字节字符的其他位置。

SPECIAL INSERT

格式 **GOLD/[integer]/GOLD + KP3**

描述 用一个字符的 ASCII 值插入该字符。此功能已经修改, 使不能输入大于 127 的值。这样避免把一个多字节字符的一部分输入文本中。

例子

```
Command: SET KEYPAD EDT
Press GOLD key; enter 10;press GOLD key and follow by KP3
```

上述例子把换行字符输入缓冲区中的当前光标位置。

SYMBOL

格式

SYMBOL *(text-string)*

描述

此功能只在 HEVE 中出现, 利用运行时间库 HSYLIB 提供的 JSY$TRA_SYMBOL 功能来画线和框。如果没有为功能调用提供一个字符串, 此功能会显示中文所有可用的符号, 这些符号您不容易通过键盘选择存取。每一个符号都映射一个 ASCII 字符的特殊序列。通过把一些特殊序列输入功能的提示, 此功能会把相应的特殊符号插入当前编辑位置的文本中。

对于不正确的特殊序列, 此功能不会进行任何翻译, 而只把那些序列插入当前编辑位置的文本中。

参数

text-string
这是一个文本字符串, 包含着获得所需的符号的特殊序列。如果任何特殊序列有任何错误, 该序列不会被翻译, 而会随着已翻译的符号插入文本中。

例子

```
Symbols: 7------------------9
```

上述例子用 40 列宽的长度画出了一个矩形框的顶部。

UPPERCASE WORD

格式 **UPPERCASE WORD**

描述 把当前字或突出显示的文本改为大写。要使用此功能:

- 任选地使用 SELECT 键或 FIND 键来突出显示文本。
- 使用 UPPERCASE WORD。突出显示取消了。如果随后还有字，光标 会移到下一个字的开始位置。

果您没有突出显示任何文本, 此功能会把当前字改为大写;如果光标在两个字之间, 下一个字会改成大写。

通常, 此功能不改变中文字符, 但属于 FULL_FORM 类别的字符则除外。此功能把小写格式的 FULL_FORM 字符转换为大写 FULL_FORM 格式 (请参阅 CHARACTER_CLASS 内设过程)。

附录 A

HTPU/HEVE 的限制

在 ASCII 内设过程中, 如果指定的整数值参数在 161 和 255 之间, 在 DECTPU 中 ASCII 会送回一个多国字符集的字符, 但在 HTPU 则不同。HTPU 把送回的字符作为双字节字符的一部分处理。因此, 用户最好不要使用如 COPY_TEXT 等命令来把 ASCII 内设过程送回的结果字符显示在屏幕上。

HTPU 中的 FAO 内设过程不能处理多字节字符。如果 FAO 的字符串参数包含多字节字符, 结果是不能预料的。

HTPU 中的 READ_LINE 内设过程不能正确地阅读多字节字符。如果有多字节字符输入, READ_LINE 会终止, 并送回终止前输入的字符。该多字节字符不会被送回。

HTPU 中的 TRANSLATE 内设过程不能处理多字节字符。如果 TRANSLATE 的参数中包含多字节字符, 结果是不能预料的。

HEVE 的 WILDCARD FIND 命令不支持 \+ 符号的使用, 因为这个符号只用于 ASCII 值为 128 至 255 的单字节字符。

此版本的 HTPU/HEVE 不支持 EVE$BUILD。

附录 B

附加的 HTPU/HEVE 信息

此附录列出 HTPU/HEVE 的附加信息。在下表中, 您会找到信息的解释和从引起那些信息的错误中恢复过来的建议措施。

有关其他信息的资料, 请参阅《DEC Text Processing Utility Reference Manual》。

B.1 HTPU 信息

表 B-1 HTPU 的附加信息及其严重程度

缩写	信息	严重程度
INVDEVTYPE	无效设备类型。 无特权存取 HTPU。	致命错误信息
RIMP_NOT_EXIST	输入法的 RIMP不存在 DECwindows 服务器上。	通信信息