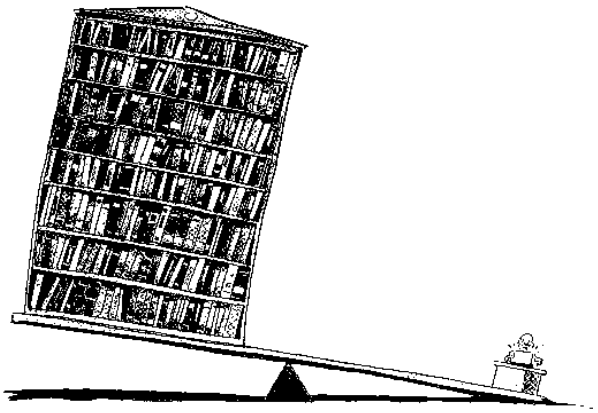
KE Texpress



Interface Guide

KE Software Pty Ltd

Contents

Chapter 1 Introduction 1-1

Input/Output Decoding 1-3
Languages and Operational Levels 1-4

Chapter 2 Keyboard Bindings 2-1

Default Bindings 2-3
Bindings Format 2-3
Determining a Command Name 2-4
Installing Bindings 2-5
Removing Bindings 2-5
Function Key Defaults 2-6
Other Considerations 2-6

Chapter 3 Messages 3-1

Default Messages 3-3
Messages Format 3-3
Printf FormatSpecifiers 3-3
European Characters 3-4
Installing Messages 3-5
Removing Messages 3-7

Chapter 4 Help Files 4-1

Default Help Files 4-3
Installing Help Files 4-3
Removing Help Files 4-4

Chapter 5 Terminal Descriptions 5-1

Terminal Identification 5-3
Viewing a Terminal Description 5-4
Character Sequence Representations 5-8
Terminal Characteristics 5-9
Terminal Name 5-9
Cursor Attributes 5-9
Screen Mode Attributes 5-13
Function Keys 5-16

European Characters 5-17
Line Drawing Characters 5-19
Installing a Terminal Description 5-21
Removing a Terminal Description 5-21
Using an Alternative Terminal Description 5-22

Chapter 1

Introduction

Input/Output Decoding..... 1-3
Languages and Operational Levels..... 1-4

Overview

KE Texpress provides several interface tailoring tools for altering the look and feel of a particular database.

In this chapter a general overview of the role of each of the interface components is provided.

Chapter 2 describes the procedure by which the DBA may change the keyboard bindings for Texpress commands. Chapter 2 also provides the default function key mappings.

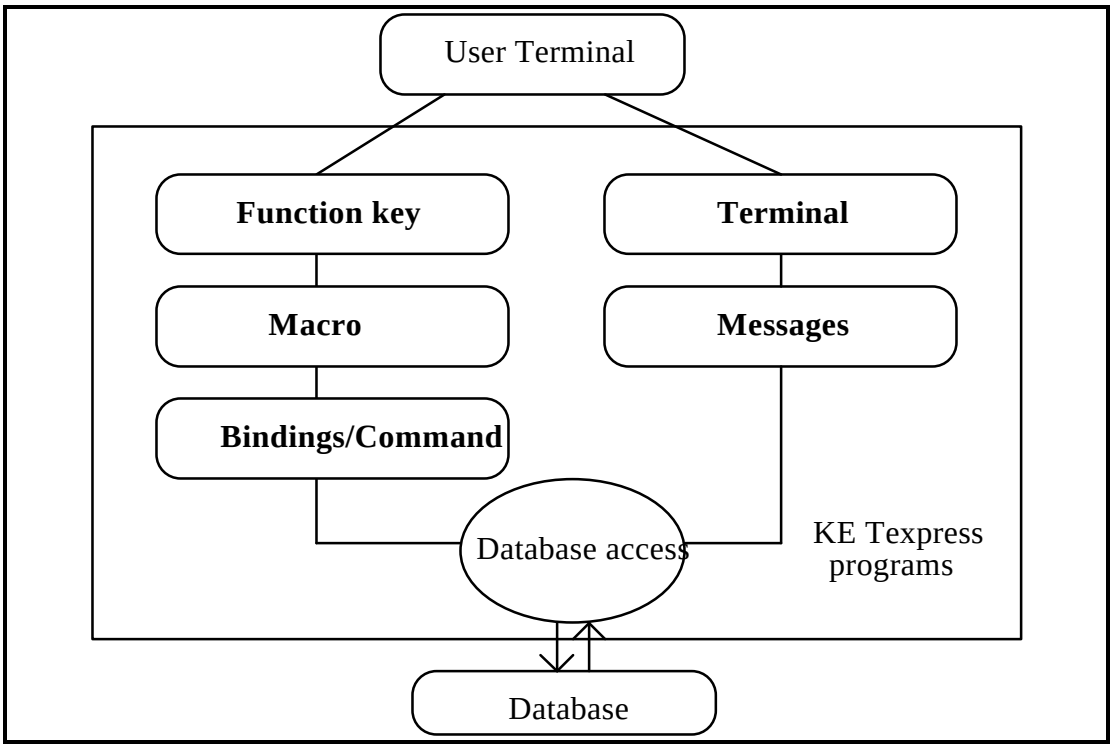
Chapter 3 describes how all display information can easily be changed by the DBA.

Chapter 4 discusses the structure of the KE Texpress on-line help files. Also described is a procedure for tailoring these help files to a particular application.

Chapter 5 describes facilities for defining the character sequences required to manipulate the screen attributes for a particular terminal type.

Input/Output Decoding

The following figure shows the interface structure of KE Texpress programs.



KE Texpress interface structure

Input decoding first determines whether the input character sequence was generated by a keyboard function key. As the function keys of different types of terminals can generate different sequences of characters, KE Texpress requires, for each terminal type used, a mapping between the terminal function keys and the actual character sequences generated by these function keys.

Next each character typed by the user is checked to determine, to determine whether it triggers a macro expansion (i.e. the input character is to be replaced by a sequence of other input characters). For more information regarding the use of macros consult the KE Texpress User Guide.

Next keyboard bindings are consulted to determine the actual command to be invoked. The user may select a command from a pull down menu or by typing the keyboard binding for the command. A keyboard binding for a command may be a function key, or combination of lower case, upper case or control characters. In order to use command bindings with function keys, a function key mapping must exist for the terminal in use.

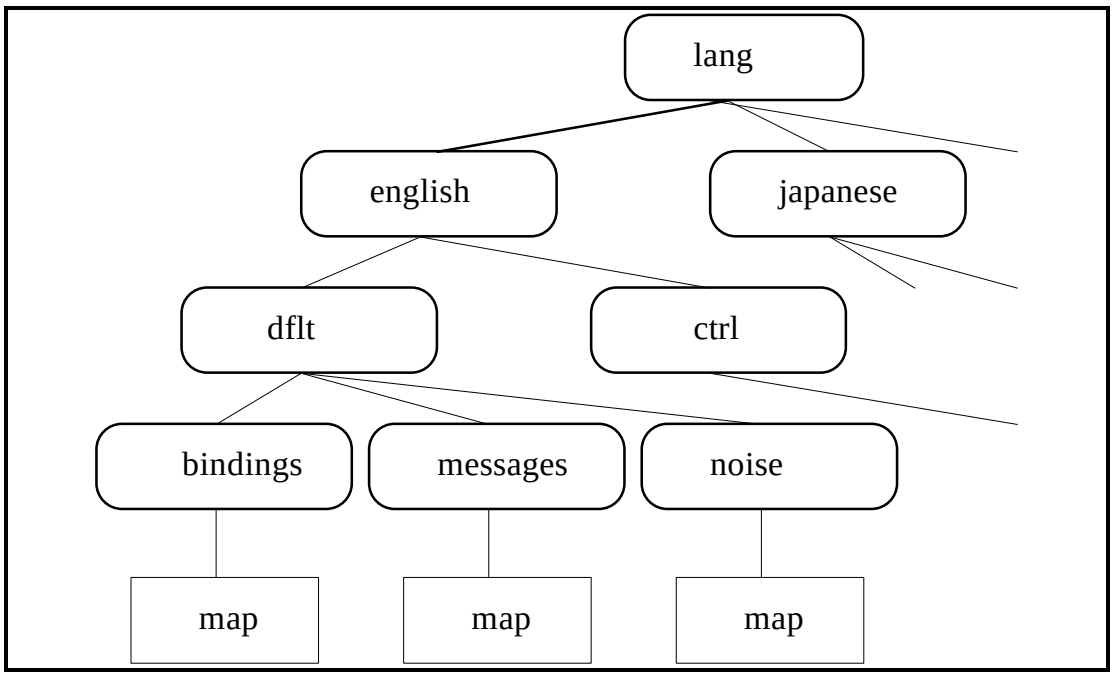
Output decoding first consults a compiled message catalgue file to determine the text to be displayed. Text such as menu titles, prompts, dialogue boxes etc. displayed by KE Texpress are not encoded within each program, but rather are extracted as required from this compiled message catalogue file.

KE Texpress may be used on a variety of terminals. Each particular terminal type expects certain distinct character sequences to be sent to the terminal to modify the display attributes. For example, a vt100 terminal requires a certain character sequence to change the current output attribute to reverse video; a wyse50 terminal requires a different character sequence to perform a similar function. The final phase of output decoding determines the correct character sequences to send to the user's terminal.

Languages and Operational Levels

The particular keyboard bindings and messages files to be used are determined by the values of two environment variables, LANG and TLEVEL. These variables indicate the language and the level of operation respectively. Multiple bindings and/or messages files may be available.

Part of the language directory structure is shown in the following figure :



Language directory structure

KE Texpress uses the **LANG** environment variable to determine which language is in use. This can be set from **csh** by the command:

```
setenv LANG language
```

or from **sh** by the commands:

```
LANG=language
export LANG
```

If this environment variable is not set, KE Texpress uses **english** as the default language.

KE Texpress uses the **TLEVEL** environment variable to determine the desired operation level. This can be set in a similar manner to the LANG environment variable. If this environment variable is not set, KE Texpress uses **dflt** as the default operation level.

Another operation level supported provided is the **ctrl** level. This level is suitable for terminals which do not support function keys. A different keyboard bindings file is utilised.

When the dflt keyboard bindings are used pull down menu options display function keys (where applicable). When ctrl keyboard bindings are used pull down menu options display control keys.

The directory structure shown in the previous figure exists under the texpress home directory and provides an example of the KE Texpress language files. A language structure specific to one database may be established by using various commands describe later in this manual. This will in fact create language directory structure under the database directory.

When invoking a database, KE Texpress has to determine which language files to use for the existing settings of LANG and TLEVEL. Texpress looks first in the database directory. If no appropriate language files are found, the KE Texpress files are used.

The selection process comprises the following squence of steps and halts as soon as a language file is found.

(1) Select the database files for the given language and the given operation level.

(2) Select the database files for the given language and the **dflt** operation level.

(3) Select the database files for the **english** language and the given operation level.

(4) Select the database files for the **english** language and the **dflt** operation level.

(5) Select the KE Texpress files for the given language and the given operation level.

(6) Select the KE Texpress files for the given language and the **dflt** operation level.

(7) Select the KE Texpress files for the **english** language and the given operation level.

(8) Select the KE Texpress files for the **english** language and the **dflt** operation level.

Other files are also stored under the language directory. The selection process for these files depends on the value of the LANG environment variable but not on the value of TLEVEL. Such files include are the noise word list in the noise sub-directory (refer to the KE Texpress Design Guide), the backup device table in the devices sub-directory (refer to the KE Texpress Maintenance Guide) and the on-line help files. The steps in the selection process for these files is similar to that described previously, without the distinction between operational levels.

Chapter 2

Keyboard Bindings

Default Bindings 2-3
Bindings Format 2-3
Determining a Command Name 2-4
Installing Bindings 2-5
Removing Bindings 2-5
Function Key Defaults 2-6
Other Considerations 2-6

Overview

The DBA has the ability to change the keyboard bindings of KE Texpress commands for a particular database.

This chapter describes the format of the source keyboard bindings file and the program, **texbind**, which is used to compile and install keyboard bindings.

Default Bindings

The default source keyboard bindings are stored in the file:

```
~texpress/ lang/ english/ dflt/bindings/text
```

where **~texpress** should be expanded to the home directory of the **texpress** Unix user account.

The default compiled bindings file (used by programs) is stored in:

```
~texpress/ lang/ english/ dflt/bindings/map
```

Another source variation of the default bindings is stored in the file:

```
~texpress/ lang/ english/ctrl/bindings/text
```

which is for the **ctrl** operation level (see Chapter 1 for information on operation levels).

Bindings Format

The keyboard bindings source file contains the names of each of the KE Texpress commands together with each of the characters which can be used to invoke the commands

In the file, the character, #, introduces a comment. Everything on the line after the # is ignored. Blank lines are also ignored.

The structure of each command definition is as follows:

```
cmdname          c1  c2  c3  ...
```

where ***cmdname*** is the name of the KE Texpress command and ***c1***, ***c2, c3*** etc. are keyboard bindings for the command.

The first character defined for a particular command name, will appear on the pull down menu for that command. Valid characters are in one of the following formats:

`'x'`	The character, *x*, where *x* can be any keyboard character except a function key.
`'^X'`	Ctrl+*X*, where *X* can be any alphabetic character.
`all(x)`	Any of the three characters, *x*, *x* or Ctrl+*X*, where *x* can be any alphabetic character.

`escape, ESC,`	The escape key.
`space, blank`	The space bar.
`newline, NL`	The newline or enter key.
`return, CR`	The carriage return key.
`BS`	The backspace key.
`DEL, RUBOUT`	The interrupt key.
`F`*`n`*	Function key number *n* where *n* is a number between 1 and 32.
`func`	One of the named function keys recognized by KE Texpress. These named function keys are described in Chapter 5 of this manual.

A shorthand form of a command binding can be specified as follows:

```
cmdname        >  previouscmdname
```

where *cmdname* is the name of the KE Texpress command being defined and *previouscmdname* is the name of a previously defined KE Texpress command. This indicates that the keyboard bindings for *previouscmdname* will also apply to *cmdname*.

Determining a Command Name

The name of a command can be determined using the KE Texpress help facility (refer to the KE Texpress User Guide or the KE Texpress Design Guide). Using the help command while an option is selected on a pull down menu, displays a help box, the last line of which has the following format:

```
Invoked by:  list of characters          Name:  cmdname
```

where *cmdname* is the name of the command. This name is used in the in the source keyboard bindings file.

Other KE Texpress commands, which do not have an explicit help file associated with them, may also appear in the bindings file.

In some circumstances, KE Texpress makes use of two character commands. If the name of a command has "**_2**" appended, the keyboard binding for the command comprises two characters - the standard lead character for this mode (also defined in the source bindings file), typically **Ctrl+T** followed by one of the

characters defined to invoke this command. Thus commands can be changed to two characters by appending "**_2**" to their names or converted to single characters by removing the "**_2**".

Installing Bindings

It is advisable not to alter the original, distributed source keyboard bindings files but rather to create new files. To achieve this, the user should first copy the original source bindings file, using a Unix command of the form:

```
cp ~texpress/lang/english/dflt/bindings/text bindingsfile
```

This creates a file, *bindingsfile*, containing a copy of the original keyboard bindings. It can be altered using any text editor.

Before installing the new bindings, the user should determine the language and operation level applicable to the bindings. If required, the **LANG** and **TLEVEL** environment variables should be set as appropriate.

User **texpress** may install the bindings for all KE Texpress databases using the command:

```
texbind bindingsfile
```

It should be ensured that no people are using KE Texpress when new keyboard bindings are installed.

The DBA may install the bindings for a particular database using the command:

```
texbind -ddbname bindsingsfile
```

The DBA should ensure that no users are currently using the database when new keyboard bindings are installed.

All subsequent users will access the new bindings file (provided that they select the appropriate language and operation level).

Removing Bindings

Keyboard bindings can be removed by installing an empty source bindings file.

To remove the bindings for the *dbname* database (assuming appropriate values for the LANG and TLEVEL environment variables) the following command can be used:

```
texbind -ddbname /dev/null
```

Function Key Defaults

Source keyboard bindings files distributed with KE Texpress bind common commands to keyboard function keys. Function key bindings must exist for the terminal type in use (refer to Chapter 5).

The function key bindings are of a general nature. The default bindings are:

Exit	F1
Help	F2
Query	F3
Insert, Write, Edit	F4
Forward	F5
Backward	F6
Look-up table	F7
Select, Choose	F8
Move left	←
Move right	→
Move up	↑
Move down	↓

Other Considerations

Caution must be exercised when changing keyboard bindings to ensure that inconsistencies are not created.

KE Texpress does not check whether the same character is a keyboard equivalent for two different commands which may be available in the same mode of operation of a particuluar program. If this situation should arise, entering keyboard binding will always select one of the commands. The other command will never be invoked by the binding (it may still be invoked by selection from a pull down menu).

Several characters used by the Unix operating system should not in general be included in keyboard bindings. These are Ctrl+S and Ctrl+Q, which are often

used for line flow control between the terminal and the computer. Also on Unix systems with job control, Ctrl+Z is used to suspend programs. Variations of these commands, such as all(s), all(q) and all(z), should also be avoided.

Chapter 3

Messages

Default Messages 3-3
Messages Format 3-3
Printf Format Specifiers 3-3
European Characters 3-4
Installing Messages 3-5
Removing Messages 3-7

Overview

Text displayed by KE Texpress is not encoded within each program but rather is extracted from a file as required. The DBA therefore has the ability to tailor menu options, prompts, dialogue boxes and error messages.

Combined with keyboard bindings, this facility enables completely different interfaces to be developed. This includes translations into languages other than English, and the development of interfaces for varying levels of operation.

Default Messages

The KE Texpress software distribution provides a default catalogue of messages for the English language. This is **dflt**, for the default KE Texpress environment.

The values of the LANG and TLEVEL environment variables are used to select the appropriate messages file (refer to Chapter 1). The default source messages files are typically stored in the files:

```
~texpress/lang/english/dflt/messages/3.3
~texpress/lang/english/dflt/messages/3.4
~texpress/lang/english/dflt/messages/5.0
```

The default compiled messages file (used by programs) is stored in:

```
~texpress/ lang/ english/ dflt/messages/map
```

Messages Format

In message files, comments are introduced with the # character. Everything on the line after the# is ignored. Blank lines are also ignored.

All other lines are of the form:

```
nnn     "text"
```

or

```
nnn     > nnn
```

where *nnn* is the number used by KE Texpress to uniquely identify the message, *text.*

Lines using the > notation indicate that the text for the first number is the same as that of the number following the > symbol (which must be specified earlier in the file).

The identifying numbers in the source message file must not be changed. The text portion of the line (enclosed in double quotes) may be edited as required. If the double quote character, **"**, is required within the text it must be preceded by the backslash character **\).**

Printf Format Specifiers

Some messages contain embedded Unix **printf** style format specifiers. These are introduced by the **%** character (refer to the Unix Programmers Manual). These

format specifiers are used for displaying information which is dependent on changing data. For example, they are used for database names, file names, record counts etc. The specification or order of these printf conversions should not be changed. It is not possible to add new printf conversions.

To assist with language translation, messages using printf style format specifiers may be altered so that arguments are accessed in an order different than the default.

To use this facility, immediately following the % character the notation ***n*$** may be used, where *n* is a number (commencing from 1) corresponding to the number of the argument in the default order. By default, argument specifiers are not required, the order 1, 2, 3, etc. is implied.

For example the message:

```
"Can't rename \"%s\" Report form as \"%s\""
```

implies:

```
"Can't rename \"%1$s\" Report form as \"%2$s\""
```

and could be changed to (presumably in another language):

```
"Name \"%2$s\" can't be used for \"%1$s\" Report form"
```

If desired, an argument may be ignored, by preceding the *n*$ notation with an **@** character. For example %@2$s would suppress printing of the second argument (it doesn't matter where this appears in the message).

When using the n$ notation care must be taken to ensure that format specifiers (normal or ignored) are supplied for all arguments in the message. At run time, if the message parsing function detects an error with format specifiers a warning message is displayed.

European Characters

European characters may be portably encoded within a message. Each european character encoding comprises two characters enclosed in curly parentheses. For example, the message:

```
"{,C}edilla"
```

would display as (on terminals that have the capability):

```
Çedilla
```

For a full list of all European character encodings refer to Chapter 5.

Installing Messages

Note that the default message catalogue is compiled from of a combination of three actual source message files:

```
~texpress/lang/english/dflt/messages/3.3
~texpress/lang/english/dflt/messages/3.4
~texpress/lang/english/dflt/messages/5.0
```

Messages are located in seperate files in order to minimise the upgrade work required when new versions of the KE Texpress software are released.

It is advisable not to alter the original source message files but rather to create new message catalogues. To achieve this, the user should first copy the original source message files using a Unix command similar to:

```
cp  ~texpress/lang/english/dflt/messages/5.0    messagefile
```

This creates a file called *messagefile* containing a copy of the original messages. It can be altered using any text editor.

Particular messages are best located by using the search facility of the text editor. Searching for a portion of text encompassed within the displayed message will quickly find the appropriate line of the file. Do not search for pieces of messages such as file names or database names as these are displayed using printf conversions and will not appear in the file.

Messages displayed on pull down menus are best located by searching for the command name. The command name can be determined using the KE Texpress help facility at the relevant point in the program (refer to Chapter 3). The actual command name is the same as the related help file name, which is defined in the message file. Located near this message in the file is the text displayed on the pull down menus for each available command. Some of the messages for command display will be set to the null value (""). These commands are not shown on a pull down menu. However, each individual command does have a unique identifying name, so it is possible to display all or any subset of the applicable commands. The order of display is pre-determined by KE Texpress and cannot be changed.

Before installing the new messages file the user should determine the language and operation levels which will apply and if required set the LANG and TLEVEL environment variables as appropriate.

User texpress can install a messages file for all databases by using the command:

```
texmessages  messagefile [messagefile ...]
```

For example to re-compile the original source message files user texpress could perform the commands (assuming use of sh):

```
LANG=english
TLEVEL=""
export  LANG  TLEVEL
cd  ~texpress/lang/english/dflt/messages
texmessages  3.3  3.4  5.0
```

It should be ensured that no people are using KE Texpress when new messages are installed.

The DBA can install a new message file for a particular database using the command:

```
texmessages  -ddbname  messagefile  [messagefile ...]
```

The DBA should ensure that no users are currently using the database when new messages are installed.

All subsequent users will access the new messages file (provided that they select the appropriate language and operation level).

It is possible to create a source message file which only contains changed messages. This file can then be combined the with the original source message files to produce a new compiled message file. The format of the source file containing the changed messages is the same as a normal source message file.

User texpress may install the new messages for all KE texpress users using the following command:

```
texmessages  3.3  3.4  5.0   newmessagefile
```

where 3.3, 3.4 and 5.0 are the original source message files (located in the directory ~texpress/lang/english/dflt/messages) and *newmessagefile* contains changes to selected messages.

The DBA may install changed messages for a particular database by adding the:

```
-ddbname
```

argument to the previous command.

Removing Messages

A compiled message file can be removed by installing an empty message file.

For example to remove the message file for the dbname database (assuming the appropriate settings of the LANG and TLEVEL environment variables) the DBA may use the command:

```
texmessages  - ddbname  /dev/null
```

Users of the *dbname* database would then revert to use of the default KE texpress message file.

Chapter 4

Help Files

Default Help Files..... 4-3
Installing Help Files..... 4-3
Removing Help Files..... 4-4

Overview

KE Texpress provides the system the ability to install help files tailored to a particular site (or language) or to a particular database.

There are numerous help files which describe the following:

- Each individual KE Texpress command. These files contain the text displayed when help is requested for the pull down menu command currently selected. The name of the help file is the same as the command name.

- Each mode of KE Texpress. These files are displayed when the help command is typed when no pull down menu is currently displayed. The information provides a overall view of the current mode of operation. To determine the name of such a file, it is necessary to search all of the help files for some of the information in this file. Mode help files all have names with a *.m* suffix.

Default Help Files

For efficient access, help files are stored under directories whose names correspond to the first letter of the help file name. These directories are all stored in the help directory:

```
~texpress/lang/english/help
```

Installing Help Files

The help files distributed with KE Texpress should not be altered. New help files can be created for translation to another language by copying the English lanugage files. The following command sequence provides an example of how this could be achieved:

```
cd ~texpress/lang/english
find help -depth -print | cpio -pduv
                          ~texpress/lang/lang
```

The name *lang* represents the new language name (e.g. japanese, french, etc.). The *lang* directory must already exist. It will exist if a bindings, messages or noise word file has previously been defined and installed for the given language. Otherwise it should be explicitly created. This type of installation can be performed only by user **texpress.**

To install new help files for a particular database (any language), the following command sequence can be used (assuming the database resides in the default location)

```
cd ~texpress/lang/english
find help -depth -print | cpio -pduv
                ~texpress/data/local/dbname/lang/lang
```

where *lang* is the new language and *dbname* is the name of the database. This directory must already exist. It will exist if a bindings file, messages file or noise word file has been compiled for the given language in the given database. This type of installation can be performed only by the DBA of the database.

If tailored help files are to be used then only those help files that have changed need be installed in the new help directory. KE Texpress will always first look for a tailored help file, and failing that, use the default help file.

Help files can be edited using text editor. When help files are installed, they immediately become active.

Removing Help Files

The help files for a particular database *dbname* and language *lang* can be removed using the following command sequence (assuming that the database resides in the default location):

```
cd  ~texpress/data/local/ dbname/lang/lang
rm  -fr  help
```

This removal of help files may only be performed by the DBA of the database.

Chapter 5

Terminal Descriptions

Terminal Identification 5-3
Viewing a Terminal Description 5-4
Character Sequence Representations 5-8
Terminal Characteristics 5-9
- Terminal Name 5-9
- Cursor Attributes 5-9
- Screen Mode Attributes 5-13
- Function Keys 5-16
- European Characters 5-17
- Line Drawing Characters 5-19

Installing a Terminal Description 5-21
Removing a Terminal Description 5-21
Using an Alternative Terminal Description 5-22

Overview

This chapter describes facilities for describing the attributes of a particular type of terminal. This compiled terminal description defines the character sequences required to correctly display information on the screen as well as the character sequences generated by particular function keys of the terminal.

A KE Texpress terminal description is similar to a Unix *terminfo* and *termcap* descriptions.

The terminal description is divided into five areas:

- Cursor management
- Screen mode attributes
- Function keys
- European characters
- Line drawing characters

Terminal Identification

Each terminal type has a name which Unix programs use for identification. Typically the name consists of the terminal brand name followed by a model identification number. Examples of common terminal type names are ansi, vt100, wyse60, ibm3151 etc.

KE Texpress provides terminal descriptions for a wide variety of terminal types. These descriptions state the characteristics of the terminal such as cursor motion, character editing sequences, methods of highlighting and other attributes. If the correct terminal name (indicating the description file to use) is not defined, KE Texpress may display text in a scrambled manner.

The directory:

```
~texpress/ etc/terms
```

contains sub-directories which hold numerous files, each with a description of a particular terminal type. Terminal description files reside in the sub-directory corresponding to the first character of the terminal name. For example the vt100 terminal description resides in the file:

```
~texpress/etc/terms/v/vt100
```

Typically the Unix system administrator will have defined the terminal type in use. This value can be verified by viewing the the **TERM** environment variable. Users of **csh** can do this by typing env or printenv. Users of sh can check the TERM value by typing the command set.

If the TERM environment variable is not set or is set incorrectly it can be defined by csh users with the command:

```
setenv  TERM   termname
```

in their .login file in their home directory (where *termname* is the appropriate terminal name).

Users of sh should have the commands:

```
TERM=termname
export  TERM
```

in their .profile file. If you modify your .login or .profile file you should logout and login again to ensure the new value is recognised.

Viewing a Terminal Description

The KE Texpress terminal description for a particular terminal *termname* can be viewed using the command:

```
texunterm   termname
```

Compiled terminal descriptions for many standard terminal types are distributed with the KE Texpress software. These include:

```
ampex
ansi
dg412
esprit
freedom
hp2382
hp2392
ibm3151
ibm3161
ibm3163
ibm3164
ibm5151
kterm
prime
sun
sun-cmd
sun3
sun4
tunix
vt100
vt200
vt220
vt300
wyse50
wyse60
xterm
```

The terminal description for the **ansi** terminal type can be viewed using the command:

```
texunterm   ansi
```

This displays (on the standard output):

```
TERM            ansi                        (terminal name)

CURSOR_MOTION   "\E[%i%p1%d;%p2%dH"         (cursor attributes)
CLEAR_BOT       "\E[J"
CLEAR_EOL       "\E[K"
INSERT_CHAR     "\E[@"
LINES           "24"
COLS            "80"
TERM_ON         "\E[44m"
TERM_OFF        "\E[37m"
                                        (screen mode attributes)
```

```
MODE1           "\E[46m\E[1m\E[37m", "\E[ m\E[44m"
MODE2           "\E[41m\E[1m\E[37m",  "\E[m\E[44m"
MODE3           "\E[32m",
MODE4           "\E[1m\E[37m", "\E[m"
MODE5           "\E[37m",
MODE6           "\E[1m\E[34m", "\E[m"
MODE7           "\E[1m\E[31m", "\E[m"
MODE8           "\E[1m\E[36m", "\E[m"
MODE9           "\E[1m\E[33m", "\E[m"
MODE10          "\E[35m",
MENU_BAR        MODE1
MENU_SELECT     MODE2
MENU_LINES      MODE3
OPT_SELECT      MODE2
OPT_VALID       MODE4
OPT_INVALID     MODE5
HELP_TEXT       MODE5
HELP_PROMPT     MODE6
HELP_LINE       MODE5
HELP_TITLE      MODE7
MESSAGE_LINE    MODE9
MESSAGE_TEXT    MODE8
MESSAGE_TITLE   MODE7
MORE_TEXT       MODE5
MORE_SHOW       MODE4
DATA            MODE4
HEADER          MODE8
PROMPT          MODE8
BORDER_TEXT     MODE7
BORDER_LINE     MODE9
COPY_TEXT       MODE8
MATCH_SHOW      MODE7
BOX             MODE8
HIGHLIGHT       MODE7
F1              "\E[M"          (function keys)
F2              "\E[N"
F3              "\E[O"
F4              "\E[P"
F5              "\E[Q"
F6              "\E[R"
F7              "\E[S"
F8              "\E[T"
F9              "\E[U"
F10             "\E[V"
F11             "\E[W"
F12             "\E[X"
UP              "\E[A"
DOWN            "\E[B"
LEFT            "\E[D"
RIGHT           "\E[C"
PREVSCREEN      "\E[I"
NEXTSCREEN      "\E[G"
INSERTCHAR      "\E[L"
DELETECHAR      "^?"
HOME            "\E[H"
END             "\E[F"
Aumlaut         "\216"          (european characters)
Aring           "\217"
AE              "\222"
```

```
Ccedilla        "\200"
Eacute          "\220"
Ntilde          "\245"
Oumlaut         "\231"
Uumlaut         "\232"
agrave          "\205"
aacute          "\240"
acircum         "\203"
aumlaut         "\204"
aring           "\206"
ae              "\221"
ccedilla        "\207"
egrave          "\212"
eacute          "\202"
ecircum         "\210"
eumlaut         "\211"
igrave          "\215"
iacute          "\241"
icircum         "\214"
iumlaut         "\213"
ntilde          "\244"
ograve          "\225"
oacute          "\242"
ocircum         "\223"
oumlaut         "\224"
ugrave          "\227"
uacute          "\243"
ucircum         "\226"
uumlaut         "\201"
yumlaut         "\230"
pound           "\234"
LINE1010        "\263"          (line drawing)
LINE1011        "\264"
LINE1012        "\265"
LINE2021        "\266"
LINE0021        "\267"
LINE0012        "\270"
LINE2022        "\271"
LINE2020        "\272"
LINE0022        "\273"
LINE2002        "\274"
LINE2001        "\275"
LINE1002        "\276"
LINE0011        "\277"
LINE1100        "\300"
LINE1101        "\301"
LINE0111        "\302"
LINE1110        "\303"
LINE0101        "\304"
LINE1111        "\305"
LINE1210        "\306"
LINE2120        "\307"
LINE2200        "\310"
LINE0220        "\311"
LINE2202        "\312"
LINE0222        "\313"
LINE2220        "\314"
LINE0202        "\315"
LINE2222        "\316"
LINE1202        "\317"
```

```
LINE2101        "\320"
LINE0212        "\321"
LINE0121        "\322"
LINE2100        "\323"
LINE1200        "\324"
LINE0210        "\325"
LINE0120        "\326"
LINE2121        "\327"
LINE1212        "\330"
LINE1001        "\331"
LINE0110        "\332"
```

The format of the output is:

```
attribute       "value"
```

where *attribute* denotes a terminal characteristic and *value* represents the character sequence applicable for the named terminal.

The command texunterm can be used to find out the terminal descriptions for a number of terminals. For example, to list the terminal descriptions for an **ansi** terminal and a**vt100** terminal the following command may be used:

```
texunterm  ansi  vt100
```

Character Sequence Representations

Character sequences for terminal attributes are enclosed within double quotes. For example:

"*abc*"

defines the sequence of characters *abc* for a particular terminal attribute. Various symbols may be used to represent non-printable characters. These symbols are:

^[, \E	escape
\n	newline
\b	backspace
\t	horizontal tab
\r	carriage return
\f	form feed
\v	vertical tab
\\	backslash
nnn	character with octal value *nnn*
^*X*	Ctrl+*X* (where *X* is in the range A-Z)
^?	DEL
^]	Ctrl+]
^\	Ctrl+\
^^	Ctrl+^
^_	Ctrl+_

For example, to represent the character sequence ESC Ctrl+A, the following could be used:

"\E^A"

When attributes are transmitted to a terminal a delay may be required between the transmission of individual characters to allow the terminal enough time to interpret the sequence. A delay may be placed anywhere in a character sequence as follows:

$<*nnn*>

where *nnn* is the required delay period in milliseconds. To transmit the sequence ESC ctrl A with a five millisecond delay after the sequence has been sent, the following could be used:

"\E^A$<5>"

Delays may be used in all character sequences other than those used to define function keys and European characters.

Terminal Characteristics

This section is broken into six parts where each part describes a distinct set of terminal attributes.

Terminal Name

Each terminal description must commence with the name of the terminal. This is achieved with the **TERM** statement. The **TERM** statement must appear on the first non comment line of the terminal definition. The character, #, introduces a comment. Everything on the line after the # is ignored. Blank lines are also ignored.

The format for defining the terminal name is:

```
TERM    termname1, termname2, ...
```

where *termname1* is the first name by which the terminal description may be accessed (i.e. a name to which the TERM environment variable may be set). More than one name may be supplied for a given terminal definition. Each name must be separated by a comma. An identical compiled terminal description is created for each of the terminal names supplied.

It is possible to compile more than one terminal description at a time. Any occurrence of the TERM statement within a description file signals the start of a new terminal description.

Cursor Attributes

The cursor attributes section of a terminal description indicates the character sequences which need to be sent to the terminal to perform screen manipulation functions. Some of these attributes are compulsory. Also some attributes require certain special character sequences for accessing parameters.

CURSOR_MOTION (compulsory, parameters)

Used to position the cursor at a specific location on the screen. Two parameters are supplied, the screen row and column numbers respectively. The top left corner of the screen is designated to be row zero and column zero.

CLEAR_BOT (compulsory)

Character sequence used to clear the screen from the current cursor position to the bottom of the screen.

CLEAR_EOL (compulsory)

Character sequence used to clear the screen from the current cursor position to the end of the line.

INSERT_CHAR

Some terminals will automatically scroll the screen when a character is explicitly written in the bottom right hand corner of the screen. To avoid this scrolling problem, this attribute is used to insert one character into a line and move the characters after the cursor one position to the right.

CURSOR_RIGHT (parameter)

This attribute takes one parameter which is the number of columns to move the cursor right from it's current position.

CURSOR_DOWN (parameter)

This attribute takes one parameter which is the number of lines to move the cursor down from it's current position.

LINES (compulsory)

To define the number of lines allowed for a particular type of terminal this attribute may be used. For systems which support variable size windows this value is ignored and the actual window size is used.

COLS (compulsory)

To define the number of columns of characters allowed for a particular type of terminal this attribute is used. For systems which support variable size windows this value is ignored and the actual window size is used.

CURSOR_ON

This attribute is used to make the cursor visible. This attribute need not be defined for terminals which do not support this feature.

CURSOR_OFF

If the terminal allows the cursor to be hidden, then this attribute should be set. For terminals which do not support this feature the cursor, "when turned off", is placed in the top right hand corner of the screen.

EXT_ON

Some terminals require different fonts to be used when displaying European characters. This attribute, if required, is used to set the appropriate font.

EXT_OFF

If a different font is required to display European characters, then this attribute is used to reset the font to the standard font.

LINE_ON

Some terminals require different fonts to be used when displaying line drawing characters. This attribute, if required, is used to set the appropriate font.

LINE_OFF

If a different font is required to display line drawing characters, then this attribute is used to reset the font to the standard font.

TERM_ON

When KE Texpress commences it first sends this attribute. This allows initialization sequences to be sent to a terminal.

TERM_OFF

Before KE Texpress terminates this attribute is the last sequence sent. This allows attribute to reset any terminal settings to be reset if necessary.

NEXT_LINE

To position the cursor at the start of the next line this attribute is used.

DISPLAY_TYPE

This attribute is generally not set. If the terminal description is for an H.P. type terminal then this attribute should be set to "1". Note other settings like chinese display & so on are now available.

The CURSOR_MOTION, CURSOR_RIGHT and CURSOR_LEFT attributes require parameters to be specified as part of the character sequence. The parameter mechanism uses a stack and special % codes for manipulation of the stack. Typically, a sequence will push one of the parameters onto the stack and then print it in some format.

The % encodings have the following meaning:

`%%`	Output '%' character.
`%d`	Pop stack and print value.
`%2d`	Pop stack and print value as two digits with leading blanks.
`%3d`	Pop stack and print value as three digits with leading blanks.
`%02d`	Pop stack and print value as two digits with leading zeros.

`%03d`	Pop stack and print value as three digits with leading zeros.
`%c`	Pop stack and print value as a single character.
`%s`	Pop stack and print value as a sequence of characters.
`%p1`	Push the first parameter onto the stack.
`%p2`	Push the second parameter onto the stack.
`%P[`*a-z*`]`	Pop stack and set to variable [*a-z*].
`%g[`*a-z*`]`	Push the value of variable [*a-z*].
`%'`*c*`'`	Push character constant *c*.
`%{`*nn*`}`	Push integer constant *nn*.
`%+ %- %* %/ %m`	Pop top two values and apply arithmetic operator pushing the result (%m is modulo).
`%& %| %^`	Pop top two values and apply bitwise operator pushing the result (%& is and, %| is or, %^ is exclusive or).
`%= %> %<`	Pop top two values and apply logical operator pushing the result (0 is false, 1 is true).
`%! %~`	Apply unary operator to top of stack (%! is logical not, %~ is complement).
`%i`	Increment first two parameters (for ansi terminals)
`%? `*expr*` %t `*thenpart*` %e `*elsepart*` %;`	If-then-else construct. The %e *elsepart* is optional.

Binary operations (%+, %-, %*, %/, %m, %&, %|, %^, %=, %<, %>) are in postfix form.

For example, to add 5 to the first parameter, one would use, "%p1%{5}%+". The value for CURSOR_MOTION from the ansi terminal description is:

```
CURSOR_MOTION              "\E[%i%p1%d;%p2%dH"
```

This states that to generate a cursor movement sequence the characters "\E[" must first be sent. The two parameters must then be incremented (%i). This is required as ansi terminals force the top left corner of the screen to be row one and column one instead of zero. The row value (%p1) is pushed onto the stack and then popped and printed as a number. A semi-colon is sent followed by the column number (%p2%d) and finally a 'H' character. So on an ansi terminal to move to row and column zero the character sequence:

```
"\E[1;1H"
```

would be sent.

Screen Mode Attributes

Screen mode attributes are used to configure the display attributes for KE Texpress objects. For example, a screen mode attribute exists for displaying prompts. On one type of terminal prompts may be defined to be displayed in blue, while on another terminal type they may be displayed in reverse video.

Screen mode attributes are optional. If a particular screen mode attribute is not defined, KE Texpress will use a suitable default display mode.

The definition of a mode is as follows:

```
modename        "on-sequence", "off-sequence"
```

where *modename* is a name used to reference the definition. Any unique name may be used. The *on-sequence* contains the characters to send to the terminal to enable the mode (e.g. if the mode is to display text in reverse video then *on-sequence* defines the character sequence required to put the terminal into reverse video). The *off-sequence* is used to reset the terminal to its default mode. Both the *on-sequence* and *off-sequence* do not have to be specified. However, the comma separator is compulsory.

To assign a mode to a screen attribute the following statement is required:

```
attribute       modename
```

where *attribute* is one of the valid screen attributes described below, and *modename* is the name of the mode to which the attribute is to be set. The *modename* must be defined before being referenced.

An example of defining a mode and setting a screen attribute to that mode is (for an **ansi** terminal type):

```
MODE1           "\E[46m\E[1m\E[37m", "\E[m\E[40m"
MENU_BAR        MODE1
```

In this example the *modename* is MODE1, the *on-sequence* is "\E[46m\E[1m\E[37m" which places an ansi terminal display into cyan background with a white foreground. The *off-sequence* is "\E[m\E[40m" which resets the screen to have a black background. The screen attribute is MENU_BAR which specifies the mode (or colour) for displaying the menu bar at the top of each screen. In this example the menu bar would therefore be displayed as a line of white characters on a cyan background.

Definable screen mode attributes are as follows:

MENU_BAR

The mode to display the menu bar which appears at the top of each screen.

MENU_SELECT

The mode in which the name of the currently pulled down menu is displayed. The name is situated in the menu bar itself.

MENU_LINES

The mode in which the lines which make up a pull down menu are shown .

OPT_SELECT

The mode in which the currently selected option is displayed.

OPT_VALID

The mode in which valid pull down menu commands are displayed.

OPT_INVALID

The mode in which invalid pull down menu options are displayed.

HELP_TEXT

Help information is displayed in this mode.

HELP_PROMPT

The messages at the bottom of a help box are displayed in this mode.

HELP_LINE

The mode in which the box surrounding the help information is shown.

HELP_TITLE

A title indicating the function on which help was required is centered in the top line of the help box. The title is displayed in this mode.

MESSAGE_LINE

This sets the mode for all boxes which appear as messages in the centre of the screen.

MESSAGE_TEXT

Any text which is shown as a message in the centre of the screen is displayed in this mode.

MESSAGE_TITLE

If KE Texpress requests the user to answer a question a title is centered in the top line of the message box. The title is displayed in this mode.

MORE_TEXT

When information is shown to the user a page at a time, the mode in which the text is displayed is determined by this attribute.

MORE_SHOW

If any text needs to be highlighted while being viewed a page at a time, the highlighted text is displayed using this attribute.

DATA

The mode of any information which the user enters is determined by this attribute.

HEADER

The mode of any headers within a form is determined by this attribute.

PROMPT

The mode of any prompts within a form, and any prompts when KE Texpress asks the user for information is determined by this attribute.

BOX

If a form contains any boxes then the mode in which the boxes are displayed is determined by this attribute.

HIGHLIGHT

When the forms editor is being used, any objects which are currently selected are displayed using this mode.

BORDER_TEXT

This attribute sets the mode for any text which is displayed in a standard box.

BORDER_LINE

This attribute sets the mode for any general purpose boxes displayed by KE Texpress.

MATCH_SHOW

This attribute determines the mode in which any terms which match a query are displayed.

Function Keys

KE Texpress allows up to 32 numbered function keys and 32 named function keys to be defined. A function key is recognised in the following manner: The statement:

```
F1      "\E[M"
```

indicates that the character sequence \E[M is mapped to F1 (function key one). This means that by pressing F1 on the keyboard, the character sequence \E[M is generated (remember that \E represents the escape character). KE Texpress then checks the terminal description and determines that this sequence corresponds to F1. It then consults the bindings (see Chapter 2) to determine the command which can be invoked by F1.

The character sequences generated by each terminal function key can generally be determined by typing from the Unix shell (for each key):

```
echo 'Fn' | cat -v
```

where *Fn* is the appropriate function key. The escape character is typically displayed as ^[, but it may be entered as \E in the terminal description file.

Recognised function key names are:

F1	Function key 1.
F2	Function key 2.
...	
F32	Function key 32.
UP	Up arrow
DOWN	Down arrow
LEFT	Left arrow
RIGHT	Right arrow
FIND	
HELP	
INSERT	
SELECT	
REMOVE	
PREVSCREEN	
NEXTSCREEN	
INSERTCHAR	
DELETECHAR	
INSERTLINE	
DELETELINE	
ERASELINE	
ERASEPAGE	
PRINT	

SEND
CLEAR
HOME

European Characters

KE Texpress provides support for the full European character set. Internally KE Texpress uses a special character representation for each European character. When the character needs to be displayed, the terminal description is consulted to find the correct sequence to be sent to the terminal. If the sequence is not defined, a default character is used. This allows European characters to be displayed by a series of default characters on terminals without the necessary font support.

The character sequence defined for a particular European character is used both to recognise the character when typed at the keyboard, and also to print the character on the screen. Delays are not permitted in European character sequences.

The following table below lists terminal description attribute name for each European character; the portable representation of the character which can be used in source bindings and messages files; the default display for terminals which do not support that particular European character; and a some further description.

Attribute	Portable Representation	Default Display	Description
Agrave	`A	A	A grave
Aacute	'A	A	A acute
Acircum	^A	A	A circumflex
Atilde	~A	A	A tilde
Aumlaut	"A	A	A umlaut
Aring	*A	A	A ring
AE	AE	A	A E ligature
Ccedilla	,C	C	C cedilla
Egrave	`E	E	E grave
Eacute	'E	E	E acute
Ecircum	^E	E	E circumflex
Eumlaut	"E	E	E umlaut
Igrave	`I	I	I grave
Iacute	'I	I	I acute
Icircum	^I	I	I circumflex

Iumlaut	"I	I	I umlaut
Ntilde	~N	N	N tilde
Ograve	`O	O	O grave
Oacute	'O	O	O acute
Ocircum	^O	O	O circumflex
Otilde	~O	O	O tilde
Oumlaut	"O	O	O umlaut
OE	OE	O	O E ligature
Ugrave	`U	U	U grave
Uacute	'U	U	U acute
Ucircum	^U	U	U circumflex
Utilde	~U	U	U tilde
Uumlaut	"U	U	U umlaut
Yumlaut	"Y	Y	Y umlaut
sharps	s		German small sharp s
agrave	`a	a	a grave
aacute	'a	a	a acute
acircum	^a	a	a circumflex
atilde	~a	a	a tilde
aumlaut	"a	a	a umlaut
aring	*a	a	a ring
ae	ae	a	a e ligature
ccedilla	,c	c	c cedilla
egrave	`e	e	e grave
eacute	'e	e	e acute
ecircum	^e	e	e circumflex
eumlaut	"e	e	e umlaut
igrave	`i	i	i grave
iacute	'i	i	i acute
icircum	^i	i	i circumflex
iumlaut	"i	i	i umlaut
ntilde	~n	n	n tilde
ograve	`o	o	o grave
oacute	'o	o	o acute
ocircum	^o	o	o circumflex
otilde	~o	o	o tilde
oumlaut	"o	o	o umlaut
oe	oe	o	o e ligature
oslash	/o	o	o slash
ugrave	`u	u	u grave
uacute	'u	u	u acute
ucircum	^u	u	u circumflex
uumlaut	"u	u	u umlaut
yumlaut	"y	y	y umlaut
pound	L-		British pound

Line Drawing Characters

If a terminal can support the drawing of lines on the screen via a special character set, KE Texpress can use this facility to display unbroken lines. If a terminal does not support line drawing facilities then no line drawing attributes need be specified. In this case KE Texpress will use a default set of characters for displaying lines and boxes.

The definition of each line drawing characters is as follows:

LINE*nesw* *value*

where *value* is the character sequence sent to the terminal to display the particular type of line. The value *nesw* is a four digit number where each of the digits may be a 0, 1 or 2. The digits in sequence represent:

n	Northerly line, either 0, 1 or 2 lines wide.
e	Easterly line, either 0, 1 or 2 lines wide.
s	Southerly line, either 0, 1 or 2 lines wide.
w	Westerly line, either 0, 1 or 2 lines wide.

Hence to represent the bottom right hand corner of a box which is one line wide the attribute name is LINE1001.

If a terminal supports either single or double lines (but not both) then the single and double line attributes should be defined the same.

The following table lists all valid line attribute names. A visual representation of the line is provided.

Attribute	Description	Visual
LINE1010	Single vertical line	|
LINE2020	Double vertical line	||
LINE0101	Single horizontal line	-
LINE0202	Double vertical line	=
LINE1111	Single line cross	-|-
LINE2222	Double line cross	=||=
LINE2121	Single horizontal, double vertical	-||-
LINE1212	Double horizontal, single vertical	=|=
LINE1011	Left single horizontal, single vertical	-|
LINE1012	Left double horizontal, single vertical	=|
LINE2021	Left single horizontal, double vertical	-||
LINE2022	Left double horizontal, double vertical	=||
LINE1110	Right single horizontal, double vertical	|-
LINE1210	Right double horizontal, double vertical	|=
LINE2120	Right single horizontal, double vertical	||-
LINE2220	Right double horizontal, double vertical	||=
LINE2220	Right double horizontal, double vertical	||=
LINE1101	Single horizontal, up single vertical	-'-
LINE2202	Double horizontal, up double vertical	="=
LINE1202	Double horizontal, up single vertical	='=
LINE2101	Single horizontal, up double vertical	-"-
LINE0111	Single horizontal, down single vertical	-,-
LINE0222	Double horizontal, down double vertical	=,,=
LINE0212	Double horizontal, down single vertical	=,=
LINE0121	Single horizontal, down double vertical	-,,-
LINE0021	Left single horizontal, down double vertical	-,,
LINE0012	Left double horizontal, down single vertical	=,
LINE0022	Left double horizontal, down double vertical	=,,
LINE0011	Left single horizontal, down single vertical	-,
LINE2002	Left double horizontal, up double vertical	="
LINE2001	Left single horizontal, up double vertical	-"
LINE1002	Left double horizontal, up single vertical	='
LINE1001	Left single horizontal, up single vertical	-'
LINE1100	Right single horizontal, up single vertical	'-
LINE2200	Right double horizontal, up double vertical	"=
LINE2100	Right single horizontal, up double vertical	"-
LINE1200	Right double horizontal, up single vertical	'=
LINE0220	Right double horizontal, down double vertical	,,=
LINE0210	Right double horizontal, down single vertical	,=
LINE0120	Right single horizontal, down double vertical	,,-
LINE0110	Right single horizontal, down single vertical	,-

Installing a Terminal Description

The simplest way to create a new terminal description is to copy and modify an existing terminal description.

For example, the ansi terminal description can be copied using the command:

```
texunterm  ansi  >   filename
```

The file, *filename*, can then be edited and the new terminal description defined. The TERM statement in the file must be changed to indicate the new terminal name. When producing a new terminal description it may be necessary to consult the actual terminal manual.

The new terminal description can be compiled and installed using the command:

```
texterm   filename
```

Error messages will be displayed if inconsistencies are detected in the file.

If several terminal types have the same terminal description the names of each terminal type may be separated by commas on the TERM line of the file. In this situation, texterm generates a dexcription for each of the listed terminal types.

Removing a Terminal Description

A terminal description can be removed by creating a description file consisting of solely the TERM line with the terminal name. The texterm program is then used as normal and determines that the terminal description is empty and removes the actual compiled description file.

A terminal description may only be removed by its creator.

Using an Alternative Terminal Description

KE Texpress (and many other Unix programs) use the TERM environment variable to determine the particular terminal type. If the use of a terminal description whose name differs from the TERM environment variable is required, a KE Texpress database system option can be utilised.

The option is called **termtype** and may be set in the standard manner of all KE Texpress database system options (refer to the KE Texpress User Guide or the KE Texpress Design Guide) using the sequence:

```
termtype=termname
```

This option can be useful if the terminal supports programmable function keys and the keys are set on a personal basis. Programmable function keys permit terminals of the same type to generate different function key character sequences.

Index

B

Bindings
- Command names, 2-4
- Default, 2-3
- Format, 2-3
- Function key defaults, 2-6
- Installing, 2-5
- keyboard, 2-2
- Other considerations, 2-6
- Removing, 2-5
- texbind, 2-5
- Two character commands, 2-4

C

Command names, 2-4
Cursor movement, 5-2

D

Database option
- termtype, 5-22

Default
- Bindings, 2-3
- Help files, 4-3
- Messages, 3-3

E

Environment variable
- LANG, 1-4
- TERM, 5-3, 5-22
- TLEVEL, 1-5

European characters, 5-2, 5-17

F

Function keys, 1-3, 5-2, 5-16

H

Help, 1-2
Help files, 1-2
- Default, 4-3
- Installing, 4-3
- Removing, 4-4
- Tailored, 4-2

I

Installing
- Help files, 4-3
- Keyboard bindings, 2-5
- Messages, 3-5
- Terminal description, 5-21

Interface tailoring, 1-2
Introduction, 1-2

K

Keyboard
- Function keys, 5-2

Keyboard bindings, 2-2
- Command names, 2-4
- Default, 2-3
- Format, 2-3
- Function key defaults, 2-6
- Installing, 2-5
- Other considerations, 2-6
- Removing, 2-5
- Two character commands, 2-4

L

LANG environment variable, 1-4
Language translation, 3-3
Languages, 1-4

Line drawing, 5-2
Line drawing characters, 5-19

M

Macro, 1-3
Messages, 1-3, 3-2
  Changing, 3-6
  Default, 3-3
  dflt, 3-3
  Format, 3-3
  Installing, 3-5
  Language translation, 3-3
  Printf format specifiers, 3-3
  Removing, 3-7

N

Names
  Command, 2-4

O

Operational Levels, 1-4

P

Printf format specifiers, 3-3

R

Removing
  Help files, 4-4
  Keyboard bindings, 2-5
  Messages, 3-7

T

TERM environment variable, 5-3, 5-22
termcap, 5-2
Terminal, 1-4
Terminal description, 5-2
  Character sequences, 5-8
  Characteristics, 5-9
  Cursor attributes, 5-9
  European characters, 5-17
  Function keys, 5-16
  Installing, 5-21
  Line drawing characters, 5-19
  Name, 5-9
  Programmable function keys, 5-22
  Removing, 5-21
  Screen mode attributes, 5-13
  Terminal identification, 5-3
  termtype, 5-22
  texterm, 5-21
  texunterm, 5-4
  Viewing, 5-4
Terminal identification, 5-3
terminfo, 5-2
termtype
  Database option, 5-22
texbind, 2-2, 2-5
texmessages, 3-5
texterm, 5-21
texunterm, 5-4
TLEVEL environment variable, 1-5
Two character commands, 2-4