EXTERNAL REFERENCE SPECIFICATION

FOR

C180 SOURCE CODE UTILITY

NOS 170 Prototype Version

Submited: _____

Approved: _____

_____

_____

_____

Disclaimer:

This document is an internal working paper only. It is subject to change, and does not necessarily represent any official intent on the part of CDC.

REVISION DEFINITION SHEET

REV	DATE	DESCRIPTION
A	08/10/76	Version 2.0 _ original release.
B	11/15/76	Version 2.1 _ NOS conversion.
C	08/15/77	Update for release under 3.3 of SES Subsystem.
D	06/30/78	Version 3.0 - Re-direction.
E	12/04/78	Version 3.1 - Utility re-written in CYBIL, dependence on SES Subsystem removed and capabilities enhanced. ERS rewritten.
F	02/06/79	ERS revised in response to DCS comments.
G	05/11/79	ERS revised in response to DCS comments.
H	11/06/79	Temporary revision of ERS in response to DCS comments etc.
J	03/24/80	This revision describes the NOS 170 prototype implementation of the utility in particular. Inputs from DCS comments and design team meetings have been included.

1.0 PREFACE

1.1 SCOPE OF DOCUMENT

This document is intended to contain the information necessary for use of the NOS 170 prototype version of SCU by a terminal or batch user. Since there are some features which are of necessity operating system or machine dependent and because not all features which will appear in the NOS/VE implementation will be present in the prototype these two implementations are being described in separate documents. The revision bars shown in this document have been generated relative to the ERS For Source Code Utility DCS number ARH1766 revision G. That was the last ERS submitted to DCS which attempted to describe both implementations. An ERS describing the NOS/VE version of the utility will be submitted to DCS under another number.

1.2 APPLICABLE DOCUMENTS

Document	DCS Number
SES User's Handbook	ARH1833
NOS Reference Manual	
NOS/VE ERS - Command Interface	ARH3609
CYBER 180 System Interface Standard	S2196

2.0 INTRODUCTION

The purpose of this document is to describe the NOS 170 prototype version of a utility which will provide for introduction and maintenance of ASCII source data on the Cyber 180. While emphasis is placed on the control and documentation necessary in large development groups, it can be used by individuals maintaining private source libraries.

The utility described below includes an interactive editor allowing for interactive development of new source units or corrections to existing source units. It provides multi_deck and build capabilities, allows for grouping of source decks and correction sets into larger identifiable units and associates descriptive information with source decks and correction sets.

The commands described in this document make use of the System Command Language (SCL) syntax which has been defined for NOS/VE, the operating system that will execute on Cyber 180.

This document describes the implementation of this utility on NOS 170. The reader should be aware that some of the parameters described below are machine dependent and will require adjustment when the final version of this utility is implemented to execute under NOS/VE. The most common examples of parameter information that will change is the length of user names and file names. Where possible the NOS 170 version will be made to emulate the final environment.

2.1 CONCEPTS

This section introduces terms used throughout later sections of this document.

2.1.1 SOURCE LIBRARY

A source library is a file containing a collection of source decks. A source library will usually represent all or part of the source code for a product at a certain stage of development.

2.1.2 SOURCE DECK

A source deck is a physical collection of lines of source text. The text may consist of source code, documentation or any other type of source data. SCU text embedded directives may be intermixed with the source data. Each line of text will have associated with it a unique (with respect to the deck) line identifier which can be used to identify the line. A line identifier consists of an alphanumeric modification name and a sequence number relative to that modification as it applies to a given deck.

2.1.3 GROUP

A group is a name associated with a collection of decks to allow for the logically related decks to be extracted from a source library together or merely to document the fact that they are logically related. A deck may have up to 255 group names associated with it.

2.1.4 MODIFICATION

In this document the term modification will be used to mean the changes introduced to a deck or series of decks by an edit session. It is possible to have an edit session be considered a continuation of a previous one. Each modification has associated with it a unique user assigned name. This name will be used to form the line identifier of each source line introduced by the modification.

Modifications are stored on the source library by SCU in terms of their effect on the source decks to which they apply, rather than in the form of the original commands that introduced the change.

2.1.5 FEATURE

A feature is a name associated with a modification. Associating the same feature name with several modifications to the same or different decks allows one the ability to group modifications together to be applied or ignored as a group. For example all the modifications necessary to make a major change in an interactive interface to the system could be associated with a common feature name. A modification may have only one feature name associated with it.

2.1.6 STATES

Every modification has a specific state assigned to it. The state reflects the degree of maturity the modification has attained. The following states are defined:

0 - Experimental	Untested code. This implies that this should probably not be included in a build by anyone except the person actually working on it or someone wanting to check for a potential conflict.
1 - Developmental	Code which has been unit tested.
2 - Stable	Code which has passed regression tests.
3 - Verified	Code which has been found to perform as advertised.
4 - Released	Code which has been completely tested and approved.

A modification in state 0 may have its text or descriptive information altered. Also if the user has proper authority, he can raise the state of a state 0 modification.

Modifications in states 1, 2 or 3 may only be raised in state or lowered to state 0. No alteration of text or descriptive information is allowed for modifications in these states.

A modification in state 4 cannot be altered in text, description or state. In other words, errors in text lines introduced by a state 4 modification must be corrected under another modification name.

2.1.7 AUTHORITY

In this document the term authority will be used to mean a user name that is validated by SCU to perform certain manipulations on a source librbary. A source library has associated with it a specification of each of four authorities necessary to change the state of a modification. The authorities specified are for 1 - Developmental, 2 - Stable, 3 - Verified and 4 - Released. Each modification has a specified state which may include any of the above or 0 -

Experimental. A user must have the specified authority for a library (or a higher one) to raise the state of a modification to a given level or to lower it from that state to state 0.

2.1.8 CRITERIA FILE

A criteria file contains directives used to limit which features and modifications at which states are to be included when blocks of source data are expanded from a source library. Similarly decks or groups may be included or excluded through directives on a criteria file. In addition values may be declared in a criteria file which can be used to trigger conditional inclusion of blocks of source data when decks are extracted from a source library and expanded for a processor.

2.1.9 TEXT EMBEDDED DIRECTIVES

When a user wishes to prepare source data for a processor, SCU will expand the source under the control of text embedded directives some of which interact with the contents of a criteria file. Text embedded directives are intermixed with the source data. Text embedded directives are recognized as distinct from source data by the occurrence of a specific character in the first postion on the line followed immediately by a keyword. In this document asterisk is shown as the key character in all examples. Some examples of text embedded directives are *COPY and *ELSE.

2.1.10 NORMAL SEARCH ORDER

Some SCU commands working with decks allow the use of multiple source libraries. In locating a deck, the source libraries are searched in the order in which they are given in the list.

2.1.11 BASIC STATUS CONCEPTS

System Command Language (SCL) under NOS/VE will require all commands to have a status parameter. This parameter will be used to try to standardize the processing of error messages and abnormal responses from command processors. In the version of this utility which is implemented to execute under NOS on CYBER 170 systems the value of this parameter must be

2.0 INTRODUCTION
2.1.11 BASIC STATUS CONCEPTS

given as the name of one of the job control regeisters, R1, R2, R3 or EF. The named register will be set non-zero if an error is encountered while processing a command. If R1, R2 or R3 is given, the value entered into the register will correspond to the condition codes shown in the section on messages.

2.1.12 NAMES

Names used in SCU will follow the NOS/VE patterns. Valid characters to be included in a name are upper and lower case letters, the characters "#", "$", "@" and "_" (underline) and decimal digits. Names must not begin with a digit. Library names, deck names, feature names and group names will contain 1 to 31 characters. The only exceptions to the pattern will be modification names which will be restricted to a maximum of 9 characters in length. Modification sequence numbers will be 6 digits long and range from 1 to 262143.

2.1.13 STRINGS

Some parameters for the SCU commands are described as being strings. System Command Language syntax requires strings to be enclosed in single quotes.

2.1.14 LOCAL FILES

SCU assumes all files it is to deal with are local with the exception of those named on the BASE and RESULT parameters below. The user may specify the user name for permanent files, if different than his own, by including it as the second value in a value set. It is the user's responsibility to acquire access to other files prior to initiating SCU commands to use them.

The NOS 170 version of SCU uses a series of scratch files internally whose names begin with the string 'SESSC'. The user of the NOS 170 version should avoid making explicit use of files with names fitting this pattern.

3.0 COMMANDS

3.0 COMMANDS

The commands listed below are available in the SCU environment. Commands for the editor, text embedded directives and criteria file directives are described later. The commands listed here function as separate job control statements and are intended to use a syntax as similar as possible to that of the System Command Language (SCL) for NOS/VE. When accessed on NOS 170, these commands must be prefixed by the characters "scu.". The commands can also be bound together in a more intimate fashion as described below under command grouping.

Parameter defaults are assigned as follows. Required parameters where possible will have static defaults assigned initially such as B*SE for the name of the base source library. The commands listed here are presented in a manner so as to emphasize the parameters of the commands. Optional parameters are enclosed with square brackets. As with NOS/VE SCL commands, commas and/or spaces can be used to separate parameters. Parameters can be given positionally, by keyword=value or a mixture. In the latter case specification of a parameter by keyword "resets" the positional pointer to the following parameter field. Where available, shortened forms for the keywords are shown. An ellipsis (two or more periods) at the end of a line indicate that the command is being continued on the next line. On NOS 170 lower case letters appearing in strings on the first physical line of an SCU command will be capitalized. To avoid this unpleasant side affect of executing on NOS 170 a user should include strings which are parameters to SCU on continuation lines.

As on NOS/VE all lower case letters appearing inside names (and references to names) are translated to their upper case counterparts by the SCL interpreter.

The short form of each command is listed with the command.

3.1 SOURCE LIBRARIES

The following commands are available for creating and manipulating source libraries.

3.1.1 CREATE_LIBRARY : CRL

This command creates the header of a source library on the specified file and includes in this header the descriptive information about the library including the library name, key character for text embedded directives and user names to be assigned authorities.

```
create_library [name=<library_name>]
               [description=<descriptive_string>]
               [authority_1=<user_name>]
               [authority_2=<user_name>]
               [authority_3=<user_name>]
               [authority_4=<user_name>]
               [key=<char>]
               [version=<version_string>]
               [result=<file_name>]
               [status=r1|r2|r3|ef]
```

name : na : This specifies the name of the source library. If this parameter is omitted the name of the result file will be used.

description : de : This 1 to 256 character string is used to describe the source code maintained on this library.

authority_1 : a1 : This is the 7 character user name to have authority 1 (developmental authority) for this library. If parameters authority_1 through authority_4 are not given, the user name that the command is executing under will be used.

authority_2 : a2 : This is the 7 character user name to have authority 2 (stable authority) for this library.

authority_3 : a3 : This is the 7 character user name to have authority 3 (verified authority) over this library.

authority_4 : a4 : This is the 7 character user name to have authority 4 (released authority) for this library.

3.0 COMMANDS
3.1.1 CREATE_LIBRARY ! CRL

key ! k ! This one character string specifies the key character used to flag text embedded directives on this source library. This defaults to *.

version ! ve ! This 1 to 31 character string is used to give a version number for the library.

result ! r ! This is the name of the file on which the source library is to be created. RESULT is used as a default value.

status ! st ! See basic status concept.

Examples:

```
scu.create_library scu_source_code "Neat stuff." a4=mjp2523 ..
r=scupl

SCU.CRL Table_Tennis_League_Records
```

In the first example the library name and description parameters are given positionally and the authority_4 and result file names are given with the keyword = value syntax. The remaining authorities would default to the user name the command was executed under and * would be used to flag text embedded directives.

In the second example only the library name is given explicitly and a source library will be created on the file RESULT.

3.1.2 DISPLAY_LIBRARY ! DIL

This command lists the descriptive information stored in the source library header and optionally the contents of deck, group, modification and feature lists for the library.

```
display_library [base=<file_name>]
                [list=<local_file_name>]
                [brief ! full]
                [status=r1!r2!r3!ef]
```

base ! ba : This is the name of the file containing the base source library. If this parameter is not given an attempt is made to access a file with the name BASE.

list ! l : This is the name of the file which receives the listing. If this parameter is not given, the file name OUTPUT will be used.

brief ! br ! full ! fu : If the command contains the keyword BRIEF, only the information from the library header will be displayed. If the keyword FULL is present, the deck, group, modification and feature lists for the library will be displayed as well. If neither is present BRIEF is assumed.

status ! st : See basic status concept.

Example:

```
scu.display_library ba = oldpl
```

This command will display only the descriptive information from the library header of the source library on file OLDPL. The display will be written to the file OUTPUT.

3.1.3 CHANGE_LIBRARY ! CHL

This command allows a user to alter fields in the library description.

```
change_library [name=<library_name>]
               [description=<descriptive_string>]
               [authority_1=<user_name>]
               [authority_2=<user_name>]
               [authority_3=<user_name>]
               [authority_4=<user_name>]
               [version=<version_string>]
               [base=<file_name>]
               [result=<file_name>]
               [status=r1|r2|r3|ef]
```

name ! na ! This specifies the name of the source library.

description ! de ! This 1 to 256 character string is used to describe the source code maintained on this library.

authority_1 ! a1 ! This is the 7 character user name to have authority 1 for this library. In order to alter fields authority_1 through authority_4, this command must be executed under the user name that has the corresponding authority or a higher one.

authority_2 ! a2 ! This is the 7 character user name to have authority 2 over this library.

authority_3 ! a3 ! This is the 7 character user name to have authority 3 over this library.

authority_4 ! a4 ! This is the 7 character user name to have authority 4 for this library.

version ! ve ! This 1 to 31 character string is used to give a version number for the source library.

base ! ba ! This is the name of the file containing the base source library. If this parameter is not given an attempt is made to access a file with the name BASE.

result ! r ! This is the file to receive the altered library. If this parameter is not given, it will be written to a file named RESULT.

3.0 COMMANDS
3.1.3 CHANGE_LIBRARY : CHL

status : st : See basic status concept.

Example:

```
scu.change_library version="1.0" ba=mylib
```

3.1.4 COMBINE_LIBRARY ! COL

This command merges the decks from a list of source libraries into a single result library. The result library will include one deck by every name encountered on any of the source libraries. If decks by the same name occur on more than one library the deck from the library listed first will be selected. Decks will have the following logical order on the result library:

1. All decks with names occurring on the last named source library will occur first in their original order.

2. Next the decks with names occurring on the next to last named library will occur in the order they occurred on that library.

3. Decks with names occurring on the remaining libraries will occur in turn in the order they occurred on those libraries.

All source libraries must have the same key character. The result source library will have the same authorities, description, etc. as the last named source library. This command will indicate an error if there are any interlock violations (see discussion of interlock under EXTRACT_LIBRARY).

```
combine_library base=<file_name>
                [result=<file_name>]
                [list=<local_file_name>]
                [brief ! full]
                [status=r1!r2!r3!ef]
```

base ! ba : This is a list of names of files containing source libraries to be combined. The names must be enclosed in parentheses and separated by commas.

result ! r : This is the file to receive the combined source library. If this parameter is not given, the combined source library will be written to a file named RESULT.

list ! l : This is the name of the file which receives the listing. The list shows from which source library decks on the result file were taken. If this parameter is not given, the file name OUTPUT will be used.

3.0 COMMANDS
3.1.4 COMBINE_LIBRARY ! COL

brief ! br ! full ! fu : These keywords can be used to select a shorter or more detailed listing. Currently both listings are the same.

status ! st : See basic status concept.

Example:

```
scu.combine_library (changed,new,base) newbase
```

In this example two source libraries on files named CHANGED and NEW were combined with one on a file called BASE to form a result on a file named NEWBASE. To illustrate how decks are selected and ordered during the processing of this command (and give an example of how this command is intended to be used) let us assume that the three libraries being combined each contained the following decks:

CHANGED	NEW	BASE	NEWBASE	Deck Taken From
CURT	SHARON	JIM	JIM	CHANGED
JIM	ED	BOB	BOB	CHANGED
MIKE		MIKE	MIKE	CHANGED
BOB		BILL	BILL	BASE
		SHERMAN	SHERMAN	BASE
		CURT	CURT	CHANGED
		JOHN	JOHN	BASE
		HARVEY	HARVEY	BASE
			SHARON	NEW
			ED	NEW

In the example above the file named CHANGED had decks that also occurred on the file named BASE. These were selected rather than their duplicates from BASE, but were placed on NEWBASE in the same relative position as their counterparts. The decks from NEW which were not duplicated were added at the end of the deck sequence from BASE in the order they occurred on NEW. Likewise had there been any decks on CHANGED that had unique names they would have been added at the end of the deck sequence on NEWBASE.

In the example above files CHANGED and NEW had no decks by the same name. Had deck BOB also occurred on NEW, the copy of BOB from CHANGED would still have been selected and placed in the same position on NEWBASE. Had deck SHARON also occurred on CHANGED the copy from CHANGED would have been selected and it would have been placed in the same position as shown in the example.

3.1.5 ADD_LIBRARY ! ADL

This command merges the decks from a list of source libraries into a single result library. The result library will include one deck by every name encountered on any of the source libraries. If decks by the same name occur on more than one of the libraries these decks will not be written on the result library and the duplication will be reported in the listing. Decks will have the following logical order on the result library:

1. All decks occurring on the last named source library will occur first in their original order.

2. Next the decks occurring on the next to last named library will occur in the order they occurred on that library.

3. Decks occurring on the remaining libraries will occur in turn in the order they occurred on those libraries.

All source libraries must have the same key character. The result source library will have the same authorities, description, etc. as the last named source library.

```
add_library base=<file_name>
            [result=<file_name>]
            [list=<local_file_name>]
            [brief ! full]
            [status=r1!r2!r3!ef]
```

base ! ba : This is a list of names of files containing source libraries to be combined. The names must be enclosed in parentheses and separated by commas.

result ! r : This is the file to receive the combined source library. If this parameter is not given, the combined source library will be written to a file named RESULT.

list ! ! : This is the name of the file which receives the listing. The list shows from which source library decks on the result file were taken. If this parameter is not given, the file name OUTPUT will be used.

brief ! br ! full ! fu : These keywords can be used to select a shorter or more detailed listing. Currently

both listings are the same.

status : st : See basic status concept.

Example:

```
scu.add_library (green,new,base) newbase
```

In this example the decks on the libraries on the files named GREEN and NEW are added to those on the library on file base to form a result on a file named NEWBASE. To illustrate how decks are selected and ordered during the processing of this command let us assume the three libraries being combined each contained the following decks:

GREEN	NEW	BASE	NEWBASE	Deck Taken From
FRED	SHARON	JIM	JIM	BASE
GORDY	ED	BOB	BOB	BASE
AL		MIKE	MIKE	BASE
JUDY		BILL	BILL	BASE
		SHERMAN	SHERMAN	BASE
		CURT	CURT	BASE
		JOHN	JOHN	BASE
		HARVEY	HARVEY	BASE
			SHARON	NEW
			ED	NEW
			FRED	GREEN
			GORDY	GREEN
			AL	GREEN
			JUDY	GREEN

In the example above there were no deck names duplicated on the three libraries. On the file NEWBASE the decks from BASE occur first in their original order followed by the decks from NEW and GREEN.

Had any deck names been duplicated, none of the duplicates would have been written to NEWBASE and the user would have been informed.

3.1.6 REPLACE_LIBRARY : REL

This command merges the decks from a list of source libraries into a single result library. The result library will include one deck by every name encountered on the last named source library. If decks by the same name occur on more than one library the deck from the library listed first will be selected. Decks with names that do not occur on the last named library will not be included in the result library and that fact will be reflected in the listing. Decks will have the following logical order on the result library:

1. All decks with names occurring on the last named source library will appear in their original order.

All source libraries must have the same key character. The result source library will have the same authorities, description, etc. as the last named source library. This command will be indicate an error if there are any interlock violations (see discussion of interlock under EXTRACT_LIBRARY).

```
replace_library base=<file_name>
                [result=<file_name>]
                [list=<local_file_name>]
                [brief : full]
                [status=r1:r2:r3:ef]
```

base : ba : This is a list of names of files containing source libraries to be combined. The names must be enclosed in parentheses and separated by commas.

result : r : This is the file to receive the combined source library. If this parameter is not given, the combined source library will be written to a file named RESULT.

list : l : This is the name of the file which receives the listing. The list shows from which source library decks on the result file were taken. If this parameter is not given, the file name OUTPUT will be used.

brief : br : full : fu : These keywords can be used to select a shorter or more detailed listing. Currently both listings are the same.

status : st : See basic status concept.

Example:
```
scu.replace_library (changed,new,base) newbase
```

In this example two source libraries on files named CHANGED and NEW were combined with one on a file called BASE to form a result on a file named NEWBASE. To illustrate how decks are selected and ordered during the processing of this command, let us assume that the three libraries being combined each contained the following decks:

CHANGED	NEW	BASE	NEWBASE	Deck Taken From
CURT	BILL	JIM	JIM	CHANGED
JIM	HARVEY	BOB	BOB	CHANGED
MIKE		MIKE	MIKE	CHANGED
BOB		BILL	BILL	NEW
		SHERMAN	SHERMAN	BASE
		CURT	CURT	CHANGED
		JOHN	JOHN	BASE
		HARVEY	HARVEY	NEW

In the example above the files named CHANGED and NEW had decks that also occurred on the file named BASE. These were selected rather than their duplicates from BASE, but were placed on NEWBASE in the same relative position as their counterparts.

Had there been decks on CHANGED or NEW that did not appear on BASE, they would not have been written to NEWBASE and the user would have been informed.

3.1.7 EXTRACT_LIBRARY ! EXL

This command is used to isolate a source library with only a subset of decks on it for manipulation by the SCU editor or maintenance as a separate source library. If the INTERLOCK option is used, the user must have write access to the file named with the BASE parameter.

```
extract_library [name=<deck_name>!all]
                [criteria=<local_file_name>]
                [interlock=<user_name> ! no_interlock]
                [base=<file_name>]
                [result=<file_name>]
                [status=r1!r2!r3!ef]
```

name ! na ! all : This is the name of a deck to be extracted. Alternatively a list of names separated by commas and enclosed by parentheses may be given or a range of decks may be indicated by giving the first and last deck names to extract separated by ellipses. Use of the alternate keyword ALL to specify this parameter has the same effect as specifying the first and last decks as a range. If this parameter is omitted, the contents of the source library to be extracted will be determined solely by the contents of the criteria file.

criteria ! cr : This names a file which can be used to further define which source data is to be included on the subset source library.

interlock ! int ! no_interlock ! nint : This parameter provides a means to allow only the people authorized to use a particular user name to alter the decks being written to the subset library. The value of the parameter may be given as a user name or if the keyword INTERLOCK only is given the user name the command is executed under will be used. When a deck is interlocked, an attempt to extract it to a subset library setting the interlock will be rejected. Also an attempt to replace the deck on a source library by doing a COMBINE_LIBRARY or REPLACE_LIBRARY with the base will be rejected, unless its from the subset library with the interlock set. If the command includes the NO_INTERLOCK keyword the user will be allowed to extract a subset source library whether or not the decks he requests are interlocked. An attempt to interlock decks with this command will be rejected if the criteria file contains directives which cause modifications to be excluded. If neither keyword

is present, the command is processed as if NO_INTERLOCK had been specified.

base ! ba ! This is the name of the file containing the base source library. If this parameter is not given an attempt is made to access a file with the name BASE.

result ! r ! This is the file to receive the subset source library. If this parameter is not given, the subset source library will be written to a file named RESULT.

status ! st ! See basic status concept.

Example:

```
scu.extract_library dsd int=johndoe r=temppl
```

In this example a subset library is extracted from the source library on the file named BASE which includes only the deck dsd. This subset source library is written to a file named TEMPPL and the deck dsd is interlocked for user name JOHNDOE.

3.2 SOURCE DECKS

These commands are available for creating and manipulating source decks.

3.2.1 CREATE_DECK : CRD

This command is used to create source deck(s).

```
create_deck name=((<deck_name>[,<local_file_name>]),...)  !
            names_on_file=<local_file_name>
            modification=<modification_name>
            [after=<deck_name>!before=<deck_name>!alphabetic]
            [author=<author_name_string>]
            [description=<deck_description_string>]
            [processor=<processor_name_string>]
            [group=<group_name>]
            [tab=<tab_character>!notab]
            [columns=<tab_column_integers>]
            [width=<integer>]
            [lid=<line_identifier_placement_keyword>]
            [base=<file_name>]
            [result=<file_name>]
            [expand ! no_expand]
            [multi_partition ! single_partition]
            [same_as=<deck_name>]
            [status=r1!r2!r3!ef]
```

name : na : names_on_file : nof : This parameter must be given in one of the two forms shown below. If the NAME or NA keyword is used (or no keyword is given), the required portion of this parameter is a 1 to 31 character deck name. It may optionally be followed by the name of a local file containing source data to be included in the deck as text. Alternatively a list of deck names or deck names and file names may be given (see the Parameter Lists section of the NOS/VE ERS). All decks created with a single CREATE_DECK command will have identical descriptive information taken from the other parameters on this command. An attempt to create a deck with the same name as one already on the base source library will be rejected. If the keyword NAMES_ON_FILE or NOF is used, this parameter names a file from which initial text for decks will be read. Deck names will be taken from *DECK text embedded directives intermixed with the initial text.

modification ! mod : This 1 to 9 character modification name will be associated with the creation of this deck and included as part of the line identifier for all initial lines of text if any are introduced with this command.

after ! af ! before ! be ! alphabetic ! alpha : This parameter can be used to name a deck on the library after or before which this deck will be added. If the keywords AFTER or BEFORE are used alone, the deck will be added at the end or beginning of the the deck sequence respectively. Alternatively the ALPHABETIC keyword may be used to indicate that the deck will be added in alphabetic sequence. If this parameter is omitted the deck will be addec in alphabetic sequence.

author ! au : This 1 to 31 character string is provided to identify the person who wrote this unit of source data.

description ! de : This 1 to 256 character string is used to describe this ceck.

processor ! p : This 1 to 31 character string is used to identify the processor that will use the source data in this deck as input.

group ! g : This is a 1 to 31 character group name to be associated with this deck. Alternatively a list of group names may be given separated by commas and enclosed in parentheses.

tab ! notab : If the TAB keyword is used or this parameter is given positionally, this one character string specifies a default tab character for use when editing this deck. If the NOTAB keyword is given tabbing will be deselected by default for this deck.

columns ! col : These integers between 1 and 256 are the default tab stops when editing this deck. A series of tab columns may be specified as a value list (see the Parameter lists section of the NOS/VE ERS). A maximum of 256 tab columns may be specified.

width ! w : This is the default line width to be used when this deck is expanded and to be used by the SCU editor. The value must be given as an integer between 0 and 256. Specifying 0 as a width selects the defaults of

supplying no trailing blanks when this deck is expanded and allowing lines up to 256 characters to be entered with the editor. If this parameter is not given a default value of 0 will be used.

lid : This parameter is the default line identifier placement when this deck is expanded. If this parameter is omitted the expanded text file will contain no line identifiers. This parameter can be given as one of 3 values.

right : Line identifiers will be placed on the right.

left : Line identifiers will be placed on the left.

none : Output lines will contain expanded text only.

base ! ba : This is the name of the file containing the base source library to which the new decks are to be added. If this parameter is omitted an attempt is made to access a file named BASE.

result : r : This is the file to receive the modified source library. If this parameter is not given, the modified source library will be written to a file named RESULT.

expand ! exp ! no_expand ! nexp : If the NO_EXPAND keyword is included on this command these decks will only be written to the expanded text file produced by the EXPAND_DECK command as a result of processing the text embedded directives *COPY and *COPYC. If the EXPAND keyword is included or this parameter is omitted entirely, these decks will be written to the expanded text file by the EXPAND_DECK command whether named on the command or on *COPY and *COPYC directives.

multi_partition ! mp ! single_partition ! sp : If the MULTI_PARTITION keyword is included on the command, each end of record encountered in the source file will be represented on the source library as a *WEOP text embedded directive. If the SINGLE_PARTITION keyword is used or this parameter is omitted entirely, only the text before the first separator on the source file will be represented on the source library.

3.0 COMMANDS
3.2.1 CREATE_DECK ! CRD

same_as ! sa : This specifies the name of a deck on the base library. Values from that deck's header will be used to supply values not given explicitly on this command for the author, description, processor, group, tab, columns, width, lid, and expand/no_expand parameters.

status ! st : See basic status concept.

Example 1:

```
scu.create_deck ..
name=((amp$get_direct,srcfile)) ..
modification=original ..
author="R. D. Palm" ..
description="Get Direct module." ..
processor="CYBIL/CI" ..
group=file_io ..
same_as=file_io_prototype
```

In this example, a new deck named amp$get_direct is added to the source library on file BASE. The enlarged library is written on the file RESULT. The initial text for the deck is read from file SRCFILE and each line will have the modification name original associated with it. The author, description, processor and group parameters were given explicitly on the command. Values for the remaining parameters will be copied from the description of the deck named file_io_prototype. In this example the ability to use an ellipsis at the end of a line to continue a command was used to make the command more readable and possibly easier to maintain or modify should it be used as part of a procedure file.

Example 2:

```
scu.crd William_Cody bill
```

In this example the deck WILLIAM_CODY is added to the library on file BASE and the result is written to file RESULT. The creation of the deck is associated with the modification name bill. No initial text is supplied for the deck (it may be added later with the editor) and default values are supplied for all the other parameters.

Example 3:

```
scu.create_deck ..
(u,v,(w,wfile),x,y,z) ..
encrypto ..
author="M. J. Perreten" ..
same_as=CYBIL_type_declarations
```

In this example decks u through z are added to the source library on file BASE. The enlarged library is written to the file RESULT. Deck w has initial text supplied from file WFILE. The modification and author fields are given explicitly on the command. Values for the remaining parameters will be copied from the description of the deck named CYBIL_type_declarations.

Example 4:

```
scu.create_deck ..
nof=august ..
number12 ..
author="Henry McGilton" ..
description="August Tools Bulletin"
```

In this example text for some deck or decks is read from a file named August which contains DECK text embedded directives. This deck will be added to those on the library on file BASE and the result written to a file named RESULT. The modification name associated with each line in the new deck will be NUMBER12. Henry McGilton is identified as the author and the deck is described as containing the August Tools Bulletin.

3.2.2 DISPLAY_DECK ! DID

This command displays descriptive information about the deck.

```
display_deck name=<deck_name>!all
             [base=<file_name>]
             [list=<local_file_name>]
             [brief ! full]
             [notext ! active ! inactive]
             [status=r1!r2!r3!ef]
```

name ! na ! all : This is the name of a deck to be displayed. Alternatively a list of names separated by commas and enclosed by parentheses may be given or a range of decks may be indicated by giving the first and last deck names to display separated by ellipses. Use of the alternate keyword ALL to specify this parameter has the same effect as specifying the first and last decks as a range.

base ! ba : This is the name of the file containing the base source library. If this parameter is not given an attempt is made to access a file with the name BASE.

list ! l : This is the name of the file which receives the listing. If this parameter is not given, the file name OUTPUT will be used.

brief ! br ! full ! fu : If the BRIEF keyword is included on this command, only the descriptive information from the deck header will be displayed. If the FULL keyword is used the list of modification names and group names associated with this deck will be displayed as well. Omitting this parameter gives the brief display.

notext ! active ! act ! inactive ! inact : If the INACTIVE keyword is included on this command both active and inactive lines from the deck will be displayed. If the keyword ACTIVE is given only the active lines in the deck will be displayed. If the NO_TEXT keyword is given or this parameter is omitted entirely none of the text lines of the deck will be displayed.

status ! st : See basic status concept.

Example:

```
scu.display_deck dsd full
```

3.0 COMMANDS
3.2.2 DISPLAY_DECK ! DID

The command in this example lists the descriptive information from the header of deck dsd from the source library on file BASE as well as the list of modification and group names associated with this deck. The text lines of the deck are not included in the listing. The listing is written to the file called OUTPUT.

3.2.3 CHANGE_DECK : CHD

This command is used to change fields in the deck description.

```
change_deck name=<deck_name>|all
            [author=<author_name_string>]
            [interlock=<user_name>]
            [description=<deck_description_string>]
            [processor=<processor_name_string>]
            [group=<group_name>]
            [delete_group=<group_name>]
            [tab=<tab_character>|notab]
            [columns=<tab_column_integers>]
            [delete_columns=<tab_column_integers>]
            [width=<integer>]
            [lid=<line_identifier_placement_keyword>]
            [base=<file_name>]
            [result=<file_name>]
            [expand | no_expand]
            [status=r1|r2|r3|ef]
```

name | na | all : This is the name of a deck to have its description altered. Alternatively a list of names separated by commas and enclosed by parentheses may be given or a range of decks may be indicated by giving the first and last deck names separated by ellipses. Use of the alternate keyword ALL to specify this parameter has the same effect as specifying the first and last decks as a range.

author | au : This 1 to 31 character string is provided to identify the person who wrote this unit of source data.

interlock | int : This parameter is included to allow the interlock for a deck (see the EXTRACT_LIBRARY command) to be changed or cleared. It is altered by supplying a new user name as the value for this parameter and it is cleared by supplying the keyword INTERLOCK only. Altering this field is restricted to the user name which has authority 4 for this library. Clearing this field is restricted to the user name which has authority 4 for the library or the user name for which the interlock is set.

description | de : This 1 to 256 character string is used to describe the contents of this deck.

processor ! p ! This 1 to 31 character string is used to identify the processor that will use the source data in this deck as input.

group ! g ! This is a 1 to 31 character group name to be associated with this deck. Aternatively a list of group names may be given separated by commas and enclosed in parentheses.

delete_group ! dg ! This parameter allows group names to be deleted from the deck header. The value of this parameter is given as a single group name or a list of group names separated by commas and enclosed in parentheses.

tab ! notab ! If the TAB keyword is used or this parameter is given positionally, this one character string specifies a default tab character for use when editing this deck. If the NOTAB keyword is given tabbing will be deselected by default for this deck.

columns ! col ! These integers between 1 and 256 are tab stops to be added to the deck header. A series of tab columns to be added may be specified as a value list (see the Parameter lists section of the NOS/VE ERS). A maximum of 256 tab columns may be specified. When the delete_columns parameter is also included on this command, the tab stops it specifies are cleared before the tab stops specified by the columns parameter are set.

delete_columns ! dcol ! These integers between 1 and 256 are default tab stops for editing to be deleted from the deck header. A series of tabs stops may be specified as a value list (see the Parameter lists section of the NOS/VE ERS).

width ! w ! This is the default line width to be used when this deck is expanded and to be used by the SCU editor. The value must be given as an integer between 0 and 256. Specifying 0 as a width selects the defaults of supplying no trailing blanks when this deck is expanded and allowing lines up to 256 characters to be entered with the editor.

lid ! This parameter is the default line identifier placement when this deck is expanded. This parameter can be given as one of 3 values.

right : Line identifiers will be placed on the right.

left : Line identifiers will be placed on the left.

none : Output lines will contain expanded text only.

base ! ba : This is the name of the file containing the base source library. If this parameter is not given an attempt is made to access a file with the name BASE.

result : r : This is the file to receive the modified source library. If this parameter is not given, the modified source library will be written to a file named RESULT.

expand ! exp ! no_expand ! nexp : If the NO_EXPAND keyword is included on this command these decks will only be written to the expanded text file produced by the EXPAND_DECK command as a result of processing the text embedded directives *COPY and *COPYC. If the EXPAND keyword is included or this parameter is omitted entirely, these decks will be written to the expanded text file by the EXPAND_DECK command whether named on the command or on *COPY and *COPYC directives.

status ! st : See basic status concept.

Example:

```
SCU.CHD scm$delete p="CYBIL"
```

In this example CYBIL is identified as the processor that will use this deck as input. The description of the deck scm$delete will be obtained from file BASE and the modified source library will be written to the file RESULT.

3.0 COMMANDS
3.2.4 PURGE_DECK ! PUD

3.2.4 PURGE_DECK ! PUD

This command is used to remove decks from a source library. This command will be rejected if the user name it is executed under does not have an authority equal to or higher than the state of the modification associated with this deck's creation.

```
purge_deck name=<deck_name>
           [base=<file_name>]
           [result=<file_name>]
           [status=r1!r2!r3!ef]
```

name ! na : This is the name of a deck to be purged. Alternatively a list of names separated by commas and enclosed by parentheses may be given or a range of decks may be indicated by giving the first and last deck names to purge separated by ellipses.

base ! ba : This is the name of the file containing the base source library. If this parameter is not given an attempt is made to access a file with the name BASE.

result ! r : This is the file to receive the modified source library. If this parameter is not given, the modified source library will be written to a file named RESULT.

status ! st : See basic status concept.

Example:

```
scu.purge_deck soup..nuts current shorter
```

In this example the source library on file SHORTER will contain all of the decks in the source library on file CURRENT except those in the range from soup to nuts.

3.2.5 SEQUENCE_DECK ! SED

This command is used to resequence a deck. Only lines associated with modifications of state 4 will be altered. The modification identifier under which the deck was created will be associated with each of these lines on the result library. All history information concerning previous modifications to these lines will be discarded, including inactive lines. Because of this, resequencing is recommended only when a deck has attained some significant milestone. This command is rejected if it is not executed under the user name which has authority 4 for the library.

```
sequence_deck name=<deck_name>!all
              [base=<file_name>]
              [result=<file_name>]
              [status=r1!r2!r3!ef]
```

name ! na ! all : This is the name of a deck to be resequenced. Alternatively a list of names separated by commas and enclosed by parentheses may be given or a range of decks may be indicated by giving the first and last deck names to resequence separated by ellipses. Use of the alternate keyword ALL to specify this parameter has the same effect as specifying the first and last decks as a range.

base ! ba : This is the name of the file containing the base source library. If this parameter is not given an attempt is made to access a file with the name BASE.

result ! r : This is the file to receive the modified source library. If this parameter is not given, the modified source library will be written to a file named RESULT.

status ! st : See basic status concept.

Example:

```
scu.sequence_deck name=(Bob,Carol,Ted,Alice,a..z) oldpl,r=joe
```

3.2.6 MOVE_DECK ! MOD

This command is used to move a deck to another logical position on the source library.

```
move_deck name=<deck_name>!all                                    ! 6
          [after=<deck_name>!before=<deck_name>!alphabetic]       ! 7
          [base=<file_name>]                                      ! 8
          [result=<file_name>]                                    ! 9
          [status=r1!r2!r3!ef]                                    !10
```

name ! na ! all : This is the name of a deck to be moved. Alternatively a list of names separated by commas and enclosed by parentheses may be given or a range of decks may be indicated by giving the first and last deck names to move separated by ellipses. An error will be reported if an attempt is made to move a range of decks to a destination within the range. The alternative keyword ALL will be allowed only if alphabetic ordering is selected and will have the effect of sorting the decks on the result library.

after ! af ! before ! be ! alphabetic ! alpha : This parameter can be used to name a deck on the library after or before which this deck will be moved. If the keywords AFTER or BEFORE are used alone, this deck will be moved to the end or beginning of the deck sequence respectively. Alternatively the ALPHABETIC keyword may be used to indicate that the deck will be moved to alphabetic sequence. If this parameter is omitted the deck will be moved to alphabetic sequence.

base ! ba : This is the name of the file containing the base source library. If this parameter is not given an attempt is made to access a file with the name BASE.

result ! r : This is the file to receive the modified source library. If this parameter is not given, the modified source library will be written to a file named RESULT.

status ! st : See basic status concept.

Example:

```
scu.move_deck r=newpl na=Anatoly after=Bobby oldpl
```

In this example the use of keywords has allowed the parameters to be specified in something other than the default

sequence.

3.2.7 EXPAND_DECK ! EXPD

This command expands a source deck into a file where it may be used as input to some processor. During the course of expansion, text embedded directives intermixed with the source will be processed. For a description of these, see the section on text embedded directives.

```
expand_deck [name=<deck_name>!all]
            [compile=<local_file_name>]
            [criteria=<local_file_name>]
            [width=<integer>]
            [lid=<line_identifier_placement_keyword>]
            [base=<file_name>]
            [list=<local_file_name>]
            [depth=<integer>]
            [brief!full]
            [library_order ! command_order]
            [status=r1!r2!r3!ef]
```

name ! na ! all : This is the name of a deck to be expanded. Alternatively a list of names separated by commas and enclosed by parentheses may be given or a range of decks may be indicated by giving the first and last deck names to expand separated by ellipses. The use of deck ranges on this command for anything other than decks occurring on the last named source library is not recommended (See description of the LIBRARY_ORDER keyword below). Use of the alternate keyword ALL to specify this parameter has the same effect as specifying all decks from a combined library as a range. If this parameter is omitted, the source data to be expanded will be determined solely by the contents of the criteria file.

compile ! c : This is the name of the file for the expanded text. This file will be rewound before and after the command is processed. If this parameter is not given, the expanded text will be written to a file named COMPILE.

criteria ! cr : This names a file which can be used to further define which source data is to be expanded. If no criteria file is used, the selected decks will be expanded with all modifications included.

width ! w : This is the length of the output lines excluding line identifiers if they are requested. A maximum value of 256 will be allowed. An additional 17

columns will be necessary for a line identifier since a blank will separate the modification identifier from the sequence number and another blank will separate the entire line identifier from the expanded text. If 0 is specified as the value for this parameter, output lines will be actual length. If this parameter is not given on the command, the default value from the deck header will be used. If the width selected results in a line being truncated, an error will be reported.

lid : This parameter can be given as one of 3 values.

right : Line identifiers will be placed on the right.

left : Line identifiers will be placed on the left.

none : Output lines will contain expanded text only.

If this parameter is not given on the command, the default value from the deck header is used.

base ! ba : This is the name of the file containing the base source library. Alternatively a list of file names may be given to allow decks to be expanded from alternate source libraries. See normal search order. If this parameter is not given an attempt is made to access a file with the name BASE.

list ! l : This is the name of the file to receive the listing. If this parameter is not given, the file name OUTPUT will be used.

depth : This non-negative integer limits how deeply nested *COPY or *COPYC text embedded directives will be processed. *COPY or *COPYC directives encountered at the next level will be included in the expanded text file as text lines. If this parameter is omitted, *COPY or *COPYC directives will be processed wherever they are encountered.

brief ! br ! full ! fu : If the BRIEF keyword is present or this parameter is omitted, the list will contain only a report of errors which may have occurred while processing the command. If the FULL keyword is present, an explanation of which library each deck was selected from will be included when the command accesses multiple

3.0 COMMANDS
3.2.7 EXPAND_DECK ! EXPD

source libraries.

library_order ! lo ! command_order ! co : If this command includes the COMMAND_ORDER keyword the decks will be written to the expanded text file in the order they are specified on the command. If the LIBRARY_ORDER keyword is included or this parameter is omitted, the decks are written to the expanded text file in the order that the decks occur on the base source library (or in the order they would occur on a combined source library if one were produced from the list of libraries specified by the base parameter).

status ! st : See basic status concept.

Example 1:

```
scu.expand_deck mydeck ba=oldpl
```

In this example the deck mydeck is expanded from the source library on the file OLDPL and written on the expanded text file named COMPILE.

Example 2:

```
scu.expand_deck scm$change ba=(myopl,(cybccmn,ses))
```

In this example the deck scm$change is expanded from the libraries MYOPL and CYCCCMN and written on the expanded text file named COMPILE. In this case CYBCCMN was located under another user name.

3.0 COMMANDS
3.2.8 EXTRACT_DECK ! EXTD

3.2.8 EXTRACT_DECK ! EXTD

This command performs a function similar to EXPAND_DECK with the exception that text embedded directives are not processed but included as text.

```
extract_deck [name=<deck_name>!all]
             [source=<local_file_name>]
             [criteria=<local_file_name>]
             [width=<integer>]
             [lid=<line_identifier_placement_keyword>]
             [base=<file_name>]
             [list=<local_file_name>]
             [brief!full]
             [library_order!command_order]
             [expand!no_expand]
             [deck_directives!no_deck_directives]
             [status=r1!r2!r3!ef]
```

name ! na ! all ! This is the name of a deck to be extracted. Alternatively a list of names separated by commas and enclosed by parentheses may be given or a range of decks may be indicated by giving the first and last deck names to extract separated by ellipses. The use of deck ranges on this command for anything other than decks occurring on the last named source library is not recommended (See description of the LIBRARY_ORDER keyword below). Use of the alternate keyword ALL to specify this parameter has the same effect as specifying all decks from a combined library as a range. If this parameter is omitted, the source data to be extracted will be determined solely by the contents of the criteria file.

source ! s ! This is the name of the file for the extracted text. Decks will be separated by end of partition (end of record on NOS170). This file will be rewound before and after the command is processed. If this parameter is not given, the extracted text will be written to a file named SOURCE.

criteria ! cr ! This names a file which can be used to further define which source data is to be extracted. If no criteria file is used, the selected decks will be extracted with all modifications included.

width ! w ! This is the length of the output lines excluding line identifiers if they are requested. A

maximum value of 256 will be allowed. An additional 17 columns will be necessary for a line identifier since a blank will separate the modification identifier from the sequence number and another blank will separate the entire line identifier from the extracted text. If 0 is specified as the value of this parameter, output lines will be actual length. If this parameter is not given on the command, the default value from the deck header will be used. If the width selected results in a line being truncated, an error will be reported.

lid : This parameter can be given as one of 3 values.

right : Line identifiers will be placed on the right.

left : Line identifiers will be placed on the left.

none : Output lines will contain extracted text only.

If this parameter is not given on the command, the default value from the deck header is used.

base ! ba : This is the name of the file containing the base source library. Alternatively a list of file names may be given to allow decks to be extracted from alternate source libraries. See normal search order. If this parameter is not given an attempt is made to access a file with the name BASE.

list ! l : This is the name of the file to receive the listing. If this parameter is not given, the file name OUTPUT will be used.

brief ! br ! full ! fu : If the BRIEF keyword is present or this parameter is omitted, the list will contain only a report of errors which may have occurred while processing the command. If the FULL keyword is present, an explanation of which library each deck was selected from will be included when the command accesses multiple source libraries.

library_order ! lo ! command_order ! co : If this command includes the COMMAND_ORDER keyword the decks will be written to the extracted text file in the order they are specified on the command. If the LIBRARY_ORDER keyword is included or this parameter is omitted, the decks are

3.0 COMMANDS
3.2.8 EXTRACT_DECK ! EXTD

written to the extracted text file in the order that the decks occur on the base source library (or in the order they would occur on a combined source library if one were produced from the list of libraries specified by the base parameter).

expand ! exp ! no_expand ! nexp : If the EXPAND keyword is present on this command, only decks without the NO_EXPAND attribute will be written to the SOURCE file. If the NO_EXPAND keyword is present, only decks with the NO_EXPAND attribute will be written to the SOURCE file. If neither keyword is present, decks will be written to the SOURCE file without regard to their NO_EXPAND attribute.

deck_directives ! dd ! no_deck_directives ! ndd : If the DECK_DIRECTIVES keyword is present, DECK text embedded directives will precede each deck on the SOURCE file. If the NO_DECK_DIRECTIVES keyword is used or neither keyword is present, no DECK directives will be written to the SOURCE file.

status ! st : See basic status concept.

Example:

```
scu.extract_deck mydeck ba=oldpl
```

In this example the deck mydeck is extracted from the source library on the file OLDPL and written on the text file named SOURCE.

3.2.9 DISPLAY_DECK_REFERENCES ! DDR

This command displays the names of the decks which reference the named deck directly or indirectly by *COPY text embedded directives. It can optionally also display the names of the decks referenced by the named deck directly or indirectly through use of the *COPY text embedded directive.

```
display_deck_references name=<deck_name>!all
                        [base=<file_name>]
                        [list=<local_file_name>]
                        [brief!full]
                        [status=r1!r2!r3!ef]
```

name ! na ! all : This is the name of a deck to be cross referenced. Alternatively a list of names separated by commas and enclosed by parentheses may be given or a range of decks may be indicated by giving the first and last deck names to cross reference separated by ellipses. Use of the alternate keyword ALL to specify this parameter has the same effect as specifying the first and last decks as a range.

base ! ba : This is the name of the file containing the base source library. If this parameter is not given an attempt is made to access a file with the name BASE.

list ! l : This is the name of the file which receives the listing. If this parameter is not given, the file name OUTPUT will be used.

brief ! br ! full ! fu : If the BRIEF keyword is included on this command or this parameter is omitted entirely, only the decks which reference this deck will be listed. If the FULL keyword is used, the list of decks referenced by this deck will be shown as well. References to decks not on the base source library will be flagged as external.

status ! st : See basic status concept.

Example:

```
SCU.DDR scc$display_default_values scupl
```

The command in this example displays the names of all the decks on the source library on file SCUPL which refer to the deck scc$display_default_values directly or indirectly through *COPY directives.

3.2.10 DISPLAY_DECK_LIST ! DDL

This command displays the list of decks on a source library.

```
display_deck_list [base=<file_name>]
                  [list=<local_file_name>]
                  [brief ! full]
                  [status=r1!r2!r3!ef]
```

base ! ba : This is the name of the file containing the base source library. Alternatively a list of file names may be given to give a combined deck list. If this parameter is not given an attempt is made to access a file with the name BASE.

list ! l : This is the name of the file which receives the listing. If this parameter is not given, the file name OUTPUT will be used.

brief : br ! full ! fu : If the command contains the key word BRIEF or this parameter is omitted, a shorter form of the listing will be produced. If the FULL keyword is present a more detailed listing will be given. Currently these listings are identical.

status ! st : See basic status concept.

Example:

```
SCU.DDL scupl sculist
```

3.2.11 DISPLAY_GROUP ! DIG

This command will display a list of the decks associated with a group and optionally the short form of the deck description (see DISPLAY_DECK) for all decks associated with this group name.

```
display_group group=<group_name>
              [base=<file_name>]
              [list=<local_file_name>]
              [brief ! full]
              [status=r1!r2!r3!ef]
```

group ! g : This is the name of the group.

base ! ba : This is the name of the file containing the base source library. Alternatively a list of file names may be given to allow a user to interrogate a set of source libraries that are frequently used together. If this parameter is not given an attempt is made to access a file with the name BASE.

list ! l : This is the name of the file which receives the listing. If this parameter is not given, the file name OUTPUT will be used.

brief ! br ! full ! fu : If the command includes the keyword BRIEF, only the list of decks associated with the group will be displayed. If the keyword FULL is present, the short form of the deck description for each deck will be given as well. The shorter display is selected by default.

status ! st : See basic status concept.

Example:

```
scu.display_group all_iou_programs oldpl
```

3.2.12 DISPLAY_GROUP_LIST ! DGL

This command will list all the group names associated with this source library as well as the decks associated with each group.

```
display_group_list [base=<file_name>]
                   [list=<local_file_name>]
```

3.0 COMMANDS
3.2.12 DISPLAY_GROUP_LIST ! DGL

```
                    [brief ! full]                                          ! 1
                    [status=r1!r2!r3!ef]                                    ! 2
                                                                              3
      base ! ba ! This is the name of the file containing the               4
      base source library. Alternatively a list of file names               5
      may be given to produce a combined group list. If this                ! 6
      parameter is not given an attempt is made to access a                 ! 7
      file with the name BASE.                                              ! 8
                                                                            9
      list ! l ! This is the name of the file which receives               10
      the listing. If this parameter is not given, the file                11
      name OUTPUT will be used.                                            12
                                                                           13
      brief ! br ! full ! fu ! If the command includes the                 14
      BRIEF keyword or this parameter is omitted entirely, the             15
      display will contain only the group names. If the FULL               16
      keyword is given, the group names as well as the decks               17
      associated with each group will be displayed.                        18
                                                                           19
      status ! st ! See basic status concept.                              20
                                                                           21
Example:                                                                   !22
SCU.DGL testpl                                                             !23
                                                                           24
                                                                           25
                                                                           26
                                                                           27
                                                                           28
                                                                           29
                                                                           30
                                                                           31
                                                                           32
                                                                           33
                                                                           34
                                                                           35
                                                                           36
                                                                           37
                                                                           38
                                                                           39
                                                                           40
                                                                           41
                                                                           42
                                                                           43
                                                                           44
                                                                           45
                                                                           46
                                                                           47
                                                                           48
```

3.3 MODIFICATIONS

The following commands together with the SCU editor are used to manipulate modifications to source libraries.

3.3.1 CREATE_MODIFICATION : CRM

This command can be used to create the descriptive information to be associated with a modification. Modifications are always created with state 0.

```
create_modification modification=<modification_name>
                    [feature=<feature_name>]
                    [author=<author_name_string>]
                    [description=<descriptive_string>]
                    [base=<file_name>]
                    [result=<file_name>]
                    [status=r1|r2|r3|ef]
```

modification : mod : This 1 to 9 character field is the modification name. Alternatively a list of names may be given.

feature : fe : This is a 1 to 31 character feature name to be associated with this modification.

author : au : This 1 to 31 character string can be used to identify the individual who wrote this modification.

description : de : This 1 to 256 character string can be used to describe the modification. It is suggested that this be used to point to official documentation for the modification such as psr numbers or design documents.

base : ba : This is the name of the file containing the base source library. If this parameter is not given an attempt is made to access a file with the name BASE.

result : r : This is the file to receive the modified source library. If this parameter is not given, the modified source library will be written to a file named RESULT.

status : st : See basic status concept.

Example:

```
SCU.CRM ..
```

3.0 COMMANDS
3.3.1 CREATE_MODIFICATION ! CRM

```
mod=mymod ..                                                          1
fe=my_appearance ..                                                   ! 2
au="M. J. Perreten" ..                                                ! 3
de="short ,fat, brown hair, beard" ..                                 4
ba=oldfact ..                                                         5
r=newfact                                                             6
                                                                      7
                                                                      8
```

3.3.2 DISPLAY_MODIFICATION ! DIM

```
                                                                      9
                                                                     10
```

This command displays descriptive information about a modification.

```
                                                                     11
                                                                     12
                                                                     13
display_modification modification=<modification_name>!all            !14
                     [name=<deck_name>]                              !15
                     [base=<file_name>]                              !16
                     [list=<local_file_name>]                        !17
                     [brief ! full]                                  !18
                     [status=r1!r2!r3!ef]                            !19
                                                                     20
```

modification ! mod ! all : This is a 1 to 9 character modification name. Alternatively a list of modification names may be given separated by commas and enclosed in parentheses or a range of modifications may be indicated by giving the first and last modification names separated by ellipses. Using the keyword ALL has the same effect as specifying the first and last modifications on the library as a range.

name ! na : This is the name of a deck. If this parameter is specified, descriptive information will be given about only the named modification as it applies to this deck.

base ! ba : This is the name of the file containing the base source library. If this parameter is not given an attempt is made to access a file with the name BASE.

list ! l : This is the name of the file which receives the listing. If this parameter is not given, the file name OUTPUT will be used.

brief ! br ! full ! fu : If the FULL keyword is included on this command, the descriptive information about this modification will be displayed as well as every line of text altered by this modification. If the BRIEF keyword is included or this parameter is omitted entirely, then only the descriptive information about this modification

is displayed.

status ! st : See basic status concept.

Example:

```
scu.display_modification mod=psr123456 na=routineb ba=syspl ..
!=fixforb full
```

In this example the descriptive information about the modification named psr123456 is listed on the file FIXFORB along with any lines altered by this modification in the deck routineb.

3.3.3 CHANGE_MODIFICATION ! CHM

This command enters or modifies descriptive information about a modification.

```
change_modification modification=<modification_name>!all
                    [feature=<feature_name>]
                    [author=<author_name_string>]
                    [description=<descriptive_string>]
                    [base=<file_name>]
                    [result=<file_name>]
                    [state=<integer>]
                    [status=r1!r2!r3!ef]
```

modification ! mod ! all : This is a 1 to 9 character modification name. Alternatively a list of modification names may be given separated by commas and enclosed in parentheses or a range of modifications to be changed may be indicated by giving the first and last modification names separated by ellipses. Using the keyword ALL has the same effect as specifying the first and last modifications on a library as a range.

feature ! fe : This is a 1 to 31 character feature name to be associated with this modification.

author ! au : This 1 to 31 character string can be used to identify the individual who wrote this modification.

description ! de : This 1 to 256 character string can be used to describe the modification. It is suggested that this be used to point to official documentation for the modification such as psr numbers or design documents.

base ! ba : This is the name of the file containing the base source library. If this parameter is not given an attempt is made to access a file with the name BASE.

result ! r : This is the file to receive the modified source library. If this parameter is not given, the modified source library will be written to a file named RESULT.

state : This is the new state for the modification. For a user to raise a modification to a given state or lower it from that state to 0, he must have the corresponding authority over the base source library or a higher one. Values this parameter may assume are:

3.0 COMMANDS
3.3.3 CHANGE_MODIFICATION : CHM

```
                0 : Experimental
                1 : Developmental
                2 : Stable
                3 : Verified
                4 : Released

          status ! st : See basic status concept.

   Example:
   SCU.CHM mymod de='short, fat, bald, beard' state=4
```

```
3.3.4 PURGE_MODIFICATION ! PUM                                          1
                                                                        2
   This command is used to remove a modification from the             ! 3
source library. In order to purge a modification a user must           ! 4
have the appropriate authority for the library to manipulate           ! 5
modifcations at this state. One will not be allowed to purge           ! 6
a modification that is associated with the creation of a               ! 7
deck. Such decks must be purged first.                                 ! 8
                                                                        9
purge_modification modification=<modification_name>                    10
                   [name=<deck_name>]                                  11
                   [base=<file_name>]                                 !12
                   [result=<file_name>]                               !13
                   [status=r1|r2|r3|ef]                               !14
                                                                       15
     modification ! mod : This is the name of a modification.          16
     Alternatively a list of modification names may be given           17
     separated by commas and enclosed in parentheses or a              18
     range of modifications may be indicated by giving the             19
     first and last modification names separated by ellipses.          20
                                                                       21
     name ! na : This is the name of a deck. If this                   22
     parameter is given, only the part of the modification             23
     that applies to this deck will be removed. Alternatively          24
     a list of names separated by commas and enclosed by               25
     parentheses may be given or a range of decks may be               26
     indicated by giving the first and last deck names                 27
     separated by ellipses.                                            28
                                                                       29
     base ! ba : This is the name of the file containing the           !30
     base source library. If this parameter is not given an            !31
     attempt is made to access a file with the name BASE.              !32
                                                                       33
     result ! r : This is the file to receive the modified             !34
     source library. If this parameter is not given, the               !35
     modified source library will be written to a file named           !36
     RESULT.                                                           !37
                                                                       38
     status ! st : See basic status concept.                           !39
                                                                       40
Example:                                                               !41
scu.purge_modification fangs na=serpents ba=beasts r=animals           !42
                                                                       43
                                                                       44
                                                                       45
                                                                       46
                                                                       47
                                                                       48
```

3.3.5 SEQUENCE_MODIFICATION ! SEM

This allows a user to resequence a modification so that sequence numbers will appear in the order that source lines appear in the deck rather than in the order they were introduced. This is only allowed for modifications in state 0. All inactive lines introduced under this modification name will be discarded.

```
sequence_modification modification=<modification_name>
                      [name=<deck_name>]
                      [base=<file_name>]
                      [result=<file_name>]
                      [status=r1!r2!r3!ef]
```

modification ! mod : This is the name of a modification. Alternatively a list of modification names may be given separated by commas and enclosed in parentheses or a range of modifications may be indicated by giving the first and last modification names separated by ellipses.

name ! na : This is the name of a deck. If this parameter is given, only the part of the modification which applies to this deck will be resequenced. Alternatively a list of names separated by commas and enclosed in parentheses may be given or a range of decks may be indicated by giving the first and last deck names separated by ellipses.

base ! ba : This is the name of the file containing the base source library. If this parameter is not given an attempt is made to access a file with the name BASE.

result ! r : This is the file to receive the modified source library. If this parameter is not given, the modified source library will be written to a file named RESULT.

status ! st : See basic status concept.

Example:

```
scu.sequence_modification psr12345
```

3.3.6 EXTRACT_MODIFICATION ! EXM

This command is used to extract a modification from a source library in the form of the editor commands INSERT, DELETE and REPLACE.

```
extract_modification modification=<modification_name>
                     edit_commands=<local_file_name>
                     [name=<deck_name>]
                     [base=<file_name>]
                     [scu|update]
                     [status=r1|r2|r3|ef]
```

modification ! mod ! This is the name of a modification. Alternatively a list of modification names may be given separated by commas and enclosed in parentheses or a range of modifications may be indicated by giving the first and last modification names separated by ellipses.

edit_commands ! ec ! This is the name of the local file to receive the edit commands generated by this command.

name ! na ! This is a deck name. If specified only that part of the modification which applies to this deck will be written to the EDIT_COMMANDS file.

base ! ba ! This is the name of the file containing the base source library. If this parameter is not given an attempt is made to access a file with the name BASE.

scu ! update ! If the UPDATE keyword is specified the modification will be externalized as a six bit display code UPDATE correction set containing *BEFORE, *INSERT and *DELETE directives. If the SCU keyword is given or this parameter is omitted entirely, the modification will be externalized as the ASCII SCU editor commands INSERT, DELETE and REPLACE.

status ! st ! See basic status concept.

Example:

```
scu.extract_modification fred ec=comfile ba=pets
```

3.3.7 DISPLAY_MODIFICATION_LIST ! DML

This command displays a list of all the modifications for a source library.

```
display_modification_list [base=<file_name>]
                          [list=<local_file_name>]
                          [brief!full]
                          [status=r1!r2!r3!ef]
```

base ! ba : This is the name of the file containing the base source library. If this parameter is not given an attempt is made to access a file with the name BASE.

list ! l : This is the name of the file which receives the listing. If this parameter is not given, the file name OUTPUT will be used.

brief ! br ! full ! fu : If the BRIEF keyword is included on the command or this parameter is omitted a shorter display will be given. If the FULL keyword is present a more detailed display will be given. Currently the displays selected by the two keywords are identical.

status ! st : See basic status concept.

Example:

```
SCU.DML !ibrary modlist
```

3.3.8 DISPLAY_FEATURE ! DIF

This command is used to produce a modification description for each modification associated with a feature.

```
display_feature feature=<feature_name>
                [base=<file_name>]
                [list=<local_file_name>]
                [brief ! full]
                [status=r1!r2!r3!ef]
```

feature ! fe : This is the name of the feature to be displayed.

base ! ba : This is the name of the file containing the base source library. If this parameter is not given an attempt is made to access a file with the name BASE.

3.0 COMMANDS
3.3.8 DISPLAY_FEATURE ! DIF

list ! l : This is the name of the file which receives the listing. If this parameter is not given, the file name OUTPUT will be used.

brief ! br ! full ! fu : If the BRIEF keyword is included on this command or this parameter is omitted entirely, the display will include only a list of the modifications associated with the feature. If the FULL keyword is used, the display will give the modification descriptions for each of these modifications as well.

status ! st : See basic status concept.

Example:

```
scu.display_feature enhanced_network_interface
```

3.3.9 DISPLAY_FEATURE_LIST ! DFL

This command produces a list of all feature names for a source library and optionally a list of which modifications are associated with each.

```
display_feature_list [base=<file_name>]
                     [list=<local_file_name>]
                     [brief ! full]
                     [status=r1!r2!r3!ef]
```

base ! ba : This is the name of the file containing the base source library. If this parameter is not given an attempt is made to access a file with the name BASE.

list ! l : This is the name of the file which receives the listing. If this parameter is not given, the file name OUTPUT will be used.

brief ! br ! full ! fu : If the BRIEF keyword is included on this command or this parameter is omitted entirely, only the list of feature names for this source library will be displayed. If the FULL keyword is used, lists of the modifications associated with each feature will be included as well.

status ! st : See basic status concept.

Example:

```
scu.display_feature_list myopl
```

3.4 CONVERSION AIDS

Two commands are available to convert source code from UPDATE and MODIFY program library format to SCU source library format. Update and Modify names which do constitute valid SCU names can be preserved as will sequence information. All modifications are set to 0 (experimental) state. This is done to allow a user the opportunity to add descriptive information to key modifications. On conversion *WEOR directives will be changed to *WEOP directives. Because only one level of file separator will be supported under NOS/VE, the *WEOF directive will be changed to a *WEOP directive as well and the user will be informed.

A command is available to convert source code from SCU source library format to UPDATE program library format. With the exception of name substitution, sequence information will be preserved.

Deck names or correction identifiers which do not qualify as valid SCU deck or modification names will be flagged as errors and the user will be expected to effect name substitution during a subsequent attempt at conversion. Put another way, these conversion utilities will reject names that do not begin with a letter or $.

During conversion a user may wish to make name substitutions either because existing names on the UPDATE or MODIFY libraries may not qualify as SCL names or because he wishes to use the longer names allowed by SCU. The conversion utilities allow for name substitution through the NAME_LIST file. Each line in this file is interpreted as the parameters of a SCL command as shown below:

```
nos_name=<update_or_modify_name>
[scl_short=<modification_name>]
[scl_long=<deck_name>|<define_name>]
```

nos_name : This is a the name of an UPDATE or MODIFY deck, correction identifier or DEFINEd name. The user should specify this parameter with the keyword if the name contains characters other than letters, digits and $.

scl_short : This is a 1 to 9 character SCU modification name.

scl_long : This is a 1 to 31 character SCU deck name or DECLARE name (see criteria file directives). If this parameter is omitted the value given for SCL_SHORT will be used. If a value is not given for at least one of the

3.0 COMMANDS
3.4 CONVERSION AIDS

parameters SCL_SHORT or SCL_LONG an error will be reported and conversion inhibited.

Example:

```
dsd dsd dynamic_display_driver
zscmere scm$ere scm$replace
nos_name=old=fool old_fool
```

On the first line the MODIFY name DSD will continue to be used on SCU as the modification name associated with lines introduced under that name, but the deck name will have the longer, more descriptive form. On the second line a MODIFY deck name is being changed to fit the NOS/VE naming pattern. On the third line the name OLD=FOOL is being changed to OLD_FOOL to make it a legal SCL name.

3.4.1 CONVERT_UPDATE_TO_SCU ! CUTS

This command converts an UPDATE program library to an SCU source library.

```
convert_update_to_scu [oldpl=<local_file_name>]
                      [result=<file_name>]
                      [list=<local_file_name>]
                      [name_list=<local_file_name>]
                      [brief ! full]
                      [ascii64 ! ascii812]
                      [criteria=<local_file_name>]
                      [status=r1!r2!r3!ef]
```

oldpl : This is the name of a file containing an UPDATE program library in sequential format. If this parameter is not given an attempt will be made to access a local file with the name OLDPL.

result ! r : This is the file to receive the new source library. If this parameter is not given, the new source library will be written to a file named RESULT.

list ! l : This is a file to receive a report describing the conversion. The default value for this parameter is OUTPUT.

name_list ! nl : See discussion of name substitution above.

3.0 COMMANDS
3.4.1 CONVERT_UPDATE_TO_SCU ! CUTS

brief ! br ! full ! fu ! If the BRIEF keyword is included on the command or this parameter is omitted, a shorter form of the listing will be produced. If the FULL keyword is used a more detailed listing will be produced.

ascii64 ! ascii812 ! These alternate keywords inform the conversion routine whether the input program library is in the 64 character set or full ascii. ASCII812 is selected if neither keyword is given. Currently the conversion from ASCII update (ASCII812) is not implemented.

criteria ! cr ! This is the name of a file to receive criteria file directives which will preserve a part of the information from the YANK$$$ deck on the UPDATE program library. Active YANK*s will be converted to the corresponding EXCLUDE*s. Active DEFINE*s will be converted to DECLARE*s. Inactive cards in the YANK$$$ deck will be ignored. If this parameter is omitted the contents of the YANK$$$ deck will be ignored.

status ! st ! See basic status concept.

Example:

```
scu.convert_update_to_scu pl1a scupl1a saga ascii64
```

3.4.2 CONVERT_MODIFY_TO_SCU ! CMTS

This command converts a MODIFY program library to an SCU source library.

```
convert_modify_to_scu [opl=<local_file_name>]
                      [result=<file_name>]
                      [list=<local_file_name>]
                      [name_list=<local_file_name>]
                      [brief ! full]
                      [ascii64 ! ascii612]
                      [criteria=<local_file_name>]
                      [status=r1!r2!r3!ef]
```

opl ! This is the name of a file containing a MODIFY program library. If this parameter is not given an attempt is made to access a local file named OPL.

result ! r ! This is the file to receive the new source

3.0 COMMANDS
3.4.2 CONVERT_MODIFY_TO_SCU : CMTS

library. If this parameter is not given, the new source library will be written to a file named RESULT.

list : l : This is a file to receive a report describing the conversion. The default value for this parameter is OUTPUT.

name_list : nl : See discussion of name substitution above.

brief : br : full : fu : If the BRIEF keyword is included on this command or this parameter is omitted, a shorter listing will be produced. If the keyword FULL is included a more detailed listing will be produced.

ascii64 : ascii612 : These alternate keywords inform the conversion routine whether the input program library is in the 64 character set or full ascii. ASCII612 is selected if this parameter is omitted.

criteria : cr : This is the name of a file to receive criteria file directives. As an example, modifications flagged as yanked by MODIFY will be entered into the criteria file as EXCLUDE_MOD directives.

status : st : See basic status concept.

Example:

```
scu.convert_modify_to_scu rel1a scu1a saga ascii64
```

3.4.3 CONVERT_SCU_TO_UPDATE : CSTU

This command converts an SCU source library to an UPDATE program library. This command is not currently implemented and the requirement for such a command is under review.

```
convert_scu_to_update [base=<file_name>]
                      [newpl=<local_file_name>]
                      [list=<local_file_name>]
                      [name_list=<local_file_name>]
                      [brief : full]
                      [ascii812 : ascii64]
                      [criteria=<local_file_name>]
                      [status=r1|r2|r3|ef]
```

base : ba : This is the name of a file containing an SCU

3.0 COMMANDS
3.4.3 CONVERT_SCU_TO_UPDATE : CSTU

source library to be converted. If this parameter is not given an attempt will be made to access a file named BASE.

newpl : This is the file to receive the new UPDATE program library. If this parameter is not given, the new program library will be written to a file named NEWPL.

list : l : This is a file to receive a report describing the conversion. The default value for this parameter is OUTPUT.

name_list : nl : See discussion of name substitution above.

brief : br : full : fu : If the BRIEF keyword is included on the command or this pararmeter is omitted, a shorter listing will be produced. If the FULL keyword is included a more detailed listing will be given.

ascii812 : ascii64 : These alternate keywords inform the conversion routine whether the output program library is in the 64 character set or full ascii. ASCII812 is used if this parameter is omitted.

criteria : cr : This is the name of a criteria file. DECLARE directives will appear as *DEFINE directives in the YANK$$$ deck on NEWPL. EXCLUDE directives will appear as *YANK *SELYANK and *YANKDECK directives.

status : st : See basic status concept.

Example:

```
scu.convert_scu_to_update scupl1a pl1a myth
```

3.5 CALLING THE EDITOR

The command described in this section is used to call the editor. All commands will be interpreted as editor commands until one exits the editor. The editor can be used in either batch or interactive mode.

3.5.1 EDIT

This command establishes an environment for the other line editor commands.

```
edit modification=<modification_name>
     [base=<file_name>]
     [result=<file_name>]
     [input=<local_file_name>]
     [list=<local_file_name>]
     [name=<deck_name>]
     [continue]
     [status=r1:r2:r3:ef]
```

modification : mod : This is the name for the modification being introduced or altered.

base : ba : This is the name of the file containing a source library containing one or more decks to be edited. If this parameter is not given an attempt is made to access a file with the name BASE.

result : r : This is the name of the file to receive the modified source library. If this is not given the modified library will be written to a file named RESULT.

input : i : This is the name of the file from which edit commands are to be read. The default value is INPUT.

list : l : This is the name of the file which receives the listing. If this parameter is not given, the file name OUTPUT will be used.

name : na : This is the name of the deck to edit first.

continue : con : If this keyword is present, the editor will allow the user to add to an existing modification in state 0.

status ! st : See basic status concept.

Example:
scu.edit improve oldpl continue

3.6 GROUPING SCU COMMANDS

Although SCU commands are available as separate job control statements that may be used entirely independently of one another, it is often desirable to associate them with one another to make a series of changes to a base source library and produce a single result source library and listable output. This is done by using the GROUP and END commands.

3.6.1 GROUP

This command is used to flag that following SCU commands are to be associated with one another and to assign default values for the names of some of the files that these commands must access. When this command is used the base source library is copied to a working copy. Subsequent SCU commands using the default base source library will modify this working copy. Using the END command causes the working copy to be written to the result source library. Nondefault base and result source libraries may be named explicitly on individual commands.

Permanent files named on the BASE and RESULT parameters on the GROUP command will remain local until the corresponding END command is executed. The commands listed above under conversion aids should not be used in group mode.

```
group [base=<file_name>]
      [result=<file_name>]
      [list=<local_file_name>]
      [status=r1|r2|r3|ef]
```

base | ba : This is the name of the file containing the base source library. If this parameter is not given an attempt is made to access a file with the name BASE.

result | r : This is the file to receive the modified source library. If this parameter is not given, the modified source library will be written to a file named RESULT.

list | l : This is the name of the file which receives the listing. If this parameter is not given, the file name OUTPUT will be used.

Example: See example below under end.

3.0 COMMANDS
3.6.1 GROUP

3.6.2 END

This command causes the working copy of the source library to be written to the result source library file and any subsequent SCU commands to operate independently of one another.

```
end [status=<status_variable>]
```

```
Example: scu.group oldlib newlib lst0212
         scu.display_library full
         attach,textfp.
         scu.create_deck ((fetch_parameters,textfp)) ..
         fetch_p g=scu
         scu.create_modification mod_to_4 au="M. J. Perreten" .
         de="Adds syntax check."
         scu.edit mod_to_4 continue i=edcoms
         scu.display_library alterna l=altlist
         scu.change_modification axle state=4
         scu.display_library full
         scu.end
```

In this example the GROUP command declares that the SCU commands that follow will be associated with one another. OLDLIB is the file which contains the base source library. All changes to this base are accumulated on a working copy and will be written to the result source library NEWLIB when an END command is encountered. Displays will be written to the file LST0212 unless another list file is explicitly named on an individual command.

The first DISPLAY_LIBRARY command gives a detailed description of the original source library.

The attach of file TEXTFP is shown to illustrate that other commands can be intermixed with SCU commands.

The CREATE_DECK command creates a new deck on the working copy of the source library using the contents of file TEXTFP as the initial text.

The CREATE_MODIFCATION command creates a new modification on the working source library.

The EDIT command calls the editor to perform the edit commands from file EDCOMS on the working copy of the source library. Lines introduced by this session will be part of the modification mod_to_4.

The second DISPLAY_LIBRARY command is included to emphasize

that individual commands can explicitly name alternate files to manipulate.

The CHANGE_MODIFCATION command raises the state of modification axle on the working source library to state 4 (released).

The last DISPLAY_LIBRARY command gives a detailed description of the working source library after all the changes introduced by the entire block of SCU commands.

The END command causes the working source library to be written to the result source library on file NEWLIB and flags that subsequent SCU commands will execute independently of one another.

4.0 DIRECTIVES

Directives are available to the SCU user to assist him in defining the contents of an expanded text file. These are text embedded directives intermixed with the source data and criteria file directives. This section describes both types of directives and their interaction with one another.

4.1 TEXT EMBEDDED DIRECTIVES

A text embedded directive (TED) will be recognized by the occurrence of a key character as the first character of a source line followed immediately by a keyword. If a TED is encountered the normal expansion will be altered in the mannner specified by the particular TED.

While processing the EXPAND_DECK command, TED's are processed and they are not copied into the expanded text file. While processing the EXTRACT_DECK command, TED's are not processed and are written to the the extracted text file.

4.1.1 COPY

This directive must be followed by a deck name. This directive will copy the text of the specified deck into the expanded text file. The same search order will be used as for decks listed on the EXPAND_DECK command. A deck may copy a deck that copies another deck and so on as long as the sequence is not recursive.

*copy name=<deck_name>

name ! na : This is the name of a deck. Alternatively a list of names may be given separated by commas and enclosed in parentheses.

Example: *copy deck1

4.1.2 COPYC (COPY CONDITIONAL)

This directive serves the same function as COPY except that the text of the deck is written to the expanded text file conditionally. In other words, if the specified deck has been copied into the expanded text file for the deck currently being expanded, it will not be included again.

```
*copyc name=<deck_name>
```

name ! na : This is the name of a deck. Alternatively a list of names may be given separated by commas and enclosed in parentheses.

Example: *copyc deck1

4.1.3 DECK

This text embedded directive will only be interpreted as such when it occurs as the first line of a partition (record on NOS 170) in the file named by the NAMES_ON_FILE parameter of the create deck command. Its purpose is to provide the deck name for the following text lines.

```
*deck name=<deck_name> [expand!no_expand]
```

name ! na : This is the name to be given to the deck which will contain the text lines that follow the directive.

expand ! exp ! no_expand ! nexp : If the NO_EXPAND keyword is included on the directive this deck will only be written to the expanded text file produced by the EXPAND_DECK command as a result of processing the text embedded directives *COPY and *COPYC. If the EXPAND keyword is included or this parameter is omitted entirely, this deck will be written to the expanded text file by the EXPAND_DECK command whether named on the command or on the *COPY and *COPYC directives.

4.1.4 ELSE

This optional directive may be used to separate lines of source data that are to be expanded if the given condition is true from those expanded if it is false. Either of the blocks between the IF and the ELSE or between the ELSE and the IFEND

may be empty.

4.1.5 IF

This directive is used to allow source lines to be written to the expanded text file conditionally. If the specified condition is true, the source lines between this directive and the next ELSE or IFEND directive (whichever occurs first) will be written to the expanded text file. Text embedded directives in the same range will be processed while processing the EXPAND_DECK command. If an ELSE directive is present before the IFEND the source lines between those two directives will be skipped as well as any text embedded directives.

If the specified condition is false, source lines and text embedded directives will be skipped until either an ELSE or IFEND directive is encountered. Source lines and text embedded directives between a following ELSE and IFEND will be processed. Normal (unconditional) processing resumes following the IFEND whether the condition is true or false.

There are two forms of this directive shown below.

```
*if $scu_type(name,type)
```

name : This is a name to be checked for the attribute described in the type field.

type : This field is used to specify one of three keywords. Any of these keywords may be prefixed by NOT. The keywords are:

- DECK This is a name of a deck on any of the libraries specified by the BASE parameter.
- MOD This is the name of a modification to any of the libraries specified by the BASE parameter.
- DECL This is a name defined by a DECLARE directive on the criteria file associated with this EXPAND_DECK command.

4.0 DIRECTIVES
4.1.5 IF

```
Example: *if $scu_type(dsd,deck)
         These source lines are expanded if dsd
         is a deck name.
         *else
         These source lines are expanded if dsd
         is not a deck name and these source
         lines are skipped if dsd is a deck name.
         *ifend
         Unconditional processing resumes with
         these source lines.

Example 2: *if $scu_type(fix2,notmod)
           These lines are expanded if there is no
           modification to the source library with
           the name fix2.
           *ifend

*if $ro(name,value)

     name : This is a name defined by a DECLARE directive
     on the criteria file associated with this
     EXPAND_DECK command. Testing a name that is not
     DECLAREd will cause an error.

     ro : This is a relational operator. acceptable
     values are:

          gt : Greater than.

          ge : Greater than or equal to.

          lt : Less than.

          le : Less than or equal.

          eq : Equal to.

          ne : Not Equal to.

     value : This is a positive decimal value between 0
     and 65535.

Example: *if $ne(fritz,35)
         These lines will be expanded if the name
         fritz has any value other than 35.
         *ifend

Conditional blocks may be nested to a maximum depth of 32
```

levels with the restriction that every IF must have a matching IFEND within the same deck.

4.1.6 IFEND

This directive marks the end of a block of source text to be expanded conditionally. An IFEND must occur in the same deck as the corresponding IF.

4.1.7 TEXT

This directive may be used to indicate that the following block of source data is to be considered as text. In other words no scan will be made for text embedded directives except for TEXTEND.

4.1.8 TEXTEND

This directive tells SCU to resume scaning source data for text embedded directives.

4.1.9 WEOP

This directive will cause an end of partition to be written on the file receiving the expanded text. (In the NOS 170 implementation of SCU, an end of record will be written.)

4.1.10 WEOPC

This directive will cause an end of partition to be written to the file receiving the expanded text, if any source data has been written to the file since the last separator. (In the Cyber 170 implementation, an end of record will be written.)

4.2 CRITERIA FILE DIRECTIVES

These directives are available to allow the SCU user to be more selective about what source data is written to the expanded text file. They are given on a separate file and are not preceded by a key character. They are processed in order of occurence.

4.2.1 DECLARE

This directive is used to define a name with a value associated with it that can be used in connection with the IF text embedded directive to conditonally include source data on the expanded text file.

declare name =<name> [value=<integer>]

name : na : This is a 1 to 31 character name.

value : This optional parameter is an integer in the range 0 to 65535 decimal. If this is omitted the value 0 is assigned.

Example: declare fritz 35

4.2.2 EXCLUDE_DECK

This directive can be used to explicitly exclude a particular deck from being written to the expanded text file.

exclude_deck name=<deck_name>

name : na : This is the name of a deck to be excluded. Alternatively a list of names separated by commas and enclosed by parentheses may be given or a range of decks may be indicated by giving the first and last deck names to exclude separated by ellipses.

4.2.3 EXCLUDE_FEATURE

This directive is used to exclude incomplete or unstable features from an expanded text file.

exclude_feature feature=<feature_name> [state=<integer>]

feature : fe : This is the name of a feature to exclude when generating an expanded text file. Alternatively a list of features may be given separated by commas and enclosed in parentheses.

state : This parameter may be given as one of five integers 0-4 (see states in the concepts section). All modifications which are part of this feature which are at or below the given state will be excluded from the expanded text file. If this parameter is omitted all modifications associated with this feature will be excluded.

Example: exclude_feature new_network_interface 0

4.2.4 EXCLUDE_GROUP

This directive is used to exclude an entire group of related decks from the expanded text file.

exclude_group group=<group_name>

group : g : This is a 1 to 31 character group name. Alternatively a list of group names may be given separated by commas and enclosed in parentheses or a range of groups may be indicated by giving the first and last group names to exclude separated by ellipses.

4.2.5 EXCLUDE_MOD

This directive can be used to exclude the effects of a particular modification.

exclude_mod modification=<modification_name> ... [name=deck_name]

modification : mod : This is the name of a modification to be excluded from the expanded text file. Alternatively a list of modifications may be given separated by commas and enclosed in parentheses.

name : na : This is the name of a deck. That part of the modification which applies to this deck will be excluded on the expanded text file. Alternatively a list of deck names may be given separated by commas and enclosed in

parentheses.

Example: exclude_mod newmethod obsolete_deck

4.2.6 EXCLUDE_STATE

This directive is used to exclude the effects of all modifications at or below a given state.

exclude_state state=<integer>

state : This parameter is given as one of the integers 0-3.

4.2.7 INCLUDE_COPYING_DECKS

This directive causes all decks which copy a named deck to be written to the expanded text file. Chains of indirect references will be followed until decks with the EXPAND attribute are found.

include_copying_decks name=<deck_name>

name ! na : This is a deck name. As on the EXPAND_DECK command the name parameter may be given as a list or range of decks.

4.2.8 INCLUDE_DECK

This directive is used to specifcally include a particular deck on the expanded text file.

include_deck name=<deck_name>

name ! na : This is a deck name. As on the EXPAND_DECK command the name parameter may be given as a list or range of decks.

4.2.9 INCLUDE_FEATURE

This directive is used to include all the modifications associated with a feature which are at a given state or higher on the expanded text file.

4.0 DIRECTIVES
4.2.9 INCLUDE_FEATURE

include feature=<feature_name> [state=<integer>]

feature : fe : This is the name of a feature to be included. Alternatively a list of features may be given separated by commas and enclosed in parentheses.

state : This optional parameter is given as one of five integers 0-4 (see states in the concept section). If this parameter is omitted all modifications associated with this feature will be included.

Example: include_feature new_network_interface 3

4.2.10 INCLUDE_GROUP

This directive is used to cause an entire group of related decks to be included on the expanded text file.

include_group group=<group_name>

group : g : This is a 1 to 31 character group name. Alternatively a list of group names may be given separated by commas and enclosed in parentheses or a range of groups may be indicated by giving the first and last group names to include separated by ellipses.

4.2.11 INCLUDE_MOD

This directive is used to explicitly include a modification in the expanded text file.

include_mod modification=<modification_name> .. [name=<deck_name>]

modification : mod : This is the name of a modification to be included. Alternatively a list of modifications my be given separated by commas and enclosed in parentheses.

name : na : This is the name of a deck. If this parameter is given only that part of the modification that applies to this deck will be included. As on the EXPAND_DECK command the name parameter may be given as a list or range of decks.

Example: include_mod accounting_fixes

4.2.12 INCLUDE_STATE

This directive is used to include the effects of all modifications at or above a given state.

include_state state=<integer>

state : This parameter is given as one of the integers 0-4.

5.0 SCU EDITOR

The SCU editor provides the capability of generating and manipulating source text. It operates on files in source library format allowing source lines to be introduced or corrected either interactively or in batch mode.

5.1 EDITOR CONCEPTS

This section introduces some concepts that are used in the SCU editor. Although most of the terms are not new they may have particular meaning in the SCU context.

5.1.1 TABBING

The TAB command defines a tab character and associated tab columns. Other commands which involve inserting text scan the inserted text for the tab character. When the tab character is encountered, spaces are inserted into the text from that point up to the next tab column where the next character is placed. The tab character is not placed in the library text. One can define a default value for the tab character and columns in each deck header.

One tab character and up 256 associated tab columns can be active at one time.

5.1.2 LINE IDENTIFIERS

Each line within an SCU library is assigned a line identifier which is unique within a deck. This identifier consists of the 1 to 9 character modification name and a six digit sequence number. The sequence number will start from 1 as the first line introduced under the modification in a deck and increment by 1 as lines are added. Continuing a modification will result in sequence numbers starting 1 greater than the last previously introduced to this deck. Sequencing starts from 1 in each deck rather than being continuous throughout the source library. To reference a line by line identifier one gives the modification name separated from the sequence number by a period. Trailing blanks in the

modification name and leading zeros in the sequence numbers will have no significance.

5.1.3 KEYWORD VALUES

On some editor commands, lines or position within a deck may be given through the use of keywords. These keywords and their short forms are listed below.

ALL : A : This indicates all active lines in an SCU deck.

FIRST : F : This indicates the first active line in an SCU deck.

LAST : L : This indicates the last active line in an SCU deck.

CURRENT : C : This indicates the current active line in an SCU deck.

On those line editor commands using the number parameter the keyword below can also be used to specify a value.

ALL : A : This specifies every active line in a range.

On those line editor commands using the occurrence parameter, the keyword below can be used to specify a value.

LAST : L : This specifies the last occurrence of the line or block.

On the DECK command a deck to be edited can be selected by giving a deck name explicitly or by the use of the keywords below.

FIRST : This keyword selects the first deck in the directory on the library as the object of the commands which follow.

LAST : This keyword selects the last deck in the directory on the library as the object of the commands which follow.

NEXT : This command selects the next deck in the directory on the library as the object of the commands which follow.

5.1.4 WIDTH

Width is a maximum number of characters to be allowed in a text line. Line overflow occurs if the user causes a line to be longer than the width declared. The editor issues a warning message and allows the overflowing line to be added or altered. It is the user's responsibility to split longer lines where necessary.

5.1.5 THE BATCH USER

The SCU line editor will in some cases discriminate between batch and interactive users. The VETO option on commands will not be available to batch users. Further restrictions on which commands and options are available to the batch user may prove desirable. For example, locating or altering source lines by matching text strings alone may lead to ambiguity which is readily apparent when editing interactively but may cause serious problems when entering corrections in batch mode.

5.1.6 UNIT

Some of the line editor commands use as a parameter the UNIT keyword, which has the short form UN. When this option is selected a text string will be considered a match with that given on the command only if it occurs as a unit, that is it is surrounded by characters other than those allowed in an SCL name. The most common example of this is to search for a character string that is enclosed in a pair of blanks.

5.1.7 VETO

Some of the line editor commands that involve text or source line alteration allow the keyword parameter VETO which has the short form V. If this parameter is specified, the first and last line of each group to be altered will be output followed by the query "CHANGE?", "DELETE?" or "REPLACE?" depending on the command. In the case of the CHANGE command the candidate line will be shown with the next string substitution already in effect. The user may make the following responses to the query:

NO ! N : This causes the current group replacement to be skipped.

QUIT : Q : This causes the command to terminate with no further replacements.

CONTINUE : C : This causes veto mode to be ended. The current and any subsequent replacements will occur.

YES : Y : This causes the current group replacement to take place.

Any other response causes the query to be repeated and veto mode to remain in effect.

5.1.8 END OF LINE BLANKS

The SCU editor deletes trailing blanks on input. For the sake of matching character strings during searches lines appear to be blank padded to "width" characters.

5.1.9 TERMINAL INTERRUPTS

Under IAF on 170 either user break 1 or user break 2 will have the same effect on the editor. If either is received while displays are being sent to the terminal the remainder of the output will be discarded and the editor will wait for a new command to be entered. If either is entered while the editor is waiting for a response to the veto query described above the editor will respond as if the user had entered QUIT. If either is entered while the editor is accepting text input, text input mode will be ended. In the last case an additional carriage return will be required before the user will be able to continue.

5.1.10 REPLACEMENT TEXT

On the INSERT and REPLACE commands described below replacement text may be supplied in three different ways each indicated by the use of a different keyword. Tab characters are processed regardless of the origin of the replacement text. Lines introduced will have line identifiers consisting of the current modification name and consecutive sequence numbers starting from the current value of the sequence number for this modification. The replacement text parameter is

described below:

[until=d!with=t3!file=<local_file_name>]

until ! u ! When the keyword UNTIL is used, replacement text is read from the edit command input file. The one character string given as the value of this parameter specifies the delimiter marking the end of the text to be inserted. The delimiter is only interpreted as such when it is the last character on a line. The delimiter is not considered part of the entered text.

with ! w ! When the with keyword is used, the string given as the value for this parameter is used as the replacement text.

file ! f ! When the file keyword is used, replacement text is read from the named file. If the keyword FILE is used and no value is given an attempt is made to access a file named MERGE. When this keyword is used the keywords below can be used as the second value of a value set for this parameter.

rewind ! r ! rewind_before ! rb ! rewind_after ! ra ! no_rewind ! nr These keywords specify positioning to be done on the file. Default is REWIND_BEFORE.

Default values are assigned as follows. If the the value of this parameter is given positionally, it is assumed to be a value for WITH. If this parameter is omitted entirely the command will be processed as if UNTIL="/" had been specified.

5.2 EDITOR COMMANDS

The SCU line editor commands employ a syntax similar to that proposed for NOS/VE System Command Language (SCL). Where ranges are indicated, such as first and last lines to be listed, the two values are separated by ellipses. Items shown in brackets are optional and those nested in brackets within brackets are an optional part of an optional item.

The short form for a command is given with the command where one is available.

5.2.1 CHANGE ! C

This changes a specified number of occurrences of a specified string to a replacement string.

```
change text=t1 [with=t3] [number=n] [lines=l1[..l2]] [unit] ..
[veto]
```

text ! t : This specifies the text string which is to be replaced on lines in the range given below.

with ! w : This specifies the text string which will replace the one searched for in the previous parameter. If this parameter is omitted the original text string will be deleted.

number ! n : This specifies the maximum number of occurences of the original string that are to be replaced. The value of this parameter may be given as a number or the keyword ALL. This defaults to ALL if a range is given explicitly for the lines parameter and to 1 occurence when it is not.

lines ! l : This specifies the range of lines to be scanned for string substitution. The value of l1 may be given as a line identifier or one of the keywords ALL, FIRST, CURRENT or LAST. The value of l2 may be given as a line identifier or the keyword CURRENT or LAST. If this parameter is omitted l1 and l2 are given the default values of CURRENT and LAST respectively.

unit ! un : See description of unit above.

veto ! v : See description of veto above. Also for this command, should the string to be replaced occur more than once in a line, the veto option will cause the line to be displayed once for each potential replacement with the next replacement in effect.

Example: change "Bill" "Sam" l=decka.153..LAST

The command above will replace every occurence of the text string "Bill" with the text string "Sam" on each line in the deck currently being edited from the line with the identifier decka.153 to the last active line in the deck.

5.2.2 CLEARTAB ! CTAB

This clears tabbing selections defined by a TAB command.

```
cleartab [columns=c1,c2...cn]
```

columns ! col : This parameter is a list of tab columns to be cleared. If this parameter is not given, all columns are cleared.

5.2.3 COPY

This copies a group of lines from an library to a local file. The copied lines are left unaltered in the original location.

```
copy [text=t1[..t2]] [file=(<local_file_name>
     [,rewind!rewind_before!rewind_after!no_rewind])] ..
[lines=l1[..l2]] [number=n] [unit]
```

text ! t : This specifies strings of text occurring in the first and last lines of text to be copied. If this parameter is omitted entirely, the lines parameter will determine the lines to be copied. If a single text string is given, a group to be copied will contain a single line. When two text strings are given, the search for the second starts immediately after the occurrence of the first.

file ! f : This is the name of a local file to which the text of the selected lines will be written. If this parameter is omitted the text lines will be written to a file named MERGE. The keywords below can be used as the second value of a value set for this parameter.

rewind ! r ! rewind_before ! rb ! rewind_after ! ra ! no_rewind ! nr :
This parameter specifies positioning for the local file. Omission causes the file to be rewound before it is written.

number ! n : This specifies the maximum number of groups of lines to be copied. Number specifies the count of occurrences of the copied block when the text parameter is given and individual lines when it is not. This parameter may have its value given as a number or the keyword ALL. This parameter defaults to ALL when a range

5.0 SCU EDITOR
5.2.3 COPY

is given explicitly for the lines parameter and to 1 when it is not.

lines ! l : This specifies a range of lines to be scanned for copying. l1 can be a line identifier or the keyword ALL, FIRST, CURRENT or LAST. l2 may be given as a line identifier or the keyword CURRENT or LAST. The default values supplied if no range is given explicitly for the lines parameter are CURRENT and LAST respectively.

unit ! un : See description of unit above.

Example: copy l=abc.4..abc.63

In this example the text lines abc.4 through abc.63 will be copied out to a file called MERGE.

5.2.4 DECK

This command names the deck to which the following edit commands are to apply.

deck name=deck_name!first!next!last

name ! na ! first ! next ! last : When the keyword NAME or NA is used or a value only is given this specifies the name of the deck to be edited. The alternative keywords FIRST, NEXT and LAST can be used to select the decks which occur in those positions in the directory on the library.

Example: deck dsd

5.2.5 DELETE ! D

This deletes specified lines from the current deck.

delete [text=t1[..t2]] [number=n] [lines=l1[..l2]] [unit] .. [veto]

text ! t : This specifies strings of text occurring in the first and last lines of text to be deleted. If this parameter is omitted entirely, the lines parameter will determine the lines to be deleted. If a single text

string is given, a group to be deleted will contain a single line. When two text strings are given, the search for the second starts immediately after the occurrence of the first.

number : n : This specifies the maximum number of groups of lines to be deleted. Number specifies the count of occurrences of matching blocks when the text parameter is given and individual lines when it is not. This parameter may have its value given as a number or the keyword ALL. This parameter defaults to ALL when a range is given explicitly for the lines parameter and to 1 when it is not.

lines : l : This specifies a range of lines to be scanned for deleting. l1 can be a line identifier or the keyword ALL, FIRST, CURRENT or LAST. l2 may be given as a line identifier or the keyword CURRENT or LAST. The default values supplied if no range is given are CURRENT and LAST respectively.

unit : un : See description of unit above.

veto : v : See description of veto above.

Examples: delete 'This line has a four letter word.' l=all
d

The first example deletes all lines in the current deck which contain the indicated string. The second example deletes the current line.

5.2.6 DISPLAY_EDITOR_STATUS : DES

This command displays the name of the file being edited, the modification name in use, the deck being edited, the currently selected width, the current tab character and columns, what window columns are in effect and whether the verify and state options are active.

display_editor_status

5.2.7 END

This command terminates the line editor.

5.2.8 FIND ! F

This finds a specified block of text in the current deck and updates the current position within the deck. If the verify option is selected the first and last lines in the block will be displayed or the line will be displayed if a single line is sought.

find [text=t1[..t2]] [occurrence=o] [lines=l1[..l2]] [unit]

text ! t : This specifies strings of text occurring in the first and last lines of the block of text to be found. If this parameter is omitted entirely, the lines parameter will determine the line to be found. If a single text string is given, the block to be found will contain a single line. When two text strings are given, the search for the second starts immediately after the occurrence of the first.

occurrence ! o : Occurrence specifies the count of occurrences of the matching block when the text parameter is given and single lines when it is not. Occurrence specifies that the oth occurrence of the line or block is to be found. The value of this parameter may be given as a number or the keyword LAST. The value of this parameter defaults to 1. When o is negative, the oth previous occurence of the line or block will be found starting with l1 from the lines parameter and scanning towards FIRST.

lines ! l : This specifies a range of lines to be scanned for the find. l1 may be given as a line identifier or the keyword ALL, FIRST, CURRENT or LAST. l2 may be given as a line identifier or the keyword CURRENT or LAST. l1 defaults to CURRENT and l2 defaults to LAST if the lines parameter is omitted entirely. If the value of the occurrence parameter is negative, l2 may not be given and the search is backwards from l1 to FIRST.

unit ! un : See description of unit above.

Example: f "title" -1

5.0 SCU EDITOR
5.2.8 FIND : F

In this example the SCU editor will start from the current line and then search for the nearest previous line which includes the text string "title".

5.2.9 INSERT : I

This command inserts text after or before the specified line.

```
insert [until=d|with=t3|file=<name>] [after=l1|before=l1] ..
[single_partition|multi_partition]
```

until : u : with : w : file : f : See discussion of replacement text above.

after : af : before : be : This specifies the line text is to be inserted after or before. This parameter may have its value given as a line identifier or one of the keyword FIRST, CURRENT or LAST. The default value for this parameter is after CURRENT.

single_partition : sp : multi_partition : mp : This parameter is only meaningful when the FILE keyword is used to indicate the source of replacement text. If the MULTI_PARTITION keyword is included on this command, each end of record encountered in the merge file will be represented on the source library as a *WEOP text embedded directive. If the SINGLE_PARTITION keyword is used or this parameter is omitted entirely, only the text before the first separator on the source file will be represented on the source library.

```
Example 1: insert af=bill.14
           This line is inserted after bill line 14./
```

```
Example 2: i "(Brief inane comment.)"
```

In this example the replacement text was given on the command.

5.2.10 LIST ! L

This lists specified blocks of lines from the current deck.

list [text=t1[..t2]] [number=n] [lines=l1[..l2]] [unit]

text ! t : This specifies strings of text occurring in the first and last lines of the block of text to be listed. If this parameter is omitted entirely, the lines parameter will determine the lines to be listed. If a single text string is given, the block to be listed will contain a single line. When two text strings are given, the search for the second starts immediately after the occurrence of the first.

number ! n : This specifies the maximum number of groups of lines to be listed. Number specifies the count of matching blocks when the text parameter is given and individual lines when it is not. This parameter may have its value given as a number or the keyword ALL. This parameter defaults to ALL when a range is given by the lines parameter and to 1 when no range is given.

lines ! l : This specifies the range of lines to be listed. A value for l1 may be given as a line identifier or one of the keywords ALL, FIRST, CURRENT or LAST. A value for l2 may be given as a line identifier or the keyword CURRENT or LAST. If this parameter is omitted, l1 defaults to CURRENT and l2 defaults to LAST.

unit ! un : See description of unit above.

Example: ! n=4
This lists 4 lines starting with the current line.

5.2.11 MASK

This command defines a character which will be considered as matching any character during string searches or clears the selection of such a character.

mask character=c!off

character ! c ! off : If the OFF keyword is used, masking will be turned off. If the CHARACTER keyword is used or no keyword is used, this must be a one character string

5.0 SCU EDITOR
5.2.11 MASK

specifying the character which is to be used to match all characters in string searches.

Example: Mask "*"
change "P*rr*t*n" "Perreten"

In this example all vowels in the name Perreten are e's. These commands will correct all misspellings of the name in the current deck which match the pattern given by the TEXT parameter.

5.2.12 REPLACE ! R

This deletes each group of lines in the specified range that starts and ends with lines containing the specified text, and replaces each group of lines with the replacement text.

```
replace  [text=t1[..t2]]  [until=d!with=t3!file=<name>]  ..
[number=n]  [lines=l1[..l2]]  [unit]  [veto]  ..
[single_partition!multi_partition]
```

text ! t : This specifies strings of text occurring in the first and last lines of text to be replaced. If this parameter is omitted entirely, the lines parameter will determine the lines to be replaced. If a single text string is given, a group to be replaced will contain a single line. When two text strings are given, the search for the second starts immediately after the occurrence of the first.

until ! u ! with ! w ! file ! f : See discussion of replacement text above.

number ! n : This specifies the maximum number of groups of lines to be replaced. Number specifies the count of matching blocks when the text parameter is given and individual lines when it is not. This parameter may have its value given as a number or the keyword ALL. This parameter defaults to ALL when a range is given explicitly for the lines parameter and to 1 when it is not.

lines ! l : This specifies a range of lines to be scanned for replacement. l1 can be a line identifier or the keyword ALL, FIRST, CURRENT or LAST. l2 can be a line identifier or the keyword CURRENT or LAST. The default values supplied if no range is given are CURRENT and

5.0 SCU EDITOR
5.2.12 REPLACE ! R

LAST.

unit ! un :See description of unit above.

veto ! v : See description of veto above.

single_partition ! sp ! multi_partition ! mp : This parameter is only meaningful when the FILE keyword is used to indicate the source of replacement text. If the MULTI_PARTITION keyword is included on this command, each end of record encountered in the merge file will be represented on the source library as a *WEOP text embedded directive. If the SINGLE_PARTITION keyword is used or this parameter is omitted entirely, only the text before the first separator on the source file will be represented on the source library.

Example: replace *call old* lines=ALL
call new(p1,p2,p3)/

5.2.13 SEQUENCE ! SEQ

This command alters the form in which lines from the library are displayed by the editor.

sequence on!off!separate

on ! off ! separate : If the ON keyword is used on this command, the line identifier will precede the text of the line on the display as part of the same line. If the OFF keyword is used on this command the line identifier will not be displayed. If the SEPARATE keyword is used the line identifier will precede the text of a line as a separate line. The initial setting of this option is taken from the value of the LID field in the header of a deck when a deck is selected for editing on the EDIT call or a DECK command. If LID=NONE then SEQUENCE OFF is selected, otherwise SEQUENCE ON is selected.

5.2.14 STATE

This command causes the state of the modification associated with the introduction of a text line to be displayed when that line is displayed.

state on!off

5.2.15 TAB : T

This command alters the tabbing environment for the INSERT and REPLACE commands. The user specifies a tab character and columns. The presence of the tab character in input text causes blank fill to the next tab column. Tab characters entered after the last tab column has been reached will be accepted as data. Use of the DECK command causes the default tab selections for the named deck to be invoked.

tab [character=c] [columns=c1,c2,...cn]

character : c : This one character string specifies the tab character. If this parameter is omitted, the currently selected tab character will remain in effect.

columns : col : This gives tab columns to be selected. A maximum of 256 tab columns may be selected at any time. These integers must be in the range 1 to 256. Multiple columns may be specified as a value list (see the Parameter Lists section of the NOS/VE ERS). If this parameter is omitted entirely the currently selected tab columns will remain in effect.

Example: tab "\" (11,18,30,36)

5.2.16 VERIFY : V

This command causes the CHANGE command to echo back to the user the altered line(s). In addition if the verify option is selected the first and last lines of a block located by the FIND command will be displayed or the line will be displayed if a single line is sought.

verify on:off

5.2.17 WIDTH : W

This command declares the number of characters the editor will allow to be entered in a text line without issuing a warning message. Use of the DECK command causes the default width from the deck header to be invoked.

5.0 SCU EDITOR
5.2.17 WIDTH : W

width width=n

width : w : The value for this parameter must be given as an integer from 0 to 256. Specifying 0 will allow lines of up to 256 characters to be entered.

5.2.18 WINDOW : WN

This command limits the range of character positions within a line to be scanned in string searches during subsequent editor commands. Only the characters in the given range are searched for a match with the string given on the commands DELETE, LIST, REPLACE, FIND, COPY and CHANGE.

window [columns=c1[..c2]]

columns : col : This specifies the starting and ending scan columns. c1 defaults to column 1 and c2 to column 256.

5.3 EDITOR FUNCTIONS

Specialized command language functions will be available within the version of the SCU editor which executes under NOS/VE. They are intended to be used in conjunction with the System Command language control statements to aid in altering the path taken through sequences of SCU editor commands. These functions will not be available in the NOS 170 version.

5.3.1 $CURRENT_DECK

This function will return the name of the currently selected deck as its value.

5.3.2 $FIRST_DECK

This function will return the name of the deck which occurs first in the directory of the library as its value.

5.3.3 $LAST_DECK

This function will return the name of the deck which occurs last in the directory of the library as its value.

5.4 EDITOR LOOPING

On NOS/VE it will be possible to combine SCU editor commands and functions with SCL control statements to perform repetitive changes that are often necssary in such applications as test base maintenance.

6.0 FEATURES UNDER CONSIDERATION

The features listed in this section are under consideration. Some are listed here because it is felt that they may prove superfluous. In other cases they are listed here because it is questionable whether they are compatible with the basic design of the utility or it is not yet clear in what form the feature should be supplied.

6.1 MODNAME EDITOR COMMAND

It may prove desirable to provide a editor command similar to the MODNAME directive in MODIFY, which would establish a default modification name allowing a shorthand form for line identifiers.

6.2 INACTIVE LINES AND THE EDITOR

Currently it is planned that the SCU editor deal with only active lines. It may prove desirable for the editor to FIND or LIST inactive lines or perform other manipulations on inactive lines.

6.3 RESTORE LINE COMMAND

It may prove desirable to provide a RESTORE command in the editor, which would clear the inactive flag for lines previously deleted.

6.4 CHANGES NOT INTRODUCED IN THE EDITOR

Some SCU commands outside the editor introduce significant changes to a source library without being associated with a modification name. No effective method for automating the control of this type of change has been devised. Those commands of concern in this respect are PURGE_DECK, SEQUENCE_DECK, PURGE_MODIFICATION and SEQUENCE_MODIFICATION.

6.5 AUTOMATIC MODIFICATION RESEQUENCING

It may be desirable to automatically resequence modifications when their state is raised to 1 (developmental). See the SEQUENCE_MODIFICATION command.

6.6 SAVING EDIT FILES

It may be desirable to provide a mechanism for preserving an library as permanent without exiting the editor.

6.7 KEY CHARACTER SUBSTITUTION

It may be desirable to allow the CHANGE_LIBRARY command to substitute the key character used on a source library.

6.8 COMPARE UTILITY

It may prove useful to supply a utility which could compare a source file with a deck on an SCU library and produce a set of editor commands to cause the deck to match the source file.

6.9 PROCESSOR BASED DECK SELECTION

It may prove useful to provide a capability to conditionally write a deck to an expanded text file based on the contents of the processor field in the deck header.

6.10 EDITOR MOVE COMMAND

It may prove useful to provide an editor command to move a block of text lines from one place in a deck to another.

7.0 PHASED IMPLEMENTATION

7.0 PHASED IMPLEMENTATION

Current plans are to implement the version of SCU that executes under NOS 170 in the steps below. Design of the version of SCU to execute on NOS/VE has begun and implementation will tentatively begin in March of 1980.

1. All SCU features will be implemented and tested by the project by April,1980.

2. SCU will be ready for release as part of the SES tools package, as part of SES Release 14, currently scheduled for July, 1980.

8.0 MESSAGES

8.1 NON-EDITOR MESSAGES

Below are listed messages issued primarily from non-editor SCU procedures. The conditon numbers are shown relative to 9000. This origin may change on 170 and will change on NOS/VE. The letters listed in the severity column stand for E - error, F - fatal, I - informative and W - warning respectively. Where the characters +T are shown below some string will be substituted in the actual message.

Condition	Severity	Message
9001	E	file +T not local
9002	E	file +T is not SCU library

The contents of the file +T are not recognizable by the utility as an SCU source library.

9003	E	empty SCU library - +T

There are no decks on the SCU library on file +T.

9004	E	deck +T not found
9005	E	mod +T not found
9006	E	conflict in file name - +T

An Example of when this message is issued is naming the same file for two different parameters on a command such as EXM MODNAME BA=FILE EC=FILE. The message will also be issued if the user attempts an operation that would be destructive to the result file named on the SCU.GROUP command.

9007	E	duplicate deck name - +T

The user has mentioned the same deck twice on the same command.

9008	E	duplicate mod name - +T

8.0 MESSAGES
8.1 NON-EDITOR MESSAGES

The user has attempted to create a modification with the same name as one already existing on the source library.

9009 E deck +T already on library

The user has attempted to create a deck with the same name as one already existing on the source library.

9010 E emoty deck - +T

No text lines have ever been introduced into deck +T.

9011 E invalid deck range - +T to +T

One of the deck names given as one of the ends of a range does not exist on the library or they are not in order of occurrence on the library.

9012 E invalid mod range - +T to +T

One of the modification names given as one of the ends of a range does not exist on the library or they are not in order of occurrence on the library. Displaying the modification list for the library should suggest which.

9013 E invalid parameter value for +T

An invalid value has been given for parameter +T on the command or directive currently being scanned.

9014 E unexpected +T encountered

+T is an unexpected text embedded directive (ELSE, IFEND or TEXTEND) which has been encountered in the text of the deck currently being processed.

9015 E expected +T not found

SCU has searched to the end of a deck without encountering a matching text embedded directive (IFEND or TEXTEND).

9016 E user without proper authority

The user has attempted to perform some commmand which is restricted to one of the user names assigned authority for the library.

8.0 MESSAGES
8.1 NON-EDITOR MESSAGES

9017 E deck +T already interlocked

9018 E illegal directive +T

The expression given on the displayed *IF text embedded directive is not recognizable.

9019 E too many nested IF*s

The user has nested his IF directives beyond the depth currently processed by SCU (32).

9020 E too many alternate bases

SCU currently allows the specification of at most 128 libraries on those commands that allow multiple libraries.

9021 E +T value too large

+T is the keyword for the parameter which was given a value too large.

9022 E mod +T not state 4

9023 E mod +T state 4

9024 E +T

In the case of this message the entire error message is substituted and should explain itself.

9025 E file +T without write permission

9026 E grouping commands. BASE specified, RESULT must be specified.

9027 E feature +T not found

9028 E group +T not found

9029 E different key character on file - +T

9030 I line +T in deck +T truncated

The indicated line in the indicated deck was truncated on an EXPAND_DECK or EXTRACT_DECK command. A larger width will have to be supplied on the command or the line will have to be split.

8.0 MESSAGES
8.1 NON-EDITOR MESSAGES

9031	E	interlock violation in deck +T

An interlock violation has occurred for the indicated deck during the processing of a COMBINE_LIBRARY or REPLACE_LIBRARY command.

9090	F	unable to allocate +T

+T refers to tables allocated in memory such as p_mod_names, used for the list of modification names applying to a deck. Short of restructuring his libraries there is little the user can do to correct this problem.

9099	F	internal error +T

SCU has detected an internal error explained in the text of +T. An example might be a library pointer out of range. There is no remedial action available to the user beyond submitting a PSR documenting the problem so that the failure wherever it may be can be sought out and corrected.

8.2 EDITOR MESSAGES

Below are listed the messages issued by the SCU editor procedures. The condition numbers are shown relative to 8000. This origin may change on NOS 170 and will change on NOS/VE. The letters given in the severity column have the following significance E - error, F - fatal, I - informative and W - warning. The characters +T will be replaced with other text in the actual messages.

Condition	Severity	Message
8001	E	Unknown command: +T

The string of characters +T was not recognizable as an SCU editor command.

8002	E	No deck active

The user has not indicated to which deck on the library editor commands are to apply.

8003	E	Base and result files must be different files

8.0 MESSAGES
8.2 EDITOR MESSAGES

8004 E Continue specified for unknown modification +T

The user has specified continue on the editor call command and the modification +T does not exist on the library. Issuing a DISPLAY_MODIFICATION_LIST command for the library may point out a misspelling.

8005 E Duplicate modification name +T

The user has called the editor naming a modification name which already exists on the library without specifying the CONTINUE keyword. He must either choose a different modification name or specify the keyword CONTINUE to add to the named modification.

8006 E Modification +T not in state 0

SCU does not allow modfications in states other than 0 to be changed.

8007 E Invalid Modification name

8008 E Invalid keyword value for NUMBER - +T

The only valid keyword value for the number parameter is ALL or A.

8009 E Value for NUMBER must be positive integer or keyword

8010 E Invalid file name

8011 E Invalid file position specified

Valid keywords for file positioning are REWIND or R, REWIND_BEFORE or RB, REWIND_AFTER or RA and NO_REWIND or NR.

8012 E Column number too large

Values for the COLUMNS parameter must be between 1 and 256.

8013 E Width greater than maximum line length

Values for width must be between 0 and 256.

8014 E Width greater than maximum width for

deck

The user has attempted to give a value for width greater than that in the deck header on the library.

8015 E Modification name +T too long

8016 E Line identifier must start with modification name

8017 E Sequence number too large

The largest value SCU allows for a sequence number is 262143.

8018 E Sequence number required after modification name

8019 E Invalid keyword value +T

An invalid keyword value has been given for the LINES or AFTER/BEFORE parameter.

8021 E Keyword ALL cannot be used in a range

8022 E Range of lines cannot be specified for a backward search

8023 E Unknown modification name +T

The SCU editor does not recognize +T as the name of a modifcation on the library.

8024 E Line at end of range is inactive

The editor operates on active lines only.

8025 E Last line in range not found after first line

8027 E First line in range not found before last line

8028 E First line in range not found

8029 E First line in range not active

The editor operates on active lines only.

8.0 MESSAGES
8.2 EDITOR MESSAGES

8030 E Text string +T not found

8031 I +T occurrences found

This message informs the user how many occurrences of a text block were found if the user asked to find all or some number greater than existed in the range of lines to be searched.

8032 E No active lines in range

The editor operates on active lines only.

8033 I +T lines to end of range

This message informs the user how many lines were found in the remainder of the range of lines to be accessed if the user asked to operate on all lines or some number greater than exist in the remainder of the range.

8034 W Line is longer than maximum line length

One of the lines the user has introduced was greater than 256 characters in length. It has been truncated to 256 characters.

8035 I Begin editing deck +T

8036 E No start of deck

The structure of the library is not intact. There is no remedial action available to the user.

8037 E History array for line has overflowed

The line in question has been modified more than 255 times.

8038 W Line is longer than current line width

This is a warning that the user has introduced a line longer than the current line width and should consider splitting the line where appropriate.

8039 E Invalid keyword value for occurrence - +T

8.0 MESSAGES
8.2 EDITOR MESSAGES

The only acceptable keyword value for the OCCURRENCE parameter is LAST or L.

8040	E	Value for occurrence must be integer or keyword

9.0 SUGGESTIONS ON SCU USE

9.0 SUGGESTIONS ON SCU USE

The degree of control necessary over corrections to source data blocks being maintained for an individual for his own convenience and over the source code for a major operating system component being maintained by an integration and evaluation group is very different. For that reason the two groups of suggestions are offered below.

9.1 PRIVATE SOURCE CODE MAINTENANCE

For the convenience of an individual maintaining a private source library it is suggested that he set all authorities to his own user name. Probably the only criteria file directive that he would need to use would be DECLARE, which is useful in allowing alternate blocks of source code to be written to an expanded text file.

9.2 GROUP SOURCE CODE MAINTENANCE

It is suggested that for maintenance of complex blocks of source data that may be required to be accessible to a large group of implementors that the authority and interlock features of SCU be used to ensure control over modifications being made. It is suggested that a user who is assigned the responsibility to implement a correction to a particular deck extract a subset library containing only the deck that he is to modify, setting the interlock for that deck in the process.

The individual could then enter his correction and make trial use of it from his subset source library satisfying needs for other decks from the base library specified as an alternate. The individual would finally submit his modified subset library for code review and combining with the base library with the new corrections in state 0. The interlock would be cleared when the combine was done.

As various stages of testing are completed the person in authority can grant that the new corrections be raised in state or if testing indicates that the correction is not

9.0 SUGGESTIONS ON SCU USE
9.2 GROUP SOURCE CODE MAINTENANCE

acceptable can lower the state of the correction to state 0 and contact the person responsible to make an adjustment before resubmitting the correction.

Expanded text files can be built with all available corrections on the base source library to check for conflicts between developers or at the other extreme can be built with only released code to produce a version of the source data at some significant milestone.

The DISPLAY_LIBRARY, DISPLAY_DECK, DISPLAY_MODIFICATION, DISPLAY_GROUP and similar commands can be used to help document and evaluate subset source libraries prior to combining them with the base source library.

Criteria files can be used to specify the contents of a complex expanded text file. The contents of these files can of course be maintained on the source library themselves.

Table of Contents