CHAPTER 9 THE SCULPTOR MENU INTERPRETER

This chapter describes the Sculptor portable menu interpreter, **menu**.

Contents **Page**

Introduction

When designing applications, it is often necessary to provide a menu of programs from which the user can make a selection. The Sculptor menu interpreter program, **menu**, provides a fully portable method by which such menus can be created.

Menu files are standard text files with a **.m** extension and may be created with any text editor. When the menu program is run, the text file is read and interpreted and the menu is displayed on the screen. Menus may be nested, each menu being contained in a separate file.

Declarations exist to control many aspects of the presentation of the menu, including thin-line graphics and video attributes. On-line help text may also be presented either as a single line of help text at the bottom of the screen or as multiple pages of help text contained in a separate file. Five different menu styles are available.

Structure of a menu source file

A menu source file has four distinct sections.

1. The first line of the file is the title line and will be displayed, centred, above the menu. This line may be left blank if a title is not required for the menu.

2. Comments may be placed anywhere in the menu file, with the exception of a command line, and should begin with a full stop.

3. Declarations control many characteristics of the menu and begin with an exclamation mark "!". The available declarations are detailed fully beginning on page 9-5. Declarations must precede the option definition section.

Option definitions

4. The fourth section of a menu source file is the menu option definition section. Each option which is displayed on the menu must have a minimum of two lines and may have up to five lines. The structure of each line of a single option definition is shown on the next two pages. Up to 99 options may exist within a single menu.

Line 1: *number* [*fkey*], "text" [**at** *row,col*]

Where *number* is a sequential menu option number in the range 0-99 and *fkey* is a function key in the range **F1**-**F32** (applicable to style 5 only). The *text* will be displayed at a location automatically determined by the menu system or at the specified *row* and *column*, if given. If *number* **0** is not used, an **exit** option will be automatically generated when the menu is executed but an alternative option number 0 may be specified. The menu will only be executed if there is an **exit** command on one of the option definitions (unless **!noexit** is used). This protects a menu with no exit route from being accidentally run.

Line 2: [-]*command*

When the user selects this option the specified system *command* is executed as a child task.

The keywords **menu** and **exit** are recognised by the menu system and do not cause the creation of a child task. **menu** loads and interprets the named menu file as a sub-menu (if you have renamed the menu program itself be careful to keep the word **menu** in such commands to take advantage of this feature). **exit** exits the menu and returns to the previous menu or the operating system.

A percent sign "%" within the operating system command line will display the text of the following line as a prompt and will substitute the entered text at the position of the percent sign (see the **!noinput** declaration on page 9-7).

Up to 9 arguments may be passed to the menu program for substitution in command lines and these are referenced as $1 through $9 ($0 is the menu file name).

The percent "%" or dollar "$" characters may be passed on to the command being performed by typing them twice, ie %% will yield % in the command line and will not cause parameter substitution.

A minus (-) sign before the command executes the command without calling a shell or command processor (in this case, no redirection or wildcard substitution may be performed).

An example of a simple menu option is shown below:

```
1,"Check all Sculptor files" at 7,50
kfcheck -f *.k
```

When the task completes, the menu is redisplayed but, if the child task returns a non-zero termination code the menu program assumes an error message is being displayed and prompts the user to press RETURN before redisplaying the menu.

Next Line: "Prompt text" **[at** *row,col*]

If the command line contains a percent sign "%", this prompt text is displayed and a response is required. The response given replaces any percentage signs "%" in the command line above. This line is only required if parameter substitution, with %, is used in the operating system command line.

In **!style 5**, one of the following two help mechanisms are available. A single option definition cannot use both types.

Next Line: **!help** *filename*

Declaring a separate help file using **!help** will display that file in pages on the vdu if the user presses **F1** when the highlight bar is on this option. Help files are text files which may also contain attribute declarations.

Next Line: "Help text for this menu option"

The text inside quotes will be displayed if the user presses **?** (question mark) when the highlight bar is on this option. The **!autohelp** declaration will display this help text automatically.

Next Line: **!reset**

If extended width and/or depth is being used for this menu, **!reset** will cause the vdu to be set to standard depth and width for the duration of the operating system command. It will be restored to extended depth and/or width afterwards.

Command line syntax

The command line syntax for the menu interpreter is:

menu [**-v**] [*filename*] [*arg1*]...[*arg9*]

filename is a text file with a **.m** extension which defaults to **menu.m** if not specified.

Version information may be shown with the **-v** option.

EXAMPLE

```
menu
menu system 9 MAINTENANCE
menu apps DEBUG
```

Menu declarations

!autocr

Enables automatic carriage return within **!style 5** if the user presses the first letter of a menu option. This declaration is ignored if the menu is not using **!style 5**.

!autohelp

Enable automatic display of help text (**!style 5** only). The help text that is displayed is that defined within quotation marks for an option definition.

!blink

Messages printed with **!display** which follow will blink.

!box *depth, width* [**at** *row,col*]

Draws a box of *depth* lines and *width* characters at location *row,col*. The box is drawn using the thin-line graphics characters defined in the vdu parameter file.

!depth 24 | **25**

Sets the screen depth for this menu to either 24 (standard) or 25 (extended) lines.

!display "*message*" [**at** *row,col*]

Displays the text *message* at either the current cursor location or at location *row,col*.

!help *filename*

Used only within an option definition, *filename* will be displayed in pages on the screen if the help key is pressed. The filename must not be enclosed in quotes.

!line *length,style* [**at** *row,col*]

Draws a horizontal line of *length* characters in *style* 0, for a normal line or *style* 1, for a line with vertical bars at either end.

!standout

Any **!display** commands which follow will be in standout video as defined in the vdu parameter file.

!nodate

Disables display of the date. If this declaration is not present, the date is displayed near the top right hand corner of the menu.

!noexit

Allows execution of a menu without a valid **exit** command. If **!noexit** is not declared and the menu does NOT contain a valid **exit** command, the menu will not be executed.

The menu program will automatically create an **exit** option and statement when the menu is run if there is no option number 0 (zero) in the menu source file.

!noinput

Return to the menu without executing the command if nothing is input to a substitution prompt.

!normal

Any **!display** commands which follow will be in normal video.

!on *fkey* next menu [at *row,col*]

Apllicable to **!style 5** only. Defines a function key *fkey*, in the range **F2** to **F32**, to prompt for a menu file name at position *row,col*. The menu is called and returns to this menu on exit. This provides a way of running "hidden" programs for the expert user.

!prompt "*Text*" [at *row,col*]

Redefines the prompt text and allows display at *row,col*. The default prompt text is "Which option do you require ? "

!reset

Only used within an option definition. Causes the vdu to be reset to standard depth and width prior to the execution of the command.

!reverse

Any **!display** commands which follow will be in reverse video.

!style *mode*

Selects the presentation mode and type of response required for the menu. Modes 1 to 4 present the menu with either letters or numbers to the left of each option. The user then types a corresponding letter or number to select the desired option. Mode 5 presents the menu option prompt text as entered. The user then selects an option by typing the first character of that option or by using the cursor control keys to move a light bar from option to option. Mode 5 also enables the use of the **!autohelp** declaration and function key support. *mode* may be one of the following:

1	Selection by number 0-99, ENTER required
2	Selection by number 0-9, single key press
3	Selection by letter A-Z, ENTER required
4	Selection by letter A-Z, single key press
5	Light bar mode, use arrow keys or first letter to move and ENTER to select (but see **!autocr**)

!underscore

Any **!display** commands which follow will be underscored.

!width 132 | 80

Sets the screen width for this menu. Only available if your vdu supports extended width as defined in the vdu parameter file.

System variables

The following system variables are available for use in a source file.

$user	The user login name
$pwd	Present working directory
$date	System date
$task	Process identifier
$0 - $9	Command line arguments
$depth	Screen depth in lines
$width	Screen width in characters

Example menu

```
Demonstration Menu

. declarations follow

!width 132
!nodate
!autohelp
!style 5
!noinput
!box 15,70 at 3,5
!standout
!display "Path: $pwd " at 5,9
!normal
!prompt "Please select an option" at 18,45

. menu option definitions

0,"Exit this menu     " at 10,32
exit
"Return to previous menu or operating system"
1,"Accounting system " at 12,32
-sage acclogin %
"Please enter the company code " at 22,9
"Multi-company accounts system"

. end of demonstration menu
```

Error messages

Error messages are preceded by the message:

> **error in** *filename* **at line** *nn*

where *filename* is the menu file being read and *nn* is the line number where the error occurred. The error may be one of the following:

too many nested menus

Menus may be nested up to 12 levels deep. You may be calling a menu recursively, using **menu** *menuname* instead of **exit**.

no exit route

No valid exit command was found and the user will not be able to exit this menu. Over-ride with the **!noexit** option if exit is to be intentionally disallowed.

can't open *filename* **: (error** *nn***)**

The specified file could not be opened. The error number returned is the operating system error code. Consult your operating system manual for further details.

no title line

The first line of a menu file must be the title. Leave the first line blank if a title is not required.

option number expected

The menu definition must always start with a number in the range 0-99.

function key not in range 1-32

The option definition can only use function keys in the range F1-F32.

option not in range 0-99

There can be a maximum of 99 options in any single menu. Multiple menus should be used if the number of options exceeds 99.

no command line

There is no operating system command with this menu definition. The second line of an option definition *must* be a command line.

no prompt line

A substitution character (%) was found in the command line, but no prompt text is defined.

syntax error in *display|box|style|prompt* **statement**

See the appropriate command for the correct syntax.

bad prompt style

The prompt style must be in the range 1-5 (see **!style** for details).

unknown command

The command is unknown or spelt incorrectly.

'next' missing in 'on' command

Check the syntax of the **!on** command.

function key out of range in 'on' command

Function key range in **!on** is 2-32.

error in 'on' command

Unknown error in **!on** - check the syntax.

terminator missing

Quote characters (") are unmatched.

CHAPTER 10 THE SCULPTOR UTILITY PROGRAMS

This chapter describes the various utility programs that are supplied with the Sculptor Development System.

Contents **Page**

Introduction

The utilities described in this chapter provide support for the Sculptor development system. The programs fall into three major categories; the keyed file utilities, the parameter file utilities and miscellaneous utilities.

The keyed file utilities are:

kfcheck	Checks the integrity of Sculptor keyed files.
kfcopy	Copies and optionally merges one keyed file to another.
kfdet	Prints information about a keyed file.
kfri	Rebuilds the index of a keyed file.
newkf	Creates new Sculptor keyed files from a data dictionary file.
reformat	Preserves data in a keyed file after changes to the record layout.

The parameter file utilities are:

decprint	Decodes an existing printer parameter file so that it may be edited.
decvdu	Decodes an existing vdu parameter file so that it may be edited.
setprint	Creates a printer parameter file.
setvdu	Creates a vdu parameter file.

The miscellaneous utilities are:

kprnt	Help screen interpreter.
lcf	Allows messages and prompts produced by the Sculptor system to be language configured.
pause	Displays a prompt and awaits a key press.
pdes	Prints a data dictionary in a variety of formats.
sageform	Prints screen form displays.
vno	Prints information about the revision of Sculptor used to compile a program.

These utilities are described in detail in the following sections.

Introduction

Printer parameter files are stored in a binary format that allows them to be quickly and easily read by the **sagerep** program at run-time. The printer parameter file holds printer specific information so that the report/batch language system need not have that information "hard-coded" into the program.

The **decprint** utility reads the printer parameter file and outputs that file in a "human-readable" form with each entry on a separate line and each line having a heading to identify it. This output may be redirected into a file (or output directly to a file using the **-f** command line option) and edited. The resulting file, including headings, may then be used as input to the **setprint** program, described on page 10-39.

Command line syntax

decprint [**-chp**][**-f**[=*outputfile*] *filename*

Decode the printer parameter file *filename*. Output from **decprint** may be redirected to a file of your choice for editing. Alternatively, the **-f** option may be used to send output directly to *filename*.**s** or *outputfile* if specified.

The **-c** option is used to decode and convert printer parameter files from version 1.16 of Sculptor. The **-c** option must also be used with **setprint** to complete the conversion.

The **-p** option will read and decode the file from the path specified by the **SCULPTOR** environment variable or, if this is not defined, the file from the default Sculptor printer file directory. If **-p** is combined with **-f**, the output file will be created in the same directory as the encoded printer parameter file.

The **-h** option disables the automatic conversion to mnemonic characters and causes all codes to be output using hexadecimal notation.

EXAMPLE

```
decprint printer >printer.out
decprint -h printer >printer.s
decprint -hpf printer
decprint -pf=newprint.txt printer
```

Error messages

Can't open output file

The output file could not be opened. Check disk space and permissions.

filename : File is not version 2.0 or later

The printer parameter file must be converted to operate with this version of Sculptor. Use the **-c** option to decode the file. Remember to also use the **-c** switch with **setprint** to complete the conversion.

filename : File is a VDU parameter file

The file is not a printer parameter file. Use **decvdu** to decode it.

filename : bad file

The named file is not in the correct format for a printer parameter file. Check that you are attempting to decode an encoded file. The file may also be corrupt.

Error in arguments

Check the command line syntax above.

Too many file names

Only one file name may be used as an argument to **decprint**.

Path too long

The path length exceeds the program's limits.

Input and output files are identical

The input and output files must have different names.

Introduction

Vdu parameter files are stored in a binary format that allows them to be quickly and easily read by the **sage** program at run-time. The vdu parameter file holds vdu specific information so that the screen form language system need not have that information "hard-coded" into the program.

The **decvdu** utility reads the vdu parameter file and outputs that file in a "human-readable" form with each entry on a separate line and each line having a heading to identify it. This output may be redirected into a file (or output directly to a file using the **-f** command line option) and edited. The resulting file, including headings, may then be used as input to the **setvdu** program, described on page 10-45.

Command line syntax

decvdu [**-chp**][**-f**[=*outputfile*] *filename*

Decode the vdu parameter file *filename*. Output from **decvdu** may be redirected to a file of your choice for editing. Altenatively, the **-f** option may be used to send output directly to *filename***.s** or to *outputfile* if specified.

The **-c** option is used when decoding a version 1.16 or earlier version of the vdu parameter file which is to be converted to the latest format. The **-c** option must also be used on **setvdu** to complete the conversion.

The **-p** option decodes the file found in the path defined in the **SCULPTOR** environment variable, or, if this is not defined, the default Sculptor vdu parameter file directory. If **-p** is combined with **-f**, the output file will be created in the same directory as the encoded vdu parameter file.

The **-h** option disables the automatic conversion to mnemonic characters and causes all codes to be output using hexadecimal notation.

EXAMPLE

```
decvdu wyse60 >wyse60.out
decvdu -h qvt119 >qvt119.s
decvdu -hpf ibmpc
decpvdu -pf=stdvdu.txt ibmpcc
```

Error messages

Can't open output file

The file named could not be opened for writing.

Can't open *filename* (system error *nn*)

The file *filename* could not be opened due to an operating system error identified as *nn*. Check your operating system manual for further details. Frequent causes are: the disk is full, no permission to write, etc.

filename is not version 2.0 or later

Use the **-c** option to convert the vdu parameter file from an older version.

filename is a printer parameter file

Use **decprint** to decode printer parameter files.

filename is a bad file

The named file is not in the correct format for a vdu parameter file. The file may not be an encoded parameter file. The file may be corrupt.

Error in arguments

Check the command line syntax above.

Too many file names

decvdu may be called with only one file name.

Path too long

The path length exceeds the program's limits.

Input and output files are identical

The input and output files must have different names.

Introduction

The Sculptor keyed file system is very robust and has been thoroughly tested over several years. The multi-level tree index can, however, be corrupted if an update routine is interrupted by power or hardware failure or by uncontrolled task termination when the system is incorrectly shut down. The update routines themselves ignore all normal keyboard interrupts.

The **kfcheck** program verifies a keyed file index by checking that all record pointers are unique and that missing pointers belong to deleted records. An additional option allows the key value recorded in the data file to be compared with the key value stored in the index file.

kfcheck should be run every time the system is switched on or, if the system is permanently on, it should be run once each day. It should also be run immediately after a system crash. The **kfri** program may be used to rebuild a damaged index (see page 10-15 for details).

Command line syntax

kfcheck [**-dfs**] *filename* [*filename*] ...

Checks the integrity of the Sculptor keyed file *filename* (the **.k** extension is added automatically).

The number of deleted records may be displayed with the **-d** switch. Sculptor automatically re-uses deleted record space, so this number will decrease to zero as insertions are made.

By default, **kfcheck** only checks the index file record pointers and the data file free chain for integrity. Use the **-f** switch to force **kfcheck** to also read the data file key values and compare them with the index key values. This provides an additional level of security, but takes longer.

For use within automated routines, the **-s** switch causes **kfcheck** to stop on error and exit with a value which defines the error found. The error codes which may be returned are:

0	no error found
1	file is damaged
nn	operating system error number *nn* encountered

If more than one filename (or a wildcard, eg *.k) is used, the error code returned will refer only to the last file processed.

One of the following messages will be shown when **kfcheck** is run:

> Checking *filename* - Okay (*n* records)
> Checking *filename* - Okay (*n* records + *n* deleted)
> Checking *filename* - DAMAGED

If damage is reported either recover an undamaged copy from your backup source or run the **kfri** program to rebuild the index.

EXAMPLE

```
kfcheck -f transact
kfcheck -fs *.k
```

Error messages

Checking *filename* - DAMAGED

The named index file does not accurately relate to the data file. Rebuild the index file from the data file using **kfri**. If the data file itself is damaged, it must be restored from a backup source.

Error *nn* reading *filename*

An operating system error occurred. The number given is the error code returned by the operating system. Check your operating system manual for further details.

Key data wrong in record

The **-f** switch was specified and the index file and data file key data does not compare exactly. Use **kfri** to rebuild the index.

Illegal option -*x*

The option -*x* is not a valid option for this command. Check the command line syntax above.

Filename too long

The filename exceeds the program's limits for file and path names.

Introduction

Copies the data in a keyed file to another keyed file with an identical record layout and field names. The new keyed file may be created empty or the records copied may be added.

This command is frequently used to archive records to a file in another directory or on another file system and also to reduce the operating system storage used when a file has undergone a substantial and permanent reduction in the number of records stored.

On the OS9 operating system, a file which is repeatedly extended will eventually fill up its sector map and cause a disk write error. This will damage the file. If an existing file is becoming full, rename it temporarily and use **newkf** to create a new, pre-extended, file of sufficient size. The existing data may then be copied into the new file with **kfcopy** (using the **-e** option).

Command line syntax

kfcopy [**-ec**] *oldfile newfile*

Reads keyed file *oldfile* and creates a new keyed file *newfile* containing all records which have not been deleted.

newfile may be extended with the **-e** switch. In this case *newfile* must exist already and may contain records. All existing records in *newfile* are preserved and all valid records from *oldfile* are inserted into *newfile*. If a key value from *oldfile* is already present in *newfile*, the record will not be inserted. The number of duplicate keys which have been ignored in this way will be reported at the end of the **kfcopy** operation. This switch may be used to archive information to another file.

The **-c** switch displays a count every 100 records copied.

If *newfile* already exists, its record layout must be identical to that of *oldfile* in terms of field type, size and position within the record.

EXAMPLE

```
kfcopy -c oldstat newstat
kfcopy -ec monthfile yearfile
```

Error messages

Illegal option *-x*

The switch *-x* is not a valid switch for this command. Check the command syntax.

INTERRUPT - copy incomplete

The **kfcopy** process was interrupted and *newfile* is not complete.

Can't open *filename* (operating system error number *nn*)

The file named could not be opened due to an operating system error number *nn*. Check your operating system manual for further details. Frequent causes are no permission or file not found.

Introduction

This utility displays information about the key length, record length and number of index levels of a Sculptor keyed file.

The number of index levels specifies the "depth" of the index tree and determines the maximum number of logical disk reads that will be required to recover a record on the file. The number of index levels will vary depending on the size of the key, the number of records in the file and the order of insertion of the records..

Command line syntax

kfdet *filename* [*filename* ...]

Where *filename* is a Sculptor keyed file (the .k extension is automatically added).

EXAMPLE

```
kfdet sales
kfdet stock1 stock2 stock3 > stockdat.txt
```

Error messages

File is damaged

The header record of the index file specified did not contain valid data. Use **kfcheck** to check the integrity of the file and then **kfri** to rebuild it if damage is reported.

Filename too long

The filename exceeds the system limits for file and path name length.

System error *nn* **reading** *filename*

An operating system error number *nn* occurred when reading the index file. Check your operating system manual for further details. Frequent causes are disk full or no permission.

Introduction

If a keyed file is found to be damaged or if its index file has been lost then, as long as the data file is still intact, **kfri** will build a new index for that file. It can do this because Sculptor stores the key data at the front of each data record.

To successfully rebuild the index of a keyed data file, **kfri** must know the key and record length of the file. Provided that the old index file is present, **kfri** always display key and record length values for confirmation. These should be checked carefully, by examining the data dictionary for the file if there is the slightest doubt about their accuracy. If the old index file is not present, **kfri** prompts you to enter the key and record length of the file to be rebuilt. The **pdes** utility may be used to examine the file's data dictionary to recover the correct values.

Having entered or confirmed the key and record lengths, the index rebuild will take place. Each record in the data file will be read and a new index entry created for that record. Duplicate key values will be ignored and usually indicate some corruption of your data file. On completion of the rebuild, the number of keys inserted into the index is displayed. This is the number of records correctly read from your data file. The number of duplicate key values is also displayed.

kfri detects the deleted records in the data file by assuming that all records which are completely null except for the first three bytes are deleted; the first three bytes are pointers in the deleted record chain. It is possible, therefore, that a file with a very small record size and no alphanumeric data after the first three bytes could have some genuine records taken as deleted. In practice there are very few cases where **kfri** cannot perfectly rebuild the index.

In the case of an index only file, the index file is renamed first, a new index is built from it by treating it as a special data file and then the renamed file is deleted.

Command line syntax

kfri [**-csp**] *filename*

Rebuilds the index file *filename*.**k** from the data file *filename* or from the index file *filename*.**k** if the file is an index only file.

The **-c** switch displays a count every 100 records rebuilt.

The **-s** switch forces operation in silent mode. No messages will be issued and Y will be automatically answered to the confirmation prompt.

The **-p** switch controls the way that **kfri** packs the index. Records are physically stored on the disk in the order that they were inserted. In this case the order is usually fairly random in relation to the key, and **kfri** creates an index which is evenly packed. If all of the records, however, were inserted in key order sequence, the index file created would be no more than half full, although it's efficiency would only be marginally affected.

If you are certain that the records were inserted in key order, the **-p** switch will instruct **kfri** to pack index entries to the leftmost index page first, and will result in a smaller index file. If the **-p** switch is used and the data is not in key order sequence, the resulting index file may, however, be larger than it would be without the **-p** switch. Exercise caution in your use of this switch.

EXAMPLE

```
kfri -c stock
kfri sales
```

Error messages

Record length not divisor of file length

The record length shown does not divide evenly into the data file length. The record length may be incorrect or the data file may be badly corrupted.

Warning: data file may be original index file

The **kfri** process was interrupted during the index rebuild of an *index only* file and the data file may actually be the previous index file which is to be rebuilt. Ensure you have a backup of the data and index file, rename the data file (without the extension) as the index file (.k extension) and re-try the index rebuild.

Filename too long

The filename given exceeds the operating system limits for file name length.

Can't create *filename* (system error *nn*)
Can't delete *filename* (system error *nn*)
Can't rename *oldfile* as *newfile* (system error *nn*)

The index file could not be created, deleted or renamed due to an operating system error identified as *nn* . Check your operating system manual for further details. Frequent causes are disk full, no permission or file is read-only.

Cannot build a good index. Data file is badly damaged.

The data file is severely damaged and some data recovery is required before the index file may be rebuilt. Refer to the data recovery section on the next page before attempting another index file rebuild.

Introduction

This program may be used to create help and information screens. It reads and displays an ordinary text file, interpreting certain special characters to give a fully portable presentation that includes graphics and video attributes.

The file may be created with a standard text editor. Care should be taken to avoid certain characters, such as TAB, which may be non-portable or confuse the display. The file can contain text to be displayed and a number of characters that have a special meaning to **kprnt**. These characters are interpreted to produce lines, boxes and video attributes on the final display.

Command line syntax

kprnt *filename*

Where *filename* is the text file containing the help or information to be displayed.

Graphics boxes

To ensure portability across a wide range of display systems and terminals, thin-line graphic boxes may be defined within an information file using dollar signs "$" to mark the four corners of the box. Horizontal and vertical lines may also be drawn within this box by marking the intersection points with "@". The example below shows the text on the left and the resulting output on the right.

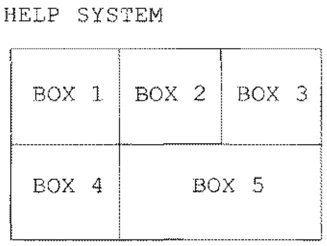
```
HELP SYSTEM
$        @        @        $

 BOX 1    BOX 2    BOX 3

@        @        @        @

 BOX 4       BOX 5

$        @                 $
```



Video attributes

Text may be highlighted with a set of video attributes. There are two versions of each command. The first set cause a change in the text being displayed but nothing else. The second prints two spaces after the command. This is useful when the text is inside a box and the two spaces are needed to keep the graphics aligned. All commands are prefixed by a \ (backslash character).

Video Effect	Command with packing	Command without packing
Normal	\n	\N
Standout	\s	\S
Reverse	\r	\R
Underscore	\u	\U
Blinking	\b	\B

Other commands

A number of other special characters may be included in the information file. These characters are shown below.

Effect	Command
Stop and wait for a key press	\?
Switch between characters sets	
as specified in the vdu parameter file	\!
The \ character	\\
The ASCII code represented by *n*	\n
The ASCII code without packing	\%*n*
Align subsequent text and graphics	*

The * command accumulates spaces that may have been lost by previously used commands (on that line) and prints them, thus re-aligning any subsequent text.

Since the dollar ($) sign is used to delimit thin-line graphic boxes it may not be directly used as a displayable character. Use the notation **\36** wherever you need to display a dollar sign.

Error messages

Can't open source file

The named file could not be opened. Check the file exists and that you have permission to read it.

Too many boxes

The maximum limit of 100 display boxes has been exceeded.

Top right '$' not found
Lower left '$' not found
Lower right '$' not found

The '**$**' character is used to define the bounds of a thin line graphic box on the screen. Each box should have four corners. One of these messages will arise if an initial '**$**' is found and the corresponding corners could not be found.

Line exceeds 142 characters

The maximum line length is 142 characters. Check that no single line in your file exceeds this limit.

Introduction

Many programs in the Sculptor system are language and date format configurable, the program load modules being designed to allow the translation of text messages into another spoken language. The **lcf** utility is provided for this purpose.

lcf scans the program load module and displays each translatable piece of text in turn. Beneath the text it indicates the maximum acceptable length with a row of dots and waits for new text to be typed. If RETURN alone is pressed, the existing text remains unaltered, otherwise the entered text is stored in the program load module.

Wherever the text **y/n** occurs it must be translated such that the characters which correspond to **y** and **n** are in exactly the same place in the text.

Wherever the text **%d** occurs in a message, the **%d** will be used to display a number. The translated message must also contain one, and only one, **%d**.

Some load modules have configurable date formats. The characters **d**, **m** and **y** are used to indicate day, month and year respectively. You may alter their order, the separator character and the number of digits in the year. If the date format is changed in one program it should be changed in the same way in all other programs.

In **sage** and **sagerep**, the single characters "." and "," are presented for alteration. These characters are used to define the decimal point and digit grouping characters, respectively. Changing them alters the format of input and output data values but you must still use the English notation ("." for decimal point and "," for digit grouping) when defining formats in your programs and in **describe**.

In sagerep, the single character **x** is presented for alteration. This does nothing when set to **x**, but if set to **r** will append a carriage return to the very end of the report. This is required for some print spoolers.

After modifying a load module on the OS9 operating system, use **verify** to reset the CRC (see the OS9 system manual)

Command line syntax

lcf *filename*

Amend the language configurable information contained within the executable file *filename*. The named file must be in the current directory.

Configurable programs

The following programs have configurable entries:

cf	**cr**	**describe**	**fql**	**kfcheck**	**kfcopy**
kfdet	**kfri**	**menu**	**newkf**	**pause**	**pdes**
reformat	**rg**	**sage**	**sageform**	**sagerep**	**sg**
spp	**sql**				

Configurable text

In the following pages, the configurable information is described using the following notation:

filename

```
Default text/format
.............................. (nn)
```

Filename is the executable file being configured. The current text for this field is shown with the dots below representing the field size available for input. The field size is shown in brackets to the right. Additional information may be shown directly beneath an entry.

cf

```
dd/mm/yy
.......... (10)
```

The default format used to interpret date constants during compilation.

cr

```
dd/mm/yy
.......... (10)
```

The default format used to interpret date constants during compilation.

describe

```
dd/mm/yyyy
.......... (10)
```

The format used when expanding date validation entries.

fql

```
dd/mm/yy
.......... (10)
```

Date format for input.

kfcheck

```
Checking
.................... (20)
Okay
................ (16)
DAMAGED
................... (19)
record
.................. (18)
records
................... (19)
deleted
................... (19)
Error %d reading
.............................. (30)
```

Please note: The %d in this text is used by the program to display the error number. The message must contain one, and only one, %d.

```
Key data wrong in record
........................................... (43)
```

kfcopy

```
records copied
............................ (28)
duplicate keys ignored
............................................. (45)
Record length not divisor of file length. Continue
............................................................................. (77)
WARNING: Data file may be original index file
....................................................................... (71)
```

menu

```
Finish
.............................. (30)
```

The text used when generating automatic program exit statements

```
Which option do you require?
........................................... (43)
Exit code %d. Press return to continue:
...................................................... (54)
```

The statement must contain one, and only one, %d part. The %d is used to display the error number generated.

```
Menu Name?
...................................................... (54)
```

newkf

```
created
..................... (21)
```

pause

```
Strike a key when ready . . .
............................................ (44)
```

pdes

```
dd/mm/yyyy
.......... (10)
```

Sets the date format used when displaying date validation information.

reformat

```
dd/mm/yy
.......... (10)
```

```
.
. (1)
```

Decimal point character

```
Key length error
.............................. (30)
Record length error
................................. (33)
Reformat in progress
..................................... (37)
records reformatted
................................. (33)
Insert field
.......................... (26)
Delete field
.......................... (26)
Modify field
.......................... (26)
Okay
.................. (18)
y/n
... (3)
```

NOTE: Any characters replacing the **y** or **n** should appear in the same position as those shown.

```
Duplicate key:
............................ (28)
INTERRUPT - reformat incomplete!
........................................................ (56)
```

rg

```
Page:
................ (16)
Date:
................ (16)
END OF REPORT
........................... (27)
```

sage

```
dd/mm/yy
.......... (10)
.
. (1)
```

Decimal point character

```
,
. (1)
```

Digit grouping character. Typically used to separate thousands.

```
[
. (1)
```

Default left field delimiter character

```
]
. (1)
```

Default right field delimiter character

```
Which option do you require?
........................................... (43)
No such record
............................ (28)
Can't open file
............................ (28)
Read error on input (get)
............................ (28)
Write error on output (put)
............................ (28)
Record already exists
................................... (35)
Waiting ...
........................ (24)
No record selected
................................ (32)
 (y/n)
........ (8)
```

The characters replacing the y or n in this text must occupy exactly the same position.

```
return with no gosub!
................................... (35)
Illegal arithmetic!
................................. (33)
Stack overflow!
............................. (29)
Attempt to access an inactive screen
...................................... (38)
<INS>
............. (13)
<OVT>
............. (13)
Range:
................................. (33)
```

sageform

```
[
. (1)
]
. (1)
Which option do you require?
........................................... (43)
```

sagerep

```
dd/mm/yy
.......... (10)
.
. (1)
```

Decimal point character

```
,
. (1)
```

Digit grouping character. Typically used to separate thousands.

```
return with no gosub!
................................... (35)
Illegal arithmetic!
.................................. (34)
Stack overflow!
.............................. (30)
Bad input data
............................. (29)
x
. (1)
```

If set to **r**, appends a carriage return character to the *very end* of the report. Sometimes necessary with some spoolers to force the last few characters out if these are control codes to reset the printer.

```
Can't open file
............................ (28)
Read error on input (get)
............................ (28)
Write error on output (put)
............................ (28)
```

sg

```
FILE MAINTENANCE
.............................. (30)
Today's date
.......................... (26)
Use BACKSPACE to finish inserting
.......................................................... (56)
```

```
Already recorded
.............................. (30)
No further matching records
................................................ (48)
All correct
......................... (25)
Record amended
............................ (28)
Are you sure
.............................. (30)
```

spp

```
Suppress zeroes in blank numeric fields.
Current setting is N. (answer Y/N)
```

The entry for **spp** is not strictly language configuration. If this entry is set to Y, zeros will be displayed as *spaces* in all screen and report language programs subsequently compiled with **spp**.

sql

```
dd/mm/yy
.......... (10)
```

The format used when interpreting date constants in **sql** queries.

Error messages

filename is not language configurable

The named file has no language configurable text.

Too long!

The entry exceeds the number of characters available. Re-enter the text for the entry or press **RETURN** to leave the entry unchanged.

System error *nn* reading *filename*

A system error identified as *nn* has occurred reading the named file. Check your operating system manual for further details.

Seek error!

lcf could not find the next section of the file. Check the integrity of your filing system.

Argument error

Check the syntax shown above.

Introduction

Once a file layout has been defined with the **describe** program, the data and index files must be created with the **newkf** utility.

A Sculptor data file has the same name as the data dictionary file but without the **.d** extension. An index file also has the same name but with a **.k** extension.

Command line syntax

newkf [**-i**] [**-r** *recno*] *filename* ...

Creates a data file from a data dictionary file which is *filename* without an extension, and an index file which is *filename*.**k**.

If the file has only key fields, the **-i** option will prevent creation of the data file and will create an index only (**.k**) file. This allows the creation of alternative indexes to another file, and minimises duplication of index storage. This option is ignored if the file has data fields.

The **-r** option is used on systems which will benefit from pre-extended files. The data and index files will be pre-extended to *recno* records. On the OS9 operating system a file which is repeatedly extended may fill up its sector map and cause a disk write error which can damage the file. For this reason, it is strongly recommended that files created on OS9 are pre-extended to the maximum number of records that they are likely to hold. This may take some extra time when the data file is created, but does not significantly affect performance when the file is accessed.

CAUTION: **newkf** will create *empty* index and data files. **Do not use if you wish to preserve any existing data**. See **reformat** on page 10-34 for details.

EXAMPLE

```
newkf control
newkf -r5000 customers
newkf -i nameidx codeidx costidx
```

-r option not available in restricted version

Pre-extension of data files is only available in the full version of **newkf**, not in the evaluation version.

Illegal option

Check the command line syntax again.

Filename too long

The filename exceeds the program's limit for file and/or path names.

Can't create *filename* (key length zero)

Sculptor data files must have a minimum of one byte for the key. Check the data descriptor file with **describe** and try again.

Can't create *filename* (key length > 195)

The maximum key length is 195 bytes. Use **describe** to amend the descriptor file accordingly and try again.

Can't create *filename* (record length <3)

The minimum data file length (including key) is three bytes. Use **describe** to amend the descriptor file accordingly and try again.

Can't create *filename* (system error *nn*)

The system error identified as *nn* has occurred. Refer to your operating system manual for further information. Frequent causes are disk full, no permission or file is read-only.

Introduction

Similar to the pause command supported by the DOS command processor. Supplied with some Sculptor systems to ensure program portability between DOS and other operating systems. The **pause** utility displays the message "Strike a key when ready" on the screen and awaits a key press from the user.

This utility is useful when an operating system command is being performed from a menu which provides information to the user. The pause utility can be placed on the command line of the menu to allow the user time to see the information.

Command line syntax

```
pause
```

Error messages

No error messages are produced.

Introduction

The **pdes** utility allows the contents of a data dictionary to be listed in a variety of formats. The compressed format is useful for summary information about the file and contains no field comments or on-line help text. The normal format is useful for most purposes, including documentation, while the long format lists all information contained in the data dictionary and displays up to five lines for each field.

The file title and comments, together with the record length of the file always appear at the top of the listing.

For long files, the output from **pdes** may be paged, with a keypress required between each page.

Command line syntax

pdes [**-lmc**] *filename* [*filename* ...]

Prints the file structure held in the data dictionary file *filename*.**d** in normal format, compressed format (**-c**) or long format (**-l**). Output may be continuous or paged (**-m**). The **.d** extension is added automatically.

EXAMPLE

```
pdes -c *.d >file.lst
pdes -l stock
pdes -m tr*.d
```

Error messages

No Files!

Please enter at least one file name or a wildcard (eg *.d).

Command line syntax error

Please check the syntax shown above. Valid options are **l**, **m** and **c**.

Introduction

When a keyed file is created with the **newkf** utility program, the key and record length is fixed and the record layout set for that file. If the record layout is subsequently changed, the **newkf** program may again be used to create a new keyed file which incorporates the changes. This, however, will destroy any existing data in the file.

If data contained in the file is to be preserved, the **reformat** utility may be used to copy the data from the old file to a new file, applying any changes to the record structure as it does so.

reformat examines both data dictionary files and assumes that only fields with the same name are to be retained. If their type and/or size differs, then a suitable conversion will take place. Fields which exist in the old file but not in the new are deleted. New fields or array elements are initialised spaces or zero according to the field type.

Existing data in the destination file may be retained if required and the data from the old file will be merged with that file. Duplicate key values from the old file are discarded and are not inserted into the destination file.

After using **reformat**, all programs which access the reformatted file must be recompiled.

When converting r8 fields to m4 or m8 fields, **reformat** multiplies by 100, and when converting m4 or m8 fields to r8, it divides by 100.

Command line syntax

reformat [**-ce**] *oldfile newfile*

The structure of the file *oldfile* is read from the data dictionary (.d) file and compared to *newfile.* Any differences are displayed and you are prompted to confirm these changes. The data in *oldfile* is copied into fields with the same name in *newfile.*

The **-c** switch will display a count every 100 records reformatted.

The **-e** switch will allow extension of the file *newfile* from *oldfile. newfile* must exist and any records within it will be retained.

Example

```
reformat temp stock
reformat -c tempdat sale_list
```

Procedure

1. Backup all files.
2. Rename the existing data file, it's index (.k) file and the data dictionary (.d) file to another name, eg OLDFILE, OLDFILE.K, OLDFILE.D.
3. Copy the data dictionary file OLDFILE.D to the original name, so that you may edit the original file using **describe** to make any layout changes required. Do NOT alter the names of any fields which are to be preserved.
4. Having made the changes required, and after making sure you have a backup copy of your original data file, use reformat as follows:

```
reformat OLDFILE NEWFILE
```

The changes you have made are listed and you are asked to confirm that you wish to proceed. The data is then read from OLDFILE and written to NEWFILE. A summary is shown below:

```
copy stock.d temp.d
rename stock temp
rename stock.k temp.k
describe stock
reformat temp stock
delete temp temp.k temp.d
```

Error messages

Filename too long

The named file exceeds the system limits for file/path name length.

Illegal option *-x*

Check the syntax shown above.

Key length error
Record length error

The key and/or record length of the original data dictionary file is incorrect when compared to the actual data file. Please ensure that the original data dictionary file is used when reformatting.

INTERRUPT - reformat incomplete

The reformat process was interrupted and the new file is not complete

Duplicate key

If a duplicate key is encountered, **reformat** will print this message, but continue. It serves as a warning that an insert into the new file failed because the key already existed, and is normally only encountered when using the **-e** switch. If this error is encountered when the **-e** switch has NOT been used, it indicates that the new index fields are not unique. If you have not changed the index, it indicates that the old file is damaged and should be rebuilt using **kfri**.

Can't open *filename* **(operating system error** *nn***)**

The named file could not be opened. Check disk space and access permissions.

Introduction

The **sageform** utility provides a portable method of producing a printed copy of each screen form for documentation and reference purposes.

Multiple screen forms within a single program may be printed.

Command line syntax

sageform [**-s**=*screen*] [**-w**=*width*] [**-d**=*depth*] *filename*

Produces a printed copy of a Sculptor screen form language screen layout by reading the compiled (**.g**) intermediate code file *filename*.**g** and re-creating the screen line by line to standard output.

Any multiple screens used in the specified program may be printed using the **-s**=*screen* option where *screen* is the number of the screen to print in the range 1 to 8. The default is screen number one.

The screen form may be printed in any *width* or *depth*, over-riding any **!width** or **!depth** declarations used within the screen form program.

EXAMPLES

```
sageform stdform
sageform -s=3 arrival
sageform -s=8 arrival
```

Error messages

Error in arguments

Check the syntax shown above.

Too many file names

Only one file may be printed at a time with **sageform**.

Program out of date. ***filename*.f must be recompiled.**

filename.**g** is not a current version and must be recompiled from *filename*.**f**, using Sculptor version 2.0 or greater, before this version of **sageform** can read it.

Introduction

Printer parameter files are stored in a binary format that allows them to be quickly and easily read by the **sagerep** program at run-time. The printer parameter file holds printer specific information so that the report/batch language system need not have that information "hard-coded" into the program.

The **setprint** utility reads either standard input or a text version of the printer parameter file created with **decprint** and creates a printer parameter file. This printer parameter file may be subsequently decoded using the **decprint** utility (see page 10-3).

If standard input is used, and input is not redirected, a series of questions about the printer is displayed and a response is awaited. The reference manual for the printer will be needed to answer these questions (see page 10-41 for a complete list of the printer parameter file entries). A more common method of creating a printer parameter file, however, is to use **decprint** to decode an existing file, edit that file using a text editor and then create a new printer parameter file from that edited text file.

Printer parameter files are less critical then vdu parameter files. The p80 and p132 parameter file, for instance, are suitable for use with almost any printer, but if you wish to use features such as double-width underlining, etc, then a special parameter file must be created. In order to make the best use of of different paper sizes, several parameter files may be required for each type of printer being used.

If you wish to use a printer feature which Sculptor does not specifically allow for, you may use one of the user define sequences.

When using the **sagerep** interpreter, a printer parameter file may be specified on the command line. The default parameter file used is called *printer*. The parameter file *pvdu* may be used to output to the vdu and should contain the codes necessary for bold, underline, etc on the vdu, if available..

Command line syntax

setprint [**-cp**][**-f**[=*inputfile*] *filename*

Input to **setprint** may be redirected from a file created by **decprint** or input may be read directly from the file *filename*.**s** or from *inputfile* if the **-f** option is used.

The **-c** option must be used if the incoming text file was decoded using the **-c** option with **decprint** (this is used to decode a Sculptor version 1.16 printer parameter file).

The **-p** option will create the parameter file in the path specified by the **SCULPTOR** environment variable or, if this is not defined, in the default Sculptor printer file directory.

Printer parameter files

The available printer parameter file entries are listed with a description of where each entry is used.

Entry format

The codes issued by each parameter may be entered in a variety of formats, any of which may be combined. These are the available formats:

#*nnn*	*nnn* is a decimal number, eg #90, #108, #9
$*xx*	*xx* is a two character hexadecimal code, eg $0d, $ff, $09
^*x*	^*x* is a control code, eg ^L, ^M, ^G, ^A
ESC	an escape character (#27, $1b)
SPC	a space character (#32, $20)
NULL	an ASCII zero character (#0, $00)
PAD *n*	a string of *n* **NULL** characters. *n* can be in the range 0-127
\$, \#, \^	send the $, # or ^ character
DEL	send the **DEL** character, normally $7f, #127

A sequence of codes for a single entry must be separated with commas. For example

```
ESC,m,#7,#1,m
#27,$8b,L
```

Printer entries

printer identification line

Used for information only to identify the printer this parameter file is intended for.

Top of form =

The code to position the printer carriage at the top of the next form. Usually the form feed code (#12, $0c), if this entry is left blank, **sagerep** will output line feeds to position the page. Sent directly by the **#tf** print item command within a **print** or **printh** statement.

Start underline =
End underline =

Enter the codes to start and end underlining. **NOTE**: some printers use two different codes for underlining, one which is permanent and another which is automatically turned off at the end of the line. Use the permanent code if available. Sent by the **#su** (start underline) and **#eu** (end underline) print item commands respectively.

Print double width =

If available, enter the code to select double width characters. Sent by the **#dw** print item command.

Print single width =

If a code has been entered for double width characters, enter the code which resets the character size to normal. Sent by the **#sw** print item command.

Print enhanced characters =

If your printer has both a draft and a high quality character set, enter the code to select the high quality set. Sent by the **#ec** print item command

Print ordinary characters =

If your printer has both a draft and a high quality character set, enter the code to select the draft quality set. Sent by the **#oc** print item command.

Set alternate character set =

Send the code for any alternate character set which your printer may support. Sent by the **#ac** print item command.

Set standard character set =

If an alternate character set code has been defined, enter the code to return the printer to the standard character set. Sent by the **#sc** print item command.

User control 0 =

There are 28 user control entries numbered 0 to 27. You may select any of these codes within a **sagerep** program using the **#c***nn* print item command in a **print** or **printh** statement where *nn* is the user control number.

Configure printer =

This sequence is sent to the printer once at the start of each **sagerep** program. Use it to perform any initialisation of the printer. If the parameter file is to be used to output to a vdu, enter the code to select the correct display mode for the terminal.

Reconfigure printer =

Used to return the printer to a normal state on exit from a **sagrep** program.

Standard page length =

Enter the number of lines on a standard page. This number should match the hardware page length in lines for the printer (usually set by switches within the printer).

Standard page width =

Used to determine which of the page widths below should be considered the standard page width. Enter the number of characters across the page.

Number of page widths =

This entry should be at least one and is used to determine how many page widths and codes will follow. The two lines below should be repeated this number of times and should start with the least compressed width and proceed to the most compressed.

Width 1 =

For each page width available on the printer, enter the number of characters across the page in that width.

Code =

Enter the code to set the page width specified above.

These are all the entries that are actually required for a printer parameter file. If your printer supports many page widths, you may have a number of further lines defining the size of each page width and the code to set that width.

Error messages

Can't open *filename* **(system error** *nn***)**
Can't create *filename* **(system error** *nn***)**

The named file could not be opened/created due to an error identified as *nn*. Check your operating system manual for further details.

Error in control codes at line *nn*

An error was found at line number *nn*.

Bad number of page widths

The number of page widths must be greater than zero.

Error in column setting parameter

An error has occurred in the code to set the column entry.

Decimal number out of range

Valid ranges for decimal character codes are 0-255.

Error in hex number

Hex numbers must be in the format **$***nn* where *n* is a character in the range 0-9 or A-F.

Illegal code at line *nn*

Check the parameter source file. Details on valid entries may be found in appendix B of this guide.

Error *nn* **writing** *filename*

The operating system error identified as *nn* has occurred while writing the output file. Check your operating system manual for further details.

Error in arguments

Check the command line syntax.

Too many file names

setprint may only be called with a single file name.

Path too long

The path length exceeds the program's limits.

Input and output files are identical

The input and output files must have different names.

Introduction

Vdu parameter files are stored in a binary format that allows them to be quickly and easily read by the **sage** program at run-time. The vdu parameter file holds vdu specific information so that the screen form language system need not have that information "hard-coded" into the program.

The **setvdu** utility reads either standard input or a text version of the vdu parameter file created with **decvdu** and creates a vdu parameter file. This parameter file may be subsequently decoded using the **decvdu** utility (see page 10-5).

If standard input is used, and input is not redirected, a series of questions about the vdu are displayed and a response is awaited. The reference manual for the vdu will be required to answer these questions (see page 10-47 for a complete list of the vdu parameter file entries). A more common method of creating a vdu parameter file, however, is to use **decvdu** to decode an existing file, edit that file using a text editor and then create a new vdu parameter file from that edited text file.

A set of vdu parameter files is supplied with the Sculptor system. If you are using a vdu for which a parameter file has not been supplied, a new file must be created as detailed above. Setting up a new parameter file often involves some trial and error as the available documentation is not always clear. The best approach is to get the cursor control and basic erase functions working first. If the terminal supprts protected fields, try using these next, as they considerably speed up erases of foreground data. It is common for protected fields to be in low intensity and this provides a good contrast between the form and the data. Finally, experiment with highlighting, but note that those terminals which have embedded attributes (ie, occupying a space on the screen) can only be used to highlight text which has a spare space at either end. Terminals with non-embedded attributes are much more flexible.

It is not possible to guarantee that Sculptor will work with a particular terminal, but in practice nearly all ASCII terminals are suitable.

Command line syntax

setvdu [**-cp**][**-f**[=*inputfile*]] *filename*

Create a vdu parameter file *filename* by answering a series of detailed technical questions about the vdu. Input to **setvdu** may be redirected from a file created by **decvdu** or input will be read from the text file *filename*.**s** or *inputfile* if the **-f** option is used.

The **-p** option will create the file in the default Sculptor vdu parameter file directory or the path specified by the **SCULPTOR** environment variable.

On a Unix® system the environment variable **TERM** is used by Sculptor to determine the name of the vdu parameter file to use. Under MSDOS® systems, the name **vdu** is used and cannot be changed. On MSNET or Novell networks, each vdu is given the name vduN where N is a number which identifies the terminal in use. If, on a DOS network, Sculptor does not determine the correct terminal number, the environment variable **TTYNO** can be used to specify the correct terminal number for vduN.

In situations where the **TERM** variable and the Sculptor vdu file used must be different, the environment variable **SCVDU** may be set to indicate the vdu parameter file to use. This will over-ride the **TERM** environment variable. The **SCVDU** environment variable has no effect in single-user operating systems such as MSDOS.

The **-c** option is used to convert from the parameter file format used in the Sculptor version 1.16. When converting, the parameter file should be decoded, using **decvdu**, with the **-c** option and then encoded, using **setvdu**, also with the **-c** option. This will ensure accurate conversion to the new format.

EXAMPLE

```
setvdu -f vdu0
setvdu -f=stdvdu.txt wyse350
```

Vdu parameter files

The following guide lists all of the available vdu parameter file entries and briefly describes where each entry is used. The line number shown may be directly referenced within a screen form language program using the command **vdu** *nnn*, where *nnn* is the line number of the sequence to issue.

Entry format

The codes issued by each parameter may be entered in a variety of formats, any of which can be combined. These are the available formats:

#nnn	*nnn* is a decimal number, eg #90, #108, #9
$*xx*	*xx* is a two character hexadecimal code, eg $0d, $ff, $09
^*x*	^*x* is a control code, eg ^L, ^M, ^G, ^A
ESC	an escape character (#27, $1b)
SPC	a space character (#32, $20)
NULL	an ASCII zero (#0, $00)
PAD *n*	send *n* NULL characters. *n* can be in the range 0-127
\$, \#, \^	send the $, # or ^ character
DEL	send the DEL character, normally $7f, #127

A sequence of codes for a single entry must be separated with commas. For example,

```
ESC,[,1,m
#27,#101,NULL
```

Vdu entries

Vdu Name

The first line of the file is the vdu name. The name must begin **ibmpc**if the PC BIOS is to be used.

Keyboard Type

The second line is the keyboard in use. The keyboard information is available within screen form programs in the **keyboard** temporary field. This entry is typically used in conjunction with the **kprnt** utility.

Standard width (in decimal) =

The normal vdu display width in characters. For most terminals this will be 80.

Extended width (in decimal) =

If the vdu supports extended width, enter the number of columns on the extended screen display here.Many terminals support 132 columns.

Standard depth (in decimal) =

The standard depth of the vdu display in lines. For most terminals this will be 24. The IBM PC® uses a 25 line display, but you may wish to set this to 24 to maintain compatability with other systems.

Extended depth (in decimal) =

If the vdu supports extended depth, enter the number of lines on the extended depth display here.

Co-ordinate first (x/y) =

Indicate which co-ordinate is required first in a cursor positioning sequence. Some vdus require the x position co-ordinate first, others require y first.

Co-ordinates sent as ASCII text (y/n) =

Reply **n** if the x and y co-ordinate values are each sent as a single character code. Reply **y** if the co-ordinates are sent as a variable number of ASCII numerals with a separator between the x and y values (ANSI standard). See numbered entries **29** and **30**

Cursor position bias (in hex) =

If your terminal requires a position bias or offset, enter the number here in hexadecimal relative to 1. Use $ff for -1.

Extended position bias (in hex) =

If your terminal requires a position bias or offset in extended mode, enter the number here in hexadecimal.

1: Position cursor =

The sequence used to start a position cursor command to the terminal. Do not enter co-ordinate separator or terminator codes.

2 : Position cursor (extended) =

The sequence used to start a position cursor command to the terminal when it is in extended width or depth.

3 : Set extended screen width =

The code used to set the terminal to extended width.This is sent by the **!width** *nnn* statement in a screen form program or menu file if *nnn* is greater than the present width. Must also include a clear screen sequence.

4 : Set standard screen width =

The code used to set the terminal to standard width.Sent by the **!width** *nnn* statement in a screen form program or menu file if *nnn* is less than the present width. Must also include a clear screen sequence.

5 : Set extended screen depth =

The code sent to set the terminal to extended depth. This is sent by a **!depth** *nnn* statement in a screen form program or menu file if *nnn* exceeds the current screen depth. Leave blank if the terminal does not support this feature.

6 : Set standard screen depth =

The code sent to set the terminal to standard depth. Sent by a **!depth** *nnn* statement within a screen form program or menu file if *nnn* is less than the current screen depth.

7 : Sent by CANCEL key =

The CANCEL key is used to interrupt an input within a screen form program and return immediately to the option line. Normally set to ^X. Leave blank to disable this feature.

8 : Backspace character sequence =

If your terminal has a destructive backspace character enter the code for that character here, otherwise the sequence ^H,SPC,^H usually works.

9 : Sent by 'End of input' and 'MODE' key =

The key that is defined here will cause an end of input trap in screen form programs, and will toggle the MODE in the **fql** program. Leave blank to disable this feature. Normally set to ESC.

10 : Home cursor =

The code to return the cursor to the top left hand corner of the screen without clearing.

11 : Configure VDU =

This code sequence is issued on initial entry to screen form programs and on return from an **exec** command. It may be used to put the vdu into a special mode, configure function keys, etc.

12 : Reconfigure VDU =

This code sequence is issued on exit from a screen form program and prior to an exec command. It may be used to reset any special modes set by **Configure VDU** or to ensure that the vdu is reset to it's normal state.

13 : Home and clear screen =

Enter a sequence which homes the cursor and clears the screen including protected characters. Screen form programs issue the **Disable Protection** sequence prior to this one in case the same code is used for protected and unprotected erases.

14 : Home and clear unprotected =

Enter the sequence which homes the cursor and clears the screen except for protected characters. Screen form programs issue the **Honour protection** sequence prior to this one. If the vdu does not have protected fields, leave this code blank and the foreground data will be erased by spacing over it.

15 : Erase to end of screen =

The sequence which erases all characters from the current cursor position to the end of the screen. If the vdu does not provide this, enter a Home and clear screen sequence.

16 : Erase unprotected to end of screen =

This sequence is not currently used and may be left blank.

17 : Erase to end of line =

The sequence which erases all characters from the current cursor position to the end of the line.

18 : Erase to end of field =

If using protected fields and the vdu has an erase to end of field or an erase field code, enter it here.Otherwise leave blank and the field will be erased by spacing over it. This sequence must leave the cursor at the beginning of the field.

19 : Honour protection and disable scroll =

If the vdu has protected fields that are effective only when protection is enabled, enter the sequence which enables it. If this does not also disable scroll and there is a separate sequence to do so, enter it here too.

20 : Ignore protection and enable scroll =

If using fields that are effective only when protection is enabled, enter the sequence here that disables it. If this does not also enable scroll and there is a separate sequence to do so, enter it here too.

21 : Start protected field =

If using protected fields, enter the sequence which defines tha start of a protected field.

22 : End protected field =

If using protected fields, enter the sequence which defines the end of a protected field.

23 : Start page title =

If a suitable highlight is available for the title line of a screen form program, enter the sequence which starts it. The **Start protected field** sequence is always sent prior to this one.

24 : End page title =

If a highlight sequence is used for the title line, enter the sequence which ends it. Screen form programs always send the **End protected field** after this one.

25 : Start Field heading =

Enter the start sequence for field headings. Screen form programs always send the **Start protected field** sequence prior to this one.

26 : End field heading =

If a highlight sequence was used to start the field heading, enter the sequence which ends it here. Screen form programs always send the **End protected field** sequence after this one.

27 : Start error message =

Normally a bell code (#07) followed by a sequence to start any suitable video attribute (eg flashing).

28 : End error message =

If a video attribute sequence is used for error messages, enter the code which ends it.

29 : XY co-ordinate separator =

It the cursor position co-ordinates are sent as ASCII text, enter the code which separates the the x and y co-ordinates.

30 : XY co-ordinate terminator =

If the cursor position co-ordinates are sent as ASCII text, enter the code which terminates the positioning sequence.

31 : Enter 1 to avoid RAW mode =

If RAW mode is avoided then the operating system will be able to check for special characters such as XON/XOFF. This may be essential if you work with modems. However, RAW mode is essential if control codes are used for cursor positioning or if an operating system control code conflicts with a terminal control code.

32 : Start normal data =

This sequence precedes each data field that is displayed with the display command in a screen form program. It's main use is to reverse the effect of the highlight command, but you may use it to display your normal data in any highlight required.

33 : End normal data =

This sequence is appended to each data field that is displayed with the display command in a screen form program.

34 : Start option list =

If a suitable highlight is available for the option list, eg reverse, enter the sequence that starts it here. Screen form programs always send the **Start protected field** sequence prior to this one.

35 : End option list =

If a highlight sequence is used for the option list, enter the sequence that ends it here. Screen form programs always send the **End protected field** sequence after this one.

36 : Start highlight data =

If a suitable highlight is available for highlighted data, enter that sequence here.

37 : End highlight data =

This sequence is appended to each data field that is displayed with the **highlight** command in screen form programs. It should reset to the normal display attribute.

38 : Enter 1 to force left/right box sequences =

If the field delimiters are set to spaces, Sculptor will not draw those spaces and so will not set any attributes that *would be sent* if the field delimiters were not blank. Setting this entry to 1 will ensure that all attributes are sent at the correct locations even if the delimiters are blank. This will also marginally slow screen redrawing.

39 : Re-display form character =

Defines a character which, if typed while a screen form program is awaiting input, repaints the screen form. This may be left blank to disable this feature. NOTE: if boxes or lines which are not declared (ie not drawn using **!at**) have been drawn using **hline/vline** or **drawbox**, these will be erased when this key is entered. Normally set to ^F.

40 : Start left box delimiter =

When the form is painted in a screen form program, this sequence precedes each left field delimiter character.

41 : End left box delimiter =

When the form is painted in a screen form program, this sequence follows each left field delimiter character.

42 : Start right box delimiter =

As Start left box delimiter.

43 : End right box delimiter =

As End left box delimiter.

44 : User sequence =

The sequences from number 44 through to number 59 inclusive are available for your use and may be called from within a screen form program with the command **vdu** *nn*, where *nn* is the appropriate sequence number.

60 : Sent by delete to end of field key =

Enter the character that will be used to delete to the end of the field within an **input** command in a screen form program. Normally set to ^Z.

61 : Sent by backspace key =

An entry here will over-ride the default setting of $08 that is normally used for the backspace key.

62 : Sent by left arrow key =
63 : Sent by right arrow key =
64 : Sent by up arrow key =
65 : Sent by down arrow key =

Enter the codes that are sent by the cursor movement keys on the vdu keyboard. These codes are used throughout the Sculptor suite to recognise cursor movement commands.

66 : Sent by RETURN key =

An entry here will over-ride the default setting of $0d that is used to identify the **Return** or **Enter** key.

67 : Sent by TAB key =
68 : Sent by back TAB key =
69 : Sent by insert character key =
70 : Sent by delete character key =
71 : Sent by insert line key =
72 : Sent by delete line key =
73 : Sent by scroll up key =
74 : Sent by scroll down key =
75 : Sent by split line key =
76 : Sent by join line key =

Define the codes sent by the above keys. These general movement and control keys are used throughout the Sculptor suite and may be user-defined to perform any action within the screen form language.

77 : Sent by toggle insert/overtype mode key =

Define the key that will be used to switch between Insert and Overtype modes in many of the Sculptor utilities and in screen form programs.

78 : Sent by page up key =
79 : Sent by page down key =
80 : Sent by home key =

Defines the keys that may be used within screen form programs to perform any programmer-defined action. The page up and page down entries are also used within the utility programs to perform paging functions.

81 : Sent by function key 1 =

The entries between 81 and 112 define the codes sent by the function keys from F1 through to F32. Many of the Sculptor suite programs use function keys F1 to F10 and will operate incorrectly if these entries are not set properly. If a function key does not seem to be operating correctly, but the code here is correct, ensure that no previously defined key has an identical sequence.

113 : Insert character right code =

The code that instructs the terminal to insert a character to the right of the current cursor position. Leave blank if your terminal does not support this feature.

114 : Delete character right code =

The code that instructs the terminal to remove the character under the cursor and move all characters to the right left one character. Leave blank if your terminal does not support this feature.

115 : Insert line down code =
116 : Delete line up code =

Reserved for future use.

117 : Bell character =

An entry here will over-ride the default value of $07.

118 : Switch cursor off =

Sent by screen form programs immediately after input has been accepted. Enter the code that turns the cursor off. On the IBM PC®, this entry should comprise two hex numbers defining the start and end scan lines of the cursor ($20,$20 will normally turn the cursor off regardless of video mode).

119 : Switch cursor on =

The code to turn the cursor back on again should be entered here. On the IBM PC® this entry should comprise two hex numbers defining the start and end scan lines of the cursor. The scan lines will depend on the video mode and adapter card in use.Check your hardware manual for details.

120 : Normal video =

Resets the video modes shown below. Enter the code sequence to return to normal video.

121 : Standout video =
122 : Reverse video =
123 : Underscore video =
124 : Blinking video =
125 : Low & Reverse =

The attributes shown above are used both within the screen form language for general attributes, but are used throughout the Sculptor application suite.

126 : User attribute 1 =

The entries from 126 through to 134 are the **User attribute** entries from 1 through to 9 inclusive. These may be used for specific video attribute control through the use of the **vdu** command within screen form programs.

135 : Start graphics mode =
136 : End graphics mode =

Sequences to begin and end graphics mode. These sequences are issued prior to, and after issuing any of the graphics characters shown below. Your terminal may require a specific sequence to enter and leave graphics mode. These sequences may also be used to set another colour for thin-line graphics.

☐ **137 : Line wrap around on =**

Re-enables the line wrap around which may have been disabled with the sequence below.

138 : Line wrap around off =

When graphics characters are displayed at the lower right hand corner of the screen, the display will normally scroll up. The sequence for this entry should disable scroll so that the display remains intact. If this entry is not set, graphics characters will *not* be displayed in the lower right hand corner of the screen when drawing boxes.

139 : Graphics horizontal bar =
140 : Graphics vertical bar =
141 : Graphics top left corner =
142 : Graphics top right corner =
143 : Graphics lower left corner =
144 : Graphics lower right corner =
145 : Graphics 'T' character =
146 : Graphics inverted 'T' character =
147 : Graphics left rotated 'T' character =
148 : Graphics right rotated 'T' character =
149 : Graphics '+' character =

These are the line graphics characters that make up the box and line drawing facilities used by the Sculptor application suite and by screen form language programs.

150 : Block graphic character 1 =

There are four block graphic characters that are used by the **hline** and **vline** commands in a screen form program. These entries represent the **hline/vline** modes **13** through to **16**.

154 : User graphic character 1 =

There are 11 user graphic characters (sequences 154 to 164 inclusive) available for the programmer. These may be defined here and issued from within a screen form program with the **vdu** command.

165 : Select alternate character set =
166 : Select standard character set =
167 : Start form highlight =
168 : End form highlight =

The sequences above are used by the **!highlight** command in the screen form language. The **Start form highlight** sequence is sent at the beginning of each field to be highlighted and the **End form highlight** is used to return to a normal video attribute.

169 : Start writing to terminals's message line (if applicable) =
170 : End writing to terminal's message line (if applicable) =

On many 24 line terminals, the 25th line may be written to using a special sequence of control characters to start and end the line. Cursor positioning commands do not normally allow writing to this line. Set this code if your terminal supports writing to the 25th line.

171 : Normal background colour =

Resets the normal background colour if an alternate has been set in the sequence below.

172 : Alternate background colour =

For clarity, the menus within **describe** and **sp** may have a different colour background. This sequence may be used to select the background colour.

173 : Enter 1 if graphics characters are protected (eg WYSE 50) =

Certain terminals (eg Wyse50) automatically set protection on graphic characters. This means some programs will leave line graphics on the screen after other data has been cleared. Setting this entry to 1 forces such displays to be cleared by printing spaces. This will slow the clearing of the screen down considerably, so must only be used if necessary.

174 : Interrupt code (Unix) =

This entry can be used to reconfigure the keyboard interrupt character used within screen form language programs. This entry is only recognised by the UNIX® and XENIX® operating systems.

175 : Enter 1 if VDU has embedded attributes =

This entry should only be used if your terminal uses character attributes which take a character space on the screen. Setting this entry to 1 will instruct the screen based programs to perform additional processing to ensure that screen titles are displayed correctly on these systems.

176 : Reserved =

The last few lines are reserved for future use.

IBM keyboard codes and video attributes

Keyboard entries and the Sent By sequences

On an IBM PC® (or compatible) keyboard, the codes sent by the special keys are preceded by a NULL character. The **Sent By** sequences should contain only the character that follows this NULL value. The function key **F1**, for instance sends the sequence $00,$3b (NULL, ;), but the **Sent by** sequence should only contain the **$3b** (**;**) value. If the key is a control code, the entry should be written as **^V** where **V** is the control character sent.

Video attributes

The easiest way to express video attributes in DOS is using a hex number in the format **$***nn*, where the first *n* is the background colour (in the range 0-7) and the second *n* is the foreground colour (in the range 0 to f). The actual colours or attributes that these numbers represent will depend on the pallete in use.

The codes for the entry **Normal backround colour** and **Alternate background colour** correctly relate to the background colour *only* and should therefore be in the range 0-7.

Error messages

Can't open input file

The input file (with a **.s** extension) could not be read. This error will only occur when using the **-f** option.

Can't open input file (operating system error number *nn*)

An operating system error identified as *nn* has occurred. Check your operating system manual for further details. A frequent cause is no permission to create or write the file or one of the directories in it's path name.

Error in coordinate order parameter

The coordinate order entry must be x or y.

Error in coordinate as ASCII specification

Valid entries are y or n.

Error in cursor position bias
Error in extended cursor position bias

Both of these values must be input as hex numbers using $nn.

Error at line *nn*

This message identifies an error in the main body section of the vdu parameter file. The line number given is the sequence number of the offending option.

linetext : number out of range

The entered number is not valid for the entry shown.

System error *nn* writing to *filename*

System error *nn* occurred writing to the named file. Check your operating system manual for details. Frequent causes are: disk is full, no write permission or file is read-only.

Parameter file NOT created

This message will accompany any of the preceding error messages and indicates that the vdu parameter file has not been created from the input file. Note the error(s) shown, and correct the parameter source file, then re-try.

Error in arguments

Check the syntax above.

Too many file names

setvdu may be called with only one file name.

Path too long

The path exceeds the program limits.

Input and output files are identical

The input and output file must have different names.

vno — Display version information about an intermediate file

Introduction

Normally only used by MPD technical support staff, **vno** displays the version and revision number of **cf** or **cr** that was used to compile an intermediate code file.

Command line syntax

```
vno filename [filename] ...
```

Returns information about Sculptor intermediate code files. The **.g** or **.q** file extension must be explicitly stated.

Error messages

No error messages are produced.

APPENDIX A RESERVED WORDS

The following words are reserved in the Sculptor screen form and report/batch languages and should not be used as field names or as manifest constants. The symbol "r" next to a word denotes the word is reserved in the report/batch language and the symbol "s" denotes the word is reserved in the screen form language.

A

ac	r
append	rs
arg	rs
asc	rs
at	s
autocr	s
autogoi	s
autohelp	s

B

BACKTAB	s
block	s
box	s
break	rs
bs	s

C

c0-c29	r
cancel	s
case	rs
center	rs
centre	rs
cfile	rs
chain	rs
chdir	rs
check	s
chr	rs
clear	s
clearbox	s
clearbuf	rs
clearkey	s
close	rs
constant	rs
continue	rs
count	r

D

date	rs
day	rs
decdate	rs
default	rs
delete	rs
DEL_LINE	s
depth	rs
dim	rs
display	rs
drawbox	s
dw	r

E

ec	r
editmode	s
else	rs
encdate	rs
end	rs
endif	rs
ending	r
endneed	rs
endrec	r
eoi	s
err	rs
errno	rs
error	s
eu	r
exclude	r
exec	rs
execu	s
exit	rs

F

F1-F32	s
field	s
file	rs
final	s
find	rs
footnote	r
for	rs
form	s
format	rs

G

gap	r
get	rs
getstr	rs
global	s
gosub	rs
goto	rs

H

handles	rs
hangup	rs
heading	r
hline	s

I

if	rs
ifdef	rs
ifndef	rs
ifneed	rs
inchar	rs
include	rs
input	rs
inputbuf	s
inputerr	s
insert	rs
instr	rs
INS_LINE	s
INTERRUPT	s
interrupts	s

K

keep	r
key	rs
keyboard	s
keycode	s

L

left	rs
let	rs
line	s
local	s
lock	rs

M

match	rs
max	r
message	s
min	r
month	rs

N

newform	s
newpage	r
next	rs
nextkey	rs
ni	s
no	s
nrs	rs
nsr	rs

O

oc	r
off	rs
on	rs
open	rs
opthelp	s
option	s
optline	s

P

pause	rs
PGDN	s
PGUP	s
power	rs
preserve	s
prev	rs
prevkey	rs
print	r
printh	r
prompt	s
put	rs

R

rand	rs
re	rs
read	rs
readkey	rs
record	rs
redraw	s
remove	rs
return	rs
rewind	rs
right	rs
riu	rs
rounding	rs

S

sc	r
screen	s
scrline	rs
SCRL_DN	s
SCRL_UP	s
scroll	rs
select	r
separator	rs
setstr	rs
skip	s
sleep	rs
spc	r
sqrt	rs
starting	r
startrec	r
strlen	rs
su	r
sw	r
switch	rs
systime	rs

T

TAB	s
tab	r
task	rs
temp	rs
testkey	rs
tf	r
then	rs
time	rs
title	r
tolower	rs
total	r
toupper	rs
tstat	rs
ttyno	rs

U

unlock	rs
user	s
userid	s

V

validhelp	s
vdu	s
vduname	s
vline	s

W

wakeup	rs
while	rs
width	rs
write	rs

X

xfile	r

Y

year	rs
yes	s

Z

zeros	rs

APPENDIX B IMPLEMENTATION DIFFERENCES

Programs written in the Sculptor screen form or report/batch language are generally machine independent. However, since the Sculptor system permits access to operating system commands, the use of such commands can make a program operating system dependant. Also, where Sculptor relies on an operating system service, there may be differences between implementations.

The following guide lists the known implementation differences by operating system. It is not guaranteed to be exhaustive.

MSDOS

1. There are special versions of Sculptor for certain networks on which file sharing and record locking are permitted, but the standard, single-user version for MSDOS cannot share files. The **unlock** command is permitted but is ignored and a record cannot be locked.

2. Up to 32 Sculptor files may be opened simultaneously under DOS version 3.3 and above only if sufficient file handles have been specified in the CONFIG.SYS file and the **!handles** declaration is used. If **!handles** is not used in the program or if the version of DOS is less than 3.3, only 8 Sculptor files may be opened simultaneously.

3. The **exec** command is available but since the operating system cannot hold programs whose cumulative size is bigger than available memory, the machine must have sufficient RAM to hold all concurrent tasks (the DOS 640K memory limit also applies). Several system commands may be given in one **exec** call, a semi-colon ";" being used as the separator character. This emulates UNIX but means that if a semi-colon is required as part of a system command that command must be executed in a batch file.

4. The special temp **tstat** recieves the termination status of a child program only if the command is prefixed with a minus sign "-" as the command processor does not pass back the status of child tasks. In other cases, **tstat** will be zero, although it may be non-zero if the **exec** failed completely.

5. Except on Novell and MSNET, the special temp **ttyno** is always zero.

6. The special temp **task** is always blank.

7. MSDOS version 1 is not supported.

1. The **exec** command is available but since the operating system cannot hold programs whose cumulative size is bigger than available memory, the machine must have sufficient RAM to hold all concurrent tasks.
2. The special temp **ttyno** relies on the terminal device module name ending with the port number, eg. "T1".
3. For 6809 OS9 (level two), only Sculptor 1.16 is available.

INDEX TO SCULPTOR REFERENCE MANUAL