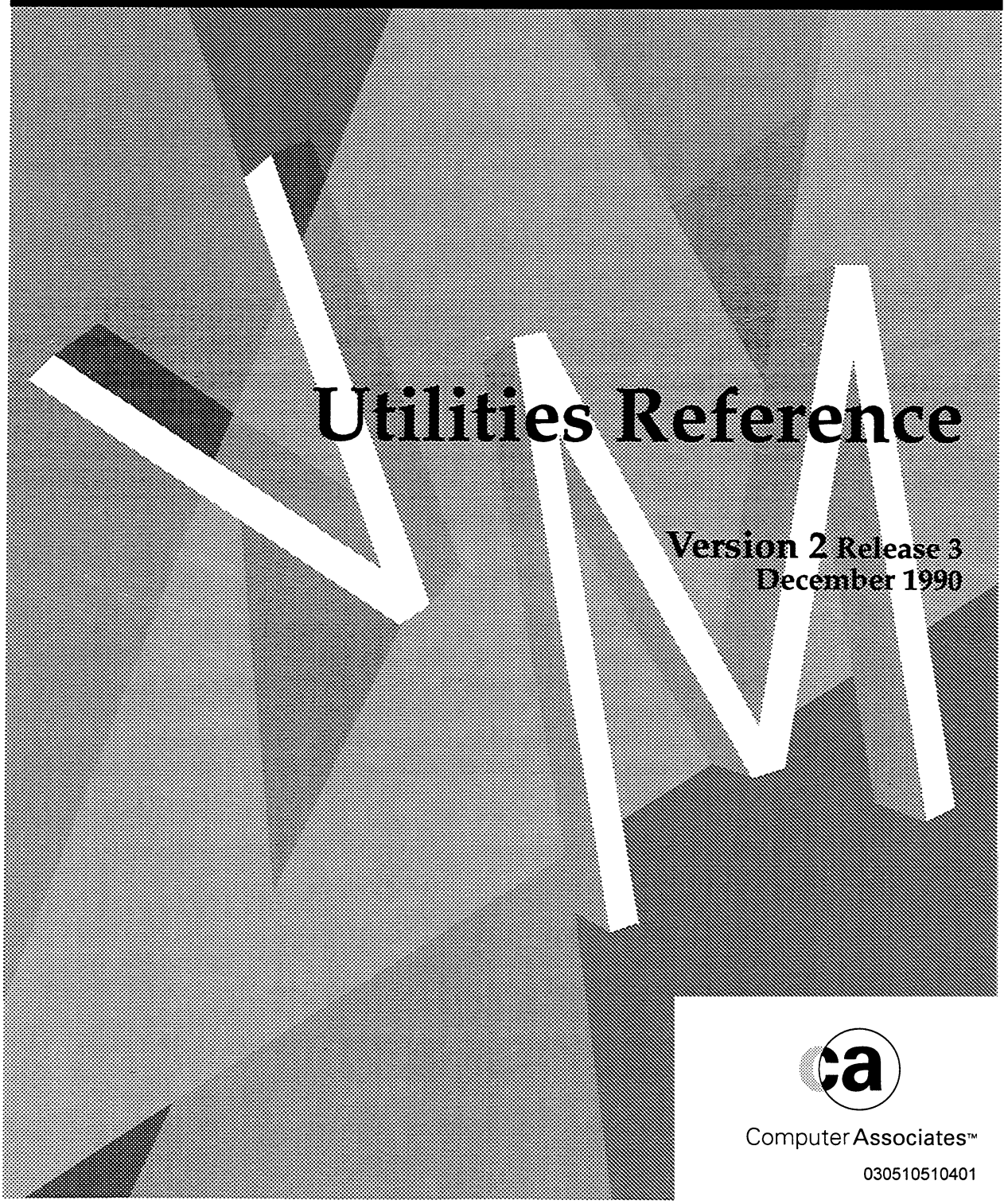
XMENU



Utilities Reference

Version 2 Release 3
December 1990

Computer Associates™

030510510401

Preface

The *XMENU Utilities Reference* contains information for use with the Relay Technology product XMENU.

Audience

This manual is intended for application programmers who want to create full-screen 327*x* applications in the VM environment using specialized XMENU utilities, and for general users who want to take advantage of XMENU's CMS utilities. The *XMENU Utilities Reference* describes the use of and command formats for the following XMENU utility programs:

CONTYPE	Returns information in the CMS return code about the type of terminal being used.
DMSCVT	Converts DMS/CMS panels to XMENU menus.
XMENUINS	Stages XMENU menus in storage to improve performance.
XPSLOAD	Loads programmed symbol sets into your 327*x* terminal.
KWAKEUP	Allows you to wait for one or more system events, such as a time of day, or one or more system interrupts, such as SMSG and IUCV.
KOLMSINS	Stages XMENU messages and other files into storage, and provides for National Language Support.
KBROWSE	Lets you browse CMS files, spool files, and tapes.
KFLIST	Lets you browse lists of CMS files.
KRDRLIST	Lets you browse lists of spooled reader files.
KPRTLIST	Lets you browse lists of spooled printer files.
KPUNLIST	Lets you browse lists of spooled punch files.

How this manual is organized

This manual contains the following chapters:

Chapter 1, "CONTYPE" on page 1, describes the use of and command syntax for the CONTYPE utility, which is used to determine what terminal type and size is being used to display an XMENU menu.

Chapter 2, "DMSCVT" on page 3, details the command format for the DMSCVT utility and describes how to generate XMENU menus from DMS/CMS panels.

Chapter 3, "XMENUINS" on page 7, details the command syntax for the XMENUINS utility, which is used to load XMENU menus into storage.

Chapter 4, "XPSLOAD" on page 11, details the command syntax for the XPSLOAD utility and describes how to load XMENU programmed symbol sets into your terminal for use with XEDIT.

Chapter 5, "KWAKEUP" on page 13, describes the use of and command syntax for the KWAKEUP utility, which is similar in function to the IPF WAKEUP utility.

Chapter 6, "KOLMSINS" on page 21, details the command format for the KOLMSINS utility, which is used to load XMENU message files into storage for faster access, and to provide for National Language Support.

Chapter 7, "KBROWSE, KOLCMSBR, and KOLCPBR" on page 23, describes the utility that lets you browse CMS files, spool files, and tapes while controlling both the browsing environment and the files being browsed via a rich array of subcommands.

Chapter 8, "KFLIST, KRDRLIST, KPRTLIST, and KPUNLIST" on page 89, introduces the utilities that allow you to list and browse CMS minidisk files, as well as spooled reader, printer, and punch files. Included is a detailed reference of the subcommands these utilities share that control both the listing and browsing environments, and the files being listed and browsed.

Notation conventions

The notation conventions used in the syntax boxes are based on, and represent a simplified version of, the conventions used by IBM in the *CMS Command Reference*:

Notation conventions for XMENU utilities

Symbol or type style	Used to indicate—
UPPERCASE	Uppercase text indicates the portion of a command that must be entered.
lowercase	Lowercase text indicates the portion of a command that may be omitted.
lowercase italics	Lowercase italics indicate a variable.
Square brackets [ ]	Square brackets enclose one or more optional items.
Vertical bar |	A vertical bar indicates choices between items.
Options:	A list of optional items can also appear stacked within the syntax box under the Options heading.

Other manuals you should have

In addition to this manual, you should also have the following XMENU documents:

- *XMENU Subroutine Library Reference*
- *XMENU Editor User's Guide and Reference*

Where appropriate, you should also have the reference manuals for any CMS EXEC or high-level programming language you are using to develop XMENU applications.

Contents

Chapter 1. CONTYPE

CONTYPE is an XMENU utility that you can invoke from the CMS command line, from within CMS EXECs, or as a callable module from within higher-level languages. CONTYPE determines the type and size of your 3270-type terminal. CONTYPE is most often used before displaying XMENU menus to determine whether the virtual console is in fact a 3270-type terminal.

The format of the CONTYPE command is shown below:

CONTYPE	[NUMLINES | NUMCOLS | DEFLINES | DEFCOLS | MODEL | ?]

Where:

No parameter specified

Returns the console device type in the CMS return code. This return code is the binary equivalent of the CP real device class and type converted into hexadecimal, then using the result as a decimal value, converted back into binary. For example, a local 3278/3279 terminal is real device class X'40' and real device type X'01'. If you enter CONTYPE while logged on to this terminal, you would get a return code of 4001. The various console device classes and types can be found in the VM/SP or HPO DMKSP MACLIB file, member DEVTYPES, or the VM XA/SP HCPOM1 MACLIB file, member DEVTYPES.

The CONTYPE utility returns data from the system-generated device type and model values. For more accurate information, use XMENU's MCTYPE and MSCRSZ subroutines in your application program—these use hardware responses whenever possible.

NUMLINES Returns the number of lines on the terminal in the return code. For example, a 3278 model 2 would return 24 in the return code. If the terminal is not a 3270-type, the return code is 0 (zero).

NUMCOLS Returns the number of columns on the terminal in the return code. For example, a 3278 model 2 would return 80 in the return code. If the terminal is not a 3270-type, the return code is 0 (zero).

DEFLINES Returns the number of lines on the terminal (in compatibility mode) in the return code. For example, a 3278 model 4 would return 43 in the return code. If the terminal is not a 3270-type, the return code is 0 (zero).

DEFCOLS Returns the number of columns on the terminal (in compatibility mode) in the return code. For example, a 3278 model 2 would return 80 in the return code. If the terminal is not a 3270-type, the return code is 0 (zero).

MODEL Returns the 327*x* model number in the return code. For example, a 3278 model 2 would return 2 in the return code. If the terminal is not a 3270-type, the return code is 0 (zero). 3279 models 2A and 2B return 2 in the return code; models 3A and 3B return 3 in the return code. If you are on a 3290, you will receive the model number used when generating the operating system (usually model 2).

? Types a short description of the CONTYPE options to your terminal.

Usage notes:

1. CONTYPE can be used in a PROFILE EXEC to take certain actions if you are on a dial-up terminal and/or on an ASCII device, for example. It can also be used by an XMENU program to test users' terminals for their ability to display XMENU menus. If you run in an environment where you have several different terminal sizes, you could create XMENU menus of different sizes, group them into different CMS XMENULIB libraries, and determine the appropriate library to use by invoking the CONTYPE NUMLINES command at the start of your EXEC. Here is a simple example using CONTYPE that tests a condition and selects the appropriate menu to display:

```
⋮
CONTYPE NUMLINES
if rc = 24
    then 'menu' = menu24
    else 'menu' = menu43
⋮
```

2. CONTYPE returns all of its responses in the CMS return code. Be careful when using it with EXECs with &ERROR conditions set.
3. This program is a transient-area module and may be called by programs running in the user area.

Return codes and messages:

0 Terminal is not a 3270.
100 (Return code following HELP messages.)
xxx Information returned (see above).

Chapter 2. DMSCVT

XMENU provides an easy-to-use utility called DMSCVT to convert DMS/CMS panels to XMENU menus. You can invoke DMSCVT from the CMS command line, from within CMS EXECs, or as a callable module from within higher-level languages.

In addition, XMENU provides a set of subroutines that are functionally and externally compatible with the DMS/CMS application interface. The DMS/CMS panel must be converted to an XMENU format menu by running the DMSCVT utility using the DMSSUBS option. Neither the DMS/CMS EXEC interfaces nor the RPG are supported.

The DMSCVT command is used to convert DMS/CMS panels created by Release 1 or 2 of the IBM DMS/CMS Program Product (5748-XXB) to XMENU-compatible menus.

To invoke the DMSCVT utility, use the following command syntax:

DMSCVT	*panelname* [(Options] [?] **Options:** NONULLS DMSSUBS NAME USE [DATA] [TEXT] [SELECT] MAP [DATA *n1 name1 n2 name2 ...*] MAP [TEXT *n1 name1 n2 name2 ...*] MAP [SELECT *n1 name1 n2 name2 ...*]

Where:

panelname Specifies the name of the DMS/CMS panel to be converted. DMSCVT creates an XMENU menu file. If an XMENU menu file with the same name exists, the previously existing menu file is replaced.

? Types a short description of DMSCVT options to your terminal.

Options:

NONULLS Specifies that any null (binary zero) characters found within panel fields are changed into blanks in the XMENU menu. This option may be specified with any other DMSCVT option.

DMSSUBS Specifies that DMS/CMS input fields be assigned XMENU field names in the form *XYYY*, where *X* is the type of the field (T for TEXT, D for DATA, S for SELECT) and *YYY* is an ascending number within each field type. In addition, DMSSUBS marks this menu so that it can be used by the XMENU DMS/CMS compatibility

subroutines. This option must be specified to use DMS/CMS compatibility. See "DMSSUBS subroutine library" on page 5 for more details.

NAME Specifies that DMS/CMS input fields be assigned XMENU field names in the form *XXXXYYY*, where *XXXX* represents the first four letters in the panel filename, and *YYY* is a unique three-digit number.

USE Assigns field names similar to the DMS/CMS USE command; i.e., T1, T2, T3, etc. for TEXT fields, D1, D2, D3, etc. for DATA fields, and S1, S2, S3, etc. for SELECT fields.

MAP Assigns specific names to fields similar to the DMS/CMS MAP command. The MAP option can be used together with the USE option.

Files

Input file: *panelname* **PANEL ***

The DMS/CMS PANEL file used to create an XMENU menu.

Output file: *panelname* **MENU A2**

The XMENU MENU file created by DMSCVT. It can be used in all the ways normal XMENU menus are used. Once this file is created, the DMS/CMS PANEL file is not needed to run XMENU applications.

If DMS/CMS subroutine compatibility is desired, this file should NOT be edited with the XMEDIT editor. If it is edited, it loses the flag designating it as eligible for compatibility mode—this is because DMS/CMS is based on field-type position.

Usage notes:

Both Release 1 and 2 DMS panels are supported by DMSCVT. This support includes extended color and highlighting.

DMS/CMS uses numeric offsets to determine which fields are to be used from an EXEC or a program, while XMENU uses symbolically named fields. There are several ways to assign names to DMS/CMS converted menus:

1. Run DMSCVT without any options. This will create an XMENU menu without naming any of the fields. Then, use the field naming feature of the XMEDIT editor to name the fields you want to use. See the *XMENU Editor User's Guide and Reference Manual* for more information on naming fields in XMEDIT.
2. Use the NAME option of DMSCVT, which will assign names to DMS/CMS input fields (assigned with the underscore character). DMSCVT assigns a name in the form *XXXXYYY*, where *XXXX* represents the first four letters of the panel name, and *YYY* is an ascending decimal number, with leading zeros where necessary.
3. Use the MAP option(s) of DMSCVT to name specific DATA, TEXT, or SELECT fields.
4. Use the DMSSUBS option to assign names of the form *XYY*, where *X* is the letter T for a TEXT field, D for a DATA field, or S for a SELECT field, and *YY* is an ascending, zero-suppressed decimal number within each field type. This form of

calling DMSCVT is required when using the menu in DMS/CMS application compatibility mode.

Once your menu is converted, either change the DMS/CMS calls in your EXEC or application to MENUEXEC calls, or take a current DMS/CMS application and load it with the XMENU subroutine library, XMENUSHR TXTLIB. Usually there is an XMENU equivalent function or set of functions for each DMS/CMS function.

Return codes and messages:

1 DMS/CMS PANEL with line length of other than 80 or 132 is not supported.
2 DMS/CMS PANEL file is not properly formatted.
3 Insufficient storage to run DMSCVT.
24 Invalid parameter list.
28 DMS/CMS PANEL file not found.
xx Error xx reading DMS/CMS PANEL file.
xx Error xx writing XMENU MENU file.
100 (Return code following HELP messages).

DMSSUBS subroutine library

XMENU provides a module that allows assembler programs written for DMS/CMS to run without change. This module is contained in the XMENU subroutine library XMENUSHR TXTLIB.

Both DMS/CMS Release 1 and 2 subroutines coexist in this module—an application using both types of calls can be written. In addition, standard XMENU subroutines (except MADD and MDEL) can be directed to the converted menu.

To convert a DMS/CMS application to XMENU, perform the following steps:

1. Convert the panel(s) to XMENU menu(s) using the DMSCVT program with the DMSSUBS option. You may want to retain the PANEL files for backup/recovery.
2. Issue the CMS `GLOBAL TXTLIB` command for the appropriate libraries needed to load your program. Replace any DMS/CMS TXTLIBs in the list with the XMENU subroutine library XMENUSHR.
3. Load your program, and, if necessary, generate a module. You should keep the old module(s) for backup/recovery.

This should be all that is required to convert to XMENU.

The following points should be noted:

- The XMENU implementation of the DMS/CMS subroutines is based on the external description of the DMS/CMS subroutines. No internal compatibility exists (i.e., you cannot depend on internal DMS/CMS entry points, offsets, or control blocks).
- This facility does not support the RPG calls, nor the DMS/CMS EXEC interface programs.
- This facility exists to enable you to continue to run old applications. It is expected that you will use the XMENU facilities for generating new applications.

- Only subroutines are provided—no DMS/CMS compile-time macros or control blocks are provided with XMENU.

Here is a list of the DMS/CMS subroutine entry points supported by DMSSUBS:

EUDMGPNL Display/Release a panel (DMS/CMS Release 1).

EUDPMNGR Display/Release a panel (DMS/CMS Release 2).

EUDWTFSR Display 327*x* data (DMS/CMS Release 1 and 2).

EUDCOBOL DMS/CMS COBOL interface (DMS/CMS Release 1 and 2).

Chapter 3. XMENUINS

The XMENUINS utility allows you to load menus into storage for subsequent use by an application using the XMENU subroutines MLOAD or MLOADX, by an application using MENUEXEC, or by the XMENU menu editor, XMEDIT. In-storage menus are always used if available (even if your call references an XMENULIB).

XMENUINS is also used to purge MENUEXEC saved menus without having them displayed, as would be the case when using the MENUEXEC PURGE option.

Although the XMENUINS utility will most often be used from within a CMS EXEC, it can be invoked from the CMS command line or as a callable module from within higher-level languages.

The format of the XMENUINS command is shown below:

XMENUINS	LOAD *menuname* [([Options] [)]] DROP *menuname*|* PURGE *menunumb*|* MAP ? **Options:** LIB *libname* ALIAS *newname* NOMSG

Where:

LOAD Specifies the menu to be loaded from a DASD file into storage.

DROP Specifies that one menu or all menus loaded into storage by the XMENUINS LOAD option are to be dropped from storage.

PURGE Specifies that one menu or all menus loaded into storage by the MENUEXEC SAVE option are to be purged from storage and from MENUEXEC's internal list of saved menus.

MAP Specifies that a map of XMENUINS in-storage menus should be displayed at the terminal.

menuname|* Specifies the menu filename to be loaded into storage. If * is used with DROP, this specifies that all menus will be dropped from storage. * cannot be used with LOAD.

menunumb|* Specifies the MENUEXEC saved menu to be purged. This value is returned by MENUEXEC in the EXEC variable, "menu". If * is used with PURGE, all MENUEXEC saved menus will be purged.

? Types a short description of the XMENUINS options to your terminal.

***** Specifies that *all* menus are to be dropped from storage. If PURGE was specified, all MENUEXEC saved menus are purged.

Options:

LIB *libname* Specifies that the menu to be loaded into storage should be loaded from the XMENULIB library *libname*.

ALIAS *newname*
Specifies that the menu to be loaded into storage should be renamed *newname*. This new name will be used when MLOAD, MLOADX, XMEDIT, or MENUEXEC attempt to load an in-storage menu.

NOMSG Specifies that no message should be displayed if this LOAD request is attempting to load a menu that is already loaded.

Usage notes:

1. You can rename a menu as it is loaded by giving it an alias. This alias is the name checked when XMENU searches the in-storage menu queue to see if it can be loaded from storage. This feature can be used to support multiple national languages or terminal sizes.
2. The menus in storage are not modified by MENUEXEC or an application—they are read/only copies. When a menu is loaded, XMENU copies the menu to a local work area for application use.
3. XMENUINS allows you to improve performance and decrease disk I/O activity when using a menu or menus that are constantly being loaded and purged.
4. XMENUINS also allows you to display a map of loaded menus. This map includes a use count, which displays the number of times a menu has been used (loaded by an application or MENUEXEC).
5. When using the MENUEXEC saved menu feature, an EXEC variable called "menu" is returned containing the internal name used to identify a saved menu. This same value is passed to the XMENUINS PURGE option. This can be done in an EXEC by passing the variable "menu" on the XMENUINS PURGE command line.
6. XMENUINS runs as a nucleus extension.

Return codes and messages:

1 8278E Library name not specified.
2 8279E Alias name not specified.
3 8280E Insufficient storage to run XMENUINS.
4 8281E Requested menu is not an XMENU menu.
5 8282E Requested library is not an XMENU XMENULIB.
6 8283E A menu with this name/alias is already loaded.
7 8284E No XMENU menus are loaded into storage.
8 8285E The menu number passed was not loaded by the MENUEXEC SAVE option.
9 8286E The MENUEXEC SAVEd menu base block does not exist.
24 8275E Missing or invalid parameter specified.
24 8276E Invalid option specified.
24 8277E Menu name not specified.

28 8289E Requested menu was not found.
xx 8287E Error interfacing to :R routine - check return code.
xx 8288E CMS file I/O error occurred - check return code.
100 (Return code following HELP messages.)

Chapter 4. XPSLOAD

The XPSLOAD utility loads an XMENU PSLOAD programmed symbol set file into the terminal for use by the CMS editor, XEDIT. PSLOAD files contain software-defined character sets that can be loaded into the terminal and used under XEDIT, with display control provided by XEDIT's SET COLOR subcommand.

XPSLOAD is an XMENU utility that you can invoke from the CMS command line, from within CMS EXECs, or as a callable module from within higher-level languages.

The format of the XPSLOAD command is shown below:

XPSLOAD	*setname* PSA | PSB | PSC | PSD | PSE | PSF

Where:

setname The name of the PSLOAD symbol set file to be loaded.

PSA | PSB | PSC | PSD | PSE | PSF
Specifies the symbol set identifier used by XEDIT.

Usage notes:

1. To use an XMENU symbol set under XEDIT follow these procedures:
   a. Use the XPSLOAD command to load the symbol set and assign it a symbol set location, for example, XPSLOAD ITALIC12 PSA.
   b. Once in XEDIT, use the SET COLOR subcommand to set appropriate XEDIT areas to the symbol set loaded in step 1, for example, SET COLOR FILEAREA BLUE PSA.
2. When you use the SET COLOR subcommand in XEDIT, be sure to set the color to something other than DEFAULT if you are using a triple-plane (color) symbol set. If the color is set to DEFAULT, the symbol set may not be displayed (the area will look all one color) if the symbol set was assigned to a triple-plane area in the terminal.
3. The terms PSA through PSF in XEDIT do not specify the terminal hardware areas PSA through PSF. The specification of symbol set PS*x* loads the symbol set into logical symbol set PS*x*—not necessarily physical area PS*x* in the terminal.
4. XPSLOAD runs as a nucleus extension.

Return codes and messages:

1 8254E Your terminal is not a 327x.
2 8255E An error occurred attempting to read the symbol set file.
3 8256E The PSLOAD file is not formatted properly.
8 8253E Insufficient storage to run XPSLOAD.
24 8250E Symbol set (PSA to PSF) not specified.
24 8251E Symbol set specified is invalid. It should be "PSA" to "PSF" only.

28 8252E The PSLOAD file specified does not exist.
xx 8257E A terminal I/O error occurred. Check the XMENU return code.
100 (Return code following HELP messages.)

Chapter 5. KWAKEUP

KWAKEUP is an XMENU utility that is used to wait for one or a combination of events to occur, such as a length of time, a specific time, or one or more system interrupts, such as IUCV, VMCF, card reader, or device. KWAKEUP can be invoked from the CMS command line, from within CMS EXECs, or as a callable module from within higher-level languages.

The format of the KWAKEUP command is shown below:

KWAKEUP	[at] [>] *hh*:*mm*[:*ss*] [(Options] +[*hh*][:][*mm*][:][*s*][*s2* ... *s5*] RESET VERSION ? **Options:** CC CONS DIAGMSG EXEC EXT FIFO FILE [(*fn* [*ft* [*fm*]])] IO IUCVMSG LIFO NOENDCMD NOEXT NOREAD QUIET RDR RESETEXT SMSG TIME USEIOUCV VMCF

Where:

hh:*mm*:*ss* **or** >*hh*:*mm*:*ss*

A specific time of day to wait for; *hh* is hours (00-23), *mm* is minutes (00-59), *ss* is seconds (00-59). If this time has already passed, KWAKEUP waits until the specified time on the following day unless > is also specified, in which case KWAKEUP will return right away.

hh and *mm* must be specified; *ss* defaults to zero if not specified. If > is specified, the second digit of *ss* is ignored (due to the standard eight-byte command token limit).

+hh:mm:ss An amount of time KWAKEUP waits before returning. The values of *hh, mm,* and *ss* are the same as described above; however, the command format is slightly different: to specify a number of minutes to wait, you just specify *mm*—for example, +15 for 15 minutes, or +1:30 for a minute and a half; for seconds, you need to specify two leading colons, for example +::99999 will wait 99,999 seconds, the maximum number of seconds; to specify a number of hours you must include a colon "place holder" for both minutes and seconds, and include, as a minimum, 0 seconds—for example, 1::0 for one hour, or 1:01:0 for one hour one minute.

RESET Resets any outstanding KWAKEUP wait conditions.

VERSION Returns the current KWAKEUP version, release, and modification level in the CMS return code. As of this writing, the returned value is 10103, meaning Version 1, Release 1, Modification level 3.

? Types a short description of the KWAKEUP command on your terminal.

Options:

CC Accepted for compatability with the IPF WAKEUP utility; KWAKEUP always uses the clock comparator for timing.

CONS Specifies that KWAKEUP exit on a CMS console interrupt (which KWAKEUP always does). It also specifies that KWAKEUP returns if one or more lines are in the CMS stack.

DIAGMSG Provides diagnostic messages about the inner working of KWAKEUP, for example, the interval of time each line in the wakeup times file (WAKEUP PARMS) implies. See "WAKEUP PARMS file contents" on page 18 for information about this file.

EXEC Specifies that information about an interrupt is provided via EXEC variables rather than stacked lines.

EXT Specifies that KWAKEUP return on any external interrupt.

FIFO Specifies that KWAKEUP stack information first-in, first-out.

FILE *(fn ft fm)* Specifies that KWAKEUP process a file containing events to wait for. *fn* is the filename (default WAKEUP), *ft* is the filetype (default PARMS), *fm* is the filemode (default A). If a specific file is specified, it must be enclosed in parentheses.

IO Specifies that KWAKEUP return on any I/O interrupt.

IUCVMSG Specifies that KWAKEUP return on an IUCV message.

LIFO Specifies that KWAKEUP stack information last-in, first-out.

NOENDCMD Specifies that KWAKEUP not reset its IUCV and VMCF interrupt handlers at CMS end-of-command processing (return to CMS command level).

NOEXT Accepted only for compatibility with the IPF WAKEUP utility; it has no function. If NOEXT is the only operand specified, KWAKEUP returns immediately with a 0 (zero) return code.

NOREAD Specifies that KWAKEUP should not pass a console interrupt attention to CMS. This avoids a possible VM READ condition when ENTER is pressed to end KWAKEUP, but may cause a console hang unless the CP RESET command is issued.

QUIET Specifies that KWAKEUP not display any messages on the terminal.

RDR Specifies that KWAKEUP return on the arrival of a reader file, or exit right away if a reader file is waiting. **Note:** The reader file class must match the virtual card reader's spooled class.

RESETEXT Specifies that KWAKEUP reset its external interrupt handling of VMCF and IUCV interupts. This option is used to explicitly end KWAKEUP handling if NOENDCMD was previously specified.

SMSG Specifies that KWAKEUP return on receipt of a SMSG interrupt.

TIME Specifies that KWAKEUP stack the current date and time. If TIME is the only operand specified, KWAKEUP returns immediately with a 0 (zero) return code.

USEOIUCV Specifies that KWAKEUP use CP IUCV services, even if running under a CMS system that provides CMS IUCV services.

VMCF Specifies that KWAKEUP return on receipt of a VMCF interrupt.

Usage notes:

1. KWAKEUP is a close equivalent to the IPF WAKEUP utility, with a few exceptions:
   - KWAKEUP always uses and requires the clock comparator for timing. Thus, KWAKEUP must be run with ECMODE ON on VM/SP and VM/SP HPO systems.
   - KWAKEUP handles all IUCV initialization before returning on the first call having IUCVMSG as a parameter: you do not need to "prime the pump" with an initial call before processing IUCV interrupts.
   - KWAKEUP provides a mechanism for returning results directly into EXEC variables.
   - KWAKEUP always calculates the earliest possible wakeup time from all time events, even if this time crosses into another day: it is not necessary to place a line in your WAKEUP PARMS file to cross date boundaries.
2. The XMENU/REXX Interface (MENUEXEC) and KWAKEUP share common VMCF interrupt-processing code. Therefore, it is possible to write an EXEC or application that uses both utilities without worrying about one interfering with the other.
3. KWAKEUP can stack multiple records; for example, if FILE, TIME, and SMSG are operands and an SMSG interrupt is received, the following lines (with real data replacing these variable notations) would be stacked in this order (that is, date/time records are written before file records, which precede interrupt records):

   ```
   * MM/DD/YY HH:MM:SS
   * nnnn File line data
   * SMSG user SMSG data
   ```

KWAKEUP event termination

KWAKEUP waits until the first of a list of events occur. On exit, KWAKEUP may stack some data, and create one or more EXEC variables. It always sets a return code to specify the type of interrupt encountered:

CONS Console interrupts cause a return code of 6 to be returned.

EXT External interrupts cause a return code of 8, and cause a line to be stacked containing the following information:

`*EXT interrupt-code`

Or, if the MENUEXEC EXEC command line option was specified, an EXEC variable called EXTCODE is created that contains the interrupt code.

FILE File timer interrupts cause a return code of 3, and cause a line to be stacked containing the following information:

`* record-number file-line`

Or, if the MENUEXEC EXEC command line option was specified, the EXEC variables FLINENO, containing the file line number, and FLINE, containing the file line, are created.

If the FILE command line option is specified, and another interrupt occurs before the file time is reached, the file line is stacked ahead of the data from the later interrupt.

IO I/O interrupts cause a return code of 7, and cause a line to be stacked containing the following information for VM/SP and VM/SP HPO machines:

`*IO device-addr CSW csw1 csw2`

For 370-mode and XA-mode machines, the following information is stacked:

`*IO device-addr SSW ssw1 ssw2 ssw3`

Or, if the MENUEXEC EXEC command line option was specified, an EXEC variable called IOCODE is created containing *device-addr*, and either CSW or SSW with its status words.

IUCVMSG Interrupts from IUCV messages cause a return code of 5, and cause a line to be stacked containing the following information:

`*type userid message-text`

type can be any of the following interrupt types (to receive any of these kinds of messages, the corresponding command(s)/directory entry(ies) must be issued):

Message type	Command/Directory entry
CP - CP responses	SET CPCONIO IUCV
EMSG - Error messages	SET EMSG IUCV
IMSG - Informational messages	SET IMSG IUCV
IUCV - Unknown IUCV type	Directory entry optional
MSG - Messages from other users	SET MSG IUCV
SCIF - Secondary Console messages	Directory entry required
SMSG - SMSGs from other users	SET SMSG IUCV
VM - VM responses	SET VMCONIO IUCV
WNG - Warnings from other users	SET WNG IUCV

Or, if the MENUEXEC EXEC command line option was specified, EXEC variables IUCVUSR, containing the userid sending the message; IUCVTYP, containing the type of IUCV received (see the above list); and IUCV, containing the message text, are created.

RDR Reader interrupts cause a return code of 4 to be returned.

TIME Causes a line in the following format to be stacked:

** MM/DD/YY HH:MM:SS*

Note: This line is stacked first; that is, it is stacked before file or interrupt lines:

** nnnn File line data*
** SMSG user SMSG data*

KWAKEUP EXEC variables

The following EXEC variables can be created by KWAKEUP:

EXTCODE The interrupt code from an external interrupt.

FLINENO The line number of the wakeup times file (WAKEUP PARMS) record causing the interruption.

FLINE The contents of the wakeup times file (WAKEUP PARMS) record causing the interruption.

IOCODE The device address and status from an I/O interrupt.

IUCV The message from the IUCV interrupt.

IUCVTYP The type of IUCV message.

IUCVUSR The user sending the IUCV message causing the interrupt.

SMSG The message from the SMSG or VMCF interrupt.

SMSGUSR The user sending the SMSG or VMCF message causing the interrupt.

WAKEUP PARMS file contents

The wakeup times file—WAKEUP PARMS A by default—is used to specify a number of timer events. KWAKEUP searches through this file and calculates the next interrupt time. This file is then date- or time-stamped with the date or time of the interrupt. Therefore, this file must be placed on a read/write minidisk.

An example of a KWAKEUP wakeup times file is presented below, followed by a detailed description of the format of the file, column by column:

Sample WAKEUP PARMS A

```
*
* Column functions:
* COL 1-8 ALL|mm/dd/yy|MON|TUES|WED|THU, etc.|M-F|S-S|WEEKDAY|WEEKEND|YEARLY|
* COL 10-17 HH:MM:SS (time of the event)
* COL 19-26 MM/DD/YY of last usage (generated by KWAKEUP)
* COL 28-... CP|CMS|EXEC|MSG01 COMMAND to be stacked
*
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
*
ALL      23:59:50 10/24/90 CP SLEEP 2 MIN
ALL      02:00:00 10/24/90 CP PURGE SYSTEM PRT CL T
ALL      02:00:00 10/24/90 CP PURGE SYSTEM PUN CL T
ALL      02:00:00 10/24/90 CP ACNT ALL CLOSE
*
==/01/== 02:00:00 10/01/90 CP AUTOLOG PROD # AUTOTIME
==/15/== 02:00:00 10/15/90 CP AUTOLOG PROD # AUTOTIME
ALL      04:10:00 10/24/90 EXEC ALARMCLN
ALL      07:00:10 10/24/90 CP AUTOLOG ACCOUNT ACCOUNT
WED      10:00:00 10/24/90 CMS COPY ALARM LOG A ALARM OLDLOG A (REP OLDD
WED      10:00:00 10/24/90 CMS ERASE ALARM LOG A
ALL      19:00:00 10/23/90 CP ACNT ALL CLOSE
```

The WAKEUP PARMS file may be either fixed or variable format and must follow this column format:

Col 1-8 Specify the date of the event. If an asterisk appears in column 1, this record is considered to be a comment and is ignored.

Otherwise, the following date formats are supported:

ALL Specifies that this record is processed every day.

MM/DD/YY Indicates a specific date. Any of *MM* (month), *DD* (day), or *YY* (year) can be specified as ==, which means that any month, day, or year matches this record. For example, 12/==/== means every day in December, 01/01/== means once a year on New Year's Day, ==/15/== means on the 15th of every month, and so on. Both specifiers for month (*MM*), day (*DD*), or year (*YY*) must be either digits or equal signs.

To improve performance, KWAKEUP changes the first slash into a period when this record can never be run again, for example, after a passed date.

MONDAY... Specifies that this record is processed once a week. Valid names, with minimum abbreviations capitalized, are SUNday, MONday, TUEsday, WEDnesda, THUrsday, FRIday, SATurday and SUNday. Because

only eight bytes are inspected, you may include the ending Y for Wednesday, if desired.

MONTHLY Specifies that this record is processed once a month (as close to the start of the month as possible).

M-F Specifies that this record is processed on weekdays. M-F is synonymous with WEEKDAY.

S-S Specifies that this record is processed on weekends. S-S is synonymous with WEEKEND.

WEEKDAY Specifies that this record is processed on weekdays. WEEKDAY is synonymous with M-F.

WEEKEND Specifies that this record is processed on weekends. WEEKEND is synonymous with S-S.

YEARLY Specifies that this record is processed once a year (as close to the start of the year as possible).

Col 10-17 Specify the time of the event.

The following time formats are supported:

HH:MM:SS Specifies an exact time.

+delta Specifies that the event occurs every *delta* time units starting at *delta* past midnight. *delta* can be specified in the following ways: for a number of minutes to wait, specify *mm*—for example, +15 for every 15 minutes, +1:30 for every minute and a half; to specify only seconds, use two leading colons—for example, +::05 for every 5 seconds, +::99999 for every 99,999 seconds, the maximum number of seconds; to specify a number of hours, include a colon "place holder" for both minutes and seconds, and include, as a minimum, 0 seconds—for example, 1::0 for every hour, or 1:15:0 for every hour and fifteen minutes.

Col 19-26 Reserved for KWAKEUP to time- or date-stamp this record. The record is date-stamped for specific time records, and time-stamped for delta time records. This field should be initially left blank.

Col 28-... Reserved for user's use. Place any data to be stacked into this area of the record, for example, a command to be processed.

Return codes:

0 Nothing to wait for.
1 A VMCF or SMSG interrupt occurred.
2 The command specified time interval has elapsed.
3 A time request from the wakeup times file has occurred.
4 A reader file exists or has arrived.
5 An IUCV message has arrived.
6 A console interrupt was generated.
7 An I/O interrupt has occurred.
8 An external interrupt has occurred.
100 Returned after help messages.
10x Error processing card reader, x is the RDCARD return code.

Other return codes match the message numbers of, and relate to, the following messages.

Messages:

9060E Non-numeric digits were specified in the HH:MM:SS time value.
9061E ">" or "+" was specified without HH:MM:SS.
9062E The hours specified in HH:MM:SS are greater than 23.
9063E The minutes specified in HH:MM:SS are greater than 59.
9064E The seconds specified in HH:MM:SS are greater than 59.
9065E There is insufficient memory to allocate a VMCPARM block.
9066E There is insufficient memory to allocate VMCF/SMSG buffers.
9067E Unable to authorize VMCF/SMSG.
9068E A VMCF/SMSG interrupt was received without a VMCPARM block.
9069E Unknown VMCF/SMSG data was received.
9070E ECMODE must be ON to use KWAKEUP's timing facilities.
9071E Unable to perform "IUCV QUERY."
9072E Insufficient storage to allocate IUCV blocks.
9073E Unable to perform "IUCV DECLARE BUFFER."
9074E Unable to perform "IUCV CONNECT" - RC contains IUCV error code.
9075E Unknown IUCV data/message received.
9076E Unable to perform "IUCV RECEIVE" - RC contains IUCV error code.
9077E Unable to preform "HNDIUCV SET" - RC contains macro error code.
9078E Unable to perform "CMSIUCV CONNECT" - RC contains macro error code.
9079E Insufficient storage to allocate an IUCV queued block.
9080E An IUCV interrupt was received but there are no queued IUCV blocks.
9081E Date specification error in line x of KWAKEUP file.
9082E Time specification error in line x of KWAKEUP file.
9083E There is insufficient storage to process the KWAKEUP file.
9084E The KWAKEUP file must be on a read/write disk.
9085E A relative time was specified with WEEKLY/MONTHLY on line x.
9086E An error occurred reading the KWAKEUP file, check FSREAD return code.
9087E An error occurred updating the KWAKEUP file, check FSWRITE return code.

Chapter 6. KOLMSINS

The KOLMSINS command loads XMENU message files, or other read/only files, into storage for faster access.

By using alias files, KOLMSINS can be used to load national language message files for XMENU.

The format of the KOLMSINS command is shown below:

KOLMSINS	DROP *filename* [*filetype* [*filemode*]] | * LOAD *filename* [*filetype* [*filemode*]] [(ALIAS *name*] MAP

Where:

DROP Specifies that a file or all files are to be dropped from storage.

filename The filename of the file to be loaded into or dropped from storage.

filetype The filetype of the file to be loaded into or dropped from storage.

filemode The filemode of the file to be loaded into or dropped from storage.

***** Specifies that all files are to be dropped from storage.

LOAD Specifies the file to be loaded into storage.

ALIAS *name* Defines an alias for the filename of the loaded file; the filetype is assumed to be MESSAGES. This filename is used to access the file once it is loaded. If not specified, KOLMSINS uses the same filename as the loaded file. This option allows you to use files of messages in different languages without requiring CMS or CP National Language Support. For example, to provide users German language XMENU messages (that you could create in a file called GX MESSAGES), you would employ the aliasing facility to load and point to the GX file whenever calls for XMENU MESSAGES are made:

```
KOLMSINS LOAD GX MESSAGES * (ALIAS XMENU
```

MAP Displays a map of files in storage.

Usage notes:

1. KOLMSINS provides fast access to messages and provides National Language Support to XMENU. Menu information is displayed on the terminal.

Chapter 7. KBROWSE, KOLCMSBR, and KOLCPBR

This chapter describes KBROWSE and its related utilities: KOLCMSBR and KOLCPBR.

These utilities permit the browsing of CMS files, tapes, and spool files, via KBROWSE, and in-storage data in the form of CMS and CP command responses, via KOLCMSBR and KOLCPBR.

The CMS command syntax for each of these utilities is given first, followed in alphabetical order by each KBROWSE subcommand.

KBROWSE command

The KBROWSE command is used to view the contents of CMS files, spool files, or tapes. CMS and CP commands may also be entered while using KBROWSE.

The individual KBROWSE subcommand descriptions are presented in alphabetical order later in this section. You may view these subcommand descriptions online in CMS HELP files by entering `HELP KBROWSE MENU`.

The format of the KBROWSE command is shown below:

KBROWSE	[*fn ft* [*fm*]] [(Options] **Options:** MEMBER *membername* OPEN | OPN [*file-number*] NOPROFIL PROFILE *filename* PRT | PRINTER *file-number* PUN | PUNCH *file-number* RDR | READER [*file-number*] SUBCMD *first-browse-subcommand* TAPE [*address*]

Where:

fn ft The optional filename and filetype of the file to be browsed. If a file with the specified filename and filetype does not exist, an error message is issued. If no filename and filetype are issued, KBROWSE enters the last file browsed. If no file has previously been browsed, an error message is issued. The BROWSE subcommand, issued by itself while under KBROWSE, is used to sequentially display the ring of browsed files.

fm The optional filemode of the file to be browsed. If omitted or specified as an asterisk (*) all accessed disks are searched. The first occurance will be the file that is browsed.

Options:

MEMBER *membername*

Specifies the member of a MACLIB, LOADLIB, TXTLIB or XMENULIB is to be browsed. MEMBER can only be specified with CMS files.

OPEN | OPN *file-number*

Specifies which of your virtual reader files is to be browsed. If *file-number* is not specified, the first file in your reader queue is browsed. Only the displayed data is read into storage; that is, KBROWSE speeds file display by only reading one screen at a time into virtual storage. The OPEN option is synonymous with the RDR option.

NOPROFIL Specifies that no browse profile should be executed.

PROFILE *filename*
Specifies the profile to be executed before this level of browse is entered. If neither this nor NOPROFIL are specified, the commands in file PROFILE KBROWSE are executed.

PRT *file-number*
Specifies which of your virtual printer files is to be browsed. Only the displayed data is read into storage; that is, KBROWSE speeds file display by only reading one screen at a time into virtual storage.

PUN *file-number*
Specifies which of your virtual punch files is to be browsed. Only the displayed data is read into storage; that is, KBROWSE speeds file display by only reading one screen at a time into virtual storage.

RDR *file-number*
Specifies which of your virtual reader files is to be browsed. If *file-number* is not specified, the first file in your reader queue is browsed. Only the displayed data is read into storage; that is, KBROWSE speeds file display by only reading one screen at a time into virtual storage. RDR is equivalent to the OPEN option.

SUBCMD *first-browse-subcommand*
Passes the first KBROWSE subcommand at KBROWSE invocation. The remainder of the command line is used to enter the subcommand; therefore, if SUBCMD is specified with other options, it should be the last option specified.

TAPE *address* Specifies that the tape attached to your virtual machine at address *address* is to be browsed. If *address* is not specified, tape 181 is browsed. Only the displayed data is read into storage; that is, KBROWSE speeds file display by only reading one screen at a time into virtual storage.

Usage notes:

1. KBROWSE should be used (rather than XEDIT, for example) if you want to view a file quickly without loading it into virtual storage.
2. KBROWSE automatically browses CMS packed files and reader RECEIVE-format files in their readable format. You can use KBROWSE and SAVE to read RECEIVE-format files, although the file will remain in the reader, and the RECEIVE logs aren't updated.
3. Any KBROWSE subcommand can be issued from KBROWSE. On return, the browsed file is redisplayed at its current position.
4. KBROWSE saves and restores the CMS read and write file pointers so that the browsing of a file doesn't affect files in use by the application being debugged. If a file is active and KBROWSE is called to display it for the first time, the current line is positioned to the active read pointer location.
5. Many files can be simultaneously browsed. When the KBROWSE command is entered by itself, without specifying a file or tape, KBROWSE moves to the next file in the browsed data ring.

Browsing data in memory

It is possible to view data within memory by using KBROWSE. This is done by passing KBROWSE a special parameter list, identical to that run in an EXEC 2 or REXX EXEC that has been loaded into storage. This interface is described in the section on Using the Extended Parameter List in the *System Product Interpreter Reference*, in the chapter on System Interfaces.

Return codes:

200 Insufficient storage to allocate KBROWSE blocks.
201 Invalid option specified.
202 Duplicate option found.
203 Profilename > eight characters.
204 File number missing or invalid for RDR, PRT, PUN.
205 File member name missing or invalid.
206 Cannot specify MEMBER with RDR, PRT, PUN or TAPE.

Return codes and messages:

40 040E File to be browsed was not found.
41 041E File to be browsed has an invalid filename.
42 042E Error xxx browsing file.
43 043E No active browsed files.
44 044E Value contains invalid decimal character(s).
47 047E Data searched for not found.
53 053E Command terminated by user interrupt.

KOLCMSBR command

The KOLCMSBR command is used to issue a CMS command or run an EXEC, trap any CMS output, and then display the output after KBROWSE exits.

The format of the KOLCMSBR command is shown below:

KOLCMSBR	*command-string*

Where:

command-string
The CMS command or EXEC to be executed.

Usage notes:

1. This utility is useful when you wish to issue a CMS command or run an EXEC that produces a significant amount of output. Use KOLCMSBR if you wish to capture and review the output later, after exiting KBROWSE.
2. KOLCMSBR traps WRTERM SVCs and BALRs, and LINEWRT SVCs. Output displayed any other way, for example, with full-screen I/O, is not trapped.
3. If you want to see CP output, use KOLCPBR instead.
4. See also the KBROWSE CMS subcommand.

Return codes:

The return code from the command issued is the final return code.

KOLCPBR command

The KOLCPBR command is used to issue a CP command trap any CP output, and then display the output after KBROWSE exits.

The format of the KOLCPBR command is shown below:

KOLCPBR	*command-string*

Where:

command-string
The CP command to be executed.

Usage notes:

1. This utility is useful when you wish to issue a CP command that produces a significant amount of output. Use KOLCPBR if you wish to capture and review the output later, after exiting KBROWSE.
2. KOLCPBR traps CP output via DIAGNOSE X'08'. Output displayed any other way, for example, CMS console output, is not trapped.
3. If you want to see CMS output, use KOLCMSBR instead.
4. See also the KBROWSE CP subcommand.

Return codes:

The return code from the command issued is the final return code.

& (Ampersand) subcommand

The & (ampersand) KBROWSE subcommand will leave itself and the text following it in the command input area of a menu.

The format of the & subcommand is shown below:

&	*text*

Where:

text The character string to be left in the command input area.

Usage notes:

1. Because the ampersand and accompanying text are left in the command input area, the & subcommand allows easy repetitive entering of a command (such as LOCATE) by simply pressing the ENTER key again.
2. *text* is executed as a command. If *text* is not a KBROWSE QUERY subcommand, the text and ampersand will remain in the command input area when the subcommand completes execution. The response of KBROWSE QUERY subcommands are placed in the command input area of the menu, thus overlaying the command.
3. The & subcommand can be removed by placing the cursor at the beginning of the command line and pressing the ERASE EOF key.

? (Question Mark) subcommand

The ? (retrieve) subcommand will retrieve the previous entry in the command stack retrieve buffer and return it to the command line of a menu.

The format of the ? subcommand is shown below:

?	

Usage notes:

1. A retrieve stack is kept internally in KBROWSE. Previous entries can be retrieved via the ? or "retrieve last subcommand" subcommand, and be returned to, or re-displayed on, the command line of the current menu. This function is equivalent to the CP RETRIEVE function. The retrieve stack is 256 bytes long. The number of entries present depends on the length of the strings entered.
2. Retrieving commands with the ? subcommand returns them to the command line, or re-displays them, and allows them to be corrected and re-entered.
3. Pressing the CLEAR key resets the RETRIEVE stack to only its most recent entry.
4. For locally attached, non-SNA, non-bisync terminals that have a TEST REQ or SYS REQ key, pressing TEST REQ or SYS REQ will also retrieve the last subcommand.
5. Any text following the question mark is ignored.
6. If more than one question mark appears on the line, line *n* of the retrieve stack is retrieved, where *n* is the number of question marks present.
7. The Retrieve (?) and Repeat (=) subcommands are not stacked for retrieval.

= (Equal Sign) subcommand

The = (repeat) subcommand will retrieve the last entry in the command stack retrieve buffer and submit it to the command processor.

The format of the = subcommand is shown below:

=	

Usage notes:

1. A retrieve stack is kept internally in KBROWSE. Entries can be re-executed via the = or "repeat last subcommand" subcommand, or returned to the command line without execution via the ? or "retrieve" subcommand. The RETRIEVE stack is 256 bytes long. The number of entries present depends on the length of the strings entered.
2. Repeating invalid commands returns the invalid command to the command line and the terminal alarm, if present, is sounded to alert the user.
3. Any text following the equal sign is ignored.
4. The Retrieve (?) and Repeat (=) subcommands are not stacked for retrieval.

BACKWARD subcommand

The BACKWARD subcommand scrolls one or more screens toward the top of the browsed data file.

The format of the BACKWARD subcommand is shown below:

BAckward	[*n* | *]

Where:

n | * — The optional number of screens to scroll backward. If not specified, *n* defaults to one. If * is specified, the top of the data file is displayed.

Usage notes:

1. This subcommand is used to move backward (toward the top) of the displayed data one screen at a time. Together, the BACKWARD and FORWARD subcommands are used to quickly scan data one screen at a time.
2. If BACKWARD is entered at the top of data file, no movement occurs.
3. BACKWARD is normally assigned to PF7/PF19.

BOTTOM subcommand

The BOTTOM subcommand makes the last line of data the current line.

The format of the BOTTOM subcommand is shown below:

BOttom	

Usage notes:

This subcommand is used to move quickly to the end of the data.

BROWSE subcommand

The BROWSE subcommand is used to view the contents of CMS files, spool files, or tapes without leaving the KBROWSE environment. Once in the KBROWSE environment, the BROWSE subcommand is equivalent to KBROWSE; that is, BROWSE and all the other KBROWSE subcommands and CMS and CP commands may be entered.

The format of the BROWSE subcommand is shown below:

BRowse	[*fn ft* [*fm*]] [(Options] **Options:** MEMBER *membername* NOPROFIL PROFILE *filename* PRT | PRINTER *file-number* PUN | PUNCH *file-number* RDR | READER [*file-number*] SUBCMD *first-browse-subcommand* TAPE [address]

Where:

fn ft The optional filename and filetype of the file to be browsed. If a file with the specified filename and filetype does not exist, an error message is issued. If no filename and filetype are issued, BROWSE enters the last file browsed. If no file has previously been browsed, an error message is issued. BROWSE issued by itself while under KBROWSE is used to sequentially display the ring of browsed files.

fm The optional filemode of the file to be browsed. If omitted or specified as an *, all accessed disks are searched. The first occurence is the file to be browsed.

Options:

MEMBER *membername*
Specifies the member of a MACLIB, LOADLIB, TXTLIB, or XMENULIB to be browsed. MEMBER can only be specified with CMS files.

NOPROFIL Specifies that no browse profile should be executed.

PROFILE *filename*
Specifies the browse profile to be executed before this level of browse is entered.

PRT *file-number*
Specifies which of your virtual printer files should be browsed. Only the displayed data is read into storage.

PUN *file-number*

Specifies which of your virtual punch files should be browsed. Only the displayed data is read into storage.

RDR *file-number*

Specifies which of your virtual reader files should be browsed. If *file-number* is not specified, the first file in your reader queue is browsed. Only the displayed data is read into storage.

SUBCMD *first-browse-subcommand*

Passes the first KBROWSE subcommand at BROWSE invocation. The remainder of the command line is treated as the subcommand; therefore if SUBCMD is specified with other options, it should be the last option specified.

TAPE *address* Specifies the tape attached to your virtual machine to be browsed. If *address* is not specified, the tape attached at virtual address 181 is browsed. Only the displayed data is read into storage.

Usage notes:

1. The BROWSE subcommand is used to view various types of data files under KBROWSE, KFLIST, KRDRLIST, KPRTLIST, and KPUNLIST.
2. Any KBROWSE subcommand can be issued from BROWSE. On return, the browsed file is redisplayed at its current position.
3. BROWSE should be used (rather than XEDIT, for example) so that you can easily go between browsing and debugging without losing your position in the browsed file.
4. BROWSE saves and restores the CMS read and write file pointers so that the browsing of a file doesn't affect files in use by the application being debugged. If a file is active and BROWSE is called to display it for the first time, the current line is positioned to the active read pointer location.
5. Many files can be simultaneously browsed. When the BROWSE subcommand is entered by itself, BROWSE moves to the next file in the browsed data ring.

Return codes:

200 Insufficient storage to allocate BROWSE blocks.
201 Invalid option specified.
202 Duplicate option found.
203 Profilename > eight characters.
204 File number missing or invalid for RDR, PRT, PUN.
205 File member name missing or invalid.
206 Cannot specify MEMBER with RDR, PRT, PUN or TAPE.

Return codes and messages:

40 040E File to be browsed was not found.
41 041E File to be browsed has an invalid filename.
42 042E Error xxx browsing file.
43 043E No active browsed files.
44 044E Value contains invalid decimal character(s).
47 047E Data searched for not found.

53 053E Command terminated by user interrupt.

BSF subcommand

The BSF subcommand will backspace a browsed tape one file.

The format of the BSF subcommand is shown below:

BSF	

Usage notes:

1. A tape can consist of one or more physical files, separated by tape marks. This command moves the tape back to the previous file (if there is one).
2. When you use BSF, the tape is positioned at the beginning of the previous physical tape file, or the beginning of the tape if there are no previous files. Use TOP or BOTTOM to go to the beginning or end of the current physical tape file.
3. Browsed tapes may be positioned using BSF, FSF, FST, and REW.

CANCEL subcommand

The CANCEL subcommand will terminate all levels of KBROWSE.

The format of the CANCEL subcommand is shown below:

CANCel	

Usage notes:

1. If any program was called between levels of KBROWSE, the CANCEL process will stop at that level. When this program ends, the CANCEL process will continue. This will continue until KBROWSE is completely ended.
2. The CANCEL subcommand is synonymous with QQUIT.

CMDLINE subcommand

The CMDLINE subcommand displays a message on the command line.

The format of the CMDLINE subcommand is shown below:

CMDLINE	*message*

Where:

message A character string to be displayed on the command line.

Usage notes:

1. This subcommand allows you to display a message on the command line, for example, the result of a macro.
2. If you want the command line's Modified Data Tag (MDT) set, for example, to allow the user to reenter the message as a command, use the CMSG subcommand.

CMS subcommand

The CMS subcommand causes CMS to execute the specified command or, if no command is specified, to enter CMS SUBSET.

The format of the CMS subcommand is shown below:

CMS	[*command*]

Where:

command The optional command to be passed to CMS. If CMS is entered without *command*, CMS subset is entered. Use the CMS command RETURN to return to XMENU from CMS subset.

Usage notes:

1. If the program you call aborts, you will also abort KBROWSE.
2. If the called program displays no output, the last KBROWSE screen is displayed. If the called program displayed line-mode output, a MORE... condition will be displayed by the terminal. If this happens, you can press either PA2 or CLEAR, or wait one minute for the KBROWSE screen to be restored.
3. If the return code from the called program was non-zero, it will be displayed in menu field CMDPREF (if it exists). The terminal alarm will be sounded and the offending command will be redisplayed in the menu's command field CMDLINE (if it exists) so that it can be corrected. If you do not wish to correct the command, place the cursor at the start of the menu's command field, press ERASE EOF and then ENTER.
4. See also "KOLCMSBR command" on page 27.

Responses:

Refer to the IBM *CMS Command Reference* manual that applies to your release of CMS.

Messages:

Refer to the IBM *System Messages and Codes* manual that applies to your version of VM.

Return codes:

Refer to the IBM *System Messages and Codes* manual that applies to your version of VM.

CMSG subcommand

The CMSG subcommand displays a message on the command line. This subcommand also sets the command line field's Modified Data Tag (MDT).

The format of the CMSG subcommand is shown below:

CMSG	*message*

Where:

message A character string to be displayed on the command line.

Usage notes:

1. This subcommand allows you to display a message on the command line, for example, the result of a macro.
2. This command also sets the command line field's Modified Data Tag (MDT) so that, if the user presses ENTER, *message* will be executed as a command. Use CMDLINE if you want to display a message on the command line without having the MDT set.

COLUMN subcommand

The COLUMN subcommand scrolls the data so that the specified column is displayed as the leftmost column of the menu.

The format of the COLUMN subcommand is shown below:

COLumn	[*n*]

Where:

n The optional column number to display as the leftmost column on the screen. If not specified, *n* defaults to 1 (one).

Usage notes:

1. This subcommand is used to quickly position data to a specific column.
2. The COLUMN subcommand is synonymous with VERIFY.

CP subcommand

The CP subcommand submits the accompanying command to CP.

The format of the CP subcommand is shown below:

CP	[*command*]

Where:

command The optional command to be passed to CP. If the CP subcommand is entered without *command*, CP is entered and `CP READ` is displayed.

Usage notes:

1. If the CP subcommand is entered without *command*, `CP READ` is entered. Enter the command `Begin` to return to KBROWSE.
2. If the command displays no output, the last KBROWSE menu is displayed. If the command displays line-mode output, a `MORE...` condition will be displayed by the terminal. If this happens, you can press either PA2 or CLEAR, or wait one minute for the BROWSE menu to be restored.
3. If the return code from the command is non-zero, it will be displayed in menu field CMDPREF (if it exists). The terminal alarm will be sounded and the offending command will be redisplayed in the command menu field CMDLINE (if it exists) so that it can be corrected. If you do not wish to correct the command, place the cursor at the start of the command field, press ERASE EOF and then ENTER.
4. See also "KOLCPBR command" on page 28.

Responses:

Refer to the IBM *System Messages and Codes* manual or the *CP Command Reference* that applies to your version of VM.

Messages:

Refer to the IBM *System Messages and Codes* manual or the *CP Command Reference* that applies to your version of VM.

Return codes:

Refer to the IBM *System Messages and Codes* manual or the *CP Command Reference* that applies to your version of VM.

CURSLINE subcommand

The CURSLINE command makes the line on which the cursor is positioned the current line.

The format of the CURSLINE subcommand is shown below:

CURSline	

Usage notes:

1. The CURSLINE subcommand is used to make the line on which the cursor is positioned the current line.
2. PF6/PF18 is normally assigned to CURSLINE.

CURSOR subcommand

The CURSOR subcommand appends the non-blank character string at the current cursor position to the command following the keyword CURSOR. The resulting string is then executed as a command.

The format of the CURSOR subcommand is shown below:

Cursor	[*command*]

Where:

command The initial portion of the command to be submitted. If this is not specified, the contents of the command line are used as the command.

Usage notes:

1. The CURSOR subcommand is normally used to pass displayed data from a higher level of BROWSE as a parameter to a called program or a lower level of BROWSE.
2. The CURSOR subcommand, together with the command to be executed, is usually assigned to a PF key. Then, when the cursor is positioned on some menu data and the PF key is pressed, the command is executed with the menu data appended as an argument, parameter, or option of the subcommand.

DICT subcommand

The DICT subcommand will display the member list of a MACLIB, TXTLIB, LOADLIB, or XMENULIB.

The format of the DICT subcommand is shown below:

DICT	

Usage notes:

1. When you display the member directory, you are placed into a new KBROWSE level. When you enter QUIT, you return to the previous level.
2. Members are shown sorted alphabetically. All KBROWSE subcommands are functional within the display of the member dictionary, such as LOCATE and FIND.
3. While displaying a member you can enter DICT to choose a different file member; you don't have to return to the highest level of KBROWSE to do so.
4. If you attempt to use DICT on a file that isn't a library, the subcommand is ignored.

DISCARD subcommand

The DISCARD subcommand leaves this level of KBROWSE and erases the browsed file if possible.

The format of the DISCARD subcommand is shown below:

DISCARD	

Usage notes:

1. The DISCARD subcommand erases the browsed file if possible and leaves this KBROWSE level.
2. DISCARD will erase CMS files and spool files. It will not erase tape files, SQL/DS database tables, or OS datasets.
3. Use the PURGE subcommand to remove this data from the browsed data ring without erasing the file.

DOWN subcommand

The DOWN subcommand scrolls one or more lines toward the bottom of the browsed data.

The format of the DOWN subcommand is shown below:

DOwn	[*n* | *]

Where:

n | * — The optional number of lines to scroll. If not specified, *n* defaults to 1 (one). If * is specified, the bottom of the data is displayed.

Usage notes:

1. This subcommand is used to move forward in (toward the bottom of) the displayed data one or more lines (rows, or records) at a time. It is equivalent to the *+nn* function of the LOCATE subcommand.
2. The DOWN subcommand is synonymous with NEXT.

ERMSG subcommand

The ERMSG subcommand displays a message on the menu title line.

The format of the ERMSG subcommand is shown below:

ERMSG	*message*

Where:

message A character string to be displayed on the menu title line.

Usage notes:

1. This subcommand allows you to display a message, for example, an error message, on the menu title line.

FADDR subcommand

The FADDR subcommand is used to find the next line in an assembly listing that starts with the specified string, ignoring carriage control.

The format of the FADDR subcommand is shown below:

FADDR	[*line*]

Where:

line The string to be searched for, to be found starting in column one. *line* must be separated from the subcommand by a single blank.

Usage notes:

1. The string to be searched for (*line*) begins following (but not including) the single blank separating it from the FADDR subcommand. Any blank character in the string will match any character in the corresponding column position. An underscore (_) character in the string will match a blank in the corresponding column position.
2. FADDR proceeds toward the end of the browsed data. If the the string is not matched when the last line of data is reached, the search wraps, a message is displayed, and the search continues until the string is found or until the line on which the search began is reached.
3. If the string is not found, an error message is displayed.
4. FADDR is not affected by SET ZONE.
5. FADDR differs from FIND in that it starts at column one even if the file contains carriage control. FADDR can be used to find addresses in assembler listings.

FIND subcommand

The FIND subcommand is used to find the next line in the file that starts with the specified string.

The format of the FIND subcommand is shown below:

Find	[*line*]

Where:

line The string to be searched for, starting in column one. *line* must be separated from the subcommand by a single blank.

Usage notes:

1. The string to be searched for (*line*) begins following (but not including) the single blank separating it from the FIND subcommand. Any blank character in the string will match any character in the corresponding column position. An underscore (_) character in the string will match a blank in the corresponding column position.
2. FIND proceeds toward the end of the browsed data.
3. If SET WRAP ON is in effect and the string is not matched when the last line of data is reached, the search wraps, a message is displayed, and the search continues until the string is found or until the line on which the search began is reached. If SET WRAP OFF is in effect, the search terminates at the end of the file.
4. If the string is not found, an error message is displayed.
5. FIND is not affected by SET ZONE.
6. FINDUP is used to search the data in a backward direction.

FINDUP subcommand

The FINDUP subcommand is used to find the next previous line in the file that starts with the specified string.

The format of the FINDUP subcommand is shown below:

FINDUp	[*line*]

Where:

line The string to be searched for starting in column one. *line* must be separated from the subcommand by a single blank.

Usage notes:

1. The string to be searched for (*line*) begins following (but not including) the single blank separating it from the FINDUP subcommand. Any blank character in the string will match any character in the corresponding column position. An underscore (_) character in the string will match a blank in the corresponding column position.
2. FINDUP proceeds toward the beginning of the browsed data.
3. If `SET WRAP ON` is in effect and the string is not matched when the first line of data is reached, the search wraps, a message is displayed, and the search continues until the string is found or until the line on which the search began is reached. If `SET WRAP OFF` is in effect, the search terminates at the top of the file.
4. If the string is not found, an error message is displayed.
5. FINDUP is not affected by `SET ZONE`.
6. FIND is used to search the data in a forward direction.

FLABEL subcommand

The FLABEL subcommand is used to find the next line in an assembly listing that matches the specified string in column 40 (the LABEL field), ignoring carriage control.

The format of the FLABEL subcommand is shown below:

FLABEL	[*line*]

Where:

line The string to be searched for, to be found starting in column 40. *line* must be separated from the subcommand by a single blank.

Usage notes:

1. The string to be searched for (*line*) begins following (but not including) the single blank separating it from the FLABEL subcommand. Any blank character in the string will match any character in the corresponding column position. An underscore (_) character in the string will match a blank in the corresponding column position.
2. FLABEL proceeds toward the end of the browsed data. If the the string is not matched when the last line of data is reached, the search wraps, a message is displayed, and the search continues until the string is found or until the line on which the search began is reached.
3. If the string is not found, an error message is displayed.
4. FLABEL is not affected by `SET ZONE`.
5. FLABEL differs from FIND in that it starts at column 40. FLABEL can be used to find assembler labels in assembler listings.

FORWARD subcommand

The FORWARD subcommand scrolls one or more screens toward the end of the browsed data.

The format of the FORWARD subcommand is shown below:

FOrward	[*n* | *]

Where:

n | * — The optional number of screens to scroll. If not specified, *n* defaults to one. If * is specified, the bottom of the data is displayed.

Usage notes:

1. This subcommand is used to move forward in (toward the bottom of) the displayed data one screen at a time. Together, the FORWARD and BACKWARD subcommands are used to quickly scan the data one screen at a time.
2. If FORWARD is entered at the bottom of the file, no movement occurs.
3. FORWARD is normally assigned to PF8/PF20.

FSF subcommand

The FSF subcommand will forward space one file in a browsed tape.

The format of the FSF subcommand is shown below:

FSF	

Usage notes:

1. A tape can consist of one or more physical files, separated by tape marks. This command allows you to go to the next file (if there is one).
2. When you use FSF, the tape is positioned at the beginning of the next physical tape file, or, if there are no files following the current file, at the end of the tape. Use TOP or BOTTOM to go to the beginning or end of the current physical tape file.
3. If you reach two consecutive tape marks, you get an end-of-tape message. Attempting to view past the end of the tape may result in I/O error messages or the tape spinning off the reel.
4. Browsed tapes may be positioned using BSF, FSF, FST, and REW.

FST subcommand

The FST subcommand will forward space to the end of a browsed tape (two consecutive tape marks).

The format of the FST subcommand is shown below:

FST	

Usage notes:

1. A tape can consist of one or more physical files, separated by tape marks. This command allows you to go to the end of all files.
2. When two tape marks are encountered, an end-of-tape message is displayed.
3. If you FST past the end of tape, it may result in an error message or in the tape spinning off the reel.
4. Tapes may be positioned using BSF, FSF, FST, and REW.

HELP subcommand

The HELP subcommand is used to display information about XMENU subcommands, keys, messages, and control file subcommands. Under certain applications the HELP PF key (PF01/PF13), together with the cursor position, is used to display more information about a menu field. The appropriate HELP information, provided it is available, is displayed.

The format of the HELP subcommand is shown below:

Help	[MENU | *topic*]

Where:

MENU Displays a menu containing a list of HELP topics. Each topic can be displayed from this menu. This is the default.

topic Specifies the subcommand for which HELP information is to be displayed.

Usage notes:

1. Abbreviations for HELP topics cannot be used. The topic menu can be displayed (HELP MENU) to determine the full name of a topic.
2. For KBROWSE, the topic name for PF key definitions is "PFKEYS", for return codes is "RETCODES", for messages is "MESSAGES", and for program options is "OPTIONS".
3. Normally, HELP is assigned to PF1/PF13.
4. KBROWSE uses the CMS HELP facility to display online product information. This information is in CMS HELP file format. If HELP is entered from the command line without any operands, the main KBROWSE HELP selection menu will be displayed. Place the cursor on a topic name on the main KBROWSE HELP selection menu and press ENTER (or PF1/PF13) to receive detailed information about that KBROWSE topic. If the optional *topic* is entered, the main selection menu is bypassed and HELP for that topic is displayed directly.
5. KBROWSE issues the CMS command HELP. If your installation has modified the IBM HELP command, the KBROWSE HELP may not function properly.

Return codes and messages:

See the CMS HELP command description.

LEFT subcommand

The LEFT subcommand shifts the menu one or more characters toward the first, or leftmost, column of the browsed data.

The format of the LEFT subcommand is shown below:

LEft	[*n* | *]

Where:

n | *
: The optional number of columns to shift. If not specified, *n* defaults to 1 (one). If * is specified, the leftmost characters of the data are displayed. LEFT 0 performs no movement.

Usage notes:

1. This subcommand is used to shift the menu to see data on lines that are too long to fit on the menu. The LEFT subcommand allows you to see data that is to the left of the first column currently displayed on the menu. Data moves to the right, exposing the specified number of columns to the left of the current first column.
2. The menu cannot be scrolled past column one, or the leftmost data.
3. Use the VERIFY and COLUMN subcommands to move to a specific data column.
4. PF10/PF22 is normally assigned to LEFT 20.
5. The RIGHT subcommand is used to move in the opposite direction.

LOCATE subcommand

The LOCATE subcommand is used to find the preceding or next occurance in the browsed data of a specified string or other search criteria.

The format of the LOCATE subcommand is shown below:

[Locate	[*pattern*]

Where:

Locate The subcommand name, but is optional. When the LOCATE subcommand is entered implicity, the starting string delimiter character (*sep1* in the *pattern* description below) must be a slash (/).

pattern The string to be located and/or the search criteria to be used. *pattern* can be entered in several forms:

1. :*nn* locates record number *nn*. If *nn* is *, the last data record is located.
2. +|– *nn* moves the current line pointer down (+) or up (–) *nn* lines. (+|– * goes to the bottom or top of the file, respectively.)
3. String search criteria take the form:

   ```
   [+|-] [¬] [sep1] ...
   [string1] [sep1] [&] [|] [¬] [sep2] [string2] [sep2]
   ```

   Where:

   + or – Specifies the direction of the search. If – is entered, the search moves toward the top of the data. If + is specified, or if the leading + or – sign is omitted, the search moves toward the bottom of the data. The first occurrence of + or – in a locate string is always used as a search direction indicator; therefore if + or – is to be used as a string delimiter, the direction character + or – must be specified.

   ¬ Denotes that this string should not be found in the line.

   sepn The string delimiter character(s). When the LOCATE subcommand is entered implicity, the starting string delimiter character (*sep1*) must be a slash (/).

   stringn The string to be located. When multiple strings are specified, each record is searched for all strings.

   & Denotes that both strings must appear in the record for the search to be successful.

   | Denotes that the appearance of either string in a record is sufficient for the match to succeed.

Usage notes:

1. LOCATE normally searches toward the end of the browsed data unless a leading - is used, in which case the search proceeds toward the start of the browsed data.
2. If SET VARblank is on, a blank in the pattern will match one or more blanks in the searched data. This is useful for finding occurrences of data that can be separated by a variable number of blanks.
3. If the search criteria are not matched when the last line of data is reached, and SET WRAP is on, the search wraps, a message is displayed, and the search continues until the operand is found or until the line on which the search began is reached. If SET WRAP is off, the search ends at the top or bottom of the file, depending on the direction of the search.
4. If SET STAY is on and if the search criteria are not matched, the current line pointer remains where it was when the LOCATE subcommand was issued. If SET STAY is off, the last line searched becomes the current line. If both SET STAY and SET WRAP are on, the current line remains the same.
5. Normally, the search is made one line at a time. If SET SPAN is on, data can be matched even if the data spans two or more lines. If SET SPAN ON Blank is specified, a blank is inserted between the lines to be searched. This can be useful when locating words or sentences in a document.
6. If SET ARBCHAR is on, a "wild card" character can be specified in LOCATE subcommands. This character represents zero or more characters in the data to be matched. For example, if $ is the arbitrary character, then the pattern "A$B" would match "AB" or "AnotherstuffB". If SET ARBCHAR OFF is specified, the arbitrary character is ignored. The default settings are OFF $, meaning that a dollar sign ($) is used as the arbitrary character, but is not enabled.
7. If SET HEX is on, hexadecimal data can be specified in the form X'*dd...*', where *dd...* represents one or more pairs of hexadecimal digits (0–9, a–f, A–F). Each pair corresponds to a single byte of data to be matched.
8. If the operand is not found, an error message is displayed.

LOFFSET subcommand

LOFFSET looks for an address in an assembler listing that is closest to—equal to or less than—the specified hexadecimal address, starting at the current line.

The format of the LOFFSET subcommand is shown below:

LOFFSET	*hex-value*

Where:

hex-value A hexadecimal value. LOFFSET 440, for example would locate a hexadecimal value equal to less than 000440.

Usage notes:

1. If no address exists that is equal to the one entered, the next lower address will be found.
2. LOFFSET starts its search at the current line. Use OFFSET to begin the search at the top of the listing.

MEMBER subcommand

The MEMBER subcommand will browse a member of a MACLIB, TXTLIB, LOADLIB, or XMENULIB.

The format of the MEMBER subcommand is shown below:

MEMBER	*name*

Where:

name The name of the member to be browsed.

Usage notes:

1. When you browse a member, you are placed into a new level of KBROWSE. When you enter QUIT, you return to the previous level.
2. You may enter MEMBER while displaying a member to view a different file member; you don't have to return to the highest level of KBROWSE to do this.
3. If you attempt to use MEMBER on a file that isn't a library, results are unpredictable (usually an "I/O error reading file" message).

NEXT subcommand

The NEXT subcommand scrolls one or more lines toward the bottom of the browsed data.

The format of the NEXT subcommand is shown below:

Next	[*n* | *]

Where:

n | * — The optional number of lines to scroll. If not specified, *n* defaults to 1 (one). If * is specified, the bottom of the data is displayed.

Usage notes:

1. This subcommand is used to move forward in (toward the bottom of) the displayed data one or more lines (rows, or records) at a time. It is equivalent to the *+nn* function of the LOCATE subcommand.
2. The NEXT subcommand is synonymous with DOWN.

NFIND subcommand

The NFIND subcommand is used to find the next line in the file that does not start with the specified data.

The format of the NFIND subcommand is shown below:

NFind	[*line*]

Where:

line The data to be "not found" starting in column one of the next data line. *line* must be separated from the NFIND subcommand by a single blank.

Usage notes:

1. The string to be searched for (*line*) begins following (but not including) the single blank separating it from the NFIND subcommand. Any blank character in the string will match any character in the corresponding column position. An underscore (_) character in the string will match a blank in the corresponding column position.
2. NFIND proceeds toward the end of the browsed data.
3. If SET WRAP ON is in effect and the string is matched when the last line of data is reached, the search wraps, a message is displayed, and the search continues until the string is not found or until the line on which the search began is reached. If SET WRAP OFF is in effect, the search terminates at the end of the file.
4. If the string is always found, an error message is displayed.
5. NFIND is not affected by SET ZONE.
6. NFINDUP is used to search the data in a backward direction.

NFINDUP subcommand

The NFINDUP subcommand is used to find the next previous line in the file that does not start with the specified data.

The format of the NFINDUP subcommand is shown below:

NFINDUp	[*line*]

Where:

line The data to be "not found" starting in column one of the previous data line. *line* must be separated from the NFINDUP subcommand by a single blank.

Usage notes:

1. The string to be searched for (*line*) begins following (but not including) the single blank separating it from the FIND subcommand. Any blank character in the string will match any character in the corresponding column position. An underscore (_) character in the string will match a blank in the corresponding column position.
2. NFINDUP proceeds toward the beginning of the browsed data.
3. If SET WRAP ON is in effect and the string is matched when the first line of data is reached, the search wraps, a message is displayed, and the search continues until the string is not found or until the line on which the search began is reached. If SET WRAP OFF is in effect, the search terminates at the top of the file.
4. If the string *line* is always found, an error message is displayed.
5. NFINDUP is not affected by SET ZONE.
6. NFIND is used to search the data in a forward direction.

OFFSET subcommand

OFFSET looks for an address in an assembler listing that is closest to—equal to or less than—the specified hexadecimal address, starting at the first line of the file.

The format of the OFFSET subcommand is shown below:

OFFSET	*hex-value*

Where:

hex-value A hexadecimal value. OFFSET 440, for example would locate a hexadecimal value equal to less than 000440.

Usage notes:

1. If no address exists that is equal to the one entered, the next lower address will be found.
2. OFFSET starts its search at the first line in the file. Use LOFFSET to begin the search at the current line.

PEEL subcommand

The PEEL subcommand will terminate all contiguous nested levels of KBROWSE.

The format of the PEEL subcommand is shown below:

PEEL	

Usage notes:

1. If this is the first level of KBROWSE, the entire application is ended. If this is a nested level of KBROWSE, the browsed data is left in the "ring" of browsed data and the previous level of the application is reached. If this level is also browsed data, the process repeats until a "non-browse" level is reached.
2. PEEL is used to quickly exit from many nested KBROWSE levels.
3. Use QQUIT to exit noncontiguous levels of KBROWSE.

KBROWSE PF key definitions

These PF key descriptions (and command line equivalents) are the default values for PF keys in KBROWSE.

PFK	Command	Description
PF01	**HELP**	Provides detailed explanations for KBROWSE functions.
PF03	**QUIT**	Exits this level of the application.
PF04	**PSCREEN**	Prints the contents of the current menu to the virtual printer or to a CMS file.
PF05	**=**	Repeats the last command.
PF06	**CURSLINE**	Makes the line containing the cursor the current line.
PF07	**BACKWARD**	Scrolls the display back one screen.
PF08	**FORWARD**	Scrolls the display forward one screen.
PF10	**LEFT 20**	Scrolls the display left 20 columns.
PF11	**RIGHT 20**	Scrolls the display right 20 columns.
PF12	**CANCEL**	Exits all levels of the application.

Usage notes:

1. The KBROWSE PF keys can be tailored by editing the PROFILE KBROWSE file.
2. PF keys can be changed dynamically by the KBROWSE QUERY PF subcommand, or set directly by the SET PF subcommand.

PRINTSCR subcommand

The PRINTSCR subcommand is used to print the contents of the current menu to the virtual printer.

The format of the PRINTSCR subcommand is shown below:

PRintscr	

Usage notes:

1. This subcommand prints the current menu image to the virtual printer. The menu, including any current user input, is printed within a numbered box. A header line displays the menu name, the date, time, and the number of lines on the menu.
2. If the MENUEXEC PRTNOH subcommand is set, the menu is printed without the header line or numbered box. (See the *XMENU/REXX Interface User's Guide and Reference* for more information on the PRTNOH subcommand.)
3. If the MENUEXEC SCRIPT option is set, the menu is printed in SCRIPT/VS format suitable for inclusion in application description documentation. (See the *XMENU/REXX Interface User's Guide and Reference* for more information on the SCRIPT option.)
4. If the menu is wider than 132 characters, it should be sent to a CMS file or a virtual 3211 printer. If sent to a virtual 1403 printer, characters exceeding the 132-character line length will not be printed.
5. The PRINTSCR subcommand is synonymous with PSCREEN.

PRTFILE subcommand

The PRTFILE subcommand writes browsed data to the virtual printer.

The format of the PRTFILE subcommand is shown below:

PRTfile	[*number* | *]

Where:

number | * Specifies the number of lines to be printed. If * is specified, all remaining lines are printed. If neither *number* or * is specified, only the current line is printed.

Usage notes:

1. When the PRTFILE subcommand completes, the current line of the browsed data will be the last printed line.

PSCREEN subcommand

The PSCREEN subcommand is used to print the contents of the current menu to the virtual printer.

The format of the PSCREEN subcommand is shown below:

PSCreen	

Usage notes:

1. This subcommand prints the current menu image to the virtual printer. The menu, including any current user input, is printed within a numbered box. A header line displays the menu name, the date, time, and the number of lines on the menu.
2. If the MENUEXEC PRTNOH subcommand is set, the menu is printed without the header line or numbered box. (See the *XMENU/REXX Interface User's Guide and Reference* for more information on the PRTNOH subcommand.)
3. If the MENUEXEC SCRIPT option is set, the menu is printed in SCRIPT/VS format suitable for inclusion in application description documentation. (See the *XMENU/REXX Interface User's Guide and Reference* for more information on the SCRIPT option.)
4. If the menu is wider than 132 characters, it should be sent to a CMS file or a virtual 3211 printer. If sent to a virtual 1403 printer, characters exceeding the 132-character line length will not be printed.
5. The PSCREEN subcommand is synonymous with PRINTSCR.

PURGE subcommand

The PURGE subcommand will terminate this level of KBROWSE and remove this file from the "ring" of browsed data.

The format of the PURGE subcommand is shown below:

PURGE	

Usage notes:

1. If this is the first level of KBROWSE, the entire application is exited. If this is a nested level of KBROWSE, the browsed data is removed from the "ring" of browsed files.
2. Use the QUIT subcommand if you want to leave this data in the browsed data ring.

PUT subcommand

The PUT subcommand appends browsed data to a CMS file.

The format of the PUT subcommand is shown below:

PUT	[*number* | *] [*filename* | =] [*filetype* | =] [*filemode* | =]

Where:

number | * Specifies the number of lines to be written. If * is specified, all remaining lines are written. If neither *number* nor * is specified, only the current line is written.

filename | = Specifies the filename of the CMS file. If not specified, the filename, filetype and filemode default to those shown in the upper-left corner of the menu. If = is specified, this filename is used.

filetype | = Specifies the filetype of the CMS file. If not specified, the filetype and filemode default to those shown in the upper-left corner of the menu. If = is specified, this filetype is used.

filemode | = Specifies the filemode of the CMS file. If not specified, the filemode defaults to that shown in the upper-left corner of the menu. If = is specified, this filemode is used.

Usage notes:

1. When the PUT subcommand completes, the current line of the browsed data is positioned at the last line written to the CMS file.
2. If the CMS file exists, the data is appended. If you are browsing a CMS file and use the PUT subcommand without using any filenaming operands, the data will be appended to the file being browsed.
3. If you attempt to PUT records into an existing fixed length file (RECFM F) with a different record length, an error will be reported.

QQUIT subcommand

The QQUIT subcommand will terminate all levels of KBROWSE.

The format of the QQUIT subcommand is shown below:

QQuit	

Usage notes:

1. If any program was called between levels of KBROWSE, the QQUIT process will stop at that level. When this program ends, the QQUIT process will continue. This will continue until KBROWSE is completely ended.
2. The QQUIT subcommand is symonymous with CANCEL.

QUERY subcommand

The QUERY subcommand displays KBROWSE environmental settings.

The format of the QUERY subcommand is shown below:

QUERY	*function*

Where:

function	**Provides this information:**
ARBchar	The setting of the LOCATE wild card character.
CASE	Whether or not data is displayed and searched for in mixed or uppercase.
CC	Whether or not carriage control is displayed, and if so, whether in machine or ASA format.
CLASS	The class of a spool file.
COPY	The copy count of a spool file.
CURLine	The current line's record number on the menu. The top title line of the menu is line 1 (one).
DIST	The distribution data for a spool file.
FLASH	The flash setting of a spool file.
HEX	Whether or not the LOCATE subcommand will accept hexadecimal data.
HEXDisp	Whether or not data is displayed in character, hexadecimal, or both formats.
HOLD	The hold status of a spool file.
IMPcmscp	Whether or not unknown commands are passed to CMS and CP for execution.
LENgth	Whether or not line lengths are displayed to the left of each line.
LINEnd	The character used to separate multiple subcommands entered on the command line.
NAME	The name information of a spool file.
NONDisp	The character displayed when non-displayable characters are found in the browsed data.
NUMber	Whether or not line numbers are displayed to the left of each line.
OFORM	The operator form of a spool file.
PF	Displays a menu allowing you to change your PF keys.
PF*nn*	The setting of PF key *nn*.
SPAN	Whether or not LOCATE will search for character strings that span data lines.

STAY	Whether or not the current line pointer moves if the subject of a LOCATE, FIND, FINDUP, NFIND, or NFINDUP is not found.
TAG	The tag information of a spool file.
TYPE	The type information of a spool file.
UFORM	The user form of a spool file.
VARblank	Whether or not multiple blanks are treated as one during LOCATE.
VERSION	The version of the KBROWSE utility.
XMENU	The storage addresses of XMENU control blocks (for debugging).
WRap	Whether or not the search wraps the file during LOCATE, FIND, FINDUP, NFIND, or NFINDUP subcommands.
Zone	The left and right character columns within which LOCATE searches take place.

Usage notes:

1. The QUERY subcommand shows you the settings of certain environmental settings of KBROWSE. The response is placed on the command line in a form that can be easily modified and entered as a SET subcommand, to change the KBROWSE settings.

QUIT subcommand

The QUIT subcommand will terminate this level of KBROWSE.

The format of the QUIT subcommand is shown below:

QUIT	

Usage notes:

1. If this is the first level of KBROWSE, the entire application is exited. If this is a nested level of KBROWSE, the browsed data is left in the "ring" of browsed files.
2. Use the PURGE subcommand if you want this data removed from the ring of browsed data.
3. Use the DISCARD subcommand if you want this data erased. DISCARD only erases CMS files and spool files.

REW subcommand

The REW subcommand will rewind a tape, positioning it at the first physical file on the tape.

The format of the REW subcommand is shown below:

REW	

Usage notes:

1. A tape can consist of one or more physical files, separated by tape marks. This command allows you to go to the very first file.
2. Browsed tapes may be positioned using BSF, FSF, FST, and REW.

RIGHT subcommand

The RIGHT subcommand shifts the menu one or more characters toward the last, or rightmost, column of the browsed data.

The format of the RIGHT subcommand is shown below:

RIght	[*n* | *]

Where:

n | *
: The optional number of columns to shift. If not specified, *n* defaults to 1 (one). If * is specified, the rightmost characters of the data are displayed. RIGHT 0 performs no movement.

Usage notes:

1. This subcommand is used to shift the menu to see data on lines that are too long to fit on the menu. The RIGHT subcommand allows you to see data that is to the right of the last column currently displayed on the menu. Data moves to the left, exposing the specified number of columns to the right of the current last column.
2. The menu cannot be scrolled past the last column, that is, past the rightmost data.
3. Use the VERIFY and COLUMN subcommands to move to a specific data column.
4. PF11/PF23 is normally assigned to RIGHT 20.
5. The LEFT subcommand is used to move in the opposite direction.

SAVE subcommand

The SAVE subcommand writes all browsed data to a CMS file.

The format of the SAVE subcommand is shown below:

SAVE	[*filename* | =] [*filetype* | =] [*filemode* | =]

Where:

filename \| = Specifies the filename of the CMS file. If not specified, the filename, filetype and filemode default to those shown in the upper-left corner of the menu. If = is specified, this filename is used.

filetype \| = Specifies the filetype of the CMS file. If not specified, the filetype and filemode default to those shown in the upper-left corner of the menu. If = is specified, this filetype is used.

filemode \| = Specifies the filemode of the CMS file. If not specified, the filemode defaults to that shown in the upper-left corner of the menu. If = is specified, this filemode is used.

Usage notes:

1. All lines in the browsed data are written. The current line pointer is not affected.
2. If the CMS file exists, an error condition is reported.
3. If SAVE is used after the XLATE subcommand, the translated file is saved.

SET subcommand

The SET subcommand sets KBROWSE environmental settings.

The format of the SET subcommand is shown below:

SET	ARBchar OFF | ON [*char*] CASE Upper | Mixed [Respect | Ignore] CC OFF | ON | ASA | MACH CURLine ON TOP | MIDDLE | *n* HEX OFF | ON HEXDisp OFF | ON | CHAR IMPcmscp OFF | ON LENgth OFF | ON LINEnd OFF | ON [*char*] NONDisp [*char*] NUMber OFF | ON PF*nn* [*/comment/*] *command* SPAN OFF | ON [[Blank | Noblank] *nn*] STAY OFF | ON VARblank OFF | ON WRap OFF | ON Zone *zone1 zone2*

Where:

ARBchar Specifies whether a "wild card" character can be specified in the LOCATE subcommand. This character represents zero or more characters in the data to be matched. For example, if $ is the arbitrary character, then the pattern "A$B" would match "AB" or "AnotherstuffB". If OFF is specified, the arbitrary character is ignored. If ON is specified, the arbitrary character is respected. The default settings are OFF $, meaning that a dollar sign ($) is used as the arbitrary character, but is not enabled.

CASE Sets whether data is displayed in mixed or uppercase. The second parameter (RespectlIgnore) specifies whether case is respected or ignored when performing searches. The default setting is MIXED IGNORE.

CC Specifies whether carriage control is displayed, and if so, whether in machine or ASA format. If ON is specified, ASA format is assumed. The default setting is OFF.

CURLine Sets the screen position of the current line on the menu. The top title line of the menu is line 1 (one). TOP specifies the line immediately after the title, command, and scale lines. MIDDLE specifies the middle line on the screen. The default setting is MIDDLE.

HEX Sets whether hexadecimal strings can be entered as parts of LOCATE strings. If ON is specified, hexadecimal data can be entered in the form X'*dd*....' where *dd* is a pair of hexadecimal characters (0-9, a-f, A-F). Each pair represents a byte to be matched. If OFF is specified, this format cannot be used. The default setting is OFF.

HEXDisp Sets whether data is displayed in character, hexadecimal, or both formats. ON specifies that the data is displayed in hexadecimal notation. OFF specifies that the data is displayed as characters. CHAR specifies both formats are to be displayed. The default setting is OFF.

IMPcmscp Sets whether or not unknown commands are passed to CMS and CP for execution. ON specifies that commands are sent to CMS and CP. The default setting is ON.

LENgth Sets whether or not line lengths are displayed to the left of each line. The default setting is OFF.

LINEnd Sets whether or not multiple subcommands can be entered in the command line, and if so, the character used to separate them. If ON is specified, multiple subcommands can be entered. If OFF is specified, multiple subcommands cannot be entered. *char* is the character used to separate the subcommands. The default setting is ON #, meaning that the pound sign (#) character is used to separate subcommands.

NONDisp Specifies the character used to represent non-displayable characters. The default setting is a double quote (").

NUMber Sets whether or not line numbers are displayed to the left of each line. The default setting is OFF.

PF*nn* Sets a PF key to a specific function. If you also specify */comment/* the string *comment* is displayed on the PF key line at the bottom of the screen. Default PF key definitions are listed in this reference section under the heading "KBROWSE PF key definitions".

SPAN Sets whether or not data searched for in LOCATE, FIND, FINDUP, NFIND, or NFINDUP subcommands can span one or more browsed data lines. If ON is specified, and the number of lines is greater than one, searched data can span lines. If Blank is specified, a blank is placed between each line before the search begins. This is useful for finding strings of words in a document. If Noblank is specified, any trailing blanks are removed before lines are concatenated. The concatenated data is subject to the SET ZONE settings. In general, searches are faster if SPAN is set OFF. The default setting is OFF Blank 2.

STAY Sets whether or not the current line pointer moves if the object of a LOCATE, FIND, FINDUP, NFIND, or NFINDUP is not found. If ON is specified, the current line pointer does not move. If OFF is specified the current line pointer moves to the last line searched. The default setting is OFF.

VARblank Specifies whether or not a blank in a LOCATE subcommand search string will match one or multiple blanks in the searched data. If OFF is specified, a blank in the pattern matches one and only one blank in the searched data. If ON is specified, a blank in the pattern matches one or more blanks in the searched data. The default setting is OFF.

WRap Sets whether or not LOCATE, FIND, FINDUP, NFIND, or NFINDUP subcommands wrap the browsed data when looking for a match. If ON is specified, the browsed data wraps, and a

informational message is displayed. If OFF is specified, the search stops at the top or bottom of the file. The default setting is OFF.

Zone Sets the left and right character columns within which LOCATE searches take place. The default setting is the leftmost and rightmost characters in each data line.

Usage notes:

The SET subcommand allows you to specify certain environmental settings for KBROWSE. The SET subcommand can be entered from the command line or from within a PROFILE KBROWSE file. To change default settings and ensure they will be in effect for every invocation of KBROWSE, create or edit PROFILE KBROWSE and add or change subcommand settings. For example, to change the CURLINE, NUMBER, and STAY default settings, you could add these records to PROFILE KBROWSE:

Records added to PROFILE KBROWSE

```
SET CURLINE ON 2
SET NUM ON
SET STAY ON
```

STACK subcommand

The STACK subcommand places portions of one or more lines into the CMS program stack.

The format of the STACK subcommand is shown below:

STAck	[[[[LIFO | FIFO] *number* | *] *startcol*] *length*]

Where:

LIFO|FIFO Specifies whether lines are stacked last in, first out or first in, first out. If not specified, FIFO is the default.

number | * Specifies the number of lines to be stacked. If * is specified, then all remaining lines are stacked. If not specified, only the current line is stacked.

startcol Specifies the starting character column to be stacked. If not specified, the line is stacked starting with column one.

length Specifies the number of characters in each line to be stacked. If not specified, the line is stacked from *startcol* to the ending column.

Usage notes:

1. When the STACK subcommand completes, the browsed data is positioned at the last line stacked.
2. LIFO should be used if you want to stack lines in reverse order.
3. The CMS program stack is limited to lines up to 255 characters long. Excess data is truncated.

TOP subcommand

The TOP subcommand makes the first line of data the current line.

The format of the TOP subcommand is shown below:

TOP	

Usage notes:

This subcommand is used to move quickly to the beginning of the data.

UP subcommand

The UP subcommand scrolls one or more lines toward the top of the browsed data.

The format of the UP subcommand is shown below:

Up	[*n* | *]

Where:

n | * — The optional number lines of data to scroll. If not specified, *n* defaults to 1 (one). If * is specified, the top of the data is displayed.

Usage notes:

1. This subcommand is used to move backward in (toward the top of) the displayed data, one or more lines (records, or rows) at a time. It is equivalent to the *-nn* function in LOCATE.

VERIFY subcommand

The VERIFY subcommand scrolls the data so that the specified character column is displayed at the leftmost position of the menu.

The format of the VERIFY subcommand is shown below:

Verify	[*n*]

Where:

n The optional character column number to display. If not specified, it defaults to 1 (one).

Usage notes:

1. This subcommand is used to quickly position data to a specific character column.
2. Use LEFT and RIGHT to move the data a number of columns relative to the current position.
3. The VERIFY subcommand is synonymous with COLUMN.

XLATE subcommand

The XLATE subcommand specifies that the browsed data is to be translated before being displayed.

The format of the XLATE subcommand is shown below:

XLATE	OFF | *table*

Where:

OFF \| *table* — The translate table file to be used to translate the data. *table* is the filename of the translate table file whose filetype must be XLATE. If OFF is specified, no translation is performed.

Usage notes:

1. XLATE is used to browse data that is in a format other than EBCDIC. Translation takes place when the file is displayed, before searches (LOCATE, FIND, FINDUP, NFIND, NFINDUP), and before lines are stacked, printed, or saved to CMS files. By using XLATE and SAVE, KBROWSE can be used to translate files from one character representation to another.
2. Several standard translate files are supplied:

AS2EBC	Performs ASCII to EBCDIC translation
AS2EBCR	Translates bit reversed ASCII to EBCDIC
EBCDIC	Translates files into basic EBCDIC, that is, non-printable characters are changed to periods (.).
EBC2AS	Performs EBCDIC to ASCII translations
EBC2ASR	Performs EBCDIC to ASCII bit reversed translations
LOW2UP	Changes lowercase EBCDIC to uppercase EBCDIC
MIRROR	Performs binary bit reversals in any format file

Chapter 8. KFLIST, KRDRLIST, KPRTLIST, and KPUNLIST

This chapter describes KFLIST and its related utilities: KRDRLIST, KPRTLIST, and KPUNLIST. These utilities allow you to list and browse CMS minidisk files, spooled reader, printer, and punch files, respectively.

The CMS command syntax for each utility is given first. This is followed by an alphabetical listing of the subcommands these utilities share—subcommands that control both the listing and browsing environments, and the files being listed and browsed.

KFLIST command

The KFLIST command is used to list the contents of CMS minidisks. Using wild card characters, one or many accessed minidisks can be searched.

The format of the KFLIST command is shown below:

KFLIST	[*fn* [*ft* [*fm*]]] [(Options] **Options:** NOPROFIL PROFILE *filename* SUBCMD *first-KFLIST-subcommand*

Where:

fn ft The optional filename(s) and filetype(s) of the file(s) to be listed. If neither is specified, all files on minidisk *fm* are listed. If nothing is specified, all files on your A disk are listed. If *ft* isn't specified, all filetypes that match *fn* are listed.

Two types of wild cards are allowed. An asterisk matches zero, one, or more characters in that position. For example, * matches all files, A* matches all files starting with "A", *E matches all files ending with "E", and so on.

A question mark matches any single replacement character. Thus, FILE? would match "FILE1", "FILE2", and "FILEX", but neither "FILEXX" nor "FILE".

fm The optional filemode of the minidisks to be listed. If omitted or specified as an asterisk (*) all accessed disks are searched.

Options:

NOPROFIL Specifies that no KFLIST profile should be executed.

PROFILE *filename*
Specifies the KFLIST profile (PROFILE KFLIST) is to be executed before this level of KFLIST is entered. If neither this option nor NOPROFIL are specified, the commands in file PROFILE KFLIST are executed.

SUBCMD *first-KFLIST-subcommand*
Passes the first KFLIST subcommand at KFLIST invocation. The remainder of the command line is used as the subcommand; therefore if SUBCMD is used with other options, it should be the last option specified.

Usage notes:

1. KFLIST has many things in common with the FLIST and FILELIST commands, and some differences, including the ones below:
   - In general, KFLIST looks and acts more like FLIST than FILELIST.
   - The KFLIST profile has a different format than the FLIST $PROFILE file. The KFLIST profile is an EXEC 2 or REXX EXEC containing KFLIST subcommands. You can, for example, specify that files come up sorted by date, by placing the subcommand SORT DATE in your PROFILE KFLIST file.
   - KFLIST has a top command line. This can be used to issue KFLIST global subcommands, CMS, and CP commands.

KFLIST command line subcommands

These commands can be specified on the KFLIST top command line, in the KFLIST profile, and/or in KFLIST macros. These commands are described in alphabetical order in the reference section that follows. They can also be found in CMS HELP files by entering HELP KFLIST MENU.

KFLIST file line subcommands

These commands are used to issue a CP, CMS, or EXEC command relating to a specific file. These commands are described under the heading "LINECMDS" in the reference section that follows.

Return codes:

200 Insufficient storage to allocate KFLIST blocks.
201 Invalid option specified.
202 Duplicate option found.
203 Profile name > eight characters.

Return codes and messages:

28 No files match the command line criteria.

KRDRLIST command

The KRDRLIST command is used to list the files in your spooled reader queue.

The format of the KRDRLIST command is shown below:

KRDRLIST	[(Options] **Options:** NOPROFIL PROFILE *filename* SUBCMD *first-KRDRLIST-subcommand*

Options:

NOPROFIL Specifies that no KRDRLIST profile should be executed.

PROFILE *filename*
Specifies the KRDRLIST profile (PROFILE KRDRLIST) is to be executed before this level of KRDRLIST is entered. If neither this nor NOPROFIL are specified, the commands in file PROFILE KRDRLIST are executed.

SUBCMD *first-KRDRLIST-subcommand*
Passes the first KRDRLIST subcommand at KRDRLIST invocation. The remainder of the command line is used as the subcommand; therefore if SUBCMD is used with other options, it should be the last option specified.

Usage notes:

KRDRLIST has many things in common with the RDRLIST command, and some differences, including the ones below:

1. In general, KRDRLIST looks and acts more like FLIST.
2. The KRDRLIST profile has a different format than the FLIST $PROFILE file. The KRDRLIST profile is an EXEC 2 or REXX EXEC containing KRDRLIST subcommands. You can, for example, specify that files come up sorted by date, by placing the subcommand SORT DATE in your PROFILE KRDRLIST file.
3. KRDRLIST has a top command line. This can be used to issue KRDRLIST global subcommands, CMS, and CP commands.

KRDRLIST command line subcommands

These commands can be specified on the KRDRLIST top command line, in the KRDRLIST profile, and/or in KRDRLIST macros. These commands are described in alphabetical order in the reference section that follows. They can also be found in CMS HELP files by entering HELP KRDRLIST MENU.

KRDRLIST file line subcommands

These commands are used to issue a CP, CMS, or EXEC command relating to a specific file. These commands are described under the heading "LINECMDS" in the reference section that follows.

Return codes:

200 Insufficient storage to allocate KRDRLIST blocks.
201 Invalid option specified.
202 Duplicate option found.
203 Profile name > eight characters.

Return codes and messages:

28 No files match the command line criteria.

KPRTLIST command

The KPRTLIST command is used to list the files in your spooled printer queue.

The format of the KPRTLIST command is shown below:

KPRTLIST	[(Options] **Options:** NOPROFIL PROFILE *filename* SUBCMD *first-KPRTLIST-subcommand*

Options:

NOPROFIL Specifies that no KPRTLIST profile should be executed.

PROFILE *filename*
Specifies the KPRTLIST profile (PROFILE KPRTLIST) is to be executed before this level of KPRTLIST is entered. If neither this nor NOPROFIL are specified, the commands in file PROFILE KPRTLIST are executed.

SUBCMD *first-KRRTLIST-subcommand*
Passed the first KPRTLIST subcommand at KPRTLIST invocation. The remainder of the command line is used as the subcommand; therefore if SUBCMD is used with other options, it should be the last option specified.

Usage notes:

KPRTLIST is similar to a print queue equivalent of the RDRLIST command. It has has many things in common with this command, and some differences, including the ones below:

1. In general, KPRTLIST looks and acts more like FLIST.
2. The KPRTLIST profile has a different format than the FLIST $PROFILE file. The KPRTLIST profile is an EXEC 2 or REXX EXEC containing KPRTLIST subcommands. You can, for example, specify that files come up sorted by date, by placing the subcommand SORT DATE in your PROFILE KPRTLIST file.
3. KPRTLIST has a top command line. This can be used to issue KPRTLIST global subcommands, CMS, and CP commands.

KPRTLIST command line subcommands

These commands can be specified on the KPRTLIST top command line, in the KPRTLIST profile, and/or in KPRTLIST macros. These commands are described in alphabetical order in the reference section that follows. They can also be found in CMS HELP files by entering HELP KPRTLIST MENU.

KPRTLIST file line subcommands

These commands are used to issue a CP, CMS, or EXEC command relating to a specific file. These commands are described under the heading "LINECMDS" in the reference section that follows.

Return codes:

200 Insufficient storage to allocate KPRTLIST blocks.
201 Invalid option specified.
202 Duplicate option found.
203 Profile name > eight characters.

Return codes and messages:

28 No files match the command line criteria.

KPUNLIST command

The KPUNLIST command is used to list the files in your spooled punch queue.

The format of the KPUNLIST command is shown below:

KPUNLIST	[(Options] **Options:** NOPROFIL PROFILE *filename* SUBCMD *first-KPUNLIST-subcommand*

Options:

NOPROFIL Specifies that no KPUNLIST profile should be executed.

PROFILE *filename*
Specifies the KPUNLIST profile (PROFILE KPUNLIST) is to be executed before this level of KPUNLIST is entered. If neither this nor NOPROFIL are specified, the commands in file PROFILE KPUNLIST are executed.

SUBCMD *first-KPUNLIST-subcommand*
Passes the first KPUNLIST subcommand at KPUNLIST invocation. The remainder of the command line is used as the subcommand; therefore if SUBCMD is used with other options, it should be the last option specified.

Usage notes:

KPUNLIST is similar to a punch queue equivalent of the RDRLIST command. It has many things in common with this command, and some differences, including the ones below:

1. In general, KPUNLIST looks and acts more like FLIST.
2. The KPUNLIST profile has a different format than the FLIST $PROFILE file. The KPUNLIST profile is an EXEC 2 or REXX EXEC containing KPUNLIST subcommands. You can, for example, specify that files come up sorted by date, by placing the subcommand SORT DATE in your PROFILE KPUNLIST file.
3. KPUNLIST has a top command line. This can be used to issue KPUNLIST global subcommands, CMS, and CP commands.

KPUNLIST command line subcommands

These commands can be specified on the KPUNLIST top command line, in the KPUNLIST profile, and/or in KPUNLIST macros. These commands are described in alphabetical order in the reference section that follows. They can also be found in CMS HELP files by entering HELP KPUNLIST MENU.

KPUNLIST file line subcommands

These commands are used to issue a CP, CMS, or EXEC command relating to a specific file. These commands are described under the heading "LINECMDS" in the reference section that follows.

Return codes:

200 Insufficient storage to allocate KPUNLIST blocks.
201 Invalid option specified.
202 Duplicate option found.
203 Profile name > eight characters.

Return codes and messages:

28 No files match the command line criteria.

& (Ampersand) subcommand

The & (ampersand) subcommand will leave itself and the following text in the command input area of a menu.

The format of the & subcommand is shown below:

&	*text*

Where:

text The character string to be left in the command input area.

Usage notes:

1. Because the ampersand and accompanying text are left in the command input area, the & subcommand allows easy repetitive entering of a command (such as LOCATE) by simply pressing the ENTER key again.
2. *text* is executed as a command. If *text* is not a KFLIST QUERY subcommand, the text and ampersand will remain in the command input area when the subcommand completes execution. The response of KFLIST QUERY subcommands are placed in the command input area of the menu, thus overlaying the command.
3. The & subcommand can be removed by placing the cursor at the beginning of the command line and pressing the ERASE EOF key.

? (Question Mark) subcommand

The ? (retrieve) subcommand will retrieve the previous entry in the command stack retrieve buffer and return it to the command line of a menu.

The format of the ? subcommand is shown below:

?	

Usage notes:

1. A retrieve stack is kept internally in KFLIST. Previous entries can be retrieved via the ? or "retrieve last subcommand" subcommand, and be returned to, or re-displayed on, the command line of the current menu. This function is equivalent to the CP RETRIEVE function that is available in VM/SP Release 2 and subsequent releases. The retrieve stack is 256 bytes long. The number of entries present depends on the length of the strings entered.
2. Retrieving commands with the ? subcommand returns them to the command line, or re-displays them, and allows them to be corrected and re-entered.
3. Pressing the CLEAR key resets the RETRIEVE stack to only its most recent entry.
4. For locally attached, non-SNA, non-bisync terminals that have a TEST REQ or SYS REQ key, pressing TEST REQ or SYS REQ will also retrieve the last subcommand.
5. Any text following the question mark is ignored.
6. If more than one question mark appears on the line, line *n* of the retrieve stack is retrieved, where *n* is the number of question marks present.
7. The Retrieve (?) and Repeat (=) subcommands are not stacked for retrieval.

= (Equal Sign) subcommand

The = (repeat) subcommand will retrieve the last entry in the command stack retrieve buffer and submit it to the command processor.

The format of the = subcommand is shown below:

=	

Usage notes:

1. A retrieve stack is kept internally in KFLIST. Entries can be re-executed via the = or "repeat last subcommand" subcommand, or returned to the command line without execution via the ? or "retrieve" subcommand. The RETRIEVE stack is 256 bytes long. The number of entries present depends on the length of the strings entered.
2. Repeating invalid commands returns the invalid command to the command line and the terminal alarm, if present, is sounded to alert the user.
3. Any text following the equal sign is ignored.
4. The Retrieve (?) and Repeat (=) subcommands are not stacked for retrieval.

BACKWARD subcommand

The BACKWARD subcommand scrolls one or more screens toward the top of the file list.

The format of the BACKWARD subcommand is shown below:

BAckward	[*n* | *]

Where:

n | *
: The optional number of screens to scroll backward. If not specified, *n* defaults to one. If * is specified, the top of the data file is displayed.

Usage notes:

1. This subcommand is used to move backward (toward the top) of the displayed data one screen at a time. Together, the BACKWARD and FORWARD subcommands are used to quickly scan data one screen at a time.
2. If BACKWARD is entered at the top of data file, no movement occurs.
3. BACKWARD is normally assigned to PF7/PF19.

BOTTOM subcommand

The BOTTOM subcommand makes the last line of the file list the current line.

The format of the BOTTOM subcommand is shown below:

BOttom	

Usage notes:

This subcommand is used to move quickly to the end of the file list.

BROWSE subcommand

The BROWSE subcommand is used to view the contents of CMS files listed by the KFLIST command, or the contents of the spool files listed by KRDRLIST, KPRTLIST, and KPUNLIST. KBROWSE subcommands and CMS and CP commands may be entered while browsing a file. Refer to Chapter 7, "KBROWSE, KOLCMSBR, and KOLCPBR" on page 23 for details on KBROWSE subcommands.

The format of the BROWSE subcommand is shown below:

BRowse	[*fn ft* [*fm*]] [(Options] **Options:** MEMBER *membername* NOPROFIL PROFILE *filename* PRT | PRINTER *file-number* PUN | PUNCH *file-number* RDR | READER [*file-number*] SUBCMD *first-browse-subcommand*

Where:

fn ft The optional filename and filetype of the file to be browsed. If a file with the specified filename and filetype does not exist, an error message is issued. If no filename and filetype are issued, BROWSE enters the last file browsed. If no file has previously been browsed, an error message is issued. BROWSE issued by itself while under BROWSE is used to sequentially display the ring of browsed files.

fm The optional filemode of the file to be browsed. If omitted or specified as an *, all accessed disks are searched. The first occurence is the file to be browsed.

Options:

MEMBER *membername*
Specifies the member of a MACLIB, LOADLIB, TXTLIB, or XMENULIB to be browsed. MEMBER can only be specified with CMS files.

NOPROFIL Specifies that no browse profile should be executed.

PROFILE *filename*
Specifies the browse profile to be executed before this level of browse is entered.

PRT *file-number*
Specifies which of your virtual printer files should be browsed. Only the displayed data is read into storage.

PUN *file-number*
Specifies which of your virtual punch files should be browsed. Only the displayed data is read into storage.

RDR *file-number*

Specifies which of your virtual reader files should be browsed. If *file-number* is not specified, the first file in your reader queue is browsed. Only the displayed data is read into storage.

SUBCMD *first-browse-subcommand*

Passes the first KBROWSE subcommand at BROWSE invocation. The remainder of the command line is treated as the subcommand; therefore if SUBCMD is specified with other options, it should be the last option specified.

Usage notes:

1. The BROWSE subcommand is used to view various types of data files under KBROWSE, KFLIST, KRDRLIST, KPRTLIST, and KPUNLIST.
2. Any KBROWSE subcommand can be issued from BROWSE except those relating to tape browsing when BROWSE is invoked from KFLIST, for example, or those relating to CMS files while browsing from KRDRLIST, and so on. On return, the browsed file is redisplayed at its current position.
3. BROWSE should be used (rather than XEDIT, for example) so that you can easily go between browsing and debugging without losing your position in the browsed file.
4. BROWSE saves and restores the CMS read and write file pointers so that the browsing of a file doesn't affect files in use by the application being debugged. If a file is active and BROWSE is called to display it for the first time, the current line is positioned to the active read pointer location.
5. Many files can be simultaneously browsed. When the BROWSE subcommand is entered by itself, BROWSE moves to the next file in the browsed data ring.

Return codes:

200 Insufficient storage to allocate BROWSE blocks.
201 Invalid option specified.
202 Duplicate option found.
203 Profilename > eight characters.
204 File number missing or invalid for RDR, PRT, PUN.
205 File member name missing or invalid.
206 Cannot specify MEMBER with RDR, PRT, PUN or TAPE.

Return codes and messages:

40 040E File to be browsed was not found.
41 041E File to be browsed has an invalid filename.
42 042E Error xxx browsing file.
43 043E No active browsed files.
44 044E Value contains invalid decimal character(s).
47 047E Data searched for not found.
53 053E Command terminated by user interrupt.

CANCEL subcommand

The CANCEL subcommand will terminate all levels of KFLIST.

The format of the CANCEL subcommand is shown below:

CANCel	

Usage notes:

1. If any program was called between levels of KFLIST, the CANCEL process will stop at that level. When this program ends, the CANCEL process will continue. This will continue until KFLIST is completely ended.

CMDLINE subcommand

The CMDLINE subcommand displays a message on the command line.

The format of the CMDLINE subcommand is shown below:

CMDLINE	*message*

Where:

message A character string to be displayed on the command line.

Usage notes:

1. This subcommand allows you to display a message on the command line, for example, the result of a macro.
2. If you want the command line's Modified Data Tag (MDT) set, for example, to allow the user to reenter the message as a command, use the CMSG subcommand.

CMS subcommand

The CMS subcommand causes CMS to execute the specified command or, if no command is specified, to enter CMS SUBSET.

The format of the CMS subcommand is shown below:

CMS	[*command*]

Where:

command The optional command to be passed to CMS. If CMS is entered without *command*, CMS subset is entered. Use the CMS command RETURN to return to XMENU from CMS subset.

Usage notes:

1. If the program you call abends, you will also abend KFLIST.
2. If the called program displays no output, the last KFLIST screen is displayed. If the called program displayed line-mode output, a MORE... condition will be displayed by the terminal. If this happens, you can press either PA2 or CLEAR, or wait one minute for the KFLIST screen to be restored.
3. If the return code from the called program was non-zero, it will be displayed in menu field CMDPREF (if it exists). The terminal alarm will be sounded and the offending command will be redisplayed in the menu's command field CMDLINE (if it exists) so that it can be corrected. If you do not wish to correct the command, place the cursor at the start of the menu's command field, press ERASE EOF and then ENTER.

Responses:

Refer to the IBM *CMS Command Reference* manual that applies to your release of CMS.

Messages:

Refer to the IBM *System Messages and Codes* manual that applies to your version of VM.

Return codes:

Refer to the IBM *System Messages and Codes* manual that applies to your version of VM.

CMSG subcommand

The CMSG subcommand displays a message on the command line. This subcommand also sets the command line field's Modified Data Tag (MDT).

The format of the CMSG subcommand is shown below:

CMSG	*message*

Where:

message A character string to be displayed on the command line.

Usage notes:

1. This subcommand allows you to display a message on the command line, for example, the result of a macro.
2. This command also sets the command line field's Modified Data Tag (MDT) so that, if the user presses ENTER, *message* will be executed as a command. Use CMDLINE if you want to display a message on the command line without having the MDT set.

CP subcommand

The CP subcommand submits the accompanying command to CP.

The format of the CP subcommand is shown below:

CP	[*command*]

Where:

command The optional command to be passed to CP. If the CP subcommand is entered without *command*, CP is entered and `CP READ` is displayed.

Usage notes:

1. If the CP subcommand is entered without *command*, and `CP READ` is displayed, enter `Begin` to return to KFLIST.
2. If the command displays no output, the last KFLIST menu is displayed. If the command displays line-mode output, a `MORE...` condition will be displayed by the terminal. If this happens, you can press either PA2 or CLEAR, or wait one minute for the KFLIST menu to be restored.
3. If the return code from the command is non-zero, it will be displayed in menu field CMDPREF (if it exists). The terminal alarm will be sounded and the offending command will be redisplayed in the command menu field CMDLINE (if it exists) so that it can be corrected. If you do not wish to correct the command, place the cursor at the start of the command field, press ERASE EOF and then ENTER.

Responses:

Refer to the IBM *System Messages and Codes* manual or the *CP Command Reference* that applies to your version of VM.

Messages:

Refer to the IBM *System Messages and Codes* manual or the *CP Command Reference* that applies to your version of VM.

Return codes:

Refer to the IBM *System Messages and Codes* manual or the *CP Command Reference* that applies to your version of VM.

DOWN subcommand

The DOWN subcommand scrolls one or more lines toward the bottom of the listed files.

The format of the DOWN subcommand is shown below:

DOwn	[*n* | *]

Where:

n | * — The optional number of lines to scroll. If not specified, *n* defaults to one. If * is specified, the bottom of the data is displayed.

Usage notes:

1. This subcommand is used to move forward in (toward the bottom of) the file list one or more lines (or rows, or records) at a time. It is equivalent to the *+nn* function of LOCATE.
2. The DOWN subcommand is synonymous with NEXT.

ERMSG subcommand

The ERMSG subcommand displays a message on the menu title line.

The format of the ERMSG subcommand is shown below:

ERMSG	*message*

Where:

message A character string to be displayed on the menu title line.

Usage notes:

1. This subcommand allows you to display a message on the menu title line, for example, an error message.

FORALL subcommand

The FORALL subcommand executes a pattern string for all files selected by this level of KFLIST.

The format of the FORALL subcommand is shown below:

FORAll	*command-string*

Where:

command-string

The command to be executed, once per line for each file selected by this level of KFLIST. KFLIST line subcommands, such as /t, /ntm, etc., which are described in the "LINECMDS" section of this reference section, can be used in the command string.

Usage notes:

1. This subcommand is used to run a command on a selected list of files (such as assembling a group of source files).
2. You can use the /d line subcommand to drop files from the selected list before running the FORALL subcommand.

FORSCRN subcommand

The FORSCRN subcommand executes a pattern string for all files currently displayed on the terminal.

The format of the FORSCRN subcommand is shown below:

FORScrn	*command-string*

Where:

command-string

The command to be executed, once per line for each file currently displayed on the terminal. KFLIST line subcommands such as /t, /ntm, etc., which are described in the "LINECMDS" section of this reference section, can be used in the command string.

Usage notes:

1. This subcommand is used to run a command on a selected list of files (such as assembling a group of source files).
2. You can use the /d line subcommand to drop files from the selected list before running the FORSCRN subcommand.

FORWARD subcommand

The FORWARD subcommand scrolls one or more screens toward the end of the file list.

The format of the FORWARD subcommand is shown below:

FOrward	[*n* | *]

Where:

n | * — The optional number of screens to scroll. If not specified, *n* defaults to one. If * is specified, the bottom of the data is displayed.

Usage notes:

1. This subcommand is used to move forward in (toward the bottom of) the displayed file list one screen at a time. Together, the FORWARD and BACKWARD subcommands are used to quickly scan the data one screen at a time.
2. If FORWARD is entered at the bottom of the file, no movement occurs.
3. FORWARD is normally assigned to PF8/PF20.

HELP subcommand

The HELP subcommand is used to display information about KFLIST subcommands, keys, messages, and control file subcommands. Under certain applications the HELP PF key (PF01/PF13), together with the cursor position, is used to display more information about a menu field. The appropriate HELP information, provided it is available, is displayed.

The format of the HELP subcommand is shown below:

Help	[MENU | *topic*]

Where:

MENU Displays a menu containing a list of HELP topics. Each topic can be displayed from this menu. This is the default.

topic Specifies the subcommand for which HELP information is to be displayed.

Usage notes:

1. Abbreviations for HELP topics cannot be used. The topic menu can be displayed to determine the full name of a topic.
2. For KFLIST, the topic name for PF key definitions is "PFKEYS", for return codes is "RETCODES", for messages is "MESSAGES", and for program options is "OPTIONS".
3. Normally, HELP is assigned to PF1/PF13.
4. KFLIST uses the CMS HELP facility to display online product information. If HELP is entered from the command line without any operands, the main KFLIST HELP selection menu will be displayed. Place the cursor on a topic name in the main KFLIST HELP selection menu and press ENTER (or PF1/PF13) to receive detailed information about that KFLIST topic. If the optional *topic* is entered, the main selection menu is bypassed and HELP for that topic is displayed directly.
5. KFLIST issues the CMS command HELP. If your installation has modified the IBM HELP command, the KFLIST HELP may not function properly.

Return codes and messages:

See the CMS HELP command description.

LINECMDS subcommand

The KFLIST, KRDRLIST, KPRTLIST, and KPUNLIST line subcommands are used to direct a command for an individually selected file.

Unless specified, the subcommand functions in all four environments.

The following subcommands can be entered on file lines of the full-screen displays:

? Show the last command entered. If reissued, the next to previous command is displayed, and so on.

Unlike the command line, the command is only displayed. It must be typed over and ENTER pressed to invoke it.

= Reissue the last command entered.

/ If specified by itself, make this file the current file.

If specified as part of a command for KFLIST, substitute the file's filename, filetype, and filemode in the command where the / subcommand appears.

If specified as part of a command for KRDRLIST, KPRTLIST, or KPUNLIST, substitute the queue type and the spool number in the command where the / subcommand appears.

/nn Position the list at file line *nn*.

/Bot Position the current line of the list on the last file.

/Cancel Cancel KFLIST.

/D Drop this file from the list of files.

/E [*pattern*] Enter another level of KFLIST based on the specified pattern. /E is synonymous with /L.

/F KRDRLIST, KPRTLIST, KPUNLIST - substitute the file number in the command where the /F subcommand appears.

/L [*pattern*] Enter another level of KFLIST based on the specified pattern. /L is synonymous with /E.

/M Substitute the file's filemode in the command where the /M subcommand appears.

/N Substitute the file's filename in the command where the /N subcommand appears.

/O Issue this command as is—don't perform any substitution or appending of the fileid.

/Quit Quit KFLIST.

/Rl KRDRLIST, KPRTLIST, KPUNLIST - scroll the screen right or left.

/SB Sort files by descending block count.

/SC KRDRLIST, KPRTLIST, KPUNLIST - sort files by spool file class.

/SCB	Scroll the file list backward one screen, toward the top of the list.
/SCF	Scroll the file list forward one screen, toward the bottom of the list.
/SD	Sort files by descending file date, then time.
/SF	KFLIST - sort files by descending record format.
/SF	KRDRLIST, KPRTLIST, KPUNLIST - sort files by spool fileid.
/SH	KRDRLIST, KPRTLIST, KPUNLIST - sort files by spool file hold status.
/SK	KFLIST - sort files by descending record count (/sc was already taken to sort the list by spool file class).
/SK	KRDRLIST, KPRTLIST, KPUNLIST - sort files by spool file class.
/SL	KFLIST - sort files by descending logical record length.
/SL	KRDRLIST, KPRTLIST, KPUNLIST - sort files by spool file load characters.
/SM	KFLIST - sort files by ascending filemode, filename, then filetype.
/SM	KRDRLIST, KPRTLIST, KPUNLIST - sort files by spool file modify.
/SN	KFLIST - sort files by ascending filename, filetype, then filemode.
/SN	KRDRLIST, KPRTLIST, KPUNLIST - sort files by spool file name.
/SO	KRDRLIST, KPRTLIST, KPUNLIST - sort files by spool file origin.
/SP	KRDRLIST, KPRTLIST, KPUNLIST - sort files by spool file copies.
/SQ	KRDRLIST, KPRTLIST, KPUNLIST - sort files by spool queue type.
/SR	KRDRLIST, KPRTLIST, KPUNLIST - sort files by spool file record count.
/ST	KFLIST - sort files by ascending filetype, filename, then filemode.
/ST	KRDRLIST, KPRTLIST, KPUNLIST - sort files by spool file type.
/SU	KRDRLIST, KPRTLIST, KPUNLIST - sort files by spool file user form.
/SV	KRDRLIST, KPRTLIST, KPUNLIST - sort files by spool file operator form.
/SW	KRDRLIST, KPRTLIST, KPUNLIST - sort files by spool file distribution.
/SZ	KRDRLIST, KPRTLIST, KPUNLIST - sort files by spool file flash count.
/T	Substitute the file's filetype in the command where the /T subcommand appears.
/TOp	Position the current line of the list on the first file.

/W	KRDRLIST, KPRTLIST, KPUNLIST - substitute the spool queue type in the command where the /W subcommand appears.

Usage notes:

1. These subcommands are used to invoke CP, CMS, and EXEC commands based on specific file characteristics.
2. When a CMS command completes, the line is changed based on the success or failure of the command. If the command executes successfully, an asterisk is displayed. If a non-zero return code is returned, a not sign is displayed, followed by the return code.
3. The data to the right of the command area can be overtyped. Any data remaining to the right is removed before the command executes.

LOCATE subcommand

The LOCATE subcommand is used to find the preceding or next occurance in the file list of a specified string or other search criteria.

The format of the LOCATE subcommand is shown below:

[Locate]	[*pattern*]

Where:

Locate The subcommand name, but is optional. When the LOCATE subcommand is entered implicity, the starting string delimiter character (*sep1* in the *pattern* description below) must be a slash (/).

pattern The string to be located and/or the search criteria to be used. *pattern* can be entered in several forms:

1. :*nn* locates record number *nn*. If *nn* is *, the last data record is located.
2. +|– *nn* moves the current line pointer down (+) or up (–) *nn* lines. (+|– * goes to the bottom or top of the file, respectively.)
3. String search criteria take on the form:

   [+|–] [¬] [*sep1*] ...
   [*string1*] [*sep1*] [&] [|] [¬] [*sep2*] [*string2*] [*sep2*]

   Where:

   + or – Specifies the direction of the search. If – is entered, the search moves toward the top of the data. If + is specified, or if the leading + or – sign is omitted, the search moves toward the bottom of the data. The first occurrence of + or – in a locate string is always used as a search direction indicator; therefore if + or – is to be used as a string delimiter, the direction character + or – must be specified.

   ¬ Denotes that this string should not be found in the line.

   sepn The string delimiter character(s). If the LOCATE subcommand is implicit, the starting string delimiter character (*sep1*) must be a slash (/).

   stringn The string to be located. When multiple strings are specified, each record is searched for all strings.

   & Denotes that both strings must appear in the record for the search to be successful.

   | Denotes that the appearance of either string in a record is sufficient for the match to succeed.

Usage notes:

1. LOCATE normally searches toward the end of the file list unless a leading – is used, in which case the search proceeds toward the start of the file list.
2. If SET VARblank is on, a blank in the pattern will match one or more blanks in the searched data. This is useful for finding occurrences of data that can be separated by a variable number of blanks.
3. If the search criteria are not matched when the last line of data is reached, and SET WRAP is on, the search wraps, a message is displayed, and the search continues until the operand is found or until the line on which the search began is reached. If SET WRAP is off, the search ends at the top or bottom of the file, depending on the direction of the search.
4. If SET STAY is on and if the search criteria are not matched, the current line pointer remains where it was when the LOCATE subcommand was issued. If SET STAY is off, the last line searched becomes the current line. If both SET STAY and SET WRAP are on, the current line remains the same.
5. Normally, the search is made one line at a time. If SET SPAN is on, data can be matched even if the data spans two or more lines. If SET SPAN ON Blank is specified, a blank is inserted between the lines to be searched. This can be useful when locating words or sentences in a document.
6. If SET ARBCHAR is on, a "wild card" character can be specified in LOCATE subcommands. This character represents zero or more characters in the data to be matched. For example, if $ is the arbitrary character, then the pattern "A$B" would match "AB" or "AnotherstuffB". If SET ARBCHAR OFF is specified, the arbitrary character is ignored. The default settings are OFF $, meaning that a dollar sign ($) is used as the arbitrary character, but is not enabled.
7. If SET HEX is on, hexadecimal data can be specified in the form X'*dd...*', where *dd...* represents one or more pairs of hexadecimal digits (0–9, a–f, A–F). Each pair corresponds to a single byte of data to be matched.
8. If the operand is not found, an error message is displayed.

NEXT subcommand

The NEXT subcommand scrolls one or more lines toward the bottom of the file list.

The format of the NEXT subcommand is shown below:

Next	[*n* | *]

Where:

n | * — The optional number of lines to scroll. If not specified, *n* defaults to 1 (one). If * is specified, the bottom of the data is displayed.

Usage notes:

1. This subcommand is used to move forward in (toward the bottom of) the file list one or more lines (rows, or records) at a time. It is equivalent to the *+nn* function of the LOCATE subcommand.
2. The NEXT subcommand is synonymous with DOWN.

KFLIST PF key definitions

These PF key descriptions (and command line equivalents) are the default values for PF keys when using KFLIST, KRDRLIST, KPRTLIST, and KPUNLIST.

PFK	Command	Description
PF01	**HELP**	Provides detailed explanations for KFLIST functions.
PF02	**KBROWSE**	Browses the file whose fileid is on the line containing the cursor.
PF03	**QUIT**	Exits this level of the application.
PF04	**XEDIT**	Edits the file whose fileid is on the line containing the cursor.
PF05	**=**	Repeats the last command.
PF06	**/SB**	Sorts the file list in decending file block size order.
PF07	**BACKWARD**	Scrolls the display back one menu's worth of rows/records.
PF08	**FORWARD**	Scrolls the display forward one menu's worth of rows/records.
PF09	**/SD**	Sorts the file list in decending file date order.
PF10	**/ST**	Sorts the file list in ascending filetype order.
PF11	**/SN**	Sorts the file list in ascending filename order.
PF12	**CANCEL**	Exits all levels of the application.

Usage notes:

1. The PF keys can be tailored by editing the appropriate profile: PROFILE KFLIST, PROFILE KRDRLIST, PROFILE KPRTLIST, or PROFILE KPUNLIST.
2. PF keys can be changed dynamically by the QUERY PF subcommand, or set directly by the SET PF subcommand.

PRINTSCR subcommand

The PRINTSCR subcommand is used to print the contents of the current menu to the virtual printer.

The format of the PRINTSCR subcommand is shown below:

PRintscr	

Usage notes:

1. This subcommand prints the current menu image to the virtual printer. The menu, including any current user input, is printed within a numbered box. A header line displays the menu name, the date, time, and the number of lines on the menu.
2. The PRINTSCR subcommand is synonymous with the PSCREEN subcommand.

PSCREEN subcommand

The PSCREEN subcommand is used to print the contents of the current menu to the virtual printer.

The format of the PSCREEN subcommand is shown below:

PSCreen	

Usage notes:

1. This subcommand prints the current menu image to the virtual printer. The menu, including any current user input, is printed within a numbered box. A header line displays the menu name, the date, time, and the number of lines on the menu.
2. The PSCREEN subcommand is synonymous with the PRINTSCR subcommand.

QQUIT subcommand

The QQUIT subcommand will terminate all levels of KFLIST.

The format of the QQUIT subcommand is shown below:

QQuit	

Usage notes:

1. If any program was called between levels of KFLIST, the QQUIT process will stop at that level. When this program ends, the QQUIT process will continue. This will continue until KFLIST is completely ended.

QUIT subcommand

The QUIT subcommand will terminate this level of KFLIST.

The format of the QUIT subcommand is shown below:

QUIT	

Usage notes:

1. If this is the first level of KFLIST, the entire application is exited.

RESCAN subcommand

The RESCAN subcommand internally rescans the CMS file system using the initial selection pattern provided on the KFLIST command line.

The format of the RESCAN subcommand is shown below:

RESCan	

Usage notes:

1. This subcommand is used to recalculate the list of files that match the initial selection criteria. To improve preformance, KFLIST keeps track of all files that it found when it first starts running. If any new files are created matching the original criteria, these can only be identified by restarting KFLIST or by using the RESCAN subcommand.

SORT subcommand

The SORT subcommand sorts the list of files by various criteria.

The format of the SORT subcommand is shown below:

SORT	Name Type Mode Length Block Date Kount Format

Where:

Name Sorts files by ascending filename, filetype, then filemode.

Type Sorts files by ascending filetype, filename, then filemode.

Mode Sorts files by ascending filemode, filename, then filetype.

Length Sorts files by descending logical record length.

Block Sorts files by descending block count.

Date Sorts files by descending file date, then time.

Kount Sorts files by descending record count.

Format Sorts files by descending record format.

Usage notes:

1. This subcommand is used to reorder the file list based on certain fields.
2. This subcommand can be put into the appropriate PROFILE files—KFLIST, KRDRLIST, KPRTLIST, KPUNLIST—if you want the list to always be sorted initially.

TOP subcommand

The TOP subcommand makes the first listed file the current line.

The format of the TOP subcommand is shown below:

TOP	

Usage notes:

1. This subcommand is used to move quickly to the beginning of the file list.

UP subcommand

The UP subcommand scrolls one or more lines toward the top of the file list.

The format of the UP subcommand is shown below:

Up	[*n* | *]

Where:

n | *
: The optional number lines of data to scroll. If not specified, *n* defaults to 1 (one). If * is specified, the top of the file list is displayed.

Usage notes:

1. This subcommand is used to move backward in (toward the top of) the file list one or more lines (rows or records) at a time. It is equivalent to the *-nn* function in LOCATE.

Index

Special Characters

A

B

C

D

E

F

H

K

L

M

N

O

P

Q

R

S

T

U

V

W

X