SCULPTOR REFERENCE MANUAL

THE SCULPTOR APPLICATION DEVELOPMENT SYSTEM

Fifth Edition issued April 1990

ISBN 0 947770 02X

Published by

Microprocessor Developments Ltd

CHAPTER 1 GLOSSARY OF TERMS

The following terms have been used in this manual and are defined here.

argument

A parameter to a function.

ascii

A set of codes used to represent characters. Sculptor supports extended, 8-bit characters. ASCII is an acronym for American Standard Code for Information Interchange.

channel

A number used to identify a particular, sequential file.

child task

A process whose execution is initiated by another (parent) process. In Sculptor, the **exec** command is used to initiate another process.

command

A word in the Sculptor programming language which is used to perform an action.

data field

A field in a file's data dictionary which is not part of the key.

data file

A system file which is used to hold records.

day number

A means of representing a date as a number of days from a fixed base-date. In Sculptor, day number 1 is the date 1/1/0001.

declaration

A statement in the Sculptor programming language which defines an option or action which is not executed in strict, procedural sequence, but is executed as part of the initialisation of the program.

descriptor

The information such as the name, type and size which is used to describe a field in a data dictionary.

field

An elementary data item stored in a record.

file id

A name or number used to identify a particular keyed file.

function

A program subroutine which computes a result or performs an action based on the values of its arguments.

index file

A system file which is used to hold an index to records in a data file. Sculptor index files have a **.k** suffix.

key field

A field in a file's data dictionary which is part of the key.

keyed file

A logical file consisting of two system files, a *data file* which holds data records and an *index file* which holds the index to those records.

label

A name given to a program line so that the line may be referenced by a **goto** or **gosub** command or a trap clause.

lock

A restriction which is temporarily placed on a record or file to prevent other processes from updating that record or file.

manifest constant

A name given to a word which may be used to represent a value in subsequent program code.

numeric expression

An expression involving fields and constants together with operators which produces a numeric value as its result.

parent task

A process which initiates the execution of another (child) process. In Sculptor, the **exec** command is used to initiate another process.

pathname

A file specification which describes the name and location of a file. In Unix, the components of a pathname are separated by a forward slash (/) and in DOS by a backward slash (\). Sculptor programs accept a forward slash on both systems.

process

A program which has been loaded and is running, or is waiting to run pending some event.

raw mode

A communication protocol with the terminal in which no characters have special meaning to the operating system. This mode has the disadvantage that interrupt characters and flow control characters cannot be used but it is occasionally necessary when using a terminal whose control codes conflict with ascii standard protocols.

record

A set of fields which comprise a logical unit, for example, a stock record.

record buffer

An area in a program into which a record from a file is read and then made available for processing. Sculptor automatically allocates a record buffer to each file declared in a program.

record field

A field which is described as part of a file's record as opposed to a temporary field declared locally within a program.

relational expression

An expression involving fields and constants together with operators which produces a true or false result. A numeric expression which produces a non-zero result is true and one which produces a zero result is false.

screen field

A record or temporary field which is linked, within a program, to a display area on the screen.

screen overlay

A set of screen fields which may be on (displayed) or off (not displayed) at any one time when a program is running.

scroll area

A set of screen fields laid out such that each field has its heading at the top of a column of data values for that field. All fields in the scroll area have the same heading line and occupy the same number of rows. (The scroll area may be used to display subscripted fields or to concurrently display data from multiple records which might, for example, represent the lines on an invoice.)

scroll line

The row number within the scroll area on which screen input and display commands currently operate. Also used as the default index value for subscripted fields which are not explicitly indexed when referenced.

sequential file

A system file from which data is either read or written in sequence. A sequential file has no index and cannot be randomly accessed.

string

A sequence of printable characters enclosed in quotation marks to form an alphanumeric constant.

subscript

An index value for a subscripted field.

subscripted field

A field which holds multiple values. The dimension of a subscripted field defines the number of values that it can hold.

temporary field

A field declared locally within a program as opposed to a field which is described as part of a file's record.

text expression

An expression involving fields and constants together with operators which produces an alphanumeric value as its result.

trap clause

A program clause which transfers execution to a labelled line if a particular exception condition occurs.

vdu

A computer terminal or PC with a screen and keyboard. VDU is an acronym for Visual Display Unit.

^X

Shorthand for the key code sent by holding down the CTRL key and pressing X. X may be any character in the ascii range @ through _ (underscore).

CHAPTER 2 AN INTRODUCTION TO SCULPTOR

This chapter provides a summary overview of the major Sculptor features.

Contents **Page**

General description

Sculptor comprises a suite of programs designed to enable the rapid development of sophisticated software for a wide variety of applications. At the heart of the Sculptor system is a powerful and flexible keyed file technique which provides very fast retrieval, insertion and deletion of stored information. Records may be accessed randomly or in ascending key sequence and there are powerful search commands for use when the exact key is not known. The indexing technique (known as a B-Tree) keeps the index permanently sorted - it never requires re-structuring.

Data processing in Sculptor is accomplished through two specifically designed high level languages. One is for interactive work through a vdu (screen form language) and the other is for report writing and batch processing (report/batch language). Both languages utilise the same basic commands and declarations, and differ only in areas specific to their particular use, and in their control logic. Both languages are also suitable for general update programs and contain powerful, high level commands. These commands take care of most normal chores, leaving the programmer free to concentrate on the application itself.

The languages are compiled to intermediate code which is then interpreted by a run-time interpreter. This allows a very swift edit-compile-execute cycle and enables full portability of both source and compiled files across the range of machines and operating systems on which Sculptor is available. The data files themselves are stored in a machine-independent fashion and are also fully portable.

A set of support programs make the development and maintenance of a complete software project both quick and easy. There are utilities to describe the structure of the files, to create a nested menu system and to check the integrity of index files and make any necessary repairs. In practice the indexing technique has proved to be extremely reliable and file damage is only likely to occur through power or hardware failure.

The Sculptor system provides full device independence through the use of vdu and printer parameter files. Utilities are supplied to define the parameters for terminal and printer characteristics so that programs may run without modification on a wide variety of systems. Many of the internal messages issued by the Sculptor system may be configured to any spoken language by the **lcf** utility.

Two program generators are supplied to automatically create screen form or report/batch programs as required.

An implementation of the popular Structured Query Language (SQL) is supplied and may be used to generate ad-hoc reports from any of the data files on the system. Fields on any data file may be protected from access by SQL.

Sculptor is ideal for a wide variety of business applications; any system involving retrieval from and updating of large or small data files, cross referencing one to another, is particularly suitable. Full data processing including all normal arithmetic calculation is handled efficiently, and Sculptor is currently managing such diverse applications as data capture in real time and complete business accounting.

The keyed file routines used within Sculptor are available separately for use within the C language to access Sculptor keyed files directly.

Documentation conventions

Throughout this manual, the following conventions are used in syntax descriptions.

1. Items in *italics* to be replaced as detailed in the text.
2. Items in **bold face** and punctuation should be entered as shown.
3. Optional items are enclosed in square brackets "[ ]". The square brackets should not be entered as part of the command.
4. An ellipsis "..." indicates that the preceding item may be repeated.
5. A list of items separated by a vertical bar character "|" indicates that a choice may be made from the selection shown.

The Sculptor program suite

The following programs are supplied in the Sculptor development system:

cf	The compiler for screen form language source files.
cr	The compiler for report/batch language source files.

describe	Define and amend the record structure for Sculptor keyed files.
kprnt	Text file interpreter used to display formatted help screens.
kfcheck,kfcopy, kfdet, kfri	Keyed file utilities to check the integrity, copy, display details and rebuild the index of Sculptor keyed files.
lcf	Language and format configuration of Sculptor programs.
menu	Text file interpreter which produces a fully portable formatted menu system.
newkf	Creates a new, empty, keyed file from a record descriptor created with **describe**.
pause	Used to display a prompt and await a key press.
pdes	Prints a data dictionary defined by **describe**.
reformat	Reorganises the data in a Sculptor keyed file after changes to the structure.
rg, sg	Report and screen form language generators for the automatic production of source files.
sage	Screen form language interpreter.
sageform	Prints a screen form layout for documentation.
sagerep	Report/batch language interpreter.
setprint, decprint	Set up and decode printer parameter files.
setvdu, decvdu	Set up and decode vdu parameter files.
sp	Interactive screen painter for screen form language programs.
spp	Source pre-processor for both screen form and report/batch language source files.
sql, eql, fql	Structured query language. Interpreter, environment and form based interfaces.

vno	Determines the version of the compiler that was used to compile a program.

Sculptor keyed files

A Sculptor keyed file is maintained as two separate disk files, one containing data records and the other the index. All data, including key information, is held in the data file. This means that if the index is lost or becomes damaged it can be rebuilt from an intact data file. A data file is initially defined with the **describe** program, which creates a data dictionary file and determines the type and size of each field within a record. The **newkf** program is then used to create the data and index files, the data file is created with the same name as the data dictionary file, but without a file extension. The index file is created with a **.k** extension.

Each keyed file has one physical key which may be logically interpreted as several separate data items. There are commands to search the index on part of the key. The index is a multi-level tree which, for efficiency, is reorganised every time a record is inserted or deleted. The technique used is very fast and means that the index is always up to date. The data file grows automatically as records are inserted and it is not necessary to predeclare the file size. To avoid unnecessary growth, the space released by deletions is re-used when new records are inserted.

The only restrictions for a Sculptor keyed file are:

1. There must be at least one key field.
2. The total record length must be at least 3 bytes.
3. The total record length must not exceed 32,767 bytes.
4. The total number of records must not exceed 16,777,216.
5. The total key length must not exceed 195 bytes

An alternative descriptor file may be used to define an alternative record layout for any keyed file. This allows different types of record to be stored in the same file (see the **!record** declaration in the screen form and report/batch language chapters).

Files which contain only key information may be defined. These can be used to provide cross-referencing and alternative lookup indexes.

vdu parameter files

The Sculptor screen form language requires a vdu parameter file for each type of vdu that is being used. Several such files are supplied with the system and new ones can be created with the program **setvdu**. See chapter 10 for further details.

The purpose of these parameter files is to make screen form language programs independent of the terminal being used. Almost any terminal with cursor positioning and a "clear screen" command may be used, although Sculptor is more effective if protected fields and thin-line graphics are available. Almost all modern terminals are suitable and many different types of terminal may be used on the same system.

Printer parameter files

The report/batch language requires a printer parameter file to describe the capabilities of the printer being used. Several such printer parameter files are supplied with the system and new ones may be created with the program **setprint**. See chapter 10 for further details.

The purpose of these parameter files is to make programs independent of the printer being used and to make it easy to use special features such as double-width characters, underlining and compressed print. If the report requires a feature that is not supported by the printer in use, the program ignores the instruction and continues to print the report.

Creating a Sculptor system

The general procedure for creating a Sculptor application is as follows:

1. Use the program **describe** to define the record layouts for each file required. Up to 32 files may be used within a single screen form or report/batch program and up to eight alternative record layouts may be defined for each file. Good design technique should avoid too much use of alternative record layouts. The **describe** program creates data dictionary files, identified by a **.d** extension, which are then used by the compilers and by some of the keyed file utilities.

2. Create new keyed files using the program **newkf** which reads the data dictionary files, calculates the required key and record lengths and initialises the new keyed files.

3. Write any required screen form programs in the screen form language. These programs are ordinary text files so you may use any text editor available on your system. The files must have a **.f** extension and may be compiled with either the screen form compiler **cf** or the source pre-processor **spp**. Standard file maintenance programs may be created with the screen program generator **sg**. The compiled programs have the same name as the the source file but with a **.g** extension.

4. Write the required report programs in the report/batch language. These are ordinary text files and may be created with any available text editor. The files must have a **.r** extension and may be compiled with either the report/batch compiler **cr** or the source pre-processor **spp**. Standard report programs may be created with the report program generator **rg**. The compiled programs have the same name as the source file but with a **.q** extension.

5. If you subsequently alter a file's record layout by deleting or inserting fields or by changing the type, size or dimension of any field then the file must either be re-initialised or reformatted and all programs which reference that file must be recompiled.

File integrity checks

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 operating system is incorrectly shut down. The update routines ignore all normal keyboard interrupts during file access.

The utility program **kfcheck** checks the integrity of keyed file indexes. It should be run as part of the start-up procedure every time the system is switched on or, if the system is permanently on, it should be run once each day. If damage is reported the **kfri** utility is normally able to repair it. Prolonged use of a damaged file can lead to serious loss of data.

See chapter 10 for further details of the keyed file utilities.

CHAPTER 3 THE SCULPTOR DATA DICTIONARY EDITOR

This chapter describes the operation of the Sculptor data dictionary editor, **describe**.

Contents **Page**

What is a data dictionary

The data dictionary stores information about a keyed file, including field name, type & size, format and validation. Default help text and comments may also be stored for each field, and a field may have attributes set to define its behaviour when accessed with SQL.

A data dictionary is used to define the record layout for each Sculptor keyed file. The data dictionary file has the same name as the data file but with a **.d** extension. It is used by the **newkf** program to create a Sculptor keyed file and by the compilers, **cf** and **cr**, when compiling a program that uses that file.

Defining all the information about a file in one place greatly simplifies the use of that file in Sculptor programs. The file details can be readily printed for documentation purposes using the **pdes** program.

Multiple data dictionaries may be defined for a single keyed file and are referred to as alternate record layouts for that file.

If a data dictionary contains only key information, it may be used as an index-only file (see **newkf** on page 10-30 for further information).

Sculptor keyed file record layout

The record layout is described in terms of fields. Each field must have a name, type and size. The field may optionally have a default heading, format, validation list and help text. These, with the exception of the validation list, may be over-ridden by the programmer within a program.

Each file must have a key. The key for a record is composed of the values for all of the key fields concatenated in the order defined. The data record is composed of all the fields in the record, including key fields, concatenated in the order defined. The record data is stored in a data file. There is also an index file which holds a B-tree index to the data records. The index file has the same name as the data dictionary but with a **.k** extension.

The order in which key fields are defined is crucial, since index sequential access depends upon the content of the key. Keys are sorted by direct byte comparison, which means that if the sequential access order is important, you should not include numeric fields which may contain negative values, or any floating point fields (r8 or m8).

Alternative record layouts may be defined for a Sculptor keyed file. These layouts may be used within programs to access the data in a keyed file using a different set of field names. A data dictionary file is created which has a name that differs from the main file. The field names must also differ. The alternative record layout must have an identical key and record length to that of the main file and it is recomended that the key structure is identical.

Field name

Field names may contain alphabetic characters in upper or lower case, the numerals 0 through 9 and the underscore character, but the first character may not be a numeral. Sculptor keywords may not be used as field names (refer to page A-1 for a list of reserved words).

Each field name must be unique within the data dictionary file. Although it is possible to use the same name in different files, the compilers will issue a warning if identical field names are encountered within a program, and only the first encountered will be accessible.

To avoid possible duplication within an application, it is advisable to adopt a naming convention which identifies the file to which a field belongs. This may be done with a file identifying prefix on each field name. For example,

```
    sl_name
    sl_accno
    sl_amount
```

may be used if the file was a sales ledger file.

Field heading

Field headings are used to identify fields on the screen or in a report. There are no restrictions on the characters that may be used in a field heading. The field heading may be over-ridden within a screen form or report/batch language program.

Field type and size

The field type & size is specified by a single alpha character indicating the type together with a number which indicates the size in bytes, eg **a20** is an alphanumeric field of 20 bytes and **m4** is a money field of 4 bytes. A data field may also be dimensioned by enclosing the number of occurrences required in parentheses after the type & size, eg **d4(10)**. Key fields may not be dimensioned.

Type	**Size**	**Description**
a	0-255	Alphanumeric field which may contain any character.
d	4	Date field stored internally as a day number starting from 1/1/0001.
i	1,2 or 4	Integer stored in binary. The range of an i1 is 0-255, the range of an i2 is -32767 to 32767 and the range of an i4 is -2147483647 to 2147483647.
m	4 or 8	Money field with a main currency unit equal to 100 of its subsidiary units. Stored internally as an integer in the lower currency unit. An m4 field is stored as an i4 type with the same range as an i4 and an m8 field is stored as an integer in a real (r8) field with the same range as an r8 type.
r	8	Floating point field stored internally in IEEE floating point format with up to 15 significant digits. Range is 10^{-37} to 10^{38}.

Since date fields are stored as day numbers, adding or subtracting an integer value from a date adjusts its value by that number of days. Dividing a date by 7 and taking the remainder (using the % operator) gves the day of the week (0=Sunday). Dates are input and displayed in standard formats, eg. 21/12/90 or 12/21/1990 with the input routines adding 1900 to any two-digit year.

Money fields are stored in the lower currency unit which helps to avoid any rounding errors, **m4** fields being stored in long integers and **m8** fields in floating point. Whenever a value is stored into an **m8** field, Sculptor adds 0.5 and then stores the "floor" value, thus removing any fractional portion. This operation is not, of course, performed on **r8** fields. When performing multiplication and division on **m4** or **m8** fields, bear in mind that the values are stored in the lower currency unit.

Field format

A format may be attached to a field and is used to determine the precise way in which the field is printed and displayed. If no format is specified, a standard default is applied to suit the field's type and size. Any format defined in the data dictionary may be overridden within a screen form or report/batch language program.

Formats apply differently to alphanumeric, numeric and date fields.

ALPHANUMERIC FIELDS

For **alphanumeric** fields data is normally output precisely as input, the number of characters equalling the field size. The following special characters convert the input and/or output as shown:

e	suppress echo of input characters (screen form only)
l	forces lower case on input
u	forces upper case on input
s	no leading spaces on printing (**sagerep** only)
t	no trailing spaces on printing (**sagerep** only)
r	remove trailing spaces when the field is used in **get, put** and **print** (**sage** and **sagerep**)
n	null terminated field (see the **get** command for details)
m+ *format*	When a numeric value is assigned to this alpha field, treat it as a money value with two implied decimal places and use this *format*.
d+ *format*	When a date value is assigned to this alpha field, treat it as a date (day number) and use this *format*.
+ *format*	When a numeric value is assigned to this alpha field, use this *format* instead of the default format.

Following the character(s) with a plus (+) sign will force conversion when assigning a value to the field. Examples of the use of formats on alphanumeric fields follow:

ut	Force upper case on input, remove trailing spaces, assignments are not affected.
l+	Force lower case on input and assignment.
m+###.##	Format the field when assigned as a money type using the shown format
d+dd/mm/yyyy	Format the field when assigned as date type using the shown format
un	Force upper case on input, spaces are preserved, field is null terminated after the last character entered (not padded to field size with spaces).

NUMERIC FIELDS

For **numeric** fields the output width equals the number of characters in the format specification. The following format characters have a special meaning, other characters in the format are output unchanged.

#	Designates a position where leading zeros are to be space filled.
0	Designates a position where leading zeros are to be printed.
*	Designates a position where leading zeros are to be filled with asterisks.
,	Designates a comma to group significant digits.
.	Position of decimal point

A numeric format is processed from right to left. Once digit selection has begun (the first # , * . or 0), the first non-special character terminates the numeric part of the format. If the last digit designator was # * or 0 , all further characters in the format are floated right and leading spaces, asterisks or zeros are used to make up the required width. Examples of the use of numeric formats is shown below:

##,###.##	Showing 9 characters, zero is shown as 0.00 unless zeros were suppressed when the program was compiled.
####0.00	Zero always displays as 0.00 regardless of zero suppression being applied.

£ ###,##0.00	Floating £, zero is always shown as £ 0.00 On most systems, £ and # are the same ascii character code and a space must be placed between the # used as £ and the # which is the most significant digit. The space causes Sculptor to treat the first # as a currency symbol.
$ 000,000.00	Fixed $, as field is always zero filled

DATE FIELDS

Date formats may use the characters **d**, **m** and **y** to designate day, month and year respectively. The year portion may be two or four digits or may be omitted altogether. However it is generally better to omit date formats and allow the default value to apply. By doing this the program becomes independant of date format and may be run without recompilation in other countries. For this reason it is wise to design screen forms and reports on the assumption that dates require space for a four-digit year. The default date format may be altered in the run-time interpreters **sage** and **sagerep** using the program **lcf**. Example date formats are shown below:

dd/mm/yy	format using 2 digit year, 1900 added to year on input
dd.mm.yyyy	format with full stops using a four digit year
mm/dd/yyyy	put the month first
dd/mm	not valid for input, display only, no year shown
no format	use the default format (configurable by **lcf**)

Default formats

If a format is not specified for a date or numeric field, a default is applied as follows:

Type & Size	**Default format**
d4	That set with **lcf**.
i1	###
i2	#####
i4	#######
m4	#######.##

m8	#######.##
r8	#######.##

There is also a default date format set in the compilers **cf** and **cr** and in **describe** itself. The only purpose of the date format in these programs is to indicate how date constants are to be interpreted in the program code and validation lists.

Field validation

Input data is always automatically validated according to field type and size, both in the screen form and the report/batch languages. Further validation within the screen form language may be defined by attaching a validation list to a field description. This will ensure that only valid data may be input by the operator. Any data valid for the type and size of the field may be assigned directly to that field. Cross-field validation is also possible by programmed testing of data input using the screen form or report/batch language.

AUTOMATIC VALIDATION

Integer fields

Only integers within the range appropriate to the field size are accepted.

Money fields

For **m4** and **m8** fields, amounts must be whole numbers or fractions with exactly two digits following the decimal point and must be within the **m4** or **m8** range.

Date fields

Day, month and year must be input in the order specified in the date format, but most punctuation characters are accepted as separators regardless of the the one used in the actual format. If a two-digit year is input then 1900 is added to the year. Input dates are fully checked with proper consideration for leap years, although a correction for the loss of eleven days in September, 1752 is not made.

VALIDATION LISTS

In addition to automatic validation, it is possible to attach a validation list to a field. The list may consist of individual values and ranges of values. Items in the list are separated by commas with ranges separated by hyphens. If used, input to the field will only be accepted if the value entered is included in the list.

Validation lists are not checked by the report/batch language.

Validation lists may also allow **no input** to the field as a valid entry. This is done by using two consecutive commas. Validation lists may not be overridden within a screen form program.

Alphanumeric data is validated by comparing the input text for equality with single items in the validation list and on a greater than or less than basis against a range. Although input data is padded with spaces to fill the length of the field, these are ignored for comparison purposes, so care must be exercised with alphanumeric validation lists. For example, if a two-byte field is to begin with a letter in the range A-M, it is not sufficient to validate the field **A-M** since **MB** will be considered invalid. Specifying **AA-MZ** overcomes this problem.

Examples are shown below for each type of field.

Alphanumeric fields

```
,,y,n
MR,MRS,MISS,MS,DR,SIR
,,A-M,X,Z
```

Numeric fields

```
0-100
,,0,100-999
11-19,21-29,31-39
,,-500--100,100-500
```

Money fields

```
,,0-9999.99
,,0-999999
1-99,500-599
```

Date fields

```
,,1/1/80-31/12/99
13/3/1990-13/4/1990
```

On displaying a validation list for a date field, **describe** will show the year as four digits.

On-line help

When a field is input in the screen form language an optional line of help text may be presented either automatically using the **autohelp on** command or when the user presses the HELP key. This on-line help may be defined for each field in the record as appropriate. The on-line help for a field may be overridden within a screen form language program.

Field comment

When a data dictionary is printed using the **pdes** utility program, the comments field may be shown. The comment entry provides information about the field, for documentation purposes, which may otherwise not be obvious from the field name or heading.

Hide from SQL

When the SQL system is used to produce ad-hoc enquiries from the file, a field may be hidden from access. This may be used to preserve the security of sensitive data while allowing users to produce reports with **sql**, **eql** or **fql**.

Reverse Sign (SQL)

When producing ad-hoc reports using SQL, a field may be defined which, although containing a positive or negative value, should have its sign reversed for the purposes of the enquiry, eg. if a field contained 100 then it would be shown and totalled as -100 and vice-versa. This is used to correctly show and total amounts which should be treated as the opposite sign, ie, debits in a bank account.

Command line syntax

The command line for **describe** has the following format:

describe [**-q**] [*filename*]

The *filename* is the name of the data dictionary file that is to be edited. The **.d** extension is automatically added.

The **-q** flag selects quiet mode so that the program name and date are not displayed on exit from **describe**.

The data dictionary file is used by the compilers **cf** and **cr**, by the **newkf** utility and by the **SQL** suite **sql, fql** and **eql**.

If changes are made to the structure of a data dictionary file then all programs that access that file must be recompiled to incorporate the changes. The **reformat** utility *must* be used to preserve existing data if the size or field order of the record is changed. If changes to the size or field order have been made and there is no existing data or the data need not be preserved, the **newkf** utility can be used to re-initialise the file (see chapter 10 for details on the **reformat** and **newkf** utilities).

EXAMPLE

```
describe
describe newfile
describe -q ledger
```

Using describe - Menu options

The describe menu is shown on the right hand side of the screen when the program begins. A description of each of the options on that menu follows.

While the cursor is positioned on any of the menu options, the current descriptor line may be changed using the plus (+) and minus (-) keys.

Edit

This selection is active on entry to **describe** and allows changes to all fields already recorded and appending of additional fields. The following keys can be used in **Edit**. As these keys are defined in the vdu parameter file, they may differ slightly on some vdus and systems.

Key	**Description**
INS	Toggle insert/overtype
PGUP/PGDN	Page up and down
ENTER	Next field
TAB	Next field
SHIFT-TAB	Previous field
ARROW KEYS	Field movement control
ESC	Return to menu
Ctrl-Z	Delete to end of field
Ctrl-X	Abandon changes
F1	Help system
F2	Copy field name to heading
F3	Quick help
F4	Expanded mode
F10	Return to menu

Insert

Inserts a blank field above the current position. Field data is entered in the expanded display. If no valid field data is entered, or **ESC** is pressed, the insert is abandoned. Pressing **F10** at any time or **RETURN** at the last field exits and inserts the new descriptor.

Delete

Deletes the field at the current position.

Undelete

Recovers the last deleted field at the current descriptor line position.

Move

The order of the fields in the descriptor file may be changed. The current field is selected and may be moved using the up and down arrow keys. Pressing **ENTER** completes the move.

IMPORTANT NOTE : Moving a key field into the data field area, or a data field into the key field area will change the field type from key to data or from data to key respectively.

WARNING : Adding, deleting or changing the order of any fields will require the reformatting (see **reformat** in chapter 10) of any existing data and the recompilation of all existing programs which use the data file.

Go to

Go directly to a specified line by entering the line number. Pressing **ESC** abandons **Go to**.

Key

Toggles the current field from Key(**K**) to Data (**D**). The current field must be either the last key field or the first data field. Use the **Move** option to place the field if necessary.

Changing the key fields will require the reformatting of any existing data and the recompilation of of all existing programs which use the data file. Existing programs may also have to be modified to correctly use the new key fields.

Title

A title is stored with the descriptor file and is used for documentation purposes. This option allows entry and amendment of that title, together with four lines of comment text.

Files

Enters the file management system. Sculptor descriptor files anywhere on the system may be viewed and the path table may be edited. See page 3-14 for further details.

Help

Displays the help menu and allows selection from a range of help topics. Press **ESC** to return to the **describe** menu.

Clear

This option clears all current fields from the workspace. Descriptor files stored on disk will remain unchanged.

Load

Enters the file management system. The file selected will be loaded for editing. The path table may also be edited. See **File management** on page 3-14 for further details. If loading a file will cause descriptor information to be lost, you will be asked if the current file is to be saved. Answering **N** here will allow you to select a file, but will give you another chance to save the changed descriptors just before the new file is loaded.

Save

The current descriptor fields are saved to disk using the current file name. If a **Clear** command has been executed or a file has not previously been loaded, a file name (without the **.d** extension) must be entered.

If a new file name is given, it will over-write an existing file of the same name after giving a warning that the existing file will be lost.

Quit

Exits **describe**. If changes have been made to the current descriptors since the last **save**, you will be prompted to save the changes.

Don't forget to use **reformat** on your data files and recompile your programs if any of the following changes have been made:

changes made	reformat	recompile
insertion, deletion or move field	yes	yes
type or size of field	yes	yes
name of field	no	yes
number of fields in key	yes	yes
heading, format or validation	no	yes❶
on line help text	no	yes❶
all other changes	no	no

❶ the program will still operate without recompilation, but the changes made will not be incorporated until the program is recompiled.

If the data file contains no records that need to be maintained, **newkf** may be used instead of **reformat** - but this will create an **EMPTY** data file.

File management

The file management system provides a method of scanning the data dictionary files on the system and selecting a file to view or load. When entered from the **Files** option on the menu, descriptor files may be viewed, but the current descriptor file is unchanged. Entry via the **Load** option on recalls the selected descriptor file for editing with **describe**.

In either mode, the functions available are the same. The current directory may be changed using the **F2** key, files in other directories may be selected with the aid of the Path Table. Using this, up to 14 directory paths can be stored and then scanned using the **F5** and **F6** keys. A complete key summary follows:

Key	**Function**
ARROW KEYS	Select a descriptor file from the list shown.
F1	Display further help list of available keys.
F2	Change to a new directory.
F3	Manually enter the name of the file to select.
F4	Edit path table
F5	Next path on path table.
F6	Previous path on path table.
F10	Return to the menu. In the load option, the file will NOT be loaded.
PGUP	View the previous page of files.
PGDN	View the next page of files.
ENTER	Return to the menu. In the load option the file will be loaded.

The Path table

When editing the **path table**, up to 14 paths may be entered. These should be the directories where your system descriptor files are located.

The keys available while editing the path table are:

Key	Function
A	Add or amend path details. Enter the line number to change.
D	Delete a path table entry. Subsequent entries are moved up.
F10	Return to the file management system.
ESC	Return to the file management system.

Paths are stored in a file called **paths.sc**. The environment variable **PATHSDIR** may be set to specify where the **paths.sc** file will be found. If this variable is NOT set, the current working directory will be used to read and write the file.

IMPORTANT NOTE. The use of relative paths within the path table *may* result in the paths being invalid if the **paths.sc** file is read while in a different directory. Always use absolute paths within the path table for consistency.

Error messages

Too many file names

Only one file name may be specified on the command line.

File name is too long

File name is greater that 35 characters.

Help files not found

On running the help system the help files could not be found. Ensure that the **SCULPTOR** environment variable points to the location of the Sculptor directory and that the help file **des.hlp** is in the **HELP** subdirectory.

Can't find file!

The file name you specified could not be found.

Field name must be entered

Each field must have a unique name.

Bad characters in field name

Only the characters a-Z, 0-9 and underscore may be used in field names.

First character in field name must be alpha

The first character of the field name must be a letter.

Field name is already used

The name you entered is not unique in this file.

First character cannot be " or '

Field names cannot use these characters.

Bad type - must be a,i,r,m,or d

Valid field types are alphanumeric (a), integer (i), real (r), money (m) or date(d).

Bad dimension

Dimension value is zero or lacks closing parenthesis.

Hide - must answer Y or N
Sign - must answer Y or N

A Y or N response is expected.

Size not specified

A size must be specified for each field.

Size must be numeric

A numeric response is expected.

Alphanumeric range is 0-255

Negative values or values greater than 255 are not allowed.

Valid integer sizes are 1, 2 and 4
Real must be 8 bytes
Valid money sizes are 4 and 8
Date must be 4 bytes long

The size of the field does not match the type entered.

Too many fields

The limit of 1000 fields has been exceeded.

Can't leave gap in key structure

An attempt has been made to convert a key field in the middle of the key to a data field. Move the field to the data area and try again.

Must be after last key entry

A data field can only be changed to a key field if it is placed immediately after the last key entry. Move the field and try again.

Keys must start at field 1

The first field must always be a key field.

No keys have been specified

The data dictionary cannot be saved without a valid key.

Dimensions not allowed in key fields

Move the field to the data area if dimensions are required.

Error! must have at least one field

Cannot save empty data dictionary file.

Error! record length must be greater than 2

The total record length must be greater than 2 bytes.

Can't open output file

The data dictionary file cannot be opened. Check your access permissions and/or disk space.

Validation list too long to save

When using a **d4** type field, the date entries in the validation list are expanded to full 4-digit year entries and, when expanded, will take more space than is available. Reduce the size of the validation list.

CHAPTER 4 PROGRAM GENERATORS

This chapter describes the report/batch and screen form program generators **rg** and **sg**.

Contents **Page**

Introduction

The normal method of creating a screen form or report/batch language program is to create it using an editor. However, the Sculptor program generators can create and compile a default screen form or report/batch language program in seconds from the data dictionary file, leaving a source file which may be edited and amended as required.

The program generators operate in two distinct modes. In silent mode (the default) they read the specified data dictionary file and produce either a screen form or a report/batch language program from that file. In active mode the program generators display a screen form which allows the programmer to select the fields that will be displayed/printed. The report title, field headings and formats may be amended as required.

Both program generators are similar in operation, but differ slightly in the fields available on the screen in active mode and, of course, in the output code produced.

The program produced can be used as a normal Sculptor screen form or report/batch program and the source file can be edited and recompiled as required.

Output code

The code that is created by the program generators is well suited to the task it has to perform. Subroutines are used for common routines, such as display and input to screen forms, and headers and totals in report/batch programs. When dealing with array fields on screen forms, each is placed on the scroll area and code is generated to display and input all elements of each array, independent of size. Although this technique is capable of dealing with a very wide variety of situations, if the code is to be used as the basis for an application program it will probably require tailoring for the application.

Command line syntax for rg

rg [**-a**] *filename*

Automatically generate a report program using record descriptions previously defined with the data dictionary editor, **describe**. *filename* is the name of the data dictionary file (with a **.d** extension automatically added) which is read to produce a Sculptor report/batch language source file. This will have the name *filename*.**r** and will then be automatically compiled using the compiler **cr** to *filename*.**q**. The resulting program can be executed by typing **sagerep** *filename.*

WARNING: Any files named *filename*.**r** or *filename*.**q** that already exists in the current directory will be overwritten.

The generated file will contain all the Sculptor commands necessary to produce a report of the fields from the data file, correctly formatted with headings and totals. This source file may then be modified using a standard text editor.

Used with the **-a** option, **rg** will display a full screen form which may be used to select the fields to be displayed, and the formats to be applied when printing those fields. The fields from the data dictionary file will be listed in turn and you may choose to select or omit them from the generated report and adjust the heading and format of the displayed field.

Screen fields for rg

Program name	The source file will be saved using this name.
Report title	The title that will be displayed centred at the top of the report.
K/D	The data type of the named field (Key or Data).
Field name	The name of the field.
T & S	The type and size of the field. Some fields may be too large to fit on a single line of the printer. These should be omitted from the generated display, or **rg** will report an error and will not create the program.
y/n	Include (y) or omit (n) the field from the generated report.

Tot	On numeric fields, a total may be printed, enter Y or N.
Heading	The heading that will be used when the field is printed.
Format	The format that will be used when the field is printed.

When you have finished selecting fields from the data dictionary file, you will be asked if you want duplicate key values suppressed. If you have not placed all of the key fields on the report, there may be duplicate values shown on the report, even though the full key is not, of course, duplicated. Selecting this option will suppress the printing of these duplicates and generally tidy the report.

Error messages for rg

Only one filename allowed on the command line

sg can only operate on a single data dictionary file at a time.

No file name given

Please enter a file name on the command line.

Can't fit the fields of *filename* on the report

All of the fields of *filename* will not fit on the report. Use the **-a** option to select the fields that *will* fit on the report.

Can't execute cr

The compiler **cr** could not be found. Check that the development system has been fully installed and that the **cr** program is available.

Can't create configuration file

sg requires that a Sculptor data file called **gen_cont** resides in the main Sculptor program directory. This error occurs if that file could not be found. Check the location of this file.

Can't execute sage

The **-a** option uses the **sage** screen form interpreter to present the screen form for input. Please check the location and availability of the **sage** program.

Can't re-open configuration file

The configuration file **gen_cont** could not be re-opened.

Mismatch on configuration file

The configuration file **gen_cont** is damaged.

Can't open *filename*

The named file could not be opened. Check access permissions

Input error on *filename*

An operating system input error has occurred. Check the integrity of the file.

Command line syntax for sg

sg [**-a**] *filename*

Automatically generate a screen form program using record descriptions previously defined with the data dictionary editor, **describe**. *filename* is the name of the data dictionary file (with a **.d** extension automatically added) which will be read to produce a Sculptor screen form language source file. This will have the name *filename*.**f** and will then be automatically compiled using the compiler **cf** to *filename*.**g**. The resulting program can be executed by typing **sage** *filename*.

WARNING: Any files named *filename*.**f** or *filename*.**g** that already exists in the current directory will be overwritten.

sg will design the screen form so that all the fields will fit on the screen, if possible, so may take some time to perform the optimisation if there are a large number of fields. If the fields cannot be fitted on the screen, **sg** will abort the process with an appropriate error message.

The generated file will contain all the Sculptor commands necessary to maintain the data file, with options to insert, find, next, match, amend, delete and exit. This source file may then be modified using a standard text editor.

Used with the **-a** option, **sg** will display a full screen form which may be used to select the fields to be displayed, and the formats to be applied when displaying those fields. The fields from the data dictionary file will be listed in turn and you may choose to select or omit them from the generated screen form and adjust the heading and format of the displayed field.

Screen fields for sg

Program name	The source file will be saved using this name.
Screen title	The title that will be displayed centred at the top of the screen.
K/D	The data type of the named field (Key or Data).
Field name	The name of the field.

T & S	The type and size of the field. Some fields may be too large to fit on the screen. These should be omitted from the generated display, or **sg** will report an error and will not create the program.
y/n	Include (y) or omit (n) the field from the generated report.
Tot	Used by **rg** only.
Heading	The heading that will be used when the field is displayed.
Format	The format that will be used when the field is displayed or input.

Error messages for sg

Only one filename allowed on the command line

sg can only operate on a single data dictionary file at a time.

No file name given

Please enter a file name on the command line.

Can't fit the fields of *filename* on the screen

All of the fields of *filename* will not fit on the screen. Use the **-a** option to select the fields that *will* fit on the screen.

Can't execute cf

The compiler **cf** could not be found. Check that the development system has been fully installed and that the **cf** program is available.

Can't create configuration file

sg requires that a Sculptor data file called **gen_cont** resides in the main Sculptor program directory. This error occurs if that file could not be found. Check the location of this file.

Can't execute sage

The **-a** option uses the **sage** screen form interpreter to present the screen form for input. Please check the location and availability of the **sage** program.

Can't re-open configuration file

The configuration file **gen_cont** could not be re-opened.

Mismatch on configuration file

The configuration file **gen_cont** is damaged.

Can't open *filename*

The named file could not be opened. Check access permissions

Input error on *filename*

An operating system input error has occurred. Check the integrity of the file.

CHAPTER 5 THE SCULPTOR SCREEN PAINTER

This chapter describes the operation of the screen painter for screen form language programs, **sp**.

Contents **Page**

Introduction

The screen painter reads a Sculptor screen form language source file and allows fields to be placed on any of the eight screens available. When an existing source file is read, screen fields are drawn on the screen and may be moved, changed or deleted as required. Fields from any file may be placed on the screen as may any temporary fields (ie, those defined with **!temp**). New temporary fields may be added and placed on the screen. The scroll area, field delimiters and form title may also be amended as required.

When the source file is saved, any changes made are written back and the source file may then be compiled in the normal manner.

As the source file may contain conditional compilation statements for the Sculptor pre-processor, **spp**, the screen painter performs the first phase of that same conditional compilation so that the sections of code that are being amended are the sections that would be compiled. Manifest constants may, therefore, be defined on the **sp** command line to provide the same level of control provided by **spp**.

When the screen painter begins, it reads the filename specified, if any, and displays the screen as it would look to the user. An additional status line at the bottom of the display and a menu on the right hand side are also displayed. Each of these menu options is detailed in the following sections.

Move fields

Allows movement using the arrow keys to any location on the screen. The cursor location is marked with a plus (+) sign. Fields displayed may be picked up and dropped at any location using the **SPACE** bar.

Movement of the field onto the start line of a defined scroll area will display the field vertically to the depth specified as the scroll area depth, with the field heading shown centred above.

If a scroll field is at the right edge of the screen, you will not be able to move it vertically out of the scroll area as doing so would display the field horizontally off the edge of the screen. Move the field to the left first.

The functions available will depend on whether the cursor is over a blank area of the screen, over a field which has not been selected or moving a field which *has* already been selected.

KEYS AVAILABLE WHEN CURSOR IS OVER A BLANK AREA

F1	Help on key usage.
F3	Select a screen to activate/deactivate.
	There are 8 available screens which may be active at any time. Fields will only be deleted from active screens. Similarly, new fields added will be added <u>only to active screens</u>. Deactivating a screen will remove, from display, all fields belonging to that screen and activating a screen will display all fields on that screen.
F7	Move the scroll area (and all scroll fields) up.
F8	Move the scroll area (and all scroll fields) down.
F10	Return to the menu.

KEYS AVAILABLE WHEN CURSOR IS OVER A FIELD

F1	Help on key usage.
SPACE	Pick up the current field (field name is displayed lower left).
F4	Display information about the current field. The information that is displayed is shown below:

Field position	The row and column position for the field.
Screens	The screens on which the field is placed.
Source	Whether the field is from a file (**!file**) or is a temporary field (**!temp**). If from a file, the filename is shown.
Name	The field name
Heading	The field heading in use.
Type & size	The type and size of the field and the number of dimensions (if any).
Format	The format being applied to the field.
Key or Data	Defines whether the field, if from a file, is a key or data field on that file.

F5	Delete the field from the display of all *active* screens.
F10	Return to the menu.

KEYS AVAILABLE WHEN MOVING A FIELD

SPACE	Drop the field at the current location.
TAB	Shift the field data area right - the heading is not moved.
Shift-TAB	Shift the field data left - the heading is not moved.

Add fields

Allows fields to be imported from a selected descriptor (**.d**) file.

Using the file management system, first select the descriptor file, then fields from that file to place on the display.

Use the arrow keys to select a file, and press **ENTER** to accept that selection. See page 5-10 for details on selecting files.

When a file has been selected, the fields in the file are displayed on the left hand side of the screen. Select or deselect a field using the **ENTER** key. As a field is selected, it's name appears in the placement list on the right section of the screen and an asterisk is shown to the left of the field name. All of the fields shown in the placement list will be transferred to the screen form in turn when you press **F2**.

When selecting the field names, **F10** may be pressed to abandon the entire option and return directly to the main menu.

Having pressed **F2**, the fields appear in turn on the display and you may place them by moving to the desired location using the arrow keys and pressing the **SPACE** bar. The next field on the placement list will then appear. After the last field has been placed you will return to the menu. If no fields have been selected you will return directly to the menu.

IMPORTANT NOTE. *The field declarations will be written to the source file in the order selected.*

A list of the currently active screens are shown in the lower right corner. All fields selected will be placed on all currently active screens in the order shown in the placement list.

Fields that appear highlighted are already displayed and may not be selected again.

The default heading set up in **describe** is automatically used, but this may be changed using the **Field Declarations** option from the main menu (see page 5-6 for further details).

Temp declarations

Allows maintenance of the temporary fields which are placed in the source file when it is saved. Temporary field declarations may be added, deleted and amended.

The field declarations are shown in a list with a menu of actions shown below. Select a declaration using the up and down arrow keys and select an action to perform using the left and right arrow keys.

The declarations are shown in the format:

!temp *fieldname*,[*heading*],*type&size*,[*format*]

where *fieldname* is the name of the field, *heading* is the heading which will be used as a default for this field (blank for no default heading), *type&size* is the field type and storage size, and *format* is the format specifier (see page 3-5 for further information).

MENU ACTIONS

Update	Amend the details of the selected declaration
Insert-Above	Insert a new declaration before the current position
Insert-Below	Add a new declaration after the current position
Delete	Delete the current declaration
Next-page	View the next page of declarations
Prev-page	View the previous page of declarations
Menu	Return to the main menu

When editing field headings, a '<' symbol is used to indicate the end of the field data. This is useful if trailing spaces are required.

When editing a declaration, the message "Linked to a field" will be displayed if the temporary field is used by a screen field declaration. If this is the case you will not be allowed to change the field name or delete the field.

Field declarations

Allows maintenance of the screen field declarations that will be placed in the source file when it is saved.

The screen field declarations are shown in a list with a menu of actions shown below. Select a declaration using the up and down arrow keys and select an action to perform using the left and right arrow keys.

The declarations are shown in the format:

+ *fieldname*, [*heading*], *row*, *column*, [*format*]

where *fieldname* is the name of the field, *heading* is the heading which will be displayed, (a space indicates no heading), *row* and *column* are the screen positions of the first character in the data field, and *format* is the optional format specifier which may over-ride the default for this field (see page 3-5 for a list of available formats).

Also shown is a list of the screen number(s) on which the highlighted field will appear and either a letter, **T** or **F**, to mark whether the declaration is for a temporary or file field.

A menu is presented at the bottom of the screen. The options available on the menu are shown below.

Update	Amend the details of the selected field declaration
Insert-Above	Insert a new declaration before the current position
Insert-Below	Insert a new declaration after the current position
Delete	Delete the current declaration
Next-page	View the next page of declarations
Prev-page	View the previous page of declarations
Menu	Return to the main menu

In **Update** or **Insert** options, these keys are active:

F3	Abandon the current operation
F10	Exit and save this declaration

No two screen field declarations may refer to the same record field or temporary field.

If an attempt is made to insert a field name from a file which has not been used in this screen form previously, the message "**needs !temp or !file declaration**" is displayed. Select the field using the **Add fields** option or create a **!temp** declaration for the field using the **Temp declarations** option.

When editing a field heading a '<' symbol is used to indicate the end of the heading data. This is useful if trailing spaces are required.

Edit title

The screen title is taken from the first line of the source file. This option allows amendment of that title. The title entered is centred and highlighted (using the **start page title** (entry 23) sequence from the vdu parameter file) on the top line of the display.

Update scroll area

The start line and depth of the scroll area may be defined. Only one scroll area may be declared for each screen form program. The total of start line and depth may not exceed the depth of the screen.

Any fields displayed on the start line are duplicated vertically the number of times defined by the scroll depth. This allows multiple copies of the same field to be displayed. The scroll area applies to all screens - not just the active screen.

Change delimiters

The characters which are used to delimit the field data may be changed with this option. Each delimiter may be any *single* character. Pressing **ENTER** alone leaves the delimiters as they are. The default delimiters used are left and right square bracket "[ ]", however, if the loaded source file contains a **!box** declaration, the characters defined there are used.

Input field delimiters are applied universally throughout all screens in the screen form program.

Change screen size

A standard and extended display depth and width may be defined in the vdu parameter file for a terminal. This option allows you to switch between standard and extended width or depth.

Select **W** to change width or **D** to change depth.

This option has no effect if your vdu does not support extended width and/or depth.

View file descriptors

Enter the file management system in READ ONLY mode, to view the descriptors of any data dictionary (**.d**) file on the system. All of the functions of file management are available including the ability to edit the **Path Table.**

Directory of files

Used to view the files in the current directory, three types of file may be listed.

Sculptor data dictionary files with a **.d** extension are displayed with the file title shown to the right of the file name. This is designed to aid identification of the file.

Screen form source (**.f**) files may also be shown, as may the screen form compiled code files (**.g**).

Load source file

Loads a source file into the workspace and displays the fields defined within that file. The vdu width and depth will be adjusted to meet the requirements of the loaded source file.

If you have not saved any changes made to your current file - you will be prompted to do so before the new source file is loaded.

This option enters the file management system where you may select from any of the source files in the current directory or use the facilities of the **Path Table** to select source files anywhere on the system. See page 5-10 for further details of how to use these facilities.

The screens that were active when you selected **Load source file** will remain active when the new file is loaded. This may result in none of the fields from that source file being displayed.

During the loading of a source file, **sp** may find errors within the file and will report them accordingly. If errors are found, the file will NOT be loaded. Use your normal editor to correct the stated errors before attempting to re-load the source file.

Save changes

Saves any changes to the current source file. If this is a new source file, you will be asked to enter a name for the file (without the **.f** extension). If a file by that name already exists you will be prompted to over-write it or enter another name for the new file.

All declarations are written back to your source file and may be edited accordingly.

IMPORTANT NOTE : The **sp** program is a tool to help ease the work involved in creating good screen layouts, but it does not generate the program code to use that layout.

An existing source file may be loaded and the layout changed without affecting any **sage** commands within that source file as **sp** will only change **declarations** as follows:

Change made in sp	**Declarations changed**
fields from new file placed	!file, +, !screen
existing fields moved	+
existing fields deleted	+, !screen
screen size changed	!width, !depth
scroll area changed	!scroll
title line changed/added/deleted	first line of file adjusted
field delimiters changed	!box
temporary fields added/amended	!temp
temporary fields placed on display	+, !screen

If multiple screens are used and fields are placed when more than one screen is active, some fields will appear simultaneously on more than one screen. When the declarations are written back to the source file the **!screen** declaration for these fields will be written first and will declare all the screens on which the fields which follow appear. ie,

```
!screen 1,3,5,8
... field declarations
```

Include files

Information which is originally read from an include file is written back to the same file when the source file is saved. When new declarations are written, however, they are written to the include file only if the new declaration immediately follows a declaration from that file. This may not always be exactly what you require, so if in doubt, check your source carefully.

The old source file is renamed as *filename*.**bak** when the changes are saved.

Command line syntax

sp [**-q**] [**-D***symbol*[=*text*]]... [*filename*]

filename is a Sculptor screen form source file with a **.f** extension (the extension is added automatically).

When used with the **-q** switch, **sp** will exit quietly and NOT display the version number header box.

The **-D** option allows manifest constants to be defined to have either a null value (without the =*text*) or with the value *text* which will be used in the program. This option is to enable the screen painter to correctly include or exclude sections of code based on manifest constants defined on the command line in the same fashion as the Sculptor pre-processor, **spp**.

The screen painter directly modifies Sculptor screen form source files. Include files declared within the source file using **!include** declarations are also read and may be modified if declarations read from those files are changed.

File management

The file management system may be entered in one of two modes. When entered from the **View Files** option on the main menu, data dictionary files may be viewed. Entry via the **Load File** option will load the selected source file for editing with **sp**.

In either mode, the functions available are the same. The current directory may be changed using the **F2** key, files in other directories may be selected with the aid of the **Path Table**. Using this, up to 14 separate directory paths can be stored and then scanned using the **F5** and **F6** keys. A complete key summary is shown below:

ARROWS	Select a descriptor file from the list shown.
F1	Display further help list of available keys.
F2	Change to a new directory.
F3	Manually enter the name of the file to select.
F4	Edit the path table.
F5	Next path on path table.
F6	Previous path on path table.
F10	Return to the menu. In the load option, the file will NOT be loaded.
PGUP	View the previous page of files.
PGDN	View the next page of files.
ENTER	Return to the menu. In the load option, the file will be loaded.

When editing the Path Table, up to 14 paths may be entered. These should contain directories where your system source files are located.

The keys available while editing the path table are:

A	Add or amend path details. Enter the line number to change
D	Delete a path table entry. Subsequent entries are moved up

F10	Return to the file management system
ESC	Return to the file management system

Paths are stored in a file called **paths.sc**. The environment variable **PATHSDIR** may be set to specify where the **paths.sc** file will be found. If this variable is NOT set, the current working directory is used to read and write the **paths.sc** file.

IMPORTANT NOTE : The use of relative paths within the path table *may* result in the paths being invalid if the **paths.sc** file is read while in a different directory. For consistency, always use absolute paths within the path table.

Error messages

These errors prevent a source file from being loaded. Many of these errors are preceded with the message:

```
    Error in filename at line [ (!include) ] nnn :
```

Where the **(!include)** will be shown to identify that the error was in an include file and where *nnn* is the line number where the error occurred.

Screen size set too large : max size 25 x 132

A **!width** or **!depth** statement was encountered which exceeds the above limits. The source file will NOT be loaded.

Fieldname too wide for the screen

The named field would not fit on the screen and has not been placed.

Fieldname adjusted

The named field would not fit on the screen at the intended position and the position has been adjusted to allow it to fit.

Too many files specified

A maximum limit of 32 **!file** and **!cfile** declarations is imposed.

Can't open file *filename*

The file name specified with a **!file** or **!record** declaration could not be read to recover field names. Check the declaration in your source, then check the file data dictionary (.d) file exists in the named location.

Too many SCULPTOR file fields

A maximum limit of 1000 fields is imposed by **sp**. This number is calculated by adding the number of fields in every file declared in a **!file**, **!cfile** or **!record** declaration, regardless of whether any of those fields are actually displayed.

Too many include files

A maximum of 20 **!include** file declarations are permitted. This includes nested **!include** declarations.

Can't find file

When selecting a file from the file management system, the named file could not be read. This error may occur if the file has been deleted by another user since the directory was read.

Syntax error in !file / !record declaration
Syntax error in !include declaration
Syntax error in !define declaration

A syntax error has been encountered in the named declaration.

Error in drawbox statement

The named declaration has a syntax error which should be corrected before the file may be loaded. This version of **sp** does not currently directly use **drawbox** statements, but does perform some syntax checking on them for future use.

Too many drawbox/hline/vline statements

Currently, **sp** supports up to 30 **drawbox**, **hline** or **vline** statements in a single screen form source file.

Missing screen number

A !screen declaration was found without a screen number.

Screen number must be non-zero

An invalid number was found in a **!screen** declaration.

Bad !define

A syntax error was encountered in a **!define** declaration.

Bad ! declaration

An unknown declaration was found.

Invalid !include path name

The path name recovered from the **!include** declaration is not in the correct format for this operating system.

Too many + field declarations

A maximum limit of 300 + field declarations is imposed. If the source file contains a very large number of + field declarations, it cannot be edited with **sp**.

fieldname : **field name not defined**

A + field declaration was encountered and the field name has not been previously defined.

Missing }

There is a missing left brace character in the source file at the specified line.

Too many !temp fields

A maximum limit of 200 temporary fields is imposed within sp.

fieldname : !temp name too long
fieldname : !temp header too long
fieldname : !temp format too long

A temporary field may have a maximum name length of 40 characters, a maximum header length of 80 characters and a maximum format string length of 30 characters.

Fieldname : dimension is zero

The named field has a zero (or invalid) dimension.

These errors may be experienced during normal operation of **sp**. They warn of adjustments to the displayed fields or give further guidance when information has been entered incorrectly.

Field too wide to adjust

When attempting to change the screen width from extended to standard, a field on the screen would exceed the bounds of the standard display width. The display width will not be changed. If the display width *must* be changed, the wide field will have to be deleted.

Field is in the extended region

When attempting to change the screen depth or width from extended to standard, a field was found in the extended region. Move the field to a location within the standard screen area.

Field too wide for the screen

The field you are trying to place on the screen is too large to be displayed.

Field already defined

When inserting a field manually within the **Field declarations** option from the menu, the name entered has already been defined and placed on the screen. A particular field name may have only one screen field declaration associated with it. If the field must be declared on several screens, activate all the screens on which it is to be placed **prior** to placement.

Field name is not a !temp or !file field

The name that you have entered for the current field declaration could not be found in the list of files or temporary fields which have been defined. If the field is from a file which has NOT been previously used, use the **Add Field** option from the menu to insert it.

Line number is off screen

The line number manually entered within a field declaration exceeds the maximum line depth for the current display size.

Field positioned too far left for heading

When manually placing fields within the **Field Declarations** option, the row and column positions must be entered. These will default to fairly sensible values, but if the heading is very long, they may not be suitable. Set the column position to an appropriate value for the size of the field heading.

Column number is off screen

The column number you entered would result in the field data being displayed off the edge of the screen.

Field name contains bad characters

A field name must start with an alphabetic character in the range a-Z and may contain any alphanumeric character and underscore. No other characters are permissible.

Name has already been used

A field of this name has already been defined as a temporary field or belongs to a field in a file which has been declared.

Cannot delete - linked to a field

The temporary field declaration that you have tried to delete is being used in a field declaration. The field declaration must be removed before the field declaration can be deleted.

Command line syntax error

The syntax given used to start **sp** is incorrect.

Linked to a field

You cannot change the name of a temporary field that has been used in a field declaration. If the name *must* be changed, delete the appropriate field declaration first then re-try.

Name MUST be given

A name must be entered for a temporary field declaration and for a field declaration.

Field size is <0 or >255

The alpha field you have specified must have a size in the range shown above.

Size must be 1, 2 or 4 for an 'i' field
Size must be 8 for 'r' fields
Size must be 4 or 8 for 'm' fields
Size must be 4 for date fields

The size entered is incorrect for the data type specified.

Unknown field type

Valid field types are **a, i, m, r** and **d**.