OpenVMS/Hangul 사용자 지침서

주문 번호 : BA322-90025

2005 년 5 월

본 지침서에는 한글 OpenVMS 소프트 웨어의 개념과 특징 , 명령어에 대한 설명이 수록되어 있습니다 .

개정 / 갱신 정보 :	본 문서는 OpenVMS/Hangul V1.5 AXP 용 OpenVMS Hangul 사용자 지침서를 대체합니다 .
소프트웨어 버전 :	OpenVMS/Hangul I64 Version 8.2 OpenVMS/Hangul Alpha Version 7.3-2

Hewlett-Packard Company

Palo Alto, California

Printed in Singapore

목 차

제 11 장 디버거 유틸리티

부록 A 프로그래밍 언어

부록 B 한글을 위한 프로그래밍 기능

그림 목록

표 목록

서 문

사용자

이 지침서는 OpenVMS/Hangul 시스템 사용자를 위한 것입니다 .

구성

이 지침서는 10 개의 장과 2 개의 부록으로 구성되어 있습니다 .

- 제 1 장은 OpenVMS/Hangul 의 개요를 설명합니다 .
- 제 2 장은 이 소프트웨어의 문자 세트인 DEC 한글 문자 세트를 상세히 설명합니다 .
- 제 3,4,6,7 및 8 장은 이 소프트웨어를 기동시키는데 도움되는 정보를 제공합니다 . 이 장들에는 DCL 명령과 유틸리티 사용에 도움이 되는 많은 예제들이 있습니다 .
- 제 5 장은 HANGULGEN 유틸리티를 중점적으로 설명합니다 .
- 제 9 장은 날짜 및 시각에서의 한글 지원에 대한 정보를 제공합니다 .
- 제 10 장은 DOOSAN 220 단말기와 프린터에서의 문자 변환을 지원하는 단말기 폴백 기능 (Terminal Fallback Facility) 에 관한 정보를 제공합니다 .
- 제 11 장은 디버거 유틸리티에서의 한글 지원에 대한 사용자 정보를 제공합니다 .

이 지침서의 2 개 부록은 숙련된 사용자를 위한 것으로서 , 관심있는 주제 사항을 제공합니다 .

- 부록 A 는 프로그래밍 언어에서 한글 문자의 사용에 관한 7 개의 예제를 수록하고 있습니다 .
- 부록 B 는 HSYSHR, HSMGSHR 및 HTPU 를 포함한 시스템에서의 한글 처리 런타임 라이브러리에 관한 정보를 제공하고 있습니다 . 또한 , 사용자가 따라 해 볼 수 있는 예제 프로그램이 있는 HTPU 호출식 루틴을 수록하고 있습니다 .

규약

OpenVMS, VMS	용어 OpenVMS 와 VMS 는 OpenVMS 운영체제를 뜻합니다 .
OpenVMS/Hangul	용어 OpenVMS/Hangul 은 한글용 OpenVMS I64 와 한글용 OpenVMS Alpha 운영 체제 모두를 뜻합니다 .
OpenVMS/Hangul I64	용어 OpenVMS/Hangul I64 는 한글용 OpenVMS I64 운영 체제를 뜻합니다 .
OpenVMS/Hangul Alpha	용어 OpenVMS/Hangul Alpha 는 한글용 OpenVMS/Hangul Alpha 운영 체제를 뜻합니다 .
한글	이 지침서에서 , 단어 " 한글 " 은 자국 음성 문자인 한글 , 중국 문자인 한자 및 기타 그래픽 문자를 포함하는 한글 문자 세트에 정의된 문자들의 집단 명칭으로 사용됩니다 .

RET	캐리지 리턴 .
	예제와 형식에서 , 별도로 언급하지 않는 한 , 사용자 입력의 각 행의 끝에서는 암시적으로 캐리지 리턴이 발생하는 것으로 합니다 . 따라서 , 입력시에는 각 행의 끝에서 캐리지 리턴을 눌러야 합니다 .
키 기호	예제에서 , 키와 순차 키는 PF2 및 CTRL/Z 같이 기호로 나타냅니다 .
CTRL	CTRL/x 는 CTRL 키를 누른 상태에서 다른 키를 동시에 누르는 것을 나타냅니다 . 예를 들어 , CTRL/C, CTRL/Y, CTRL/O 등 입니다 .
. . .	수직 생략 부호는 특정 명령에 대해 시스템의 응답 데이터를 화면에 전부 표시하지 않거나 , 또는 사용자가 입력해야 하는 것을 생략했다는 것을 뜻합니다 . 즉 , 논의중인 내용과 직접 관련이 없는 정보를 생략할 때에 사용합니다 .
...	수평 생략 부호는 추가의 매개변수 , 값 , 또는 정보가 입력될 수 있다는 것을 뜻합니다 .
()	형식 설명에서 , 괄호 () 는 사용자가 괄호 내의 것을 선택해야 한다는 것을 나타냅니다 .
[]	형식 설명에서 , 괄호 [] 는 사용자가 임의로 선택할 수 있음을 나타낼 때에 사용합니다 . 이 경우 , 사용자는 아무것도 선택하지 않거나 또는 그 중 하나만 선택하거나 또는 그 중 모두를 선택할 수 있습니다 (그러나 , 괄호가 파일 명세의 디렉토리명 구문에서 사용되거나 또는 할당문의 부문자열 명세에서 사용된 경우에는 선택적이지 않습니다).
{}	형식 설명에서 , 괄호 {} 는 사용자의 택일을 나타낼 때에 사용합니다 . 즉 , 괄호 내에 표시된 것 중 반드시 하나를 선택해야 합니다 .

관련 책자

- OpenVMS 문서 세트
- HEVE 사용자 지침서
- HTPU 와 HEVE 참조서
- OpenVMS/Hangul RTL Korean Processing (HSY$) Manual
- OpenVMS/Hangul RTL Korean Screen Management (SMG$) Manual

제 1 장
서론

OpenVMS/Hangul 은 기존의 OpenVMS 소프트웨어 기능의 최상위 버전으로서 한글 문자 처리 기능을 제공합니다 . 사용자는 OpenVMS/Hangul 의 다양한 기능들을 통하여 DEC 한글 코드와 함께 ASCII 코드도 처리할 수 있습니다 .

OpenVMS/Hangul 은 정보 교환을 위한 그래픽 문자 세트인 한국 표준 (KS C 5601-1978) 에 따르는 2- 바이트 8 비트 형식의 DEC 한글 데이터 (한글 문자) 를 지원합니다 . DEC 한글 문자 세트에 대한 상세한 설명은 제 2 장을 참조하십시오 .

OpenVMS/Hangul 에서는 , 대부분의 OpenVMS DCL 명령이 한글 문자를 지원합니다 . 이들 명령에 대해서는 제 3, 4, 6, 7,8,9,10 및 11 장에서 보다 상세히 설명되어 있습니다 . 또한 , 다양한 프로그래밍 언어로 작성된 프로그램들에서도 한글 문자를 사용할 수 있습니다 . 이와 관련된 예제는 부록 A 를 참조하십시오 .

OpenVMS/Hangul 은 다음과 같은 한글 문자를 처리할 수 있는 많은 유용한 유틸리티들을 제공합니다 .

- HANGULGEN 유틸리티는 한글 단말기 유형을 설정하여 보여줍니다 . 또한 , 유틸리티의 도움말 원문과 시스템 메시지에서 사용할 언어를 선택하는 방법을 제공합니다 .
- HDUMP 는 파일 , 디스크 볼륨 또는 마그네틱 테이프 볼륨의 내용을 10 진수 , 16 진수 또는 8 진수 형식으로 보여줄 뿐만 아니라 ASCII 문자를 한글 문자로 변환해 주는 유틸리티입니다.
- HMAIL 은 우편 유틸리티입니다 . 이것은 폴더명과 주제명을 한글로 지원합니다 .
- HTPU 는 한글 문자를 위한 원문 처리 유틸리티입니다 . HEVE 는 단어 처리 (word processing) 및 선 그리기 유틸리티입니다 .
- TERMINAL FALLBACK FACILITY 는 DOOSAN 220 단말기와 프린터에서 문자 변환을 지원합니다 .

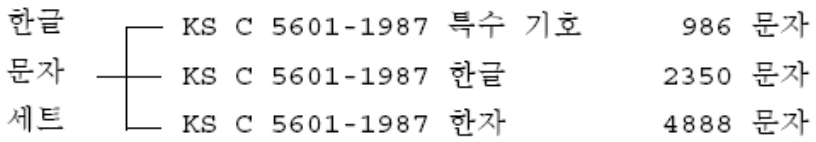
제 2 장
DEC 한글 문자 세트

이 장에서는 DEC 한글 문자 세트의 구성에 대하여 설명합니다 .

2.1 서론

DEC 한글 문자 세트는 KS C 5601-1987 규격을 따르는 한글 , 한자 및 특수 그래픽 문자들로 구성되는 기본 문자 세트입니다 . 그림 2-1 에 DEC 한글 문자 세트의 구성을 나타내었습니다 .

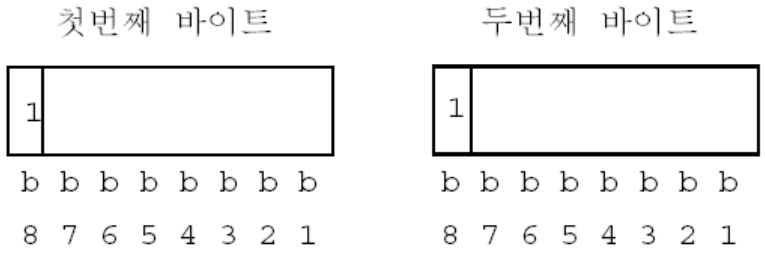
그림 2-1 DEC 한글 문자 세트



2.2 한글 코드

한글 코드에서는 그래픽 문자를 2 바이트로 표현합니다 . 한글 코드 를 일반 ASCII 코드와 구분하기 위하여 첫번째 바이트와 두번째 바 이트의 최상위 비트 (Most Significant Bit) 를 1 로 설정합니다 . 그림 2-2 에 한글 문자 표현을 나타내었습니다 .

그림 2-2 한글 문자의 2- 바이트 표현



2.2.1 코드 구성

한글 코드는 KS C 5601-1987 표준을 따릅니다 . 코드 표는 94 행 x 94 열로 배열된 8,836 개의 코드 위치로 구성됩니다 . 행과 열은 1 부터 94 까지 번호가 부여됩니다 .

한글 코드의 첫번째 바이트(b7-b1)는 코드의 행 번호를 결정하고두번째 바이트(b7-b1)는 열 번호를 결정합니다. 첫번째 바이트와 두 번째 바이트의 코드 범위는 16 진수 A1 부터 FE 까지입니다.

다음의 예제는 행과 열 번호로 나타낸 한글 문자의 2- 바이트 코드 입니다.

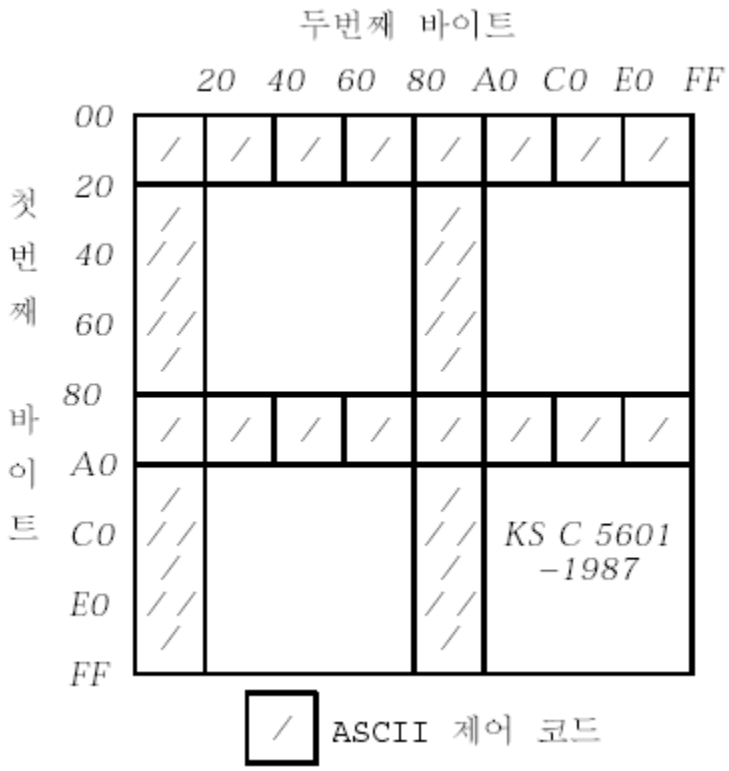
```
행 번호(10진수) = 첫번째 바이트(16진수) - A0(16진수) 및
열 번호(10진수) = 두번째 바이트(16진수) - A0(16진수)
예  16 = B0 - A0 및
    01 = A1 - A0 이므로
```

행 16, 열 1은 한글 코드로 B0A1입니다.

2.2.2 한글 코드 표

그림 2-3 은 2- 바이트 코드 표의 배치와 한글 문자 세트의 위치를 보여줍니다.

그림 2-3 2- 바이트 코드 표의 DEC 한글 문자 세트

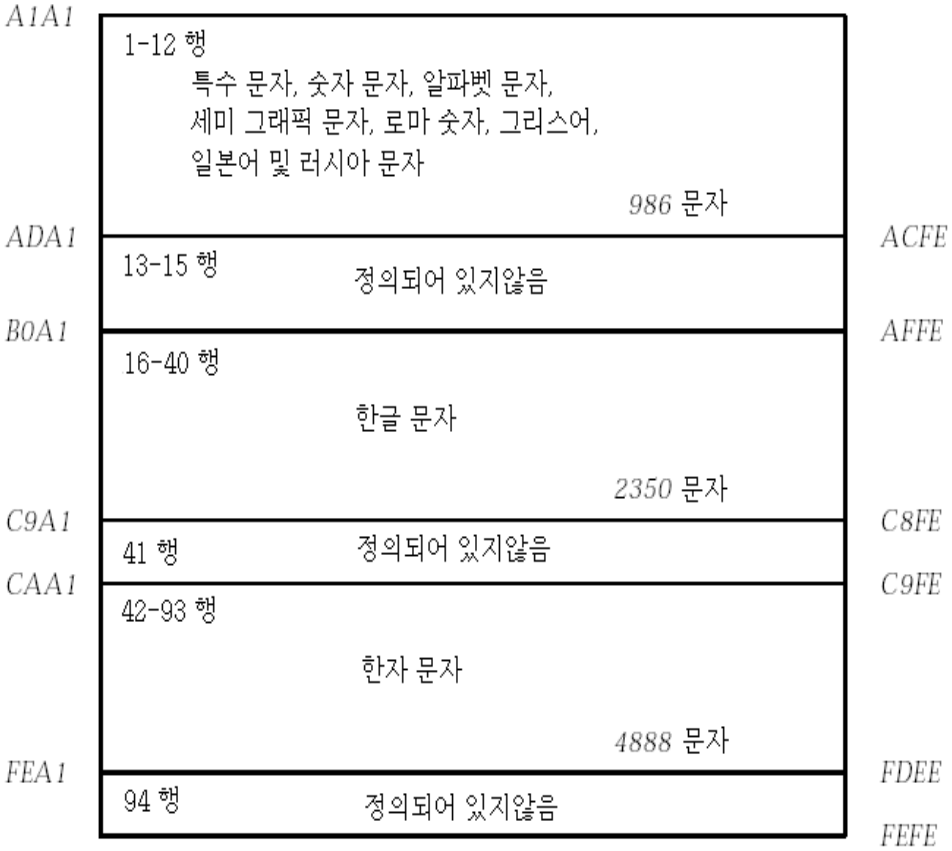


주

줄 무늬 표시 영역은 ASCII 제어 코드로 할당된 영역입니다 . 한글 코드가 이 영역에 할당되지 않음으로써 , ASCII 코드와의 상충이 없습니다 .

그림 2-4 는 한글 코드 표의 상세한 구성을 보여 줍니다 .

그림 2-4 한글 코드 표의 구성



제 3 장
한글 문자의 단말기 I/O 지원

OpenVMS/Hangul 은 한글 문자의 단말기 입력과 출력을 위하여 다음의 기능들을 지원합니다 .

- 한 / 영 혼합 문자 입력중 편집
- 한글 및 영어 문자 입력시 검사

3.1 입력 문자의 편집

OpenVMS/Hangul 은 커서 이동 , 삭제 , 삽입 , 겹쳐쓰기 및 행 끝에서의 줄 바꿔쓰기와 같이 OpenVMS 에서 지원되는 모든 행 편집 기능들을 지원합니다 . 이들 지원되는 기능들의 완전한 목록을 보려면, OpenVMS 자료를 참조하십시오 .

단말기 화면에 표시되는 한글 문자는 2 개 이상의 열을 차지합니다. 따라서 , OpenVMS/Hangul 은 편집 기능에서 다음과 같이 2 개의 동작 모드를 제공합니다 .

1. ASCII 문자 모드

   이 모드에서는 삭제 키가 단말기 화면 상에서 한번에 1 열 만을 삭제합니다 . 커서이동 키는 한번에 1 화면표시 셀 만큼 이동합니다 . 이 모드는 OpenVMS 에서만 제공됩니다 .
2. 한글 문자 모드

   이 모드에서는 삭제 키가 한번에 한 자의 한글 문자를 삭제합니다 . 커서 이동 키는 한 번에 한 자의 한글 문자만큼 이동합니다. 단어 삭제 키 (CTRL/J) 는 한 개의 영어 단어나 또는 한 개의 한글 문자열을 삭제합니다 . 만일 , 입력이 한 행의 끝 부분까지 다 채워지지 않은 상태에서 , 채워지지 않고 남은 부분 보다 넓은 문자가 입력되는 경우 (예를 들어 , 행 끝에 1 개의 화면표시 셀 밖에 여유가 없는데 한글 문자가 입력되는 경우) 에는 입력문자가 무시됩니다 .

HANGULGEN 유틸리티가 이들 두 가지의 편집 모드간을 전환하는데 사용됩니다 . 모드 전환 명령에 대해서는 이 지침서의 제 5 장 HANGULGEN 유틸리티를 참조하십시오 .

3.2 읽기 검증

이 기능은 어플리케이션에서 규정된 규칙에 따라 입력 데이터를 검증합니다 .

어플리케이션은 입력 필드를 2 개 이상의 입력 영역으로 분리하는 표시기를 사용하여 입력 필드를 지정할 수 있습니다 . KS C 5601-1987 문자 세트 내의 한글 문자가 표시기에 의해 정의된 경계에 걸치게 되는 경우 (즉 , 한글 문자가 다른 영역으로 삽입되기 위하여 두 개로 분리되어야 하는 경우) 이 문자는 무효로 간주됩니다 .

이 기능은 HANGULGEN 유틸리티를 사용하여 명료하게 OFF 시킬 수 있습니다 . 또한 어플리케이션이 한번에 1 바이트씩 입력 데이터를 읽을 때에는 자동적으로 이 기능이 OFF 됩니다 .

제 4 장
DCL 명령어 및 유틸리티

이 장은 한글 단말기와 프린터에서 요구되는 장치 특성을 수록하고있으며 , DCL 명령과 유틸리티에서의 한글 문자의 사용에 대하여 설명합니다 .

4.1 입문

4.1.1 단말기 특성의 설정

4.1.1.1 한글 단말기의 설정

한글 단말기를 요구하는 OpenVMS/Hangul 의 대부분의 유틸리티들은 HANGULGEN 유틸리티를 사용하여 다음과 같이 적절한 장치 유형으로 초기화됩니다 .

```
HANGULGEN> SET /DEVICE_TYPE=VT382
```

4.1.1.2 한글 프린터의 설정

한글 프린터는 HANGULGEN 유틸리티를 사용하여 다음과 같이 요

구되는 특성으로 초기화되어야 합니다 .

```
HANGULGEN> SET /DEVICE_TYPE=LA380/PERMANENT LTA100:
```

4.2 DCL 명령

이 절에서는 DCL 명령에서 한글 특성의 사용에 대하여 설명합니다 .

4.2.1 명령 프로시저의 인수

이 인수에는 한글 문자가 포함된 문자열을 사용할 수 있습니다 . 예를 들면 다음과 같습니다 .

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

4.2.2 SHOW 명령에서의 한글 사용

시스템 관련 정보를 화면에 한글로 표시하기 위하여 다음의 SHOW 명령을 사용할 수 있습니다 .

- $SHOW USER

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

OpenVMS 도움말 메시지와 마찬가지로 , SHOW LOGICAL 명령과 SHOW TRANSLATION 명령을 제외하고는 HANGULGEN 유틸리티를 사용하여 영어 또는 한글 메시지를 선택할 수 있습니다 .

4.2.3 APPEND, BACKUP, CONVERT, COPY, CREATE 및 TYPE 명령에서의 한글 사용

파일에 한글 데이터를 포함시킬 수 있습니다 . 또한 APPEND, BACKUP, CONVERT, COPY, CREATE 및 TYPE 명령에 한글을 사용할 수 있습니다 . 예를 들면 다음과 같습니다 .

```
$ CREATE HANGUL.DAT
한글 문자를 사용할수 있습니다 ^Z
$
$ TYPE HANGUL.DAT
한글 문자를 사용할수 있습니다
$
```

4.2.4 ASSIGN, DEASSIGN 및 DEFINE 명령에서의 한글 사용

논리명과 등가 문자열에 한글을 사용할 수 있습니다 . 또한 ASSIGN, DEASSIGN 및 DEFINE 명령에서도 한글을 사용할 수 있습니다 .

ASSIGN 명령과 DEFINE 명령을 사용하여 한글 문자로 정의된 논리명을 화면표시하기 위하여 SHOW LOGICAL 명령과 SHOW TRANSLATION 명령을 사용할 수 있습니다 . 그리고 , 렉시칼 함수 F$LOGICAL 은 한글로 변환될 수 있습니다 . 예를 들면 다음과 같습니다 .

```
$ DEFINE 논리명 그림
$ SHOW LOGICAL 논리명
  " 논리명 " = " 그림 "  (LNM$PROCESS_TABLE)
$ _ 변수 =F$LOGICAL(" 논리명 ")
$ SHOW SYMBOL _ 변수
  _ 변수 = " 그림 "
$
```

4.2.5 DIRECTORY 명령에서의 한글 사용

DIRECTORY 명령에서는 2 개 국어 시스템 메시지가 지원됩니다 . HANGULGEN 유틸리티를 사용하여 영어 또는 한글 메시지를 선택할 수 있습니다 .

4.2.6 MESSAGE 명령에서의 한글 사용

메시지 파일의 기능 필드 (facility field) 와 원문에서 한글을 사용할수 있습니다 . 그러나 , 식별 필드 (ident field) 에서는 한글이 허용되지 않습니다 .

4.2.7 OpenVMS 도움말에서의 한글 사용

HELP 명령 사용시 OpenVMS HELP 기능을 호출하여 OpenVMS 명령에 관한 정보를 화면에 표시하거나 또는 HANGULGEN 유틸리티를 사용하여 한글 또는 영어로 주제를 화면에 표시할 수 있습니다 .

출력 장치가 HANGUL_MSG 로 설정되어있다 하더라도 다음 명령을 사용하면 영어로된 도움말을 직접 화면표시할 수 있습니다 .

$HELP @HELPLIB

더우기 , OpenVMS/Hangul 전용 도움말 주제를 다음의 명령을 사용하여 화면에 표시할 수 있습니다 .

$HELP @HSYHLP

4.2.8 READ 명령과 WRITE 명령에서의 한글 사용

데이터의 입력과 출력 원문에 한글 문자를 포함할 수 있습니다 . 예를 들면 다음과 같습니다 .

```
$ TYPE HANGUL.DAT
한글 문자 읽기 쓰기 테스트 1
$ OPEN/READ INPUT HANGUL.DAT
$ OPEN/WRITE OUTPUT HANGUL1.DAT
$ READ INPUT REC
$ WRITE OUTPUT REC
$ WRITE OUTPUT " 한글 문자 읽기 쓰기 테스트 2"
$ CLOSE INPUT
$ CLOSE OUTPUT
$ TYPE HANGUL1.DAT
한글 문자 읽기 쓰기 테스트 1
한글 문자 읽기 쓰기 테스트 2
$
```

4.2.9 REPLY 명령에서의 한글 사용

메시지 원문에 한글을 사용할 수 있습니다 . 예를 들면 다음과 같습니다 .

```
$ REPLY/TERMINAL=TT "REPLY 에 한글 문자 사용 "
Reply received on HANGUL from user JOHN at _HANDVF$LTA5236:
15:19:48
REPLY 에 한글 문자 사용
$
```

4.2.10 SET 명령에서의 한글 사용

프롬프트 지정에 한글을 사용하거나 또는 명령 설명 파일의 동작어에 한글을 사용할 수 있습니다 . 그러나 , 한정자나 레이블에는 한글을 사용할 수 없습니다 .

4.2.11 SEARCH 명령에서의 한글 사용

탐색될 문자열에 한글을 사용할 수 있습니다 . 예를 들면 다음과 같습니다 .

$ TYPE HANGUL.DAT

그 그림이 아름답습니다

```
그 화가는 매우 유명합니다
$
$ SEARCH/HIGHLIGHT=UNDERLINE HANGUL.DAT 그림이  그 그림이 아름답습니다
$
```

4.2.12 한글 기호

기호명은 알파벳 문자 (_ 와 $ 포함) 로 시작해야 합니다 . 두번째와 그 이후 문자부터는 한글을 사용할 수 있습니다 . 또한 , 할당될 문 자열에 한글이 포함될 수도 있습니다 . 예를 들면 다음과 같습니다 .

```
$ _ 변수 =" 한글 데이터 "
$ SHOW SYMBOL _ 변수
  _ 변수 = " 한글 데이터 "
$
```

4.2.13 명령 프로시저에서의 레이블

명령 프로시저의 레이블명에 한글을 사용할 수 있습니다 . 따라서 GOTO 명령에 한글 레이블을 사용할 수도 있습니다 .

4.3 WWPPS(World-Wide PostScript Printing Subsystem)

WWPPS(World-Wide PostScript Printing Subsystem) 는 Alpha 와 I64 플랫폼 모두에서 일반 텍스트 파일에 대한 고품질의 인쇄 기능을 제공합니다 . 이 유틸리티를 사용하면 단일 바이트와 다중 바이트의 아시아 문자로 구성된 일반 텍스트 파일을 한번에 인쇄할 수 있습니다 . 유틸리티를 호출하려면 DCL 프롬프트에서 다음 명령을 사용하십시오 .

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

유틸리티는 다음 프롬프트로 응답합니다 .

```
$ WWPPS>
```

또한 , 다음과 같이 외부 명령을 정의하여 호출을 간략화할 수도 있습니다 .

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

다음은 일반 텍스트 파일을 인쇄하는 예입니다 .

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

또는

```
$ WWPPS PRINT/QUEUE=hp$printer/LOCALE=ko_kr_korean-
_$hangul-text_file.txt
```

WWPPS 에 대한 자세한 내용은 OpenVMS 사용자 안내서를 참조하십시오 .

제 5 장
HANGULGEN 유틸리티

이 장은 HANGULGEN 유틸리티를 설명합니다 .

5.1 서론

HANGULGEN 유틸리티는 한글 단말기 유형을 설정하고 보여줍니다 . 또한 사용자가 유틸리티 도움말 원문과 시스템 메시지에 사용할 언어를 선택하는 방법을 제공합니다 .

HANGULGEN 유틸리티의 SET 명령과 SHOW 명령은 한글 단말기와 관련된 속성을 처리하는 것을 제외하고는 DCL 명령에서와 유사합니다 .

5.2 기동 순서

HANGULGEN 명령은 다음과 같이 기동시킵니다 .

```
$ RUN HSY$SYSTEM:HANGULGEN
HANGULGEN>
```

5.3 명령 요약

SET	한글 단말기 특성의 설정
SHOW	한글 단말기 특성 보기
HELP	HANGULGEN 명령 목록
EXIT	HANGULGEN 종료후 DCL 명령 레벨로 복귀

5.3.1 SET 명령

SET 명령은 한글 단말기 특성을 설정합니다 . 이러한 특성에는 장치 유형 , 입력 유형 및 출력 유형이 포함됩니다 . 영어 및 한글 도움말 / 오류 메시지를 제공하는 유틸리티에서는 , 해당되는 출력 유형을 지정하여서 출력 언어를 선택합니다 .

5.3.1.1 명령 형식

SET 명령 형식은 다음과 같습니다 .

```
SET [device-name[:]][/DEVICE_TYPE=device-type]
                    [/INPUT=input-type]
                    [/OUTPUT=output-type]
                    [/PERMANENT]
```

[/SYSTEM]

여기에서

device-name	한글 단말기에 접속된 물리적인 장치명을 지 정합니다.	
/DEVICE_TYPE=terminal-type	HANGULGEN 이 지원하는 다음의 유효한 장치 유형을 지정합니다.	
	VT382	VT382-K 한글 단말기 장치를 식별합니다. 다른 장치 유형이 하나도 지정되지 않으면 이것이 기본값이 됩니다.
	VT100	VT100 단말기 장치를 식별합니다. ASCII 문자를 한글 장치에 설정합니다.
	VT300-series	VT300 계열 단말기 장치를 식별합니다. 이 한정자는 VT382 를 VT300 계열 단말기 특성으로 재설정하는데 사용합니다.
	LA380	장치를 LA380-K 한글 인쇄 장치로 식별합니다.
	DOOSAN 220	DOOSAN 220C 단말기 유형을 식별합니다.
/INPUT=input-type	단말기를 한글 또는 ASCII 단말기로 설정합니다. 입력 유형의 유효한 값은 다음과 같습니다.	
	HANGUL	단말기가 한글을 입력하고 또한 한글 단말기로 동작합니다.
	ASCII	단말기가 표준 ASCII 단말기입니다
	DOOSAN 220C 와 VT382-K 에 대한 내정 입력 유형은 HANGUL 입니다.	
/OUTPUT=output-type	단말기가 한글 또는 ASCII 메시지를 화면표시하도록 설정합니다. 출력 유형의 유효한 값은 다음과 같습니다.	
	HANGUL_MSG	한글 메시지가 화면표시됩니다.
	ASCII_MSG	ASCII 메시지가 화면표시됩니다.
	DOOSAN 220C 와 VT382-K 에 대한 내정 출력 유형은 HANGUL_MSG 입니다.	
/PERMANENT	영구적으로 변경 처리되는 특성을 지정합니다 이 한정자는 사용자가 소유하지 않은 장치의 특성을 설정하는 경우에 필요합니다.	
	/PERMANENT 는 /OUTPUT 과는 함께 사용되지 않습니다. 이 한정자를 사용하려면 PHY_IO 특권과 LOG_IO 특권이 필요합니다.	

/SYSTEM	특성 변경을 시스템 기본으로 취급되도록 지정합니다 . /SYSTEM 은 / DEVICE_TYPE 및 /OUTPUT 과 함께 사용될 수 없습니다 . 이 한정자를 사용하려면 PHY_IO 특권이 필요합니다 .

5.3.1.2 SET 명령의 제한사항

다음 표에 /PERMANENT 와 /SYSTEM 한정자 사용시 제한사항을 갖는 명령 한정자를 요약해 놓았습니다 .

표 5-1 /SYSTEM 및 /PERMANENT 한정자 사용시의 제한사항

	/SYSTEM	/PERMANENT
/DEVICE_TYPE	제한사항 없음	제한사항 있음
/INPUT	제한사항 있음	제한사항 있음
/OUTPUT	제한사항 없음	제한사항 없음

5.3.2 SHOW 명령

SHOW 명령은 한글 단말기 특성을 보여주는데 , 각 장치 당 1 행씩 다음 필드들의 정보를 화면표시합니다 .

- 현재 SYS$INPUT 장치의 장치명
- 장치 유형
- 입력 유형 (HANGUL 또는 ASCII)
- 출력 유형 (HANGUL_MSG 또는 ASCII_MSG)

HANGULGEN 에서는 DOOSAN 220C 와 VT382 의 장치 유형이

HANGUL_VDU 로 표시됩니다 .

그리고 LA380 의 장치 유형은 HANGUL_PRT 로 표시됩니다 .

5.3.2.1 명령 형식

SHOW 명령의 형식은 다음과 같습니다 .

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

여기에서 ,

device-name	HANGULGEN 에 의해 표시될 물리적인 장치명을 지정합니다 .
/ALL	모든 단말기 장치의 특성을 화면표시합니다 . 이 한정자를 사용하려면 PHY_IO 및 SHARE 특권이 필요합니다 . 현재 장치가 아닌 장치에 대해서는 출력 유형이 "Unknown" 으로 표시됩니다 . (다음 페이지의 예 참조)

/PERMANENT 단말기의 영구적인 특성을 화면표시합니다 . 이 한정자를 사용하려면 PHY_IO 또는 LOG_IO 특권이 필요합니다 .

/SYSTEM 시스템의 내정 특성을 화면표시합니다 . 이 한정자를 사용하려면 PHY_IO 특권이 필요합니다 .

5.4 명령 예

1. 한글 단말기로서의 모든 단말기 장치의 특성을 보려면 다음과 같이 하십시오 .

```
HANGULGEN>SHOW
Device Name      Type          Input         Output
_RTA2:       HANGUL_VDU   ASCIIH      ANGUL_MSG

HANGULGEN>SHOW/ALL
Device Name      Type            Input         Output
_OPA0:          LA36            ASCII        Unknown
_RTA1:          VT300_Series    HANGUL   Unknown
_RTA2:          HANGUL_VDU   ASCII      HANGUL_MSG
_LTA5006:      VT200_Series    HANGUL   Unknown
_LTA5007:      HANGUL_VDU   ASCII        Unknown
_LAT5008:      HANGUL_VDU   ASCII        Unknown
Total : 6 Terminals
```

2. VT382 단말기에 한글 메시지가 화면표시되게 설정하려면 다음과 같이 하십시오 .

```
$ RUN HSY$SYSTEM:HANGULGEN
HANGULGEN> SHOW
Device name      Type          Input         Output
_TTA0:        VT200_series ASCII         ASCII_MSG

HANGULGEN> SET /DEVICE_TYPE=VT382

HANGULGEN> SHOW
Device Name      Type          Input         Output
_TTA0:HANGUL_VDUHANGULHANGUL_MSG
```

3. LA380-K 한글 프린터의 특성을 설정한 후 , 그것을 보려면 다음과 같이 하십시오 .

```
HANGULGEN> SET TXA1: /DEVICE_TYPE=LA380 /PERMANENT
HANGULGEN> SHOW TXA1:
Device Name     Type             Input          Output
_TXA1:          HANGUL_PRT     HANGUL       Unknown
```

4. 단말기를 ASCII 특성으로 다시 설정한 후 , 그것을 보려면 다음 과 같이 하십시오 .

```
HANGULGEN> SHOW TXA2:
Device Name     Type             Input          Output
_TXA2:          HANGUL_VDU  HANGUL       ASCII_MSG
HANGULGEN> SET TXA2:/DEVICE_TYPE=VT100 /PERMANENT
HANGULGEN> SHOW TXA2:
Device Name     Type          Input          Output
```

```
_TXA2:          VT100     ASCII      ASCII_MSG
```

5.5 행 이상 계속되는 명령

DCL 명령과 같이 , 명령어 문자열을 2 행 이상에 걸쳐 계속 입력하려면 HANGULGEN 명령 레벨에서 행의 마지막 문자로 연속 표시 문자인 하이픈 ("-") 을 입력하십시오 . 예를 들면 다음과 같습니다.

```
HANGULGEN> SET TTA0: /DEVICE_TYPE=VT382 -
_HANGULGEN> /PERMANENT
```

5.6 주석 행 지원

DCL 명령과 같이 , HANGULGEN 명령 레벨에서 주석 행을 입력하려면 주석의 맨 앞에 느낌표 "!" 를 입력하십시오 . 예를 들면 다음과 같습니다 .

```
HANGULGEN> ! set tta0 as VT382 as Hangul terminal
HANGULGEN> SET TTA0: /DEVICE_TYPE=VT382 /PERMANENT
```

제 6 장
HMAIL 유틸리티

이 장에서는 HMAIL 유틸리티의 기능에 대해서 설명합니다 .

6.1 서론

HMAIL 은 MAIL 유틸리티에 다음의 기능들을 향상시킨 것입니다 .

- 개인명에서의 한글 지원
- 제목명에서의 한글 지원
- 폴더명에서의 한글 지원
- 문자열 탐색시 한글 지원
- HMAIL 에서의 내정 편집기

6.2 HMAIL 세션의 기동

HMAIL 은 DCL 명령을 다음과 같이 사용하여 호출할 수 있습니다 .

```
$ HMAIL
HMAIL>
```

6.3 HMAIL 의 기능

OpenVMS MAIL 유틸리티에서 HMAIL 의 추가 기능에 대해서는 다음의 소 절들에서 하나씩 설명합니다 . OpenVMS MAIL 의 기능에 대해서는 VMS 사용자 지침서를 참조하십시오 .

6.3.1 한글 개인명

HMAIL 에서는 개인명에 한글을 사용할 수 있습니다 . 예를 들어 , 사용자 KIM 이 파일 FILE.TXT 를 한글 개인명 사용자 HANGUL::LEE 에게 발송하고자 하는 경우 , 명령을 다음과 같이 입력할 수 있습니다 .

```
HMAIL> set personal_name " 김 영근 , 컴퓨터실 "
HMAIL> exit
$ HMAIL/SUBJECT="Meeting Agenda" FILE.TXT HANGUL::LEE
```

또는 사용자 KIM 이 한 개인명을 현재 메시지에만 적용되게 설정할 경우 명령을 다음과 같이 입력할 수 있습니다 .

```
HMAIL> send/personal_name=" 김 영근 , 컴퓨터실 " file.txt
수신 :  hangul::lee
주제 :  Meeting agenda
```

노드 HANGUL 에 있는 사용자 LEE 는 다음의 메시지를 수신합니다 .

```
HANGUL 노드의 HANGUL::KIM 에서온 새 전자우편 " 김 영근 , 컴퓨터실 "
```

6.3.2 한글 제목명

HMAIL 에서는 제목에 한글을 사용할 수 있습니다 . 다음은 그 사용 예를 보여줍니다 .

HMAIL>

수신 : hangul::lee

주제 : 회의 의사록

6.3.3 한글 폴더명

폴더명에 한글을 사용할 수 있습니다 . 다음의 예들은 폴더명에 한글을 사용하는 방법을 보여줍 니다 .

HMAIL> select mail

%MAIL-I-SELECTED, 5 메시지가 선택되었습니다

HMAIL> move/all 회의 의사록

회의 의사록 폴더가 없습니다 .

폴더를 만드시겠습니까 ?(Y/N, 기본값은 N) y

%MAIL-I-NEWFOLDER, 회의 의사록 폴더가 생성되었습니다

6.3.4 한글 문자열 대조

HMAIL 에서는 SEARCH 명령과 SELECT, SET FOLDER, DIRECTORY 및 READ 명령의 / SUBJECT_SUBSTRING 한정자 같이 명령의 탐색 문자열에 한글을 사용할 수 있습니다 .

예를 들면 다음과 같습니다 .

HMAIL> select 회의 의사록

%MAIL-I-SELECTED, 5 메시지가 선택되었습니다

HMAIL> directory

회의 의사록

# 발신	날짜	주제
1 HANGUL::ZQCHEN	8-JUL-1991	부서장 회의
2 LIU	17-AUG-1991	17-AUG-1991
3 SYSTEM	12-OCT-1991	시스템 데이터베이스 갱신
4 LEE	14-OCT-1991	긴급 회의
5 SYSTEM	20-OCT-1991	시스템 데이터베이스

HMAIL> directory/subject_substring= 데이터베이스

회의 의사록

# 발신	날짜	주제
1 LIU	17-AUG-1991	데이터베이스 정보
2 SYSTEM	12-OCT-1991	시스템 데이터베이스 갱신
3 SYSTEM	20-OCT-1991	시스템 데이터베이스

6.3.5 HMAIL 에서의 내정 편집기

HMAIL 에서는 HTPU 가 내정 편집기로 호출됩니다 . 그러나 , SET EDITOR 명령을 사용하여 이 내정값을 무시하고 다른 편집기를 선택할 수 있습니다 . 예를 들어 , 명령을 사용하여 내정 편집기를 다른 편집기로 설정하려면 다음과 같이 할 수 있습니다 .

HMAIL> set editor [other editor][1]

6.4 명령 요약

다음 표는 한글 처리와 관련된 HMAIL 명령을 요약한 것입니다 .

명령 / 한정자	설명
SET PERSONAL_NAME, /PERSONAL_NAME	한글 개인명을 사용할 수 있습니다 .
/SUBJECT	한글 제목명을 사용할 수 있습니다 .
COPY, DIRECTORY, FILE,MOVE, READ, SELECT,SET/SHOW FOLDER,SET WASTEBASKET	한글 폴더명을 지정할 수 있습 니다 .
SEARCH, /SUBJECT_SUBSTRING	한글 문자열을 탐색하도록 , 탐색 문자열에 한글을 포함시킬 수 있습니다 .
HELP	HANGULGEN 에서 단말기 출력이 HANGUL_MSG 로 설정되어 있으면 도움말 원문이 한글로 화면표시됩니다 .

1. [other editor] 는 OpenVMS 하에서 실행되는 다른 원문 편집기를 말합니다 . (예 , TPU)

제 7 장
HTPU 와 HEVE 유틸리티

이 장에서는 HTPU/HEVE 유틸리티의 기능에 대하여 설명합니다 .

7.1 서론

HTPU 와 HEVE 는 HEDT 의 편집 기능 외에 , 한글 원문 편집을 지원하기 위해 DECTPU/EVE 를 향상시킨 것입니다 . 이것은 문자셀 단말기 (CCT) 와 DECwindows Motif/Hangul 사용자 인터페이스를 모두 지원합니다 .

HTPU 와 HEVE 는 , DECTPU/EVE 에 기초하여 HTPU 와 HEVE 의 두 계층으로 구성되었습니다 . HEVE 는 HTPU 가 제공하는 내장 프로시저를 통해 한글 편집 기능을 구현합니다 .

7.2 HTPU 와 HEVE 의 사용

CCT 인터페이스를 사용하는 내정 편집기로서 HEVE 와 함께 HTPU 를 호출하려면 다음과 같이 명령을 입력하십시오 .

$ EDIT/HTPU TEXT.TXT

또한 , 다음과 같이 외부 명령을 정의하여 간단하게 호출할 수도 있습니다 .

$ HEVE :== $EDIT/HTPU

DECwindows Motif/Hangul 인터페이스를 사용하는 내정 편집기로서 HEVE 와 함께 HTPU 를 호출하려면 다음과 같이 명령을 입력하십시오 .

$ SET DISPLAY/CREATE/NODE=< 사용자 워크스테이션의 노드명 >
$ HEVE/DISPLAY=MOTIF TEXT.TXT

TEXT.TXT 는 편집할 파일입니다 . 일단 HEVE 가 기동되면 , 키보드의 키를 눌러서 원문을 입력할 수 있습니다 . 또한 , HEVE 명령과 사전 정의된 기능 키를 사용하여 문을 편집할 수 있습니다.

표 7-1 에 사전에 정의되어있는 키를 수록해 놓았습니다 .

표 7-1 HEVE 의 사전 정의된 편집 키와 그 기능

키	기능	키	기능
CTRL/A	모드 변경 (삽입 / 겹쳐쓰기)	FIND	지정된 문자열 탐색
CTRL/B	이전 HEVE 명령 재호출	INSERT HERE	현재 커서 위치에 선택된 범위의 원문 삽입
CTRL/E	행 끝으로 커서 이동	REMOVE	선택된 범위의 원문 제거
CTRL/H	행의 맨 앞으로 커서 이동	SELECT	일정 범위의 원문 선택

표 7-1 HEVE 의 사전 정의된 편집 키와 그 기능

CTRL/J	현재 단어의 삭제	PREV SCREEN	이전 화면
CTRL/L	페이지 구분 표시의 삽입	NEXT SCREEN	다음 화면
CTRL/U	행의 처음까지 삭제	F10	HEVE 종료
CTRL/W	화면 재생	F11	방향 변경 (전진 또는 후진 방향)
CTRL/Z	HEVE 종료	F12	행 단위로 커서 이동
DELETE	이전 문자 삭제	F13	현재 단어 삭제
UP arrow	위로 이동	F14	모드 변경 (삽입 / 겹쳐쓰기)
DOWN arrow	아래로 이동		
RIGHT arrow	오른쪽으로 이동		
LEFT arrow	왼쪽으로 이동		

HEVE 와 그 명령어들에 대한 자세한 내용을 보려면 HEVE 사용자 지침서를 참조하거나 또는 HEVE 내에서 DO 키를 누른후 HELP 를 입력하십시오 .

7.3 HTPU 의 기능

HTPU 는 일단의 내장 프로시저와 한글 편집기를 개발하기 위한 호 출식 인터페이스를 제공합니다 . 사용자는 HTPU 의 내정 편집기인 HEVE 와 같이 HTPU 언어만으로 작성된 사용자 편집기와 그것의내장 프로시저를 가지거나 또는 호출식 인터페이스를 통하여 사용자의 어플리케이션 프로그램으로부터 HTPU 를 호출할 수 있습니다 .

DECTPU 에 대하여 HTPU 의 향상된 기능은 다음과 같습니다 .

- 문자 분류를 위한 새로운 내장 프로시저
- 한글 문자 처리를 지원하기 위하여 수정된 내장 프로시저
- DECwindows Motif/Hangul 창 환경하에서의 한글 문자 입력

신규 및 수정된 내장 프로시저의 자세한 목록은 HTPU/HEVE 참조서를 살펴보십시오 . DECTPU 내장 프로시저에 대한 상세한 사항은 DEC Text Processing Utility Reference Manual 을 참조하십시오 . Guide to the DEC Text Processing Utility 에서는 HTPU 언어를 설명하고 있습니다 .

HTPU 호출식 인터페이스에 고유한 정보는 이 책의 부록 B.4 에 있습니다 . DECTPU 의 호출식 인터페이스에 대해서는 VMS Utility Routines Manual 의 제 14 장을 참조하십시오 .

7.4 HEVE 의 기능

HTPU 언어와 내장 프로시저를 구성할 때에 , HEVE 는 ASCII 문자와 함께 한글 문자도 편집할 수 있습니다 . HEVE 는 EVE[1]의 모든 기능을 제공하는 외에 다음의 확장된 기능도 가지고 있습니다 .

- 기호 구성 (symbol formation) 과 선 및 상자 그리기 기능
- 전자 및 반자 문자의 상호 변환
- HEDT 기능 시뮬레이션
- DECwindows Motif/Hangul 창 환경에서의 한글 풀다운 / 팝업 메뉴 사용

HEVE 전용 명령은 HTPU 와 HEVE 참조서를 보십시오 . EVE 명령은 EVE 참조서를 참조하십시오 .

1. MCS 문자 편집을 지원하지 않는 것은 제외 .

제 8 장
HDUMP 유틸리티

HDUMP 는 DUMP 의 한글 버전입니다. 이것은 DUMP 유틸리티에 DEC 한글 문자 세트 내의 파일 내용의 화면표시와 인쇄 기능을 보강하여 향상시킨 것입니다.

HDUMP 와 DUMP 간의 주요 차이점은 HDUMP 가 2 바이트 한글 문자를 화면표시할 수 있다는 점입니다. DUMP 는 행에 1 바이트만 남아 있는 상태에서 2 바이트의 한글 문자가 오는 상황을 처리하지 못합니다. 이 경우, DUMP 는 한글 문자의 첫번째 바이트만을 행에 남기고 두번째 바이트는 다음 행으로 넘깁니다. 따라서, 오류 결과를 야기합니다. 그러나, HDUMP 에서는 어떠한 상황에서도 해당 한글 문자의 첫번째 바이트를 두번째 바이트와 함께 다음 덤프 행 (dump line)으로 넘기고, 화면 표시 필드의 왼쪽 여백에서 한 개의 ASCII 문자 자리를 줄여서 화면표시 또는 인쇄됩니다.

적합한 한글 범위를 벗어나는 모든 2- 바이트 코드는 마침표(.)로 화면표시 또는 인쇄됩니다.

HDUMP 의 명령 형식은 다음과 같습니다.

$ HDUMP file-name

모든 명령 한정자는 DUMP 와 같습니다.

다음은 HDUMP 의 예입니다.

```
$ TYPE KOREAN.DAT
미국 ＤＥＣ 컴퓨터 주식회사   전화 번호:662-5211
미국 DEC 컴퓨터 주식회사     전화  번호 : 662-5211
$
$ HDUMP/RECORD KOREAN.DAT
WRKD$:[HANGUL.SRC]KOREAN.DAT;1 파일 덤프  9-DEC-1992 19:22:28.64
파일 ID (8433,4,0)  파일 끝 블록 1 / 할당량 3
레코드 번호 1 (00000001), 48 (0030) 바이트
BBC7C4C4 20C3A3C5 A3C4A320 B9B1CCB9   미국 ＤＥＣ 컴퓨 000000
FCC02020 20E7BBB8 C8C4BDD6 C120CDC5   터 주식회사 전   000010
31313235 2D323636 3AA3C8F8 B920ADC8 화 번호:662-5211  000020
레코드 번호 2 (00000002), 45 (002D) 바이트
20CDC5BB C7C4C420 43454420 B9B1CCB9 미국 DEC 컴퓨터  000000
20ADC8FC C0202020  E7BBB8C8 C4BDD6C1 주식회사   전화  000010
        31 3132352D  3236363A A3C8F8B9 번호:662-5211... 000020
```

제 9 장
한글 날짜 및 시각 지원

이 장에서는 한글 날짜 및 시각의 지원에 대하여 설명합니다 .

9.1 서론

입력 및 출력 형식에서 한글 날짜 및 시각이 지원됩니다 . 한글 언어표 및 미리 정의되어있는 일 - 시 출력 형식을 사용할 수 있습니 다 . 시스템과 사용자 프로세스가 서로 맞게 설정되어 있는 경우에는 DCL 명령 DIR 및 DCL 유틸리티 MAIL 과 HMAIL 에서 한글 일 - 시 출력 형식을 사용할 수 있습니다 . 뿐만 아니라 , 사용자 자신의 일 - 시 입 / 출력 형식을 정의할 때에도 한글 언어표를 사용할 수 있습니다 . 이 일 - 시 입 / 출력 형식은 OpenVMS RTL 의 일 - 시 처리 루틴에서 사용 가능합니다 . 일 - 시 처리 루틴과 그 사용법에 대해서는 VMS RTL Library(LIB$) Manual 을 참조하십시오 .

9.2 입문

한글 일 - 시를 사용하려면 , 그 전에 먼저 시스템 관리자 (또는 CMEXEC, SYSNAM 및 SYSPRV 특권이 있는 사용자) 가 명령 프로시저 SYS$MANAGER:LIB$DT_STARTUP.COM 을 실행하여 일 - 시를 위한 몇가지 출력 형식 (미리 정의된 한글 출력 형식 포함) 을 정의해 주어야 합니다 . 그리고 , 한글 일 - 시의 항목을 정의해 주어야 합니다 . 이에 대한 명령 사용 예는 다음과 같습니다.

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

이상적으로 , 상기 명령은 OpenVMS/Hangul 시스템에서는 SYSTARTUP_VMS.COM 에서 실행되어야 합니다 .

9.3 사전에 정의되어 있는 한글 출력 형식

다음의 표는 사전에 정의된 한글 일 - 시의 출력 형식을 보여줍니다 .

표에 사용된 연상 코드 (mnemonics) 에 대해서는 VMS RTL Library(LIB$) Manual 을 참조하십시오 .

표 9-1 사전에 정의되어 있는 출력 날짜 형식

날짜 형식 논리명	형식	예
LIB$DATE_FORMAT_046	!Y4 년 !MNB 월 !DB 일 !WAU	1988 년 11 월 10 일 (목)
LIB$DATE_FORMAT_047	!Y4 년 !MNB 월 !DB 일 !WA	1988 년 11 월 10 일 목요일

표 9-2 사전에 정의되어 있는 출력 시각 형식

시각 형식 논리명	형식	예
LIB$TIME_FORMAT_023	!MIU !HB2 시 !MB 분 !SB 초	오후 2 시 16 분 26 초

9.4 사전에 정의되어 있는 한글 언어표

한글 언어표는 몇 개의 논리명을 정의합니다 . 이 논리명은 9.6 절에서 설명하는 사용자 정의 출력 형식에서 사용될 수 있습니다 . 참조를 위하여 이들 논리명을 보면 다음 페이지와 같습니다 .

표 9-3 한글 언어표

논리명	출력 연상코드	등가명
LIB$WEEKDAYS_[U|L|C]	W[U|L|C]	" 월요일 "," 화요일 "," 수요일 "," 목요일 "," 금요일 "," 토요일 "," 일요일 "
LIB$WEEKDAY_ABBREVIATIONS_[U|L|C]	WA[U|L|C]	"(월)","(화)","(수)","(목)","(금)","(토)","(일)"
LIB$MONTHS_[U\|L\|C] LIB$MONTHS_ABBREVIATIONS_[U|L|C]	MA[U|L|C],MAA[U|L|C]	"1 월 ","2 월 ","3 월 ","4 월 ","5 월 ","6 월 ","7 월 ","8 월 ","9 월 ","10 월 ","11 월 ","12 월 "
LIB$MI_[U|L|C]	MI[U|L|C]	" 오전 "," 오후 "

9.5 실행중 한글 출력 형식의 선택

날짜 , 시각 또는 그 둘 모두에 특유한 한글 형식을 선택하려면 , 9.3 절에서 정의한 논리명을 사용하여 LIB$DT_FORMAT 논리명을 정의해야 합니다 . 예를 들면 다음과 같습니다 .

$DEFINE SYS$LANGUAGE HANGUL

$DEFINE LIB$DT_FORMAT LIB$DATE_FORMAT_046 , LIB$TIME_FORMAT_023

위의 두 개의 논리명의 순서는 그것들이 출력되는 순서를 결정합니다 . 위와 같이 정의하면 , 아래와 같이 지정된 형식으로 날짜가 출력되고 , 그 다음에 한 개의 공백과 , 지정된 형식의 시각 순서로 출력됩니다 .

1992 년 11 월 10 일 (목) 오후 2 시 16 분 26 초

9.6 사용자 정의 한글 출력 형식

사용자는 자신의 실행 모드 논리명을 정의하여 자신의 출력 형식을 한글로 정의할 수 있습니다 . 예를 들면 다음과 같습니다 .

$ DEFINE/EXEC/TABLE=LNM$DT_FORMAT_TABLE LIB$DATE_FORMAT_601 -

_$ " 시스템시간 : !Y4 년 !MNB 월 !DB 일 "

$ DEFINE/EXEC/TABLE=LNM$DT_FORMAT_TABLE LIB$TIME_FORMAT_601 -

_$ "!MIU !HB2 시 !MB 분 !SB 초 !WU"

원하는 형식을 정의한 후에 , 사용자는 다음의 명령들을 사용할 때 그 형식을 사용할 수 있습니다 .

$ DEFINE SYS$LANGUAGE HANGUL

$ DEFINE LIB$DT_FORMAT LIB$DATE_FORMAT_601,LIB$TIME_FORMAT_601

이 결과로 출력되는 한글 일시 형식은 다음과 같습니다 .

시스템 시간 : 1992 년 11 월 10 일 오후 2 시 16 분 26 초 목요일

9.7 실행 중 한글 입력 형식의 선택

어플리케이션에서 사용할 날짜 및 시각 입력 전용의 한글 형식을 선택하려면 , LIB$DT_INPUT_FORMAT 논리명을 정의해야 합니다 . 이 때 , LIB$DT_INPUT_FORMAT 의 등가명으로는 9.4 절에서 설명한 사전 정의된 한글 언어표로부터 연상코드를 포함시킬 수 있습니다 .

9.8 예

9.8.1 HMAIL

HMAIL> SET PERSONAL " 권경화 " 주제

HMAIL> SELECT MAIL

HMAIL> DIR

#	발신	날짜
1	NODEA::KIM	1992 년 11 월 16 일 업무 보고서

```
HMAIL> 1
#1   1992 년 11 월 16 일  수요일  오전 11 시 20 분 53 초       MAIL
발신 :    NODEA::KIM          " 권경화 "
수신 :    LEE
참조 :
주제 :    업무 보고서
총무 부장 귀하
금년도 업무 현황을 아래와 같이 보고드립니다
                  :
                  :
                  :
HMAIL> EXIT
$
```

9.8.2 DIR 명령

```
디렉토리 USER$:[LEE]
TEST.DAT;1                파일 ID: (2229,2,0)
크기 :           1/3          소유자 :    [SYSTEM]
생성일 :  1992 년  11 월 15 일  목요일  오후 6 시 35 분 25 초
개정일 :  1992 년  11 월 15 일  목요일  오후 6 시 35 분 28 초 (1)
만료일 :    < 지정된 것이 없음 >
백업일 :    < 백업이 기록되지 않았음 >
파일 구성 :          순차
파일 속성 :   할당 : 3,  확장 : 0, 글로벌 버퍼 수 : 0, 버전 한계 없음
레코드 형식 :   가변길이
레코드 속성 :   Carriage return carriage control
RMS 속성 :       없음
Journaling enabled:    없음
파일 보호 :      System:RWED, Owner:RWED, Group:RE, World:
액세스 제어 리스트 : 없음
합계 : 1 파일 , 1/3 블록
$ DIR/DATE TEST*.DAT
디렉토리 USER$:[LEE]
TEST.DAT;1       1992 년  11 월 15 일  화요일  오후 6 시 35 분 25 초
TEST1.DAT;1      1992 년  11 월 15 일  화요일  오후 6 시 36 분 19 초
TEST2.DAT;1      1992 년  11 월 15 일  화요일  오후 6 시 36 분 22 초
```

```
TEST3.DAT;1     1992 년  11 월 15 일  화요일  오후 6 시 36 분 26 초
TEST4.DAT;1     1992 년  11 월 15 일  화요일  오후 6 시 36 분 31 초
합계 : 5 파일
$
```

9.8.3 RTL 날짜 / 시각 처리 루틴

```
$ 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_047,LIB$TIME_FORMAT_023
$ DEFINE SYS$LANGUAGE HANGUL
$
```

```
$ DEFINE LIB$DT_INPUT_FORMAT -
_$"!Y4 년 !MNB 월 !DB 일 !MIU!HB2 시 !MB 분 !SB.!C2 초 "
$ RUN DATETIME
Input date time:1992 년 10 월 11 일 오후 2 시 16 분 26.02 초
Output date time: 1992 년 10 월 11 일 화요일 오후 2 시 16 분 26 초
$
$ DEFINE LIB$DT_INPUT_FORMAT -
_$"!Y4 년 !MAU 월 !DB 일 !HB4 시 !MB 분 !SB.!C2 초 "
$ RUN DATETIME
Input date time:1992 년 10 월 11 일 17 시 12 분 22.02 초
Output date time: 1992 년 10 월 11 일 화요일 오후 5 시 12 분 22  초
$
```

제 10 장
단말기 폴백 기능

이 장에서는 TFF 2- 바이트 문자 변환표를 사용하기 위하여 시스템을 설정할 때의 단말기 폴백 기능 (TFF) 유틸리티의 사용법에 대하여 설명합니다 .

10.1 단말기 폴백 기능

단말기 폴백 기능 (TFF) 은 DOOSAN 문자세트 (2- 바이트로만 구성) 만을 화면표시할 수 있는 DOOSAN 단말기에서 문자 변환을 지원하도록 확장되었습니다 . 이는 DOOSAN 220 단말기가 단말기로 / 부터의 문자 전환을 통하여 KS C 5601-1987 문자 세트에서 한글 문자 (한자는 지원되지 않음) 의 입 / 출력을 가능하게 합니다 . 문자 변환은 문자표를 통하여 어플리케이션 소프트웨어에 명료하게 수행됩니다 . 따라서 , DOOSAN 문자 세트 단말기를 사용하는 사용자는 OpenVMS/Hangul 상에서 KS C 5601-1987 문자 세트를 사용하여 개발된 소프트웨어를 사용할 수 있습니다 .

TFF 에 관한 보다 상세한 사항은 VMS Terminal Fallback Utility Manual 을 참조하십시오 .

10.1.1 TFF 환경 설정

이하의 소절들에서는 OpenVMS/Hangul 운영체제 상에서 TFF 를 사용 할 수 있기 전에 , 사용자가 수행해야 하는 기본적인 단계를 서술합니다 . TFF 환경의 설정과 유지보수 방법에 관해서는 VMS Terminal Fallback Utility Manual 을 참조하십시오 .

10.1.1.1 사용자 시스템에 TFF 의 설치

사용자의 시스템에 OpenVMS/Hangul 을 설치한 후에는 TFF 를 가동시켜야 합니다 . TFF 를 가동시키려면 , 사용자 시스템 디스크 상의 디렉토리 SYS$MANAGER 에 위치한 TFF$STARTUP.COM 을 (VAX 시스템에서) 또는 TFF$SYSTARTUP.COM 명령 프로시저를 (Alpha 에서) 호출해야 합니다 .

시스템이 재시동될 때마다 TFF 가 가동되게 하려면 , OpenVMS/Hangul 에서는 SYS$MANAGER:SYSTARTUP_VMS.COM 사이트 특정 기동 명령 프로시저를 편집하여 다음의 명령을 포함시키십시오 .

VAX

$ @SYS$MANAGER:TFF$STARTUP.COM

Alpha

$ @SYS$MANAGER:TFF$SYSTARTUP.COM

10.1.1.2 TFF 표

TFF 폴백 표인 HANGUL_DS 는 DOOSAN 문자 세트만을 화면표시할 수 있는 DOOSAN 220 단말기에서 KS C 5601-1987 문자 세트를 지원하기 위하여 제공됩니다 .

TFF 가 시스템에서 가동된 후에는 , 사용자가 액세스하기 전에 비페이지형 동적 기억장소 영역 (nonpaged dynamic memory pool) 에 있는 시스템의 물리적인 기억장소에 필요한 폴백표를 로

드해야 합니다 . 이를 위해서는 , 사용자가 직접 단말기 폴백 유틸리티 (TFU) 를 사용하여 표를 라이브러리에 포함시킨 후 , 그 표를 시스템의 물리적인 기억장소에 로드시켜야 합니다 . HANGUL_DS 를 로드하는 명령은 다음과 같습니다 .

```
$ RUN SYS$SYSTEM:TFU

VAX/VMS Terminal Fallback Facility (TFF)

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

TFU> LOAD TABLE HANGUL_DS
```

10.1.1.3 시스템 내장 표의 설정

TFF 가 가동되면 ASCII 가 내정 폴백 표로 정의됩니다 . 이 표는 비페이지형 동적 기억 장소 영역에 영구적으로 로드됩니다 . 따라서 HANGUL_DS 표를 비페이지형 동적 메모리 영역에 로드한 다음에 다음의 TFU 명령을 입력하여 그것을 새로운 내정 표로 설정할 수 있습니다 .

```
TFU> SET DEFAULT_TABLE HANGUL_DS
```

10.1.1.4 TFF 표의 가동

사용자 단말기에서 폴백 표 HANGUL_DS 를 가동하기 전에 , 8- 비트 문자의 입력이 가능하도록 해야 합니다 . 그렇게 하지 않으면 , TFF 가 입력 문자의 최상위 비트 (Most Significant Bit) 를 삭제하게 됩니다 .

8- 비트 문자의 입력을 가능하게 하는 명령은 다음과 같습니다 .

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

10.1.2 물리적 기억장소 요건

TFF 는 고가의 시스템 자원 (비페이지형 동적 기억장소 영역) 을 사용하기 때문에 , 사용자는 시스템 레벨에서의 환경을 조심스럽게 준비하여야 합니다 . HANGUL_DS 폴백 표를 로드하는 데에는 최소한 14 KBytes 의 비페이지형 동적 기억장소가 요구됩니다 .

제 11 장
디버거 유틸리티

OpenVMS 디버거는 기호 디버거로서 사용자 모드 코드를 디버깅하는 데 주로 사용됩니다. 문자 셀 사용자 인터페이스나 그래픽 (DECwindows Motif) 사용자 인터페이스를 통해 모든 OpenVMS 프로그래밍 언어에 대한 기호 디버깅을 지원합니다. 두 사용자 인터페이스 모두는 DEC 한글 문자 세트의 입력과 화면표시를 지원합니다. 프로그램 변수, 소스 코드 주석 또는 OpenVMS 디버거 오류 메시지에서의 한글 문자는 바르게 표시됩니다.

주

OpenVMS 디버거에서의 DEC 한글 문자 세트 지원은 XPG4 국제화 모델로 구현되었습니다. 이 로캘 데이터베이스는 OpenVMS 와 함께 제공되는 OpenVMS I18n 저장 세트에 있습니다.

11.1 사용자 환경 설정

이 절에는 OpenVMS 디버거의 DECwindows Motif 사용자 인터페이스와 문자 셀 사용자 인터페이스를 모두 한글 문자로 설정하는 방법이 설명되어 있습니다.

11.1.1 문자 셀 사용자 인터페이스

OpenVMS 디버거의 문자 셀 사용자 인터페이스는 DEC 한글 문자 세트를 지원하기 위해 한글 SMG 와 DEC C's XPG4 로컬라이제이션 유틸리티를 함께 사용합니다. 다음 논리명이 정의되어야 DEC 한글 문자 세트 지원이 가능합니다.

- DBG$SMGSHR

  이 논리명은 OpenVMS 디버거가 화면 모드 지원을 위해 사용하는 SMG 공유 가능 이미지의 이름을 정의합니다. DEC 한글 문자 세트가 지원되도록 하려면 이 논리명을 다음과 같이 정의하십시오.

  ```
  $ DEFINE/JOB DBG$SMGSHR HSMGSHR
  ```

- SMG$DEFAULT_CHARACTER_SET

  이 논리명은 한글 SMG 가 지원하는 기본 문자 세트를 정의합니다. DEC 한글 문자 세트가 지원되도록 하려면 이 논리명을 다음과 같이 정의하십시오.

  ```
  $ DEFINE/JOB SMG$DEFAULT_CHARACTER_SET HANGUL
  ```

- LANG

  이 논리명은 DEC C's XPG4 로컬라이제이션 유틸리티 및 런타임 라이브러리의 모든 로캘 범주에 대한 기본 로캘 설정을 정의합니다. DEC 한글 문자 세트가 지원되도록 하려면 이 논리명을 다음과 같이 정의하십시오.

```
$ DEFINE/JOB LANG KO_KR_DECKOREAN
```

11.1.2 DECwindows Motif 사용자 인터페이스

OpenVMS 디버거의 DECwindows Motif 사용자 인터페이스는 DEC 한글 문자 세트의 문자를 표시할 수 있습니다. 디버거의 리소스 파일인 VMSDEBUG.DAT 는 DECW$SYSTEM_DEFAULT 또는 DECW$USER_DEFAULTS 디렉토리에 있으며 각 디버거 윈도우에 대한 글꼴을 명시적으로 지정할 수 있습니다. VMSDEBUG.DAT 에 있는 DebugDefault.font 리소스는 모든 디버거 윈도우에 대한 기본 글꼴을 지정합니다. 또한 각 윈도우는 이 기본값과 다른 글꼴을 사용할 수도 있습니다. 자세한 내용은 VMSDEBUG.DAT 파일을 참조하십시오.

OpenVMS 디버거 윈도우에서 DEC 한글 문자 세트 지원을 작동시키려면 해당 윈도우가 DEC 한글 문자 세트 글꼴을 사용해야 합니다. 다음 방법 중 하나를 사용해 글꼴을 지정합니다.

1. 디버거 윈도우의 리소스에 의해 명시적으로 지정
2. 위 방법으로 글꼴이 명시적으로 지정되지 않은 경우, 디버거 의 기본 글꼴 리소스인 DebugDefault.font 로 지정
3. 디버거의 기본 글꼴 리소스나 디버거 윈도우의 글꼴 리소스 모두가 글꼴을 지정하지 않은 경우에는 시스템 기본 글꼴 리소스를 사용해 지정

11.2 디버거 명령

이 절에서는 디버거 명령에서의 한글 문자 사용에 대해 설명합니다.

11.2.1 EXAMINE 명령

EXAMINE 명령은 새 한정자인 /WCHAR_T:n 을 지원합니다. 이 명령은 검사된 각각의 엔터티를 n 롱워드의 다중 바이트 파일 코드 시퀀스 (n 자) 로 해석하여 표시합니다. n 의 기본값은 1 입니다.

검사된 문자열이 변환되면 OpenVMS 디버거는 디버거가 실행되는프로세스의 로캘 데이터베이스를 사용합니다. 기본값은 C 로캘입니다.

11.2.1.1 DEPOSIT 명령

DEPOSIT 명령은 새 한정자인 /WCHAR_T:n 을 지원합니다. 이 명령은 변환된 다중 바이트 파일 코드 시퀀스의 n 롱워드를 지정된 위치에 넣어 둡니다. n 의 기본값은 1 입니다.

DEPOSIT 명령으로 지정된 문자열이 변환되면 OpenVMS 디버거는 디버거가 실행되는 프로세스의 로캘 데이터베이스를 사용합니다. 기본값은 C 로캘입니다.

부록 A
프로그래밍 언어

OpenVMS/Hangul 에서는 주석 , 문자 및 문자열에 한글을 사용할 수 있고 , 언어에서 한글 데이터를 입 / 출력 처리할 수 있습니다 .

일반적인 프로그래밍 언어에서의 한글 사용을 보여주는 예제 프로그램들을 다음의 소절들에 수록하였습니다 . 이 예제 프로그램들은, OpenVMS/Hangul 시스템의 HSY$EXAMPLE 디렉토리에 있습니다 .

A.1 MACRO-32 에서의 한글 사용 예

```
$ 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  한글 문자로 사용 /
DATA2:      STRING <VAX MACRO 데이터에 한글 사용 >
inp:        string < 데이터 입력 : >
oup:        .long b3-b1
            .address b1
buff:       .long b3-b2
            .address b2
b1:         .ascii / 데이터 출력 : /
b2:         .blkb 40
b3:
.PSECT CODE
.IF DF VAX_MACRO
```

```
                .ENTRY SAMPLE,  ^m<>   ; 주석에 한글 사용 .
.IFF
SAMPLE::
.CALL_ENTRY PRESERVE=<R2,R3,R4,R5,R6,R7,R8,R9,R10,-
                 R11,R12,R13,R14,R15>  ; 주석에 한글 사용 .
.ENDC           PUSHAQ INP
                PUSHAQ DATA1
                CALLS #1, G^LIB$PUT_OUTPUT
                PUSHAQ DATA2
                CALLS #1, G^LIB$PUT_OUTPUT
LOOP:            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
VAX MACRO  한글 문자로 사용
VAX MACRO 데이터에 한글 사용
데이터 입력 : 한글 데이터
데이터 출력 : 한글 데이터
데이터 입력 : ^Z
```

A.2 DEC FORTRAN 에서의 한글 사용 예

```
$ TYPE SAMPLE.FOR
        program sample
        character*40 buff
!       주석에 한글 사용
        type *,'VAX FORTRAN   한글 문자로 사용 '
```

```
          do while (.true.)
            write (*,'(a)') '$ 데이터 입력 : '
            read  (*,'(a)',end=100) buff
            write (*,'(a)') '$ 데이터 출력 : '//buff
          end do
100       continue
          end
$
$
$ FORTRAN SAMPLE
$ LINK SAMPLE
$ RUN SAMPLE
VAX FORTRAN   한글 문자로 사용
데이터 입력 : 한글 데이터
데이터 출력 : 한글 데이터
데이터 입력 : ^Z
```

A.3 BASIC 에서의 한글 사용 예

```
$ TYPE SAMPLE.BAS
100 rem 주석에 한글 사용
110 print "VAX BASIC 한글 문자로 사용 "
120 on error go to 180
130 while 1
140         print " 데이터 입력 :";
150         input buff$
160         print " 데이터 출력 :";buff$
170 next
180 resume 190
190 end
$
$ BASIC SAMPLE
$ LINK SAMPLE
$ RUN SAMPLE
VAX BASIC 한글 문자로 사용
```

```
데이터 입력 :? 한글 데이터
데이터 출력 : 한글 데이터
데이터 입력 :? ^Z
```

A.4 PASCAL 에서의 한글 사용 예

```
$ TYPE SAMPLE.PAS
program sample(input,output);
var   buff:packed array [1..40]  of char;
begin (*  주석에 한글 사용  *)
                writeln ('VAX PASCAL  한글 문자로 사용 ');
                write ('  데이터 입력 :');
                while not eof do begin
                                readln (buff);
                                writeln('  데이터 출력 :',buff);
                                write ('  데이터 입력 :')
                end;
end.

$
$ PASCAL SAMPLE
$ LINK SAMPLE
$ RUN SAMPLE

VAX PASCAL  한글 문자로 사용
        데이터 입력 : 한글 데이터
        데이터 출력 : 한글 데이터
        데이터 입력 : ^Z
$
```

A.5 COBOL 에서의 한글 사용 예

```
$ TYPE SAMPLE.COB

* 주석에 한글 사용
identification division.
program-id. sample.
```

```
author. 김 영근 .
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  한글 문자로 사용   ".
loop.
                display " 데이터 입력 :" with no advancing.
                accept buff at end go to eof.
                display " 데이터 출력 :",buff.
                go to loop.
eof.
                stop run.
$
$ COBOL SAMPLE
$ LINK SAMPLE
$ RUN SAMPLE

VAX COBOL  한글 문자로 사용
데이터 입력 : 한글 데이터
프로그래밍 언어 A-7
데이터 출력 : 한글 데이터
데이터 입력 : ^Z
```

A.6 PL/I 에서의 한글 사용 예

```
$ TYPE SAMPLE.PLI

sample:proc options (main); /* 주석에 한글 사용 */
dcl buff char(40);
```

```
                put list ('VAX PL/I  한글 문자로 사용 ');
                on endfile (sysin) stop;
                do while ('1'b);
                put skip edit (' 데이터 입력 : ') (a);
                        get edit (buff) (a(40));
                        put edit (' 데이터 출력 : ',buff) (a,a);
                        end;
                end;
$
$ PLI SAMPLE
$ LINK SAMPLE
$ RUN SAMPLE
VAX PL/I  한글 문자로 사용
데이터 입력 : 한글 데이터
데이터 출력 : 한글 데이터
데이터 입력 : ^Z
```

A.7 DEC C 에서의 한글 사용 예

```
$ TYPE SAMPLE.C
#include <stdio.h>
main () /* 주석에 한글 사용 */
{
      char buff[40];
      printf("₩nVAX C 한글 문자로 사용 ");
      while (printf("₩n 데이터 입력 : "), gets(buff) != NULL)
           printf(" 데이터 출력 : %s",buff);
}
$
$ CC SAMPLE
$ LINK SAMPLE
$ RUN SAMPLE

VAX C 한글 문자로 사용
데이터 입력 : 한글 데이터
```

데이터 출력 : 한글 데이터
데이터 입력 : ^Z

주 의

#define 으로 정의되는 정의와 참조용 인수에는 한글이 포함될 수 없습니다 .

부록 B
한글을 위한 프로그래밍 기능

B.1 HSYSHR

HSYSHR 은 OpenVMS/Hangul 시스템의 한글 처리 런타임 라이브러리에 포함되어 있습니다 . 이 라이브러리는 기본적인 한글 처리 함수를 제공하는 루틴들을 포함하고 있는 범용 라이브러리입니다.

라이브러리에서 제공되는 온라인 도움말은 시스템에서 사용할 수 있습니다 . 이들 루틴의 함수 , 호출 형식 , 인수 전달 방법 및 복귀값에 관한 보다 상세한 정보를 보려면 다음 명령을 입력하십시오 .

$ HELP @HSYHLP HSYSHR

런타임 라이브러리는 공유 이미지 라이브러리 HSYIMGLIB.OLB 에 있습니다 . 이들 라이브러리 함수는 OpenVMS/Hangul 이 지원하는 모든 프로그래밍 언어에서 호출할 수 있습니다 . 프로그램을 런타임 라이브러리에 링크시키려면 다음 명령을 입력하십시오 .

$ LINK PROGRAM, SYS$LIBRARY:HSYIMGLIB/LIB

상세한 사항은 OpenVMS/Hangul RTL Korean Processing (HSY$) Manual 을 참조하십시오 .

B.1.1 호출할 수 있는 HSYSHR 루틴의 사용 예

```
$  TYPE HSY$EXAMPLE:SAMPLE_HSYSHR.C
/* SAMPLE_HSYSHR.C
  ---------------
  This is a C language example to demonstrate the ability
  of the HSY$ facility of the OpenVMS Run time Library,
  HSYSHR.  The program accepts an input file from the
  command line and formats the text within it in the
  following ways:
  - Left margin = 0, Right margin = 40
  - Convert all half form ASCII characters to  their full forms.
  - Remove leading and trailing blanks of each input line.
  - Remove all embedded controls or space charact
*/
#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*/
!++
! FUNCTIONALITY
!
!   It simply prints out the integer parameter int_param
!   and the string parameter str_param by means of
    tpu$message
!
!--
   BEGIN
   EXTERNAL ROUTINE
      lib$sget1_dd,
      lib$get_vm;
   LOCAL
      buffer : VECTOR [80, BYTE],
      buffer_desc : BLOCK [8, BYTE],
      buffer_len,
      str_out_len : WORD,
      status;
    !
    ! Form a fixed-length string descriptor
    !
    buffer_desc [DSC$W_LENGTH] = 80;
    buffer_desc [DSC$B_DTYPE] = DSC$K_DTYPE_T;
    buffer_desc [DSC$B_CLASS] = DSC$K_CLASS_S;
    buffer_desc [DSC$A_POINTER] = buffer;
    !
    ! Construct the ini_param message and print it out
    !
    status = $FAO ($DESCRIPTOR('Integer: !UL'), buffer_len,
                buffer_desc,  ..int_param);
    IF NOT .status
```

```
    THEN
        RETURN .status;
    buffer_desc [DSC$W_LENGTH] = .buffer_len;
    status = tpu$message (buffer_desc);
    buffer_desc [DSC$W_LENGTH] = 80;
    !
    ! Construct the str_param message and print it out
    !
    status = $FAO ($DESCRIPTOR('String: !AF'), buffer_len,
                buffer_desc, .str_param [DSC$W_LENGTH],
                                .str_param [DSC$A_POINTER]);
    IF NOT .status
    THEN
        RETURN .status;
    buffer_desc [DSC$W_LENGTH] = .buffer_len;
    status = tpu$message (buffer_desc);
    buffer_desc [DSC$W_LENGTH] = 80;
    !
    ! Construct return string
    ! We need to use lib$sget1_dd to allocate the memory
    ! for the return string since TPU useslib$sfreel_dd
    ! to free our string when it needs to.
    !
    str_out_len = %CHARCOUNT(%STRING('Success'));
    status = ;ob$sget1_dd (str_out_len, str_out [0,0,0,0]);
    IF NOT .status
    THEN
        RETURN .status;
    CH$MOVE (.str_out_len,
            UPLIT BYTE ('Success'),
            .str_out [DSC$A_PINTER]);
    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
```

B.2 HSMGSHR

한글 화면 관리 런타임 라이브러리 HSMGSHR 은 OpenVMS/Hangul 시스템에 포함되어 있습니다 . 이 라이브러리는 영상 화면 상의 이미지의 설계 , 구성 및 트랙 유지를 보조하는 일단의 루틴입니다 .

화면 관리 기능은 두 가지의 중요한 서비스 , 즉 단말기 독립성과 구성의 용이성을 제공합니다 .

이들 라이브러리 함수는 OpenVMS/Hangul 이 지원하는 모든 프로그래밍 언어에서 호출 할 수 있습니다 . 프로그램을 런타임 라이브러리에 링크시키려면 다음 명령을 입력하십시오 .

$ LINK PROGRAM, SYS$INPUT/OPTION

SYS$LIBRARY:HSMGSHR.EXE/SHARE

상세한 사항은 VMS RTL Korean Screen Management (SMG$) Manual 을 참조하십시오 .

B.3 호출식 HTPU 루틴

호출식 HTPU 루틴을 사용하면 , 모든 프로그래밍 언어와 어플리케이션에서 HTPU 를 액세스할 수 있습니다 . HTPU 는 OpenVMS Procedure Calling and Condition Handling Standard 를 사용하여 호출을 생성하는 모든 프로그래밍 언어로 작성된 프로그램에서 호출될 수 있습니다 . 또한, HMAIL 같은 OpenVMS 유틸리티에서도 HTPU 를 호출할 수 있습니다 . 호출식 HTPU 를 사용하면 사용자의 프로그램 내에서 한자 원문 처리 기능을 수행시킬 수 있습니다 .

HTPU 는 DECTPU 에서 지원하는 호출식 인터페이스 Simplified Callable Interface 및 Full Callable Interface 모두를 동일하게 지원합니다 . DECTPU 의 호출식 인터페이스에 대한 상세한 사항은 VMS Utility Routines Manual 의 14 장을 참조하십시오 .

HTPU 와 DECTPU 호출식 인터페이스의 차이점을 요약하면 다음과 같습니다 .

1. TPU$CLIPARSE 는 "TPU" 명령 동작어 대신에 "HTPU" 명령 동작어를 받아 들입니다 .
2. SYS$SHARE:TPUSHR.EXE 대신에 SYS$SHARE:HTPUSHR.EXE 를 사용하여 링크시켜야 합니다 .
3. TPU$EXECUTE_COMMAND 루틴은 DECTPU 내장 기능 뿐만 아니라 , HTPU 특유의 내장 기능도 실행할 수 있습니다 . HTPU 특유의 내장 기능에 대해서는 HTPU / HEVE 참조서를 살펴보고 , DECTPU 내장 기능에 대해서는 DEC Text Processing Utility Reference Manual 을 참조하십시오 .

B.3.1 HTPU 의 호출식 루틴

다음의 호출식 루틴들은 HTPU 동작시 사용할 수 있습니다 .

1. Simplified Callable Interface(간이 호출식 인터페이스)

TPU$TPU	DCL 명령에서의 HTPU 와 등가의 HTPU 를 호출합니다 .
TPU$EDIT	입력 파일의 판독 및 편집 , 그리고 수정된 버퍼를 출력 파일에 기입하기 위하여 HTPU 를 호출합니다 .

2. Full Callable Interface(완전 호출식 인터페이스)

TPU$CLEANUP	HTPU 의 내부 데이터 구조를 지우고 , 추가의 호출을 준비합니다 .
TPU$CLIPARSE	TPU$INITIALIZE 를 위하여 명령행을 해석하고 항목 목록을 구성합니다 .

PU$CLOSE_TERMINAL	이 루틴이 사용자 호출 루틴 내에서 호출되는 시점과 사용자 호출 루틴의 복귀 시점 동안에 단말기에 연결된 HTPU 채널을 닫습니다 .
TPU$CONTROL	TPU$INITIALIZE 호출 후에 , 사용자가 HTPU 를 종료할 때까지 제어를 HTPU 에 넘겨 줍니다 .
TPU$EXECUTE_COMMAND	TPU$INITIALIZE 호출 후에 , HTPU 의 명령문을 실행합니다 .
TPU$EXECUTE_INIFILE	TPU$INITIALIZE 호출 후 즉시 초기화 파일을 실행합니다 .
TPU$FILEIO	내정 파일 I/O 루틴입니다 .
TPU$HANDLER	내정 조건 처리기입니다 .
TPU$INITIALIZE	예외 처리기를 설정한 후에 HTPU 를 초기화합니다 .
TPU$MESSAGE	내장 프로시저 MESSAGE 를 사용하여 오류 메시지와 문자열을 작성합니다 .
TPU$PARSEINFO	CLI$DCL_PARSE 를 사용하여 명령을 해석한 후에 , TPU$INITIALIZE 에 대한 항목 목록을 구성합니다 .

B.3.2 예제 프로그램

다음은 Full Callable Interface 를 사용하여 HTPU 를 호출하기 위한 BLISS 로 작성된 예제 프로그램입니다 .

```
!++
!
! 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 (INDENT = '001',
            MAIN = test_main,
```

```
            ADDRESSING_MODE (EXTERNAL = GENERAL)) =
BEGIN
!+ +
!
! FUNCTIONALITY:
!
!   To test the callable interface of HTPUSHR
!
!--
REQUIRE
   'SYS$LIBRARY:STARLET';
LITERAL
   TEST_USER_ARG = %X'F0';     ! an arbitrary pattern
!
! Table of Content
!
FORWARD ROUTINE
   test_main,           ! Module entry
   test_callback,       ! Callback for tpu$initialize
   test_call_user,      ! Call user routine
   test_edit;           ! Edit using HTPU
!
! HTPU callable routines
!
EXTERNAL ROUTINE
   tpu$fileio,                  ! HTPU file routine
   tpu$message,                 ! Displays HTPU message
   tpu$cliparse,                ! HTPU CLI parser
   tpu$handler,                 ! HTPU handler
   tpu$initialize,              ! Initialize HTPU
   tpu$execute_inifile,         ! Execute initial commands
   tpu$execute_command,         ! Execute HTPU statements
   tpu$control,                 ! HTPU main control loop
   tpu$cleanup;                 ! Clean up HTPU
```

```
!
!  HTPU Literals for cleanup
!
EXTERNAL LITERAL
    tpu$m_last_time,
    tpu$m_delete_journal,
    tpu$m_delete_exith,
    tpu$m_delete_buffers,
    tpu$m_delete_windows,
    tpu$m_execute_file,
    tpu$m_execute_proc,
    tpu$m_delete_context,
    tpu$m_reset_terminal,
    tpu$m_kill_processes,
    tpu$m_close_section;
MACRO
    $test_set_bpv (bpv, proc) =
      (BEGIN
          BIND
                $test_bpv = bpv : VECTOR [2];
          $test_bpv [0] = proc;
          $test_bpv [1] = 0;
          END) %;
ROUTINE test_main =
!+ +
! FUNCTIONALITY:
!
!    Main routine to call HTPU
!
!--
    BEGIN
    LOCAL
              callback_bpv : BLOCK [8, BYTE],
              user_arg,
```

```
        cleanup_flag,
        status;
ENABLE
        tpu$handler;
!
! Setup the callback procedure
!
$test_set_bpv (callback_bpv, test_callback);
!
! Set the user argument for test_callback, but ignored
! there
user_arg = TEST_USER_ARG;
!
! Initialize HTPU. Initialization options are specified
! in test_callback
!
status = tpu$initialize (callback_bpv, .user_arg);
IF NOT .status
THEN
        RETURN .status;
!
! Set up the default section file
!
status = tpu$execute_inifile();
IF NOT .status
THEN
        RETURN .status;
!
! Perform editing
!
status = test_edit();
IF NOT .status
THEN
        RETURN .status;
```

```
    !
    ! Finally, we clean up the HTPU and then return
    !
    cleanup_flag = tpu$m_delete_context;
    RETURN tpu$cleanup (cleanup_flag);
    END;
ROUTINE test_callback (user_arg) =
!++
! FUNCTIONALITY:
!
!   Sets up the item list for tpu$initialize
!
!--
    BEGIN
    LOCAL
            fileio_bpv : BLOCK [8, BYTE],
            call_user_bpv : BLOCK [8, BYTE],
            status;
    !
    ! Construct fileio bpv
    !
    $test_set_bpv (fileio_bpv, tpu$fileio);
    !
    ! Construct call_user bpv
    !
    $test_set_bpv (call_user_bpv, test_call_user);
    !
    ! Calls tpu$cliparse to construct item list for
    ! tpu$intialize passing it tpu$fileio as the fileio
    ! routine and test_call_user as the call_user routine
    !
    RETURN tpu$cliparse (
            $DESCRIPTOR ('HTPU htpu$testing:test.txt'),
            fileio_bpv,
```

```
            call_user_bpv);
    END;
ROUTINE test_call_user (int_param,
                        str_param : REF BLOCK [,BYTE],
                        str_out   : REF BLOCK [,BYTE]) =
!++
! FUNCTIONALITY
!
!   It simply prints out the integer parameter int_param
!   and the string parameter str_param by means of
!   tpu$message
!--
    BEGIN
    EXTERNAL ROUTINE
        lib$sget1_dd,
        lib$get_vm;
    LOCAL
        buffer : VECTOR [80, BYTE],
        buffer_desc : BLOCK [8, BYTE],
        buffer_len,
        str_out_len : WORD,
        status;
    !
    ! Form a fixed-length string descriptor
    !
    buffer_desc [DSC$W_LENGTH] = 80;
    buffer_desc [DSC$B_DTYPE] = DSC$K_DTYPE_T;
    buffer_desc [DSC$B_CLASS] = DSC$K_CLASS_S;
    buffer_desc [DSC$A_POINTER] = buffer;
    !
    ! Construct the ini_param message and print it out
    !
    status = $FAO ($DESCRIPTOR('Integer: !UL'), buffer_len,
                   buffer_desc, ..int_param);
```

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

```
    status = tpu$execute_command ($DESCRIPTOR (command));
    IF NOT .status
    THEN
            RETURN .status;
    END)%;
  LOCAL
    status;
  !
  ! It first copies two lines to the MAIN buffer
  !
  $test_execute ('COPY_TEXT("This is the first statement"
                 );    SPLIT_LINE');
  $test_execute ('COPY_TEXT("This is the second statement
                 ");   SPLIT_LINE');
  !
  ! It then passes control to HTPU
  !
  RETURN tpu$control (1);
  END;
END
ELUDOM
```