# SEMINAR NO. DA3850

# NOS SES TOOLS

## STUDENT HANDOUT

# REVISION RECORD

| REVISION | DESCRIPTION |
|---|---|
| A | Manual Release. |
| (07-22-83) | |
| B | Manual Updated.(B2) |
| (02-02-88) | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |

Publication No.
DA3850-1

REVISION LETTERS I, O, Q AND X ARE NOT USED

NOS SES TOOLS

Address comments concerning this manual to:

CONTROL DATA CORPORATION
PS CYBER Training
ETCSEM
1450 Energy Park Dr.
St. Paul, MN 55108

or use Comment Sheet in the back of this manual.

# TABLE OF CONTENTS

# GENERAL COURSE DESCRIPTION

Course Title: NOS SES Tools

Course Number: DA3850

Course Length: Two days

Description:

The Tools course introduces the student to the set of procedures developed by the Software Engineering Services (SES) group to support software development under the NOS operating system. Several tools, utilities and products, which are also supported by the SES group, are introduced as well.

Prerequisites: NOS Usage is highly recommended.

Course Objectives:

Generally, the student should become familiar with and be able to use the SES procedures. The student should also be able to invoke and use the tools, utilities, and system software that the procedures access.

Specific objectives are listed at the beginning of each chapter. Note that these objectives should be met with the aid of class notes and the documents used in class.

Reference Material:

A list of reference documents is included at the beginning of Lesson 2 of this book.

# COURSE CHART

| HOUR | DAY 1 | DAY 2 |
|---|---|---|
| 1 | INTRODUCTION | C170 ENVIRONMENT |
| 2 | GETTING INFORMATION | DOCUMENT FORMATTING SYSTEM |
| 3 | LAB<br>FILES | LAB<br>CONVERSION UTILITIES |
| 4 | COMPILERS<br>SOURCE LIBRARIES | MISCELLANEOUS PROCEDURES<br>LAB |
| 5 | | |
| 6 | LAB | |

# COURSE CHART

<table>
<tr><th>HOUR</th><th>DAY 1</th><th>DAY 2</th></tr>
<tr><td rowspan="2">1</td><td rowspan="3">INTRODUCTION</td><td rowspan="2">C170<br>ENVIRONMENT</td></tr>
<tr></tr>
<tr><td rowspan="2">2</td><td rowspan="2">DOCUMENT<br>FORMATTING<br>SYSTEM</td></tr>
<tr><td rowspan="2">INFO<br>LAB</td></tr>
<tr><td rowspan="2">3</td><td>LAB</td></tr>
<tr><td>FILES</td><td>CONVERS</td></tr>
<tr><td rowspan="2">4</td><td>COMPILERS</td><td>MISC</td></tr>
<tr><td rowspan="3">SOURCE<br>LIBRARIES</td><td rowspan="5">LAB</td></tr>
<tr><td rowspan="2">5</td></tr>
<tr></tr>
<tr><td rowspan="2">6</td><td rowspan="2">LAB</td></tr>
<tr></tr>
</table>

# LESSON 1

## SES INTRODUCTION

# SES TOOLS COURSE OUTLINE

1. SES Introduction

   A. SES Procedures
   B. Procedure Syntax
   C. Profile Files
   D. Documentation

2. Getting Information

   A. Documents
   B. General Information
   C. SES Information

3. Files

   A. Printing Files
   B. File Manipulation

4. Compilers and Assemblers

   A. Compilers
   B. Assemblers
   C. CYBIL Reformatter — PASCAL EXTENSION LANGUAGE -

5. Source Libraries

   A. Module Maintenance
   B. Correction Set Maintenance
   C. Utilities
   D. UPDATE Modification Sets

6. C170 Environment

   A. Link and Execute
   B. Object Code Maintenance
   C. Debugging

# LESSON 1

## SES INTRODUCTION

Lesson Preview:

This lesson consists of an introduction to the course, an introduction to the SES procedures. Profile files and communication system.

Objectives:

After completing this lesson, the student will be able to:

- State who maintains the SES procedures and how the procedures are accessed.
- Describe the mode of operating that is established by the use of the procedures and the rationale for this particular set of tools.
- Describe the syntax of SES procedures and the kinds of parameters they accept.
- State what a profile is, how to create one, and what kind of information it contains.
- Explain the parameters which are valid on batch procedures.
- Send messages between users.

References:

- NOS V2 Reference Set, Volume 3, System Commands
- SES User's Handbook - Chapters 1 and 14
- SES User's Handbook - Appendix A, B, C, D, E

# SES

- A group
  - Responsible for software development tools, design, and implementation.
- A project
  - To build a system which will provide tools and access to them.
- An access method
  - Interactive
  - Run under NOS
  - Oriented to software development
- A set of procedures
  - Providing access to development tools, utilities, and standard software.

# TYPES OF PROCEDURES

- Formatting documents
- Printing files
- Maintaining source text files
- Maintaining library (object) files
- Retrieving information
- Manipulating files
- Accessing utilities (for example, conversion routines)
- Compiling, linking, and debugging C170 programs

# SES COMMANDS

```
FAMILY:    KAS
USER NAME: DL0127
PASSWORD:  ########
APPLICATION: IAF

JSN: AAUF, NAMIAF

 READY
ASCII
 READY
BATCH
RFL,0
/SES.FTN5 I=SES1
*    COMPILING  SES1
REVERT.   END FTN5
/LGO
SUM IS   30
     0.003 CP SECONDS EXECUTION TIME.
/SES.LINK170
REVERT.    END LINK170    LGOB
/LGOB
SUM IS   30
     0.003 CP SECONDS EXECUTION TIME.
/REWIND,LISTING
REWIND,LISTING.
/COPY,LISTING
1             PROGRAM ONE                 OPT=0,ROUND= A/ S/ M/-D,-DS     FTN 5
VERSION    YY/MM/DD. HH.MM.SS       PAGE     1
TWO MORE LINES OF
OPTIONS AND PARAMETERS

        1                    PROGRAM ONE
        2                    IA=10
        3                    IB=20
        4                    IC=IA+IB
        5                    PRINT 10,  IC
        6              10    FORMAT ('SUM IS ',I4)
        7                    END
```

# SES COMMANDS

```
FAMILY:    kas
USER NAME: jhw127
PASSWORD:  ########
APPLICATION: iaf

JSN: AAUF, NAMIAF

 READY
ascii
 READY
batch
RFL,0
/ses.ftn5 i=ses1
*    COMPILING  SES1
REVERT.   END FTN5
/lgo
SUM IS   30
     0.003 CP SECONDS EXECUTION TIME.
/ses.link170
REVERT.    END LINK170    LGOB
/lgob
SUM IS   30
     0.003 CP SECONDS EXECUTION TIME.
/rewind,listing
REWIND,LISTING.
/copy,listing
1            PROGRAM ONE                OPT=0,ROUND= A/ S/ M/-D,-DS     FTN 5
VERSION    YY/MM/DD. HH.MM.SS       PAGE     1
TWO MORE LINES OF
OPTIONS AND PARAMETERS

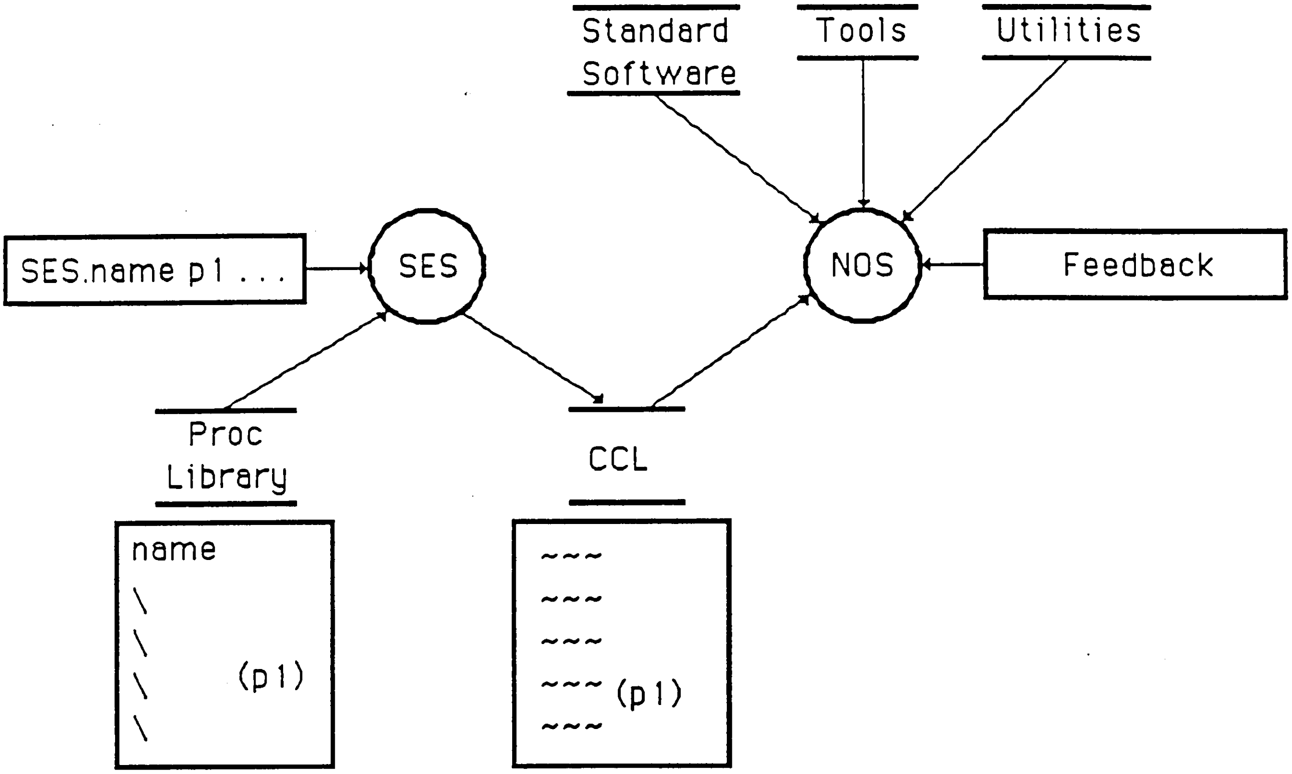
        1                   PROGRAM ONE
        2                   IA=10
        3                   IB=20
        4                   IC=IA+IB
        5                   PRINT 10,  IC
        6             10    FORMAT ('SUM IS ',I4)
        7                   END
```

WHEN USING SES MAKE CERTAIN THAT YOU ARE IN ASCII MODE

XP=1 DB=ID

TURN ON DEBUG.

## SES PROCESSOR


Standard Software
Tools
Utilities
SES.name p1 . . .
SES
NOS
Feedback
Proc Library
CCL
name
\
\
\ (p1)
\
~~~
~~~
~~~
~~~ (p1)
~~~

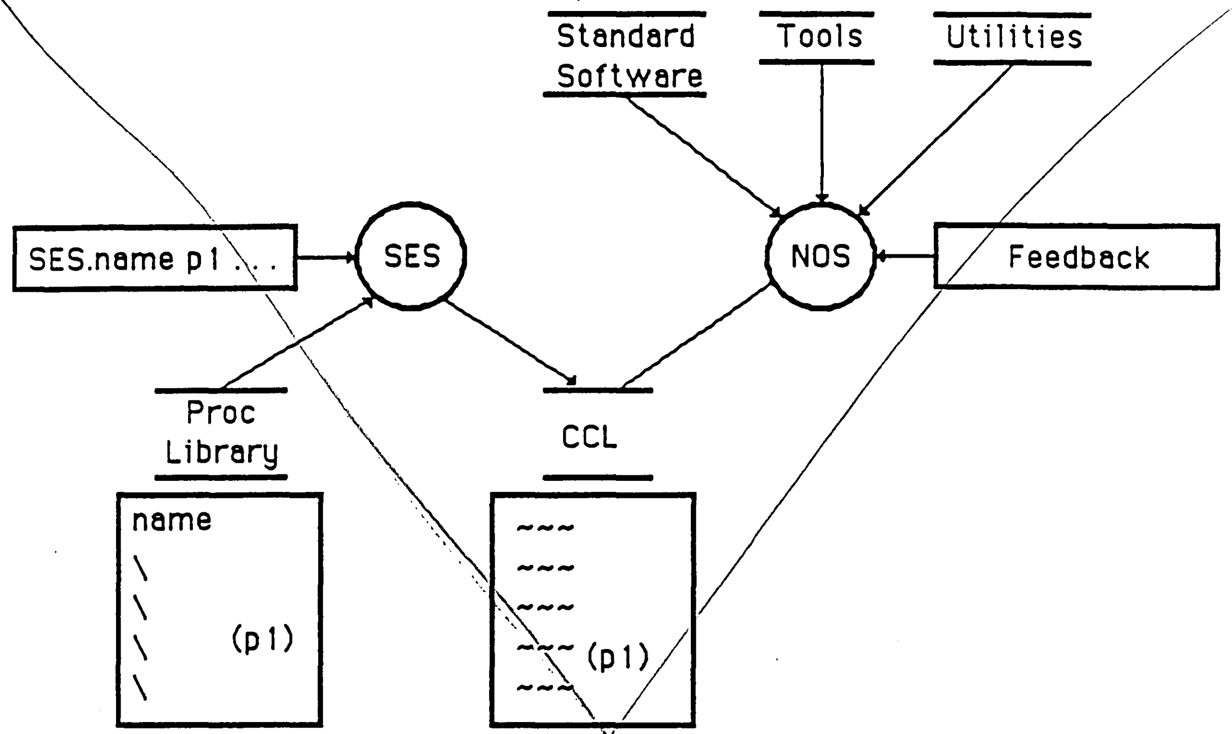
1. SES.name p1. . . is a file name call. The program SES is loaded and executed.

2. SES finds the proc, substitutes values based on the passed parameters, and produces CCL (CYBER Control Language) to do the job.

3. SES sends these commands to NOS on behalf of the initiator.

4. The procedure file contains commands to perform the function and provide feedback.
~~~

# SES PROCESSOR


Standard Software
Tools
Utilities
SES.name p1 . . .
SES
NOS
Feedback
Proc Library
CCL
name
\
\
\ (p1)
\
~~~
~~~
~~~
~~~ (p1)
~~~

1. SES.name p1. . . is a file name call. The program SES is loaded and executed.

2. SES finds the proc, substitutes values based on the passed parameters, and produces CCL (CYBER Control Language) to do the job.

3. SES sends these commands to NOS on behalf of the initiator.

4. The procedure file contains commands to perform the function and provide feedback.
~~~

# SCL SYNTAX

SYS CONTROL LANGUAGE

- Command/Parameter separators
  - Comma (,)
  - Blank
  - Both

- Parameter specification
  - Keyword
  - Positional
  - Both

- Keyword/value separators
  - Equal sign (=)
  - Blank
  - Both

- Value specification
  - Name
  - Number
  - String
  - Range
  - Value list

# PARAMETER SPECIFICATION

- Keyword

```
SES.COPYACR i=inpf o=outf cols=2. .72

SES.PRINT f=glomp n=3 , ,='JERRY'
```

- Positional

```
SES.COPYACR inpf outf

SES.PRINT glomp h='JERRY'
```

- Mixed

ASCII CODED RECORD

```
SES.COPYACR o=outf 2. .72 i=inpf
```

RANGE

```
SES.PRINT glomp h='JERRY' n=3
```

# PARAMETER SPECIFICATION

- Keyword

```
SES.COPYACR i=inpf o=outf cols=2. .72

SES.PRINT f=glomp n=3 h='JERRY'
```

- Positional

```
SES.COPYACR inpf outf

SES.PRINT glomp h='JERRY'
```

- Mixed

```
SES.COPYACR o=outf 2. .72 i=inpf

SES.PRINT glomp h='JERRY' n=3
```

# PARAMETER VALUES

- Name

  - Names are currently 1 to 7 characters long
  - The first character must be a letter, the rest, letters or numbers
  - Example: SES.PRINT f=glomp

- Number

  - Must be integers base 10, 8, or 16. 10 is default.
  - Numbers must start with a digit.
  - Example: SES.PRINT c=0A3(16) f=glomp

- String

  - Enclosed in apostrophe (').
  - Maximum of 80 characters.
  - Example: SES.PRINT glomp h='Jerry'

- Range

  - The upper and lower bound are separated by . .
  - Example: SES.GENCOMP name1 . . name2

- Value list

  - The list is in parentheses.
  - The list can be any of the first 4 (basic) types.
  - Example: SES.GENCOMP (namea, nameb, namec)

- KEY

  - No value is associated with the keys.
  - The keys will select or deselect some feature.
  - Example: SESPRINT f=glomp shift

# PARAMETER COMBINATIONS

- Multiple calls

```
SES.PRINT myfile noshift; PRINT ourfile

SES.PRINT myfile; 'PURGE,myfile'; 'BYE'
```

- Continuation lines

```
SES.PRINT myfile . .

. .?noshift c=3

SES.GETMOD g=glomp b=uspx; EDT glomp;. .

. .?REPMOD b=uspx g=glomp
```

SES
PROMPT  @@@>

TRY
SES. FINS
THEN WITH PROMPT.

# BATCH PARAMETERS

- Most procedures run in local mode.
- Some procedures have a batch option.
- Some procedures have batch default.
- Batch parameters

| | |
|---|---|
| JOBUN = | User name |
| JOBPW = | Password |
| JOBFMLY = | Family |
| JOBCN = | Charge number |
| JOBPN = | Project number |
| JOBFL = | Field length |
| JOBTL = | Time limit |
| JOBPR = | Priority |
| Key: | LOCAL |
| | BATCH |
| | BATCHN |
| | DEFER |
| Key: | NODAYF |
| | DAYFILE |
| | DF |
| | DAYFILE=filename |
| | DF=filename |

- Dayfile option is not available on some procedures.

# PROFILE

- The profile file is created by, and belongs to, the user (you). It is used to 'tell' SES things about you and the things you are working on. Profile variables provide default values in many procedures.

- There are profile variables for:

  - Source code files
  - Library files
  - Print options
  - Mailbox name
  - Batch job parameters
  - And many more

- A profile file can be build using SES.BLDPROF. Additional variables can be added using an editor.

# USING PROFILES - 1

```
/ses.format sesl
PASSWORD NOT GIVEN
```

## BUILD PROFILE

```
/ses.bldprof
WHAT IS YOUR NAME? (NOT YOUR USER NAME BUT YOUR REAL NAME!)
? david olson
WHAT IS YOUR PASSWORD?
? icr6
WHAT IS YOUR CHARGE NUMBER?
? 9059
WHAT IS YOUR PROJECT NUMBER?
? 6rd20303
ENTER TERM.TYPE(CDC713,CDC751,CDC752,DX132A,NCR260,TI745,TTY43,
DECW3,TEK4014, IST3, CDC721 OR Z19)
? cdc722_30
ENTER GRAPHICS TERMINAL TYPE(CDC721, IST3 OR TEK4014)
?
REVERT.    PROFILE AND MAILBOX CREATED
```

## LIST PROFILE

```
/get,profile
/list,f=profile
PROFILE
\  MYNAME  = 'david olson'
\  PASSWOR = 'icr6'
\  CHARGE  = '9059'
\  PROJECT = '6RD20302'
\  IF '&PROCNAM&' = 'IAF' THEN
\    TERM    = 'cdc722_30'
\  IFEND
```

## BATCH JOB SUBMITTAL

```
/ses.format sesl
 09.19.00.  SUBMIT COMPLETE.  JSN IS ADAI.
REVERT.    JOB  FORMAT  SUBMITTED
```

# USING PROFILES - 2

```
/ses.edt
 BEGIN TEXT EDITING.
? end
 END TEXT EDITING.
REVERT.    END EDT  GROUP
```

## ADD A VARIABLE

```
/ses.edt profile
 BEGIN TEXT EDITING.
? add;*.1;*
 ENTER TEXT.
? /\  group = 'dlo'/
 READY.
PROFILE
\  MYNAME  = 'david olson'
\  PASSWOR = 'icr6'
\  CHARGE  = '9059'
\  PROJECT = '6RD20302'
\  IF '&PROCNAM&' = 'IAF' THEN
\    TERM    = 'cdc722_30'
\  IFEND
\  group = 'dlo'
 -END OF FILE-
? end
 END TEXT EDITING.
REVERT.    END EDT  PROFILE
```

## USE NEW VARIABLE

```
/ses.edt
 BEGIN TEXT EDITING.
? end
 END TEXT EDITING.
REVERT.    END EDT  DLO
```

# MAIL

- Procedures

    - SES.MAIL file TO name

        + Sends mail to user-names or profile-names.

    - SES.GETMAIL file SEQ

        + Display mail and optionally get sequence numbers added.

    - SES.ANYMAIL

        + Find out how much mail you have received.

    - SES.WHOMAIL

        + Find out who sent you mail.

    - SES.SAVMAIL title

        + Save mail on direct file OLDMAIL with your record name of Mmmddyy.

    - SES.NEWMAIL

        + Purge mailbox and make a new one.

# LESSON 2

# GETTING INFORMATION

# LESSON 2
## GETTING INFORMATION

Lesson Preview:

This lesson discusses retrieving information about your local environment on NOS and retrieving information about SES documents and procedures.

Objectives:

After completing this lesson, the student will be able to:

- Use SES procedures to get status and list information about files.
- List procedures and parameters of SES procedures.
- Get current information from the SES project.
- Use help mode and the experimental catalog.
- Print current documents.

# GETTING INFORMATION

- Documents

  SES.TOOLDOC

- File information

  SES.CATLIST
  SES.CATALOG
  SES.FILES
  SES.PERMIT
  SES.DAYFILE
  SES.LIMITS
  SES.DISPLAY
  SES.TIME
  SES.DAYWEEK
  SES.EXPLAIN

- SES Information

  SES.INFO
  SES.SESPROC
  SES.SESPARM

# RESOURCE DATA

- Used in class:

  - SES User's Handbook (60457250)
  - TXTCODE User Guide (60460280)

- Reference:

  - NOS V2 Reference Set, Volume 3, System Commands (60459680)
  - CYBER Loader V1 Reference Manual (60429800)
  - CYBER Loader V1 User's Guide (60482300)
  - Modify V1 Reference Manual (60450100)
  - Update Reference Manual (60449900)
  - TXTFORM (60460290)
  - Text Editor V1 Reference Manual (60436100)

- CYBIL Related:

  - CYBIL Reference Manual (60455280)
  - NOS CYBIL Student Handout (DA3800)

# TOOLDOC

- Procedure

  SES.TOOLDOC documents listing

- Examples:

## FILE INFORMATION

- SES.CATLIST un,pn,o,fpl

  - Lists the type, security category, access mode, and length of files in your catalog. Optionally, you can select username, packname, and the number of files to describe on each output line.

- SES.CATALOG l or b, un,o,LONG

  - Lists the records in a file. The long option is the way NOS does it.

- SES.FILES o,ALL

  - Displays local file information.

- SES.PERMIT o,f

  - Displays the permission or security information about the files in your catalog.

# GENERAL INFORMATION

- SES.DAYFILE selections from your dayfile

```
n,list = number_of_lines [14]
find='search  string'
o,output=output_file  [OUTPUT]
```

- SES.LIMITS NOS limits on your job

```
o,output=output_file  [OUTPUT]
```

- SES.DISPLAY display any or all of the following:

```
expr,e=expression
KEYS: date,time,clock,
      fl,
      r1,r2  r3,ef,
      jcr,
      sw,
      all
```

- SES.TIME talk the time

- SES.DAYWEEK calculate the day of the week

```
day=n
month=n or MMM [current]
year=nnnn [current]
o,output=output_file  [OUTPUT]
```

## DAYFILE

```
/ses.dayfile
 -END OF FILE-
 12.57.33.$UNLOAD(ZQCNX3M,ZQCNX3Q)
 12.57.33.$COPYBR(ZQCNX3F,OUTPUT)
 12.57.33. COPY COMPLETE.
 12.57.33.$UNLOAD(ZQCNX3F)
 12.57.33.$RFL(0)
 12.57.33.$SKIP(ZQCNX49)
 12.57.33.SENDIF(ZQCNX49)
 12.57.33.REVERT.    END LIMITS
 12.58.12.SES.DISPLAY ALL
 12.58.13. -\SES\ JHW127 DISPLAY SESPLIB SES-
 12.58.14.- SES         00.00.02  0.512     0.325
 12.58.14.BEGIN,,ZQSES02.
 12.58.14.$RETURN(ZQSES02)
 12.58.14.REVERT.    END DISPLAY
REVERT.    END DAYFILE
/ses.dayfile n=1 find 'un=ses'
       3  OCCURRENCES OF PHRASE FOUND.
 11.39.33.EDT(ZQ9VMDM,0,OUTPUT)ASCII.S,?UN=SES?;*.L;1
REVERT.    END DAYFILE
```

## DISPLAY

```
/ses.display all
 FL = 0(8),     R1 = 0,     R2 = 0,     R3 = 0,     R1G = 0
 EF = 2, 2(8),     EFG = 2, 2(8),     SW = (F,F,F,F,F,F)
   JANUARY 15, yyyy,     FIVE TO ONE

/ses.limits
S
1LIMITS FOR DL0127  FM/KAS                    yy/mm/dd. hh.mm.ss.

  INTERACTIVE PASSWORD EXPIRATION DATE............ yy/mm/dd.

 MAXIMUM NUMBER OF -
  MAGNETIC TAPE UNITS THAT MAY BE ASSIGNED........          3
  REMOVABLE AUXILIARY DEVICES THAT MAY BE ASSIGNED          2
  CPU SECONDS ALLOWED FOR EACH JOB STEP........... UNLIMITED
  CENTRAL MEMORY WORDS ALLOWED/100B...............      3037B
  EXTENDED MEMORY WORDS ALLOWED/1000B.............       200B
  JOBS THAT CAN BE DETACHED.......................          3
  DEFERRED BATCH FILES............................         16
```

# SES INFORMATION

- SES.INFO information from the SES project

  o,output=output_file  [OUTPUT]

- SES.SESPROC SES procedure names

  un=(user names)
  o,output=output_file  [OUTPUT]

- SES.SESPARM parameters of procedures

  l=library_name
  un=user_name
  o,output,listing=output_file  [LISTING]
  Batch job parameters
  print=(print procedure parameters)

  - See handbook for interpretation of parameter list

# SES STATEMENT

- Format:

  SES,parameter.ses proc,proc params

  - parameters:

    A. MODES:
    HELP
    STATUS
    TEST
    RUN

    B. PROFILE CONTROL:
    PUN=
    P=

    C. LIBRARY CONTROL:
    UN=
    LPFN=

    D. DEFAULT CATALOG CONTROL:
    DUN=
    DPN=

# SES,MODES

## HELP

```
/ses,help.catlist

 CATLIST produces a sorted and compressed form of the NOS CATLIST command.
 ---PARAMETER---DEFAULT-------------ALLOWABLE VALUE(S)-----------
     un          current user        username
     pn          current packname    packname
     o           output              filename
     fpl         3                   1..10 (files per output line)
 HELP FOR CATLIST ON FILE OUTPUT
/ses,h=cathlp.catlist
 HELP FOR CATLIST ON FILE CATHLP
/ses,h.permit

    PERMIT produces a listing of the permit information for
 all or a selection of a user's permanent files.  Parameters are:
 ---PARAMETER---DEFAULT----------------ALLOWABLE VALUE(S)---
     o           output                   filename
     f,fn        omitted (all files)      filename(s)
 HELP FOR PERMIT ON FILE OUTPUT
```

## TEST

```
 /ses,test.permit ses1
 TEST OF PERMIT ON FILE SESTEST
 /copy,sestest
 .PROC,SESTEST.
 ACQUIRE(ZQ6D8Y1=SES5151/A,PO,M=E,UN=SES)
 ZQ6D8Y1(ZQ6D8ZF,,,1)
 EDT(ZQ6D8ZF,ZQ6D8YD,ZQ6D8ZQ)
 $BEGIN(ZQ6D8ZF,ZQ6D8ZF)
 $PACK(ZQ6D8ZK)
 EDT(ZQ6D8ZK,ZQ6D8YH,ZQ6D8ZQ)
 $UNLOAD(ZQ6D8YH,ZQ6D8ZF,ZQ6D8ZQ)
 $COPYCF(ZQ6D8ZK,SES1)
 $UNLOAD(ZQ6D8ZK)
 REVERT.    END PERMIT
  EOI ENCOUNTERED.
```

# PROFILE SEARCH DIRECTIVE

- Format: \SEARCH search_spec,search_spec. . .

  search_spec:

  - user_name

    or
  - (lib_name,lib_name . . .,user_name)

- Example:

```
/ses.getbase
GETBASE NOT FOUND
/ses,libpfn=prjplib,un=lib.getbase
 ** F CL 11007:  REQUIRED PARAMETER MISSING "SS" ON COMMAND STATEMENT
SES,LIBPFN=PRJPLIB,UN=LIB.GETBASE
                                 ↑
/ses.edt profile
 BEGIN TEXT EDITING
? add;*.1;*
 ENTER TEXT
? /\  search    (prjplib,lib) dbs, ses
 READY
PROFILE
\  MYNAME  = 'david olson'
\  PASSWOR = 'E372232'
\  CHARGE  = '9059'
\  PROJECT = 'test'
\  IF '&PROCNAM&' = 'IAF' THEN
\    TERM     = 'cdc722_30'
\  IFEND
\  group = 'dlo'
\  search    (prjplib,lib) dbs, ses/
 -END OF FILE-
? end
 END TEXT EDITING
REVERT.    END EDT  PROFILE
/replace,profile
/ses,h.getbase
     GETBASE gets Base PLs from the default, or
 a specified user catalog.  The default base user is
 ARTECS.

 ---PARAMETER------DEFAULT-----ALLOWABLE VALUES----PROFILE
    ss, pref        none/REQ    ARTECS PL subsys. ids
    un              *ARTECS     base pl user name   PROJUN
    pn              *LIB        base pl pack name   PROJPN
    r               *DL         base pl device type PROJDT
    abort,noabort    noabort  . failure action
    msg,nomsg       *msg        keyword             MSGCTRL
 HELP FOR GETBASE ON FILE OUTPUT
```

# LESSON 3

# FILES

# LESSON 3
## FILES

Lesson Preview:

This lesson covers printing files and dumping and reloading files.

Objectives:

After completing this lesson, the student will be able to:

- Use the print procedures
- Describe the profile variables which affect printing.
- Dump files from your catalog to tape and reload them.

References:

- SES User's Handbook; Chapters 3, 8
- SES User's Handbook; Appendix I, J

# PRINTING FILES

- Procedures
  - SES.PRINT
  - SES.PRINTID
  - SES.COPYSAF

- Features
  - Multiple files
  - Multiple copies
  - Heading banners
  - Printer selection
  - Character set selection
  - Shift lines automatically
  - Shift lines as specified
  - Date and time format

- Profile variables
  - Banner formatting
  - Character set
  - Time and date format
  - Number of copies
  - Printer selection
  - Shift option

# PRINT PROCEDURES

```
SES.PRINT
   f,i=(files to print)
   copies,c,n=number_of_copies [* or 1]
   h,id='heading banner' [* or filename, date, time]
   hn,idn=number_of_headings [* or 2]
   shn,sidn=number_of_sub-headings [* or 2]
   printer,pr=printer_type [* or ascii]
   KEY: compres
   KEY: [* or cs612]
   KEY: shift [*]
   KEY: DATE(ddMMM,yy) or ETAD (mm/dd/yy) [date]
   KEY: TIME(hh.mm.ss) or AMPM(h:mmXM) [ampm]
   un=
   fc=
```

*=profile variable

- Print profile variables

| | |
|---|---|
| MYID | no (eliminate banner) |
| COPIES | n |
| HN | n |
| SHN | n |
| PRINTER | ascii or ascii64 or cdc |
| INCSET | cs612 or cs64 or cs812 |
| SHIFT | shift or noshift or cmps |
| PRINTER_UN | |
| PRINTER_FC | |

- Examples

```
SES.PRINT (filea,fileb,filec)

SES.PRINT formout, c=3, NOSHIFT

SES.PRINT suh id='SES/USER''S/HANDBOOK'
```

# PRINT PROCEDURES (CONT.)

- SES.PRINTID h,n,o,date,time,ocset
  - Builds the large print heading banner. It is used by SES.PRINT.

- SES.COPYSAF i,o,icset,ocset,shift
  - Prepare a file for printing on the ASCII printer, for example, convert it to the ASCII8(CS812) character set by default.

## FILE MANIPULATION

- SES.RETAIN

  - RETAIN submits a job that will access all files in the user's catalog. If the c-parameter is specified, a catalog (list of records) will be produced. If the s-parameter is specified, a short SES style catlist is produced.

- SES.REWRITE

  - REWRITE is a building block for other procedures. An intermediate file is used to reduce the risk of loss of the input file.

```
i,rewriti=input_file
o,rewrito=intermediate_output
KEY: msg, nomsg [*,msg]
un,rewriteu=
KEY: status
```

* Profile variable name is MSG CTRL

# DUMPPF/LOADPF

- SES.DUMPPF and SES.LOADPF

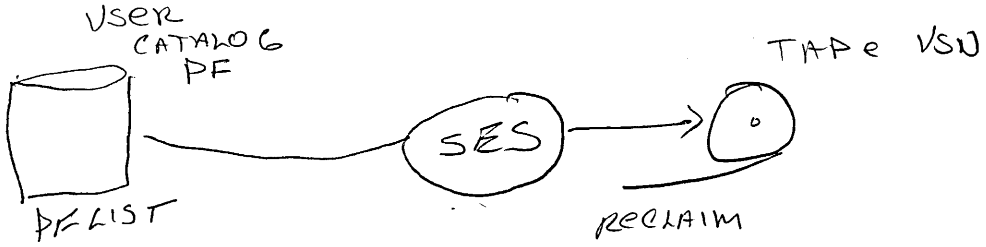
```
f,file,files=(files)   [ALL]
tape,vsn=visual_serial_number
KEY: seven,mt,nine,nt [* or NT]
dn,density=
fnt,format=
KEY: tapelist
ty,type=ALL or I or D [ALL]
KEY: lo (local files only)
   : na,noabort
xp = 'extra_parameters'
Batch Job Parameters
```

- Examples

```
SES.DUMPPF tape=666

SES.DUMPPF (conly,baker),vsn=ABC123,mt

SES.LOADPF tape=666, nopurge, xp='ct=spriv'
```



# LESSON 4

# COMPILERS AND ASSEMBLERS

# LESSON 4

## COMPILERS AND ASSEMBLERS

Lesson Preview:

This lesson covers general information about SES supported compilers and assemblers. The CYBIL formatter is also discussed.

Objectives:

After completing this lesson, the student will be able to:

- Use the SES procedures that run the compilers and assemblers supported by the SES project.

- Describe the capabilities of the CYBIL formatter, and be able to invoke it.

References:

- SES User's Handbook - Chapters 9, 13

# COMPILERS AND ASSEMBLER

- Processors

    - CYBIL
    - FTN
    - COMPASS
    - FTN5
    - COBOL5

- Preprocessors

    - CYBFORM
    - F5FORM — NESTED INDENTS
      SES.F5FORM I=,INPUT FILE, O=OUTPUT FILE

- Advantages

    - Same call format (SCL) for all processors.

    - Common parameters for all processors.

    - Procedures acquire files, provide feedback, and so forth.

# COMPILERS

| | |
|---|---|
| SES.CYBIL | i, l, b, cs, chk, lo, debug, pad, opt, msg |
| SES.FTN | i, l, b, fl, xp, debug, msg |
| SES.FTN5 | i, l, b, fl, xp, debug, msg |
| SES.COBOL5 | i, l, b, fl, xp, debug, msg |

# ASSEMBLER

| | |
|---|---|
| SES.COMPASS | i, l, b, fl, xp, msg |

- Common parameters

  i,f=source_file [COMPILE]
  l=listing [LISTING]
  b=binary_file [LGO]
  KEY: msg, nomsg [MSG]
  KEY: debug, nodebug [NODEBUG]
  xp=extra_parameters (See appropriate reference manual.)
  fl=field_length

# COMPILERS

| | |
|---|---|
| SES.CYBIL | i, l, b, cs, chk, lo, debug, pad, opt, msg |
| SES.FTN | i, l, b, fl, xp, debug, msg |
| SES.FTN5 | i, l, b, fl, xp, debug, msg |
| SES,COBOL5 | i, l, b, fl, xp, debug, msg |

# ASSEMBLER

| | |
|---|---|
| SES.COMPASS | i, l, b, fl, xp, msg |

- Common parameters

  i,f=source_file [COMPILE]<br>
  l=listing [LISTING]<br>
  b=binary_file [LGO]<br>
  KEY: msg, nomsg [MSG]<br>
  KEY: debug, nodebug [NODEBUG]<br>
  xp=extra_parameters (See appropriate reference manual.)<br>
  fl=field_length

# SES.CYBIL

```
/ses.cybil i=reform l=output cc
*     COMPILING  REFORM
1SOURCE LIST OF reform                          NOS    CYBIL/CC 1.0  version
               January 15, yyyy      1:26 PM      PAGE 1

        0      1 module   reform;
        0      2 const max=100;
        0      3 var i,j,k: integer :=0;
        0      4 procedure [xref] error;
        0      5 program main;
       13      6  /133/ begin
       17      7   if i=max
       17      8   then exit /133/
       23      9   ifend;
       23     10 end;
       23     11 error;
       24     12 procend main;
       24     13 modend;

    **** I=REFORM  L=OUTPUT B=LGO   D=OFF  LO=S  CHK=RS  OPT=0  A=1  LF=CS612
  PAD=0
0
    **** NO DIAGNOSTICS
REVERT.    END CYBIL REFORM -> OUTPUT, LGO
```

# CYBIL REFORMATTER

- SES.CYBFORM

```
i,f=input_file
o=output_file  [i,f]
```

- CYBFORM highlights
  - Capitalize keywords.
  - Lowercase identifiers.
  - Blanks squeezed to one.
  - Add spaces around delimiters and operators.
  - Indent structured statement two spaces.
  - Separate declaration statements with a blank line.
  - Put declarations on a separate line.
  - Put labels on a separate line, unindented two spaces.

- CYBFORM PRAGMATS

```
?? RIGHT:=n??
?? LEFT:=n??
??  FMT(toggle:=condition)??
```

| TOGGLE | CONDITION | DEFAULT |
|---|---|---|
| FORMAT | on/off | on |
| KEYWORD | upper/lower | upper |
| IDENT | upper/lower | lower |

# SES.CYBFORM

```
/ses.cybform i=reform o=compile
*    BEGINNING CYBFORM    REFORM --> COMPILE
*    END CYBFORM          REFORM --> COMPILE
$REVERT.CCL
/ses.cybil l=output cc
*    COMPILING  COMPILE
1SOURCE LIST OF reform                          NOS    CYBIL/CC 1.0  version
                January 15, yyyy      1:36 PM      PAGE 1

        0      1 MODULE reform;
        0      2
        0      3   CONST
        0      4     max = 100;
        0      5
        0      6   VAR
        0      7     i,
        0      8     j,
        0      9     k: integer := 0;
        0     10
        0     11   PROCEDURE [XREF] error;
        0     12
        0     13   PROGRAM main;
        0     14
       13     15   /133/
       17     16     BEGIN
       17     17       IF i = max THEN
       21     18         EXIT /133/
       23     19       IFEND;
       23     20     END /133/;
       23     21     error;
       24     22   PROCEND main;
       24     23 MODEND reform;

    **** I=COMPILE  L=OUTPUT   B=LGO   D=OFF  LO=S  CHK=RS  OPT=0  A=1
  LF=CS612  PAD=0
0
    **** NO DIAGNOSTICS
REVERT.    END CYBIL COMPILE -> OUTPUT, LGO
```

# SES.F5FORM

# LESSON 5

# SOURCE LIBRARIES

# LESSON 5
## SOURCE LIBRARIES

Lesson Preview:

This lesson covers creating and changing base libraries, applying correction sets, and listing contents of source libraries. Generating and applying UPDATE modifications is also discussed. An overview of UPDATE procs is presented.

Objectives:

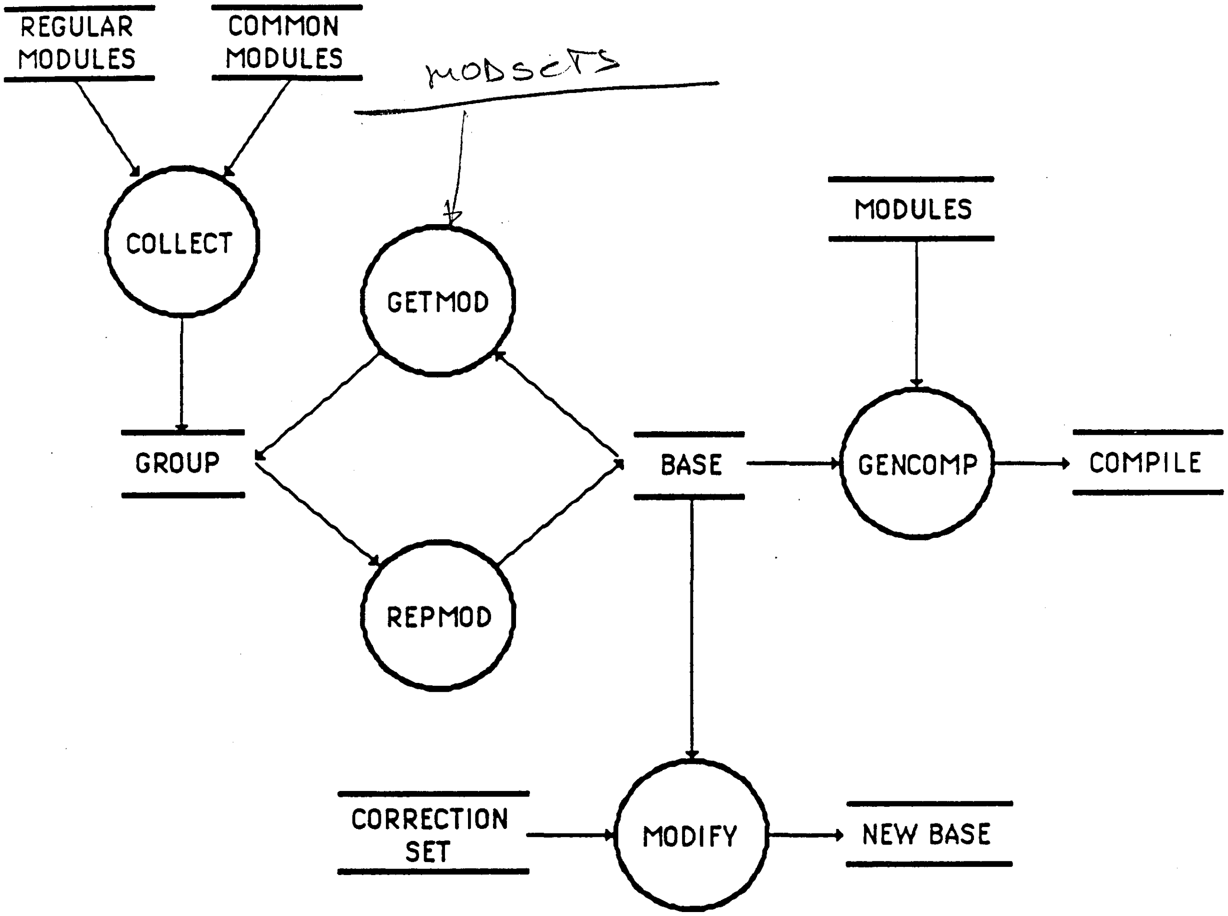
After completing this lesson, the student will be able to:

- Describe the types of source files and explain how those files are used.
- Manipulate (create, add, delete) modules of source text using SES procedures.
- Manipulate correction set files and make corrections using SES procedures.
- Use the various procedures which list, sort, and build source files.
- Describe the profile variables which affect source file maintenance.
- Describe the current capabilities of UPDATE procs.

References:

- SES User's Handbook, Chapters 5, 17
- NOS V.1 MODIFY Reference Manual
- UPDATE Reference Manual

# SOURCE LIBRARIES



- A MODULE is a single record on a base or group file. Usually modules are common or regular text.

- A GROUP is a collection of modules. Each module occupies one record.

  SOURCE CODE

- A BASE is a file in MADIFY format. A base can contain corrections as well as text modules.

# FEATURES

- Procedures

| | |
|---|---|
| SES.GETMOD | Base --> Group |
| SES.REPMOD | Group --> Base |
| SES.COLLECT | Miscellaneous Files --> Group |
| SES.GENCOMP | Base --> Compile file |
| SES.GENCOR | Generate a correction set. |
| SES.TEMPCOR | Make temporary corrections. |
| SES.MODIFY | Apply a correction set to a base. |
| SES.CATBASE | Short list of base contents. |
| SES.LISTMOD | Catalog and cross-reference and optional listing of all modules in a base. |
| SES.SORTMOD | Sort a base. |
| SES.WIPEMOD | Delete modules from a base. |
| SES.XREFMOD | Produce a cross-referenced list of common and regular modules. |
| SES.GETCOMN | Acquire CYBCCMN in MADIFY form. |
| SES.SCOOP | Compare MADIFY or UPDATE source files. |

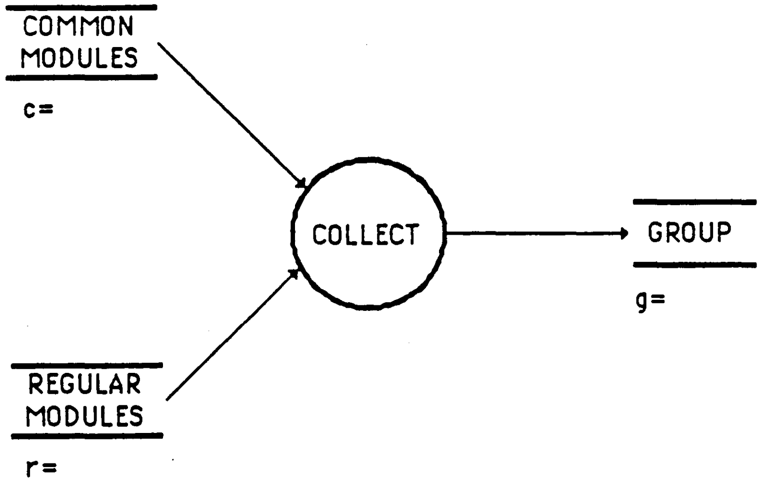
- Profile variables

| | | |
|---|---|---|
| \BASE | = | Base library name |
| \BASEOWN | = | User number |
| \NEWBASE | = | New base library name |
| \CORS | = | Correction set file name |
| \GROUP | = | Group file name |
| \CFSEQ | = | SEQ (sequence numbers on compile file) |
| \CFWIDTH | = | Compile file width |
| \SESTMPB | = | Temporary base |
| \INTRLOK | = | Interlock file name |
| \LOKMODE | = | Lock,nolock |

- Common parameters

b,l: Base name
c,com,oplc: Common modules
r,reg,opl: Regular modules
m: Module
g,group: Group file name
cors,c: Correction set file
nb,nl: New base library
lock,nolock: Lock control
msg,nomsg: Informative message control
status,sts: Retrieve NOS status for own use

# MAKE A GROUP



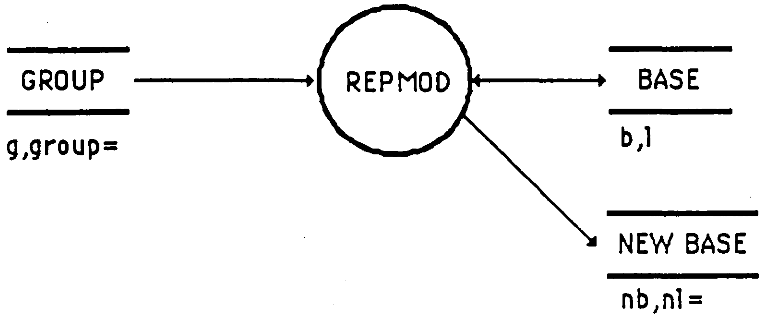
SES.COLLECT

```
    m,mem,mems,member,members
    c,com,oplc=(common  modules)
    r,reg,opl=(regular  modules)
    g,group=group_file  [*,GROUP]
    KEY: msg,nomsg [*,msg]
```

*   =  profile  variable

c and r cause names to be added to records

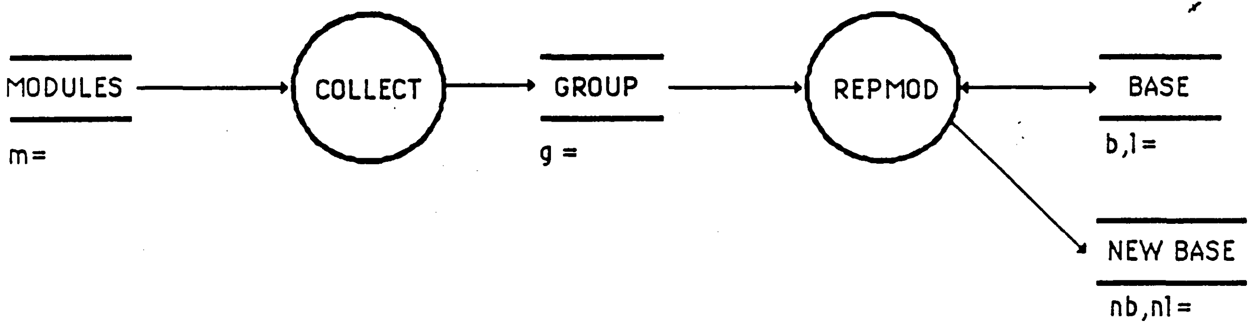
# REPLACE/ADD MODULES



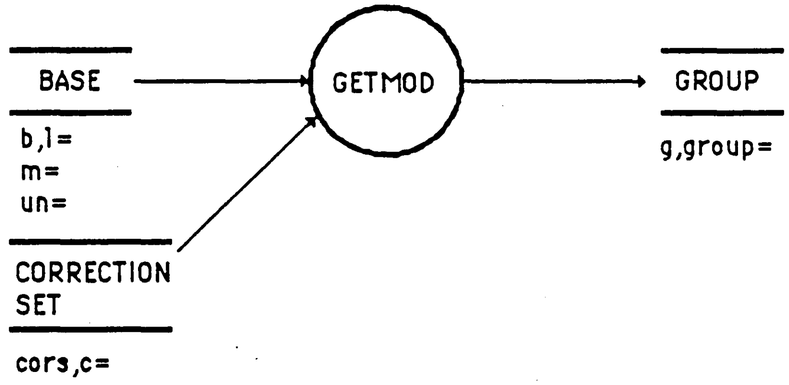
SES.REPMOD

m=(modules to replace)
g,group=group_file [*,GROUP]
b,l=base_file [*,BASE]
nb,nl=new_base_file [*,b]
un=user_name [*, current]
w,width,inwidth=width_of_input_modules
KEY: lock,nolock [*, #]
Batch Job Parameters
KEY: msg,nomsg [*, msg]

* = profile variable
# = Lock if un < > current of nolock if un=current



# GET MODULES

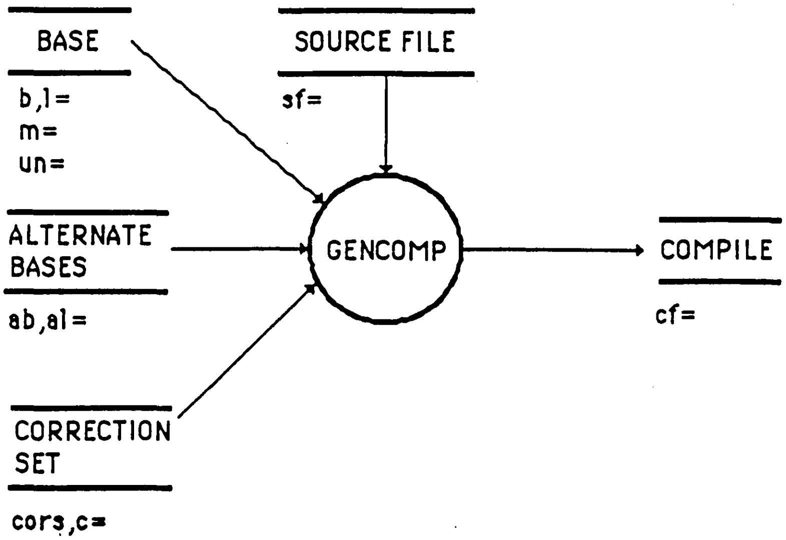


SES.GETMOD

m=(modules to extract) KEY: ALL
g,group=group_file [*, GROUP]
b,l=base_file [*, BASE]
un=user_name [*, current]
cors,c=correction_set_file
w,width=with_of_output_modules
KEY: status,sts
KEY: msg,nomsg [*, msg]

* = profile variable

# GENERATE A COMPILE FILE



```
SES.GENCOMP
   m=(modules to extract); KEY: all
   cf=compile_file [COMPILE]
   b,l=base_file [*, BASE]
   un=user_name [*, current]
   ab,al=(alternate base files)
   sf=source_file
   cors,c=correction set file [*, CORS]
   seq,noseq=width [*, no sequencing]
   def,define=(defined_names)
   status,sts=R1, R2, R3, or EF
   KEY: cybccmn,cybicmn
       : nest,nonest
       : msg,nomsg [*, msg]
       : nodrop
```

* = profile variable

Note: Alternate bases are searched right to left then base.

Note: Either parameter M or parameter SF must be given.

# MADIFY DIRECTIVE

- MADIFY directives
  - call module
  - callc module
  - ifcall name module
  - nifcall name module

## DEFINE

```
/ses.getmod all b=defpl
REVERT.   END GETMOD  DLO <- DEFPL
/copy,dlo
VECSRCH
COMMON
  PROCEDURE [XREF] vector_search;
  ?VAR
     theta: BOOLEAN := TRUE?;



LINSRCH
COMMON
  PROCEDURE [XREF] linear_search;
  ?VAR
     theta: BOOLEAN := FALSE?;
```

## SOURCE

```
DEFEX
MODULE defex;
*ifcall theta vecsrch
*nifcall theta linsrch

  PROCEDURE finder;
  ?IF theta THEN
    vector_search;
  ?ELSE
    linear_search;
  ?IFEND;
  PROCEND finder;
MODEND defex;
 EOI ENCOUNTERED.
/ses.gencomp all b=defpl def=theta
*    GENERATING COMPILE FILE  COMPILE
REVERT.    END GENCOMP    COMPILE <- DEFPL
```

## LISTING

```
/ses.cybil cc l=output
*    COMPILING  COMPILE
1SOURCE LIST OF defex                           NOS    CYBIL/CC 1.0  87103
                  January dd, yyyy      4:42 PM      PAGE 1

       0     1 MODULE defex;
       0     2   PROCEDURE [XREF] vector_search;
       0     3   ?VAR
       0     4      theta: BOOLEAN := TRUE?;
       0     5
       0     6   PROCEDURE finder;
       0     7   ?IF theta THEN
      10     8     vector_search;
      10     9   ?ELSE
            10     linear_search;
      10    11   ?IFEND;
      15    12   PROCEND finder;
      15    13 MODEND defex;
```

# DEFINE

## SOURCE

```
DEFEX
MODULE defex;
*ifcall theta vecsrch
*nifcall theta linsrch

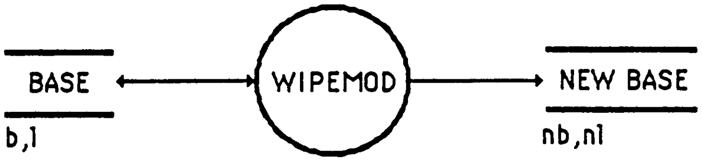
  PROCEDURE finder;
  ?IF theta THEN
    vector_search;
  ?ELSE
    linear_search;
  ?IFEND;
  PROCEND finder;
MODEND defex;
 EOI ENCOUNTERED.
/ses.gencomp all b=defpl def=theta
*    GENERATING COMPILE FILE  COMPILE
REVERT.    END GENCOMP    COMPILE <- DEFPL
```

## LISTING

```
/ses.cybil cc l=output
*    COMPILING  COMPILE
1SOURCE LIST OF defex                              NOS    CYBIL/CC 1.0  87103
                  January dd, yyyy       4:42 PM       PAGE 1

        0     1 MODULE defex;
        0     2   PROCEDURE [XREF] vector_search;
        0     3   ?VAR
        0     4      theta: BOOLEAN := TRUE?;
        0     5
        0     6   PROCEDURE finder;
        0     7   ?IF theta THEN
       10     8     vector_search;
       10     9   ?ELSE
             10     linear_search;
       10    11   ?IFEND;
       15    12   PROCEND finder;
       15    13 MODEND defex;
```

# DELETE MODULES

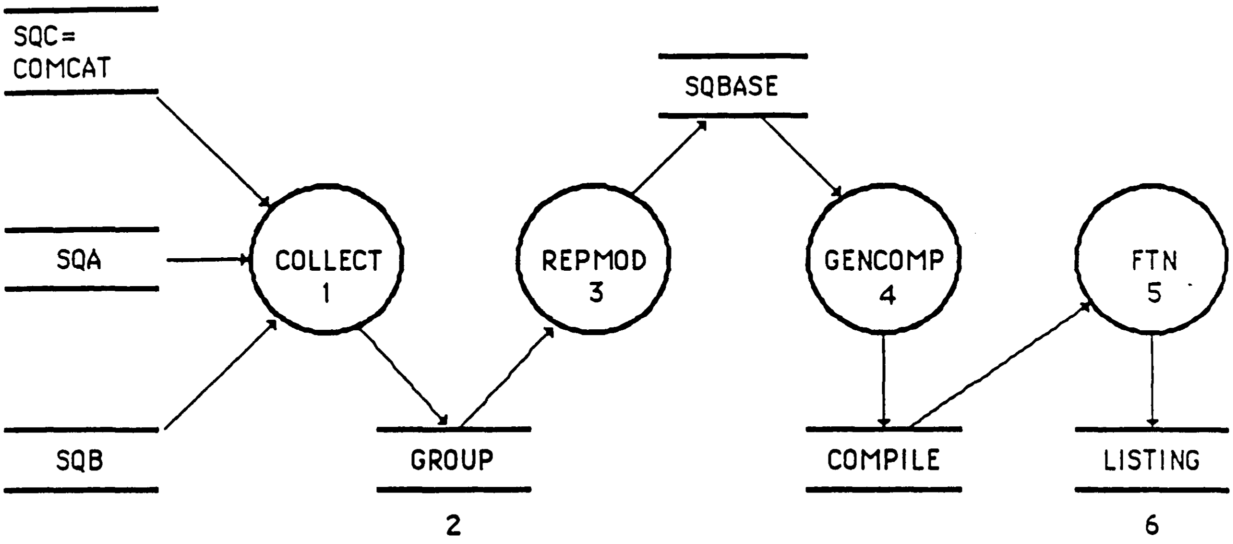


SES.WIPEMOD

b,l=base_file [*,BASE]
nb,nl=new_base_file [*,b]
un=user_name
c,com,opic=(common modules to delete)
r,reg,opl=(regular modules to delete)
KEY: lock,nolock
Batch Job Parameters
KEY: msg,nomsg

* = profile variable

## CREATE BASE



## MAKE GROUP FILE (1)

```
/get,comcat=sqc
/ses.collect c=comcat r=(sqa,sqb)
*    COMCAT/COM -> DLO
*    SQA/REG -> DLO
*    SQB/REG -> DLO
REVERT.    END COLLECT DLO
```

# CREATE BASE (CONT.)

## SOURCE FILE (2)

```
/copy,dlo
COMCAT
COMMON
      PARAMETER (MAX=20)
      INTEGER BASE(MAX)
      COMMON BASE
SQA
      PROGRAM MAIN (OUTPUT)
*CALLC COMCAT
      DO 10 I=1,MAX
        BASE(I) = I
10    CONTINUE
      PRINT *, 'INITIAL ARRAY'
      PRINT 100, BASE
C
      CALL SQUARER
C
      PRINT *, 'SQUARED ARRAY'
      PRINT 100, BASE
C
100   FORMAT (1X,I4)
      END
SQB
      SUBROUTINE SQUARER
*CALLC COMCAT
      DO 10 J=1,MAX
        BASE(J) = BASE(J) * BASE(J)
10    CONTINUE
      END
 EOI ENCOUNTERED
```

## CREATE BASE (CONT.)

### MAKE BASE (3)

```
/ses.repmod b=sqbase
*    CREATING NEW BASE SQBASE
*    NEW BASE ON SESTMPB
*    NEW BASE NOW ON SQBASE
*    SESTMPB PURGED
REVERT.   END REPMOD  DLO -> SQBASE
```

### GENERATE COMPILE FILE (4)

```
/ses.gencomp all sqbase
*    GENERATING COMPILE FILE  COMPILE
REVERT.    END GENCOMP    COMPILE <- SQBASE
```

### COMPILE (5)

```
/ses.ftn5
*    COMPILING  COMPILE
REVERT.   END FTN5
/lgo
 INITIAL ARRAY                    SQUARED ARRAY
    1                                 1
    2                                 4
    3                                 9
    4                                16
    5                                25
    6                                36
    7                                49
    8                                64
    9                                81
   10                               100
   11                               121
   12                               144
   13                               169
   14                               196
   15                               225
   16                               256
   17                               289
   18                               324
   19                               361
   20                               400
```

# CREATE BASE (CONT.)

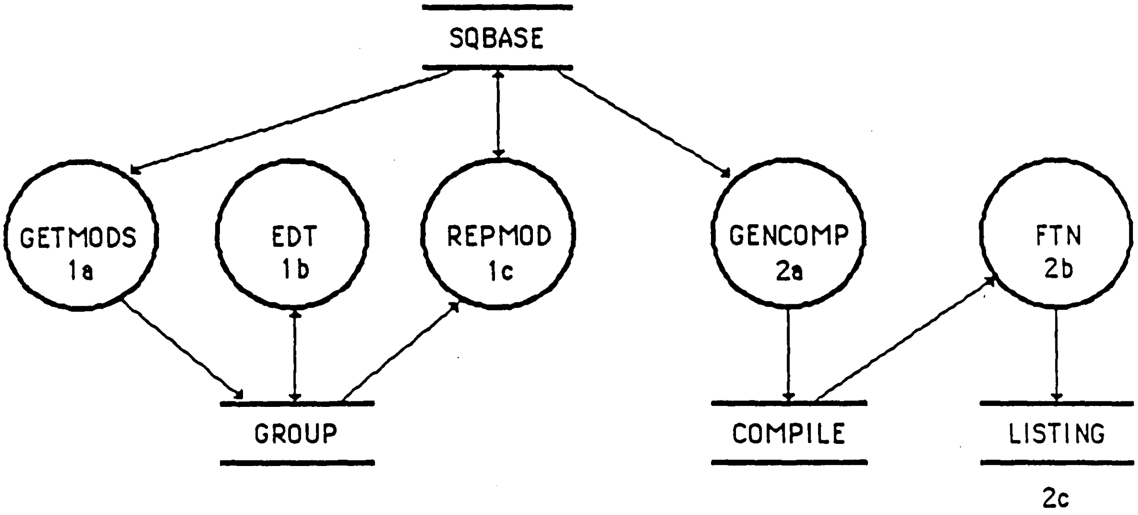
## LIST PROGRAM (6)

```
/rewind,listing
REWIND,LISTING.
/copy,listing
S
1            PROGRAM MAIN        model    OPT=0,ROUND= A/ S/ M/-D,-DS
 FTN 5.version      yy/mm/dd. hh.mm.ss        PAGE      1
         DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/
 ER/-ID/-PMD/-ST,-AL,PL=5000
         FTN5,I,L=LISTING,ET=F.


       1                 PROGRAM MAIN (OUTPUT)
       2                 PARAMETER (MAX=20)
       3                 INTEGER BASE(MAX)
       4                 COMMON BASE
       5                 DO 10 I=1,MAX
       6                   BASE(I) = I
       7           10    CONTINUE
       8                 PRINT *, 'INITIAL ARRAY'
       9                 PRINT 100, BASE
      10           C
      11                 CALL SQUARER
      12           C
      13                 PRINT *, 'SQUARED ARRAY'
      14                 PRINT 100, BASE
      15           C
      16           100   FORMAT (1X,I4)
      17                 END
1        SUBROUTINE SQUARER    model    OPT=0,ROUND= A/ S/ M/-D,-DS
FTN 5.version      yy/mm/dd. hh.mm.ss        PAGE      1


       1                 SUBROUTINE SQUARER
       2                 PARAMETER (MAX=20)
       3                 INTEGER BASE(MAX)
       4                 COMMON BASE
       5                 DO 10 J=1,MAX
       6                   BASE(J) = BASE(J) * BASE(J)
       7           10    CONTINUE
       8                 END
```

## CHANGE BASE



## MAKE CHANGES (1)

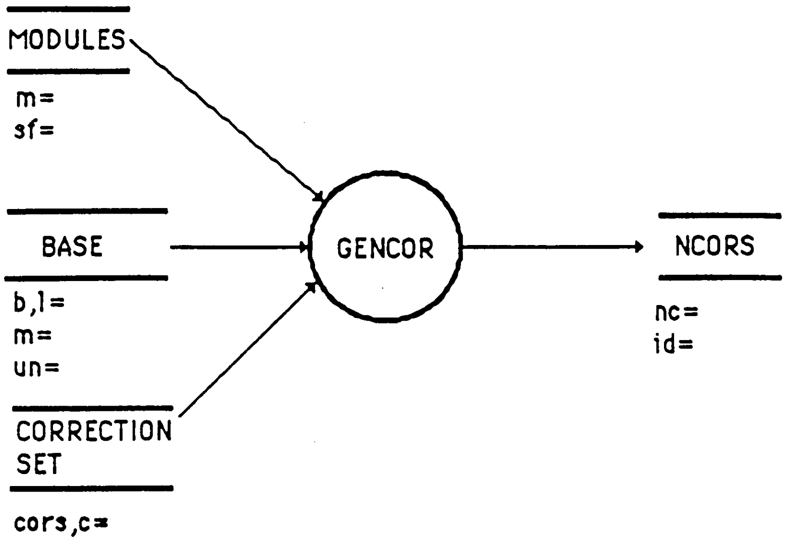
```
/ses.getmods comcat b=sqbase; edt; repmod b=sqbase
 BEGIN TEXT EDITING
? rs:/20/,/10/
? 1;*
COMCAT
COMMON
      PARAMETER (MAX=10)
      INTEGER BASE(MAX)
      COMMON BASE
 -END OF FILE-
? end
 END TEXT EDITING
*    REPLACING/ADDING MODULES ON SQBASE
*    NEW BASE ON SESTMPB
*    NEW BASE NOW ON SQBASE
*    SESTMPB PURGED
$REVERT.CCL
```

# CHANGE BASE (CONT.)

## TEST CHANGES (2)

```
/ses.gencomp all b=sqbase; ftn5
*    GENERATING COMPILE FILE  COMPILE
*    COMPILING  COMPILE
SREVERT.CCL
/lgo
 INITIAL ARRAY
    1
    2
    3
    4
    5
    6
    7
    8
    9
   10
 SQUARED ARRAY
    1
    4
    9
   16
   25
   36
   49
   64
   81
  100
     0.006 CP SECONDS EXECUTION TIME.
```

# GENERATE CORRECTION SET

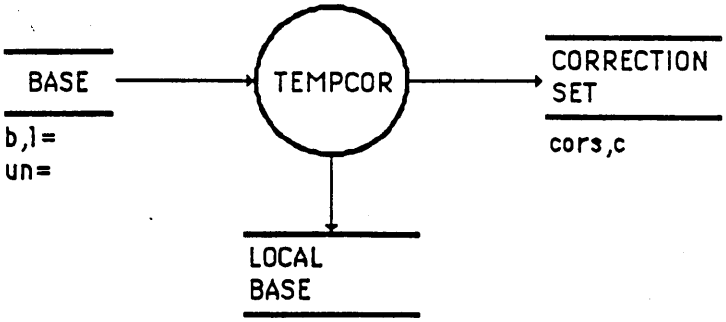


SES.GENCOR

m=module_or_file_name
b,l=base)file [*, BASE]
un=user name [*, current]
cors,c=correction_set
ncors,nc=new_correction_set [*,NCORS]
id='correction_set_ident' [fabricated ident]
sf=source_file
KEY: ls,idnorls
msg,nomsg
fl=field-length [100 K] [*, msg]

* = profile variable

# MAKE TEMPORARY CORRECTIONS

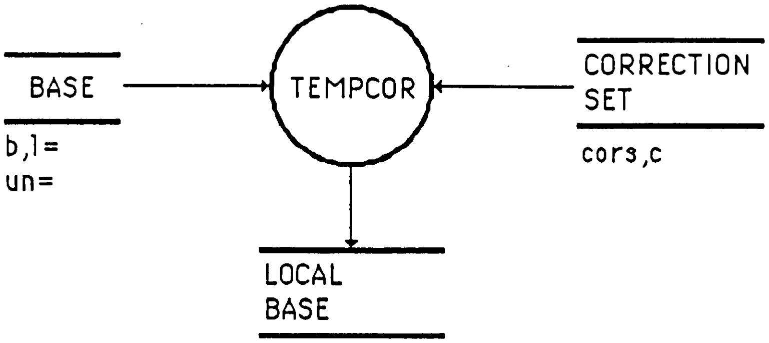


SES.TEMPCOR

b,l=base_file [*, BASE]
un=user_name [*, current]
cors,c=correction_set
KEY: msg,nomsg [*, msg]

* = profile variable

# MAKE TEMPORARY CORRECTIONS

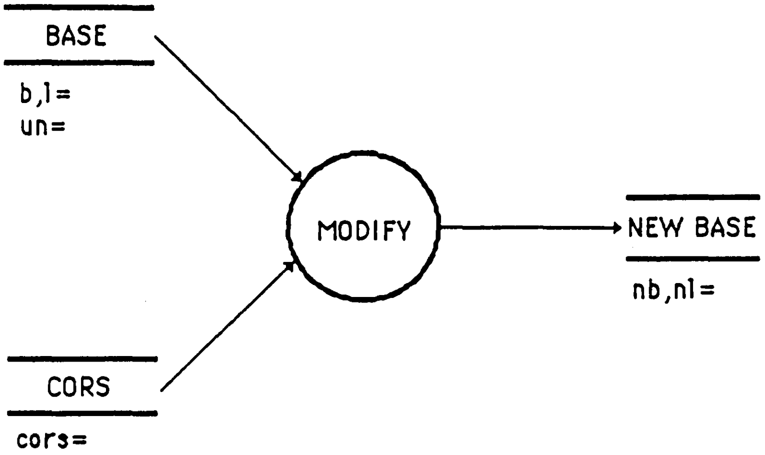


SES.TEMPCOR

b,l=base_file [*, BASE]
un=user_name [*, current]
cors,c=correction_set
KEY: msg,nomsg [*, msg]

* = profile variable

# MODIFY A BASE LIBRARY



SES.MODIFY

cors,c,i=(correction sets) [CORS]
b,l=base_file [*, BASE]
nb,nl=new_base_file [*, b]
un=user_name
KEY: lock,nolock
Batch Job Parameters
KEY: msg,nomsg [*, msg]

* = profile variable

# PRODUCE CORRECTION SET

## NEW MODULE

```
/ses.getmod comcat g=comcat b=sqbase; edt comcat
 BEGIN TEXT EDITING
? rs:/10/,/5/
? 1;*
COMCAT
COMMON
      PARAMETER (MAX=5)
      INTEGER BASE(MAX)
      COMMON BASE
 -END OF FILE-
? end
 END TEXT EDITING
SREVERT.CCL
```

## GENERATE CORRECTION SET

```
/ses.gencor m=comcat b=sqbase id=corsql
*    GENERATING CORRECTION SET FOR COMCAT ON NCORS
REVERT.   END GENCOR
/copy,ncors
*IDENT    CORSQ1
*DECK     COMCAT
*DELETE COMCAT.1
      PARAMETER (MAX=5)
*EDIT     COMCAT
 EOI ENCOUNTERED.
```

## OLD MODULE

```
/ses.getmod b=sqbase m=comcat
REVERT.   END GETMOD  DLO <- SQBASE
/copy,dlo
COMCAT
COMMON
      PARAMETER (MAX=10)
      INTEGER BASE(MAX)
      COMMON BASE
 EOI ENCOUNTERED.
```

# APPLY CORRECTION SET

## CHANGE MODULE

```
/ses.getmod m=comcat b=sqbase c=ncors
*    CREATING TEMPORARY BASE
*    TEMPORARY BASE NOW BEING USED
REVERT.   END GETMOD  DLO <- SQBASE
/copy,dlo
COMCAT
COMMON
      PARAMETER (MAX=5)
      INTEGER BASE(MAX)
      COMMON BASE
 EOI ENCOUNTERED.
```

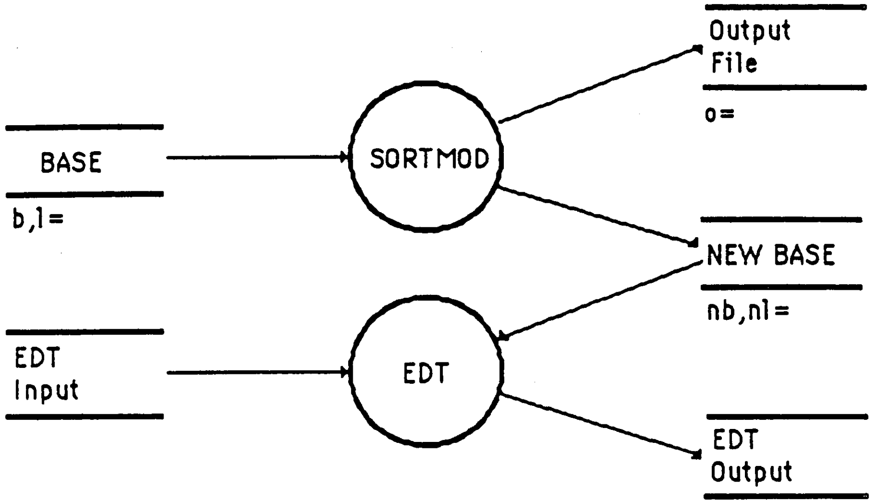
## MODIFY THE BASE

```
/ses.modify b=sqbase c=ncors
*   APPLYING NCORS TO SQBASE
*    NEW BASE ON SESTMPB
*    NEW BASE NOW ON SQBASE
*    SESTMPB PURGED
REVERT.    END MODIFY  NCORS -> SQBASE
```

## NEW BASE MODULE

```
/ses.getmod m=comcat b=sqbase
REVERT.   END GETMOD  DLO <- SQBASE
/copy,dlo
COMCAT
COMMON
      PARAMETER (MAX=5)
      INTEGER BASE(MAX)
      COMMON BASE
 EOI ENCOUNTERED.
```

# SORT MODULES



SES.SORTMOD

b,l=base_file [*, BASE]
nb,nl=new_base_file [*, b]
un=user_name
o=output_file [none]
edtsort=(EDT-input,EDT-output)
KEY: lock,nolock
Batch Job Parameters
KEY: msg,nomsg

* = profile variable

- Sort is alphabetical
- Regular and common modules are mixed.

## LIST UTILITIES

- SES.CATBASE

  - List the name and type of all modules in the base.

    b,l=base_file [*, BASE]
    un=user_ name [*, current]
    o=output_file [OUTPUT]
    KEY: short, long

- SES.LISTMOD

  - List the cross-reference of the base, and optionally list all the modules.

    b,l=base_file [*, BASE]
    o=output_file [fabricated name]
    un=user_name [*, current]
    Batch Job Parameters
    print=(PRINT procedure parameters)
    KEY: short, common

- SES.XREFMOD

  - List the cross-reference of the base. This procedure is used primarily as a building block.

## LISTMOD

```
1CROSS REFERENCE OF SOURCE LIBRARY sqbase      January dd, yyyy
                                                   PAGE 1
 ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬
¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬
ODECK NAME
 ¬¬¬¬¬¬¬¬¬
      DECKS REFERENCING THIS DECK
           DECKS REFERENCED BY THIS DECK
      ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬
           ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬
-comcat    yy/mm/dd  COMMON   LINES =  5     POS =  1
+comcat
0    sqa        sqb
-sqa       yy/mm/dd  PROGRAM  LINES =  16    POS =  2
+sqa
0
           comcat
-sqb       yy/mm/dd  UNKNOWN  LINES =  7     POS =  3
+sqb
0
           comcat
1CROSS REFERENCE OF SOURCE LIBRARY sqbase      January dd, yyyy
                                                   PAGE 2
 ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬
¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬
OEXTERNAL DECKS REFERENCED BY THIS LIBRARY
 ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬
0
-   NUMBER OF EXTERNAL DECKS REFERENCED =  0
1CROSS REFERENCE OF SOURCE LIBRARY sqbase      January dd, yyyy
                                                   PAGE 3
 ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬
¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬
OSUMMARY OF DECKS BY TYPE
 ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬
0   TYPE        LINES     DECKS
0   COMMON         5         1
    MODULE         0         0
    IDENT          0         0
    PROGRAM       16         1
    TXTFORM        0         0
    KCL            0         0
    CCL            0         0
    UNKNOWN        7         1
    SESPROC        0         0
-   TOTALS        28         3
1COMCAT
 COMMON
       PARAMETER (MAX=5)
       INTEGER BASE(MAX)
       COMMON BASE
1SQA
       PROGRAM MAIN (OUTPUT)
 *CALLC COMCAT
```

# UPDATE PROCEDURES

- Procedures

| | |
|---|---|
| GENUPCF | Generate UPDATE Compile file |
| GENUPSF | Generate UPDATE Source file |
| GETDECK | Get decks for editing |
| GENMODS | Generate modification set |
| UPDATE | Apply modification sets |

- Profile variables

| | |
|---|---|
| OLDPL | Old Program Library |
| NEWPL | New Program Library |
| PLOWNER | User Name |
| UPDTMCC | Master Control Character |

- Notes:

  - In dealing with these procedures, you may assume that an UPDATE PL exists. It provides a means of generating and applying modification sets. Also, you can assume that the programmer will change decks using an editor and generate mods. The programmer will apply these mods to the PL for test. Periodically, the PL will be updated to reflect the changes.

# UPDATE PROCEDURES

- Procedures

| | |
|---|---|
| GENUPCF | Generate UPDATE Compile file |
| GENUPSF | Generate UPDATE Source file |
| GETDECK | Get decks for editing |
| GENMODS | Generate modification set |
| UPDATE | Apply modification sets |

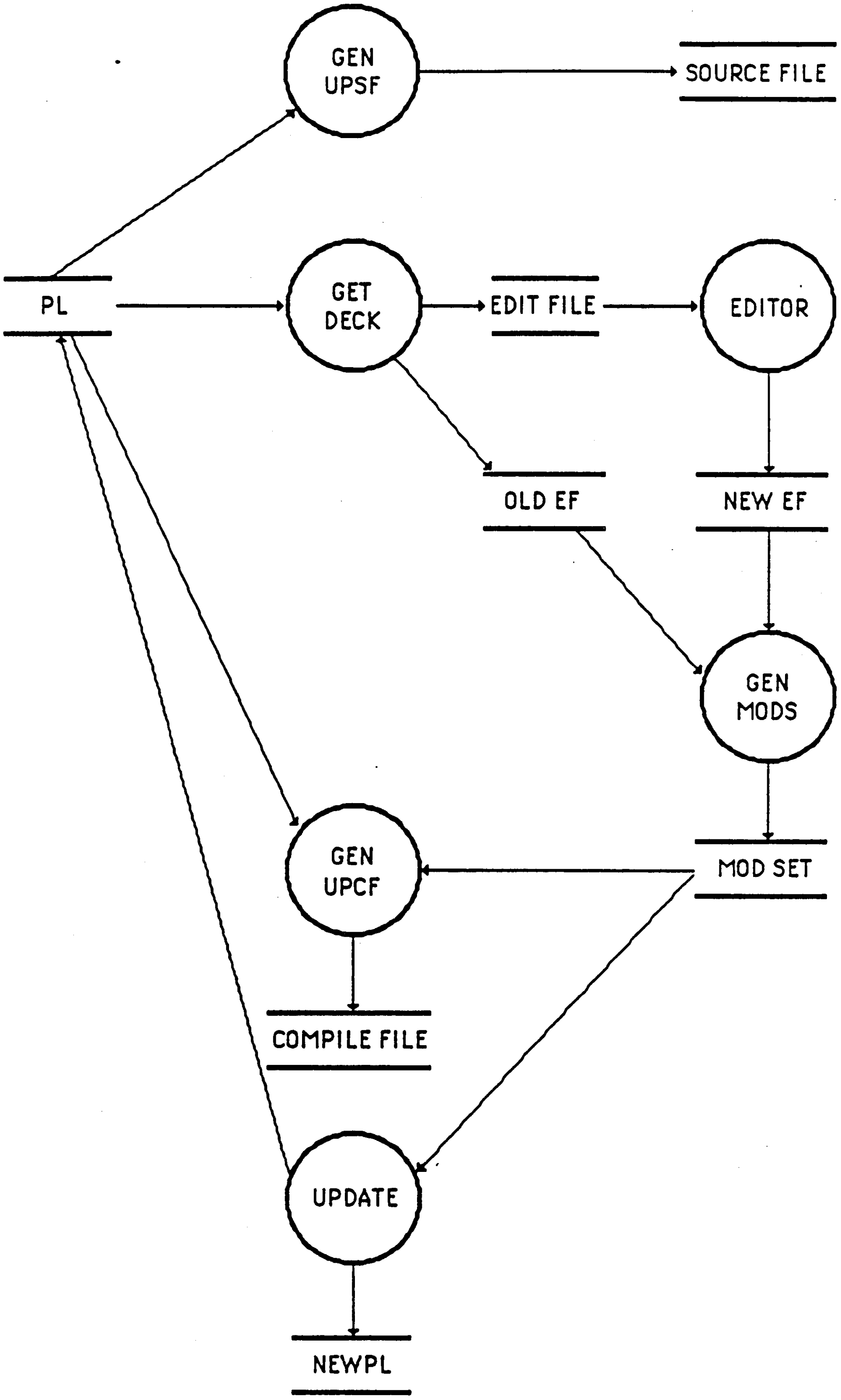
- Profile variables

| | |
|---|---|
| OLDPL | Old Program Library |
| NEWPL | New Program Library |
| PLOWNER | User Name |
| UPDTMCC | Master Control Character |

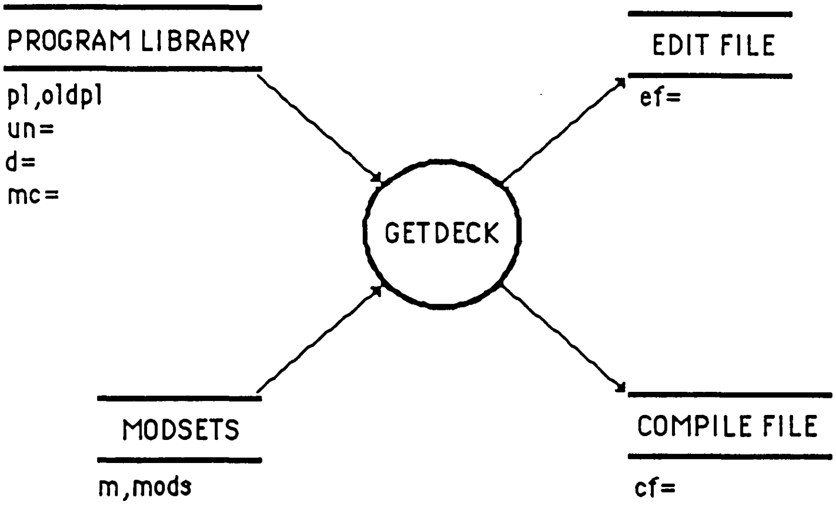
- Notes:

  - In dealing with these procedures, you can assume that an UPDATE PL exists. It provides a means of generating and applying modification sets. Also, you can assume that the programmer will change decks using an editor and generate mods. The programmer will apply these mods to the PL for test. Periodically, the PL will be updated to reflect the changes.

# UPDATE FILE USE



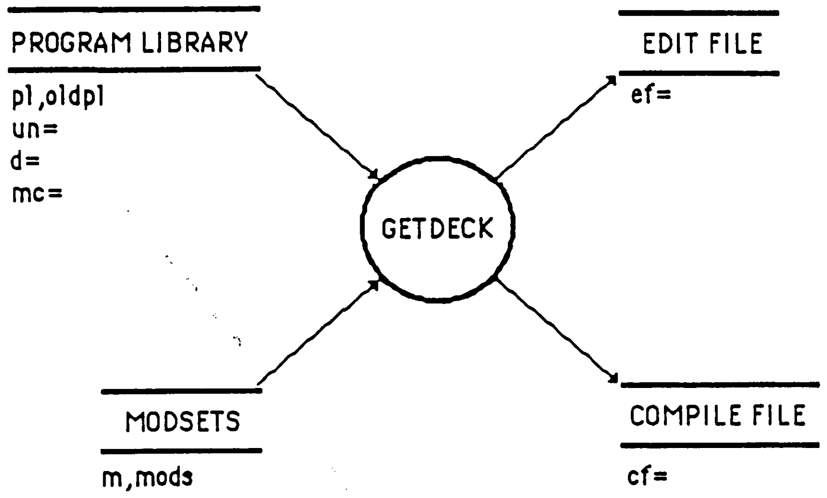
# GENERATE EDIT DECKS



SES.GETDECK

```
m,mods=(modset  files)
d,decks,c,corms,ALL=(deck_names)
ef=edit_file  [EDITFILE]
pl,oldpl=program  library  [*,current]
un=user_name  [*,  current]
cf= compile_file  [COMPILE]
KEY:  status,sts
      msg,nomsg
mc=master_control_char  [[*,  '*']
```

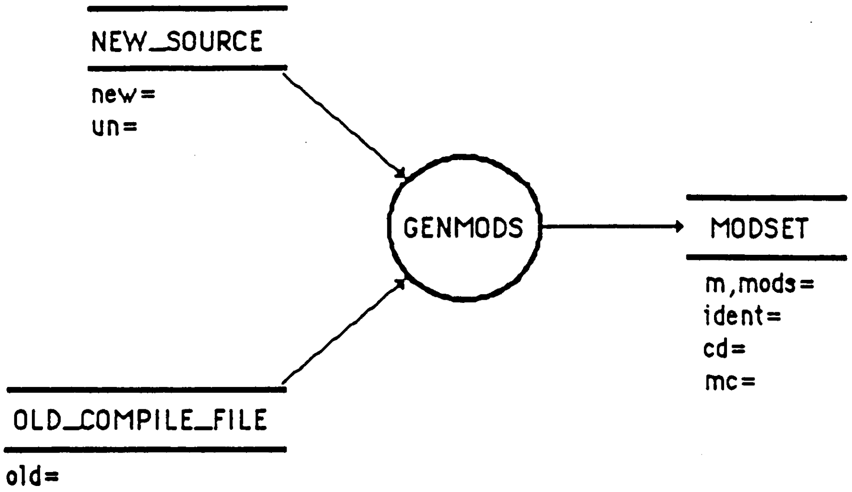
# GENERATE EDIT DECKS



SES.GETDECK

```
m,mods=(modset  files)
d,decks,c,corms,ALL=(deck_names)
ef=edit_file  [EDITFILE]
pl,oldpl=program  library  [*,current]
un=user_name  [*,  current]
cf= compile_file [COMPILE]
KEY:  status,sts
      msg,nomsg
mc=master_control_char  [[*,  '*']
```

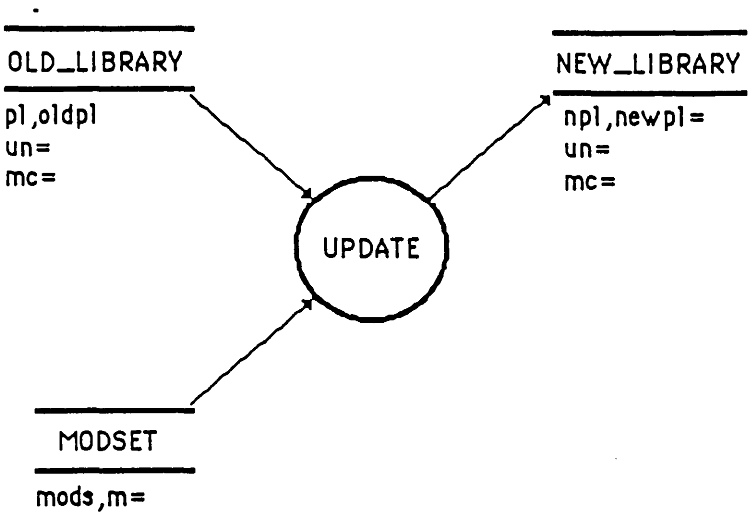
# GENERATE MODSET



SES.GENMODS

```
m,mods=modset  file
i,id,ident=correction_ident  [m]
c,cd=(*compile_deck_names)
n,new=new_source [EDTFILE]
o,old=old_compile_file  [COMPILE]
mc=master_control_char  [*,  '*']
```

# GENERATE UPDATE COMPILE FILE

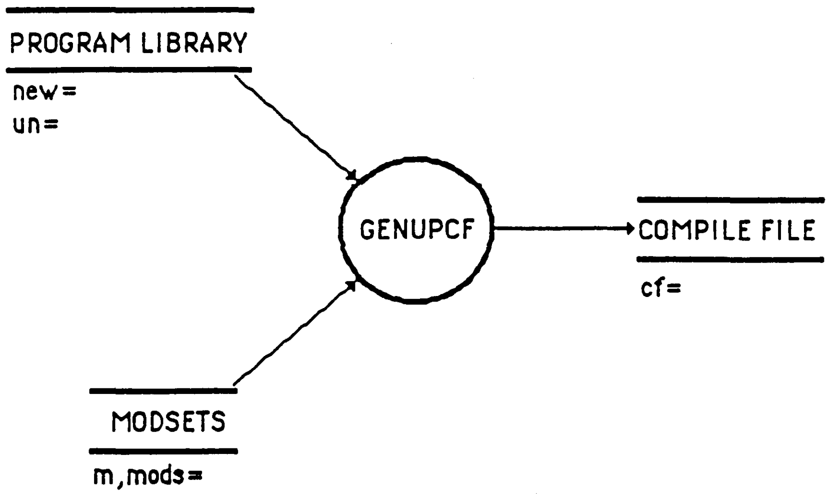


```
SES.GENUPCF

    m,mods=(modset  files)
    d,ALL=(deck  names)
    cf=compile_file  [COMPILE}
    pl,oldpl=program_library  [OLDPL]
    un=user_name  of  pl  [*,  current]
    KEY:   status,sts
           msg,nomsg
    mc=master_control_char  [*,  '*']
```

DIAGRAM IS ON PAGE

# MAKE NEW PROGRAM LIBRARY



SES.UPDATE

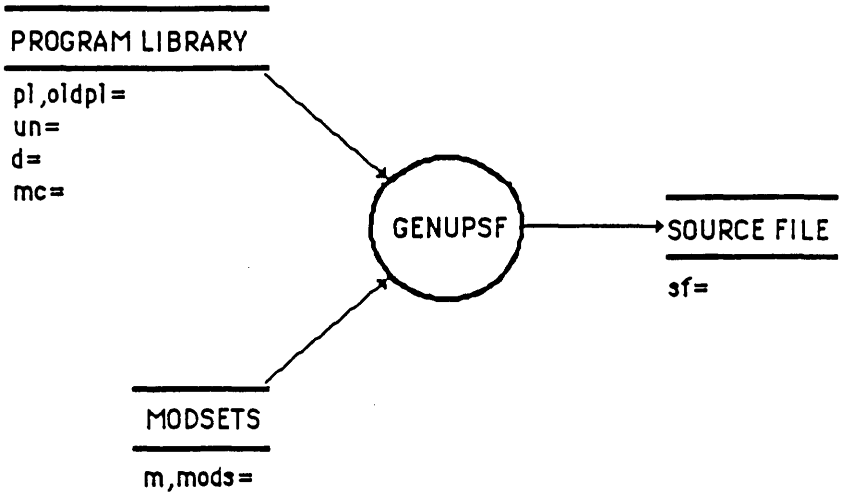
```
    mods,m=(modset  files)
    pl,oldpl=old_program_library  [*,  OLDPL]
    npl,newpl=new_program_library  [*,oldpl]
    un=user_name  [*,  current]
    KEY:   lock,nolock
           status,sts
           msg,nomsg
    mc=master_control_char  [*,  '*']
```

DIAGRAM IS ON PG.

5-29

~~5-30~~

# GENERATE SOURCE FILE



SES.GENUPSF

```
m,mods=(modset  files)
d,decks,c,coms,ALLDECK,ALLCOM=(decknames)
sf=source_file [SOURCE]
pl,oldpl=program_library  [*,  OLDPL]
un=user_name  [*,  current]
KEY:   status,sts
       msg,nomsg
mc=master_control_char  [*,  '*']
```

# EDIT A DECK

## SES.GENUPCF

```
/ses.genupcf all pl=sqpl
*    GENERATING COMPILE FILE  COMPILE
REVERT.    END GENUPCF    COMPILE <- SQPL
```

## SES.CYBIL

```
/ses.ftn5
*    COMPILING  COMPILE
REVERT.   END FTN5
/list,f=listing
1             PROGRAM MAIN    version  OPT=0,ROUND= A/ S/ M/-D,-DS    FTN5
and so forth (more heading)
        1                    PROGRAM MAIN (OUTPUT)
                         SQA         2
        2                    PARAMETER (MAX=5)
                         COMCAT      2
        3                    INTEGER BASE(MAX)
                         COMCAT      3
        4                    COMMON BASE
                         COMCAT      4
        5                    DO 10 I=1,MAX
                         SQA         4
        6                      BASE(I) = I
                         SQA         5
        7                10    CONTINUE
                         SQA         6
        8                    PRINT *, 'INITIAL ARRAY'
                         SQA         7
```

## SES.GETDECK

```
/ses.getdeck c=comcat pl=sqpl
*    GENERATING COMPILE FILE  COMPILE
*    GENERATING EDIT FILE EDTFILE
REVERT.    END GETDECK    EDTFILE, COMPILE <- SQPL
/ses.edt edtfile
 BEGIN TEXT EDITING.
? 1;*
      PARAMETER (MAX=5)
      INTEGER BASE(MAX)
      COMMON BASE
 -END OF FILE-
?  rs:/5/,/10/
?  end
 END TEXT EDITING
REVERT.    END EDT  EDTFILE
```

# MAKE A MODSET

## SES.GENMODS

```
/ses.genmods m=sqmod
SREVERT.CCL
/ses.getdeck m=cpmcat m=sqmod pl=sqpl
*    GENERATING COMPILE FILE  COMPILE
*    GENERATING EDIT FILE EDTFILE
REVERT.    END GETDECK    EDTFILE, COMPILE <- SQPL
/copy,edtfile
      PARAMETER (MAX=10)
      INTEGER BASE(MAX)
      COMMON BASE
 EOI ENCOUNTERED.
/rewind,sqmod
SREWIND,SQMOD.
/copy,sqmod
*IDENT  SQMOD
*DELETE,COMCAT.2
      PARAMETER (MAX=10)
 EOI ENCOUNTERED.
```

## SES.GENUPCF

```
/ses.genupcf all m=sqmod pl=sqpl
*    GENERATING COMPILE FILE  COMPILE
REVERT.    END GENUPCF    COMPILE <- SQPL
```

## SES.CYBIL

```
/ses.ftn5
*    COMPILING  COMPILE
REVERT.   END FTN5
/list,f=listing
1            PROGRAM MAIN    version  OPT=0,ROUND= A/ S/ M/-D,-DS    FTN5
and so forth (more heading)
        1                   PROGRAM MAIN (OUTPUT)
                      SQA         2
        2                   PARAMETER (MAX=10)
                      SQMOD       1
        3                   INTEGER BASE(MAX)
                      COMCAT      3
        4                   COMMON BASE
                      COMCAT      4
        5                   DO 10 I=1,MAX
                      SQA         4
        6                     BASE(I) = I
                      SQA         5
        7               10  CONTINUE
                      SQA         6
        8                   PRINT *, 'INITIAL ARRAY'
                      SQA         7
```

# LESSON 6

# C170 ENVIRONMENT

# LESSON 6

## C170 ENVIRONMENT

Lesson Preview:

This lesson covers linking binary files, object code maintenance, object library utilities, and the CYBIL interactive debug.

Objectives:

After completing this lesson, the student will be able to:

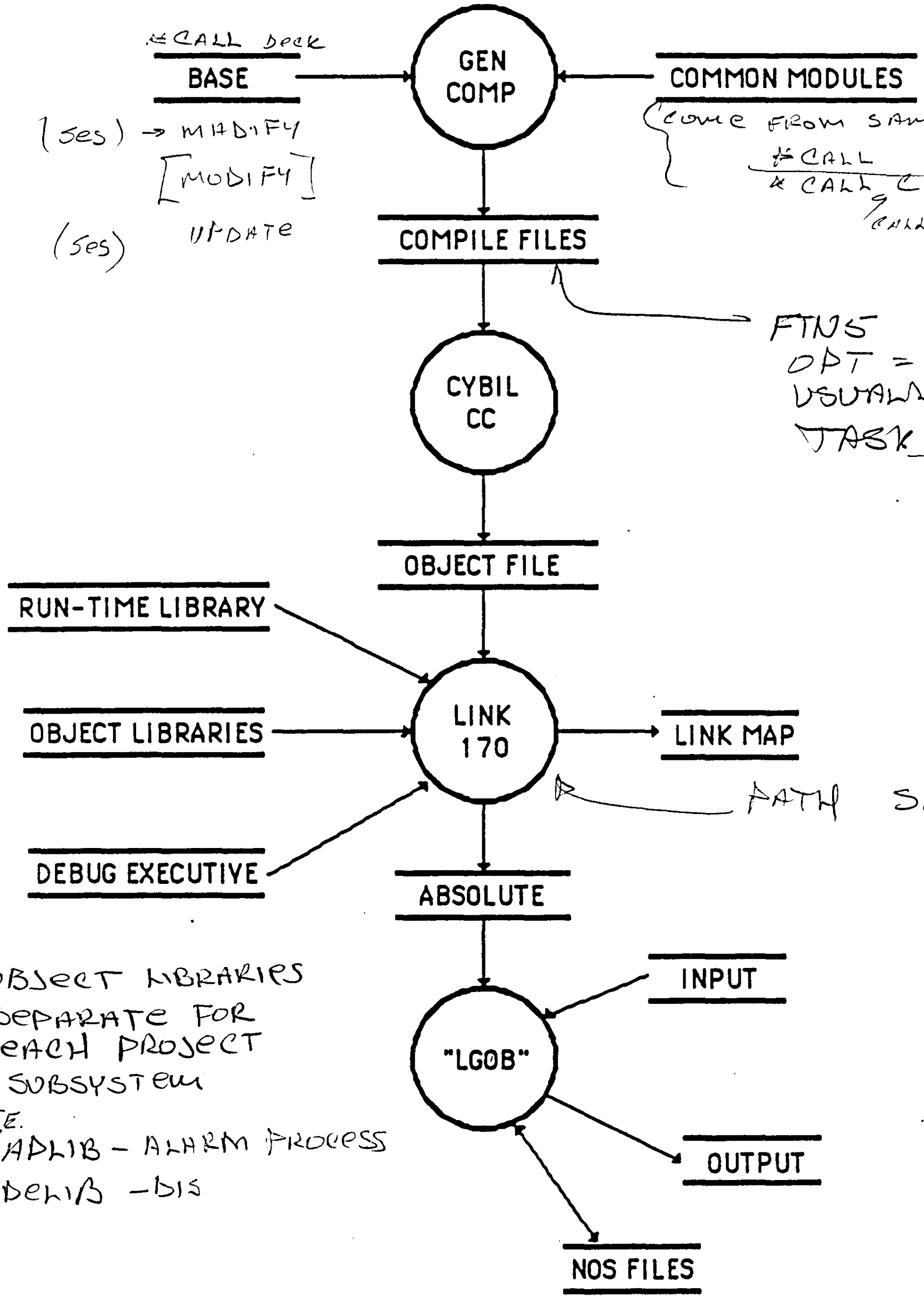
- Use the various procedures which support compiling, executing, and debugging programs.
- Link programs using SES procedures.
- Manipulate (create, add, delete) members of object libraries.
- Use the procedures which list and sort object library files.
- Use the available capabilities for debugging CYBIL programs in the C170 environment.
- Describe the capabilities of the CYBIL interactive debugger and invoke it.

TAF CANNOT DEAL WITH:

- RELOCATABLES
- CAPSULES

IT CAN ONLY OPERATE ON ABSOLUTES.

# C170 ENVIRONMENT



*CALL deck

(ses) → MADIFY

[MODIFY]

(ses) UPDATE

(come from same or other bases)
*CALL
*CALLC
call conditional
[CYBIL]

FTN5
OPT =1 OR HIGHER
USUALLY =2 FOR R.T.
TASK

PATH SOURCE → ABSOLUTE

OBJECT LIBRARIES
SEPARATE FOR
EACH PROJECT
SUBSYSTEM
I.E.
APLIB - ALARM PROCESS
DELIB - DIS

# C170 ENVIRONMENT

- CYBIL
    - CYBIL will produce a symbol table for CCDBG if the DEBUG option is selected.

- LINK170
    - The LINK170 procedure accepts relocatable binary and produces absolute binary.
    - External references can be satisfied from user libraries.
    - If the CYBCLIB option is used, LINK170 will find the CYBIL runtime library.
    - If the DEBUG option is selected, the CCDBG supervisor will be linked with the other binaries.

- OCM (Object Code Maintenance)
    - This set of procedures is similar to the set of maintaining source text. If the REPULIB procedure is used to build and change the libraries, they can be used as input to LINK170.

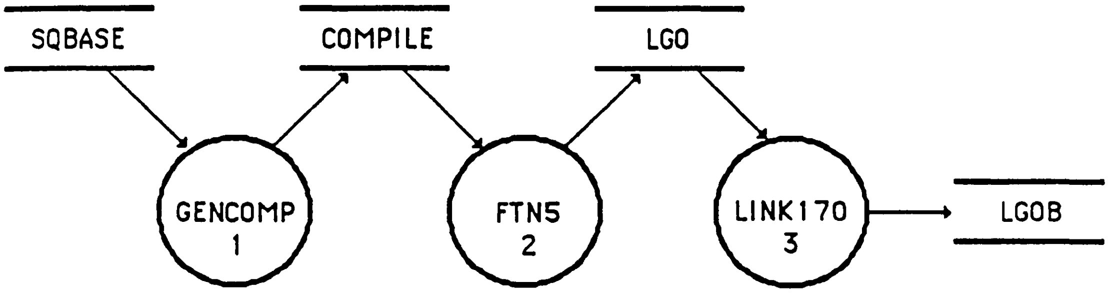
# CYBER 170 LINKER

SES.LINK170

```
f=(relocatable files) [LGO]
b=absolute_file [LGOB]
p=(library files for patching externals)
l=map_listing [OUTPUT]
lo=list_options [NOS default]          See CYBER Loader 1 Reference Manual
ep='entry_point_name'
xld=extra_load_parameters              See CYBER loader 1 Reference Manual
KEY:  Debug,nodebug [nodebug]
      cybclib
el=NONE or FATAL or ALL [ALL]
```

- Advantages
  - Provides an interactive interface to the CYBER loader.
  - Provides a convenient way to make absolute files.
  - Acquires the appropriate library for CYBIL.

## SES.LINK170



### GENERATE COMPILE FILE (1)

```
/ses.gencomp all b=sqbase
*    GENERTING COMPILE FILE  COMPILE
REVERT.   END GENCOMP    COMPILE <- SQBASE
```

### COMPILE (2)

```
/ses.ftn5
*    COMPILING  COMPILE
REVERT.   END FTN5
```

### LINK AND EXECUTE (3)

```
/ses.link170; 'lgob'
 INITIAL ARRAY
    1
    2
    3
    4
    5
 SQUARED ARRAY
    1
    4
    9
   16
   25
$REVERT.CCL
```

# USER LIBRARY TOOLS

SES. COLLECT - COLLECT MEMBER
(FILES → GROUP)

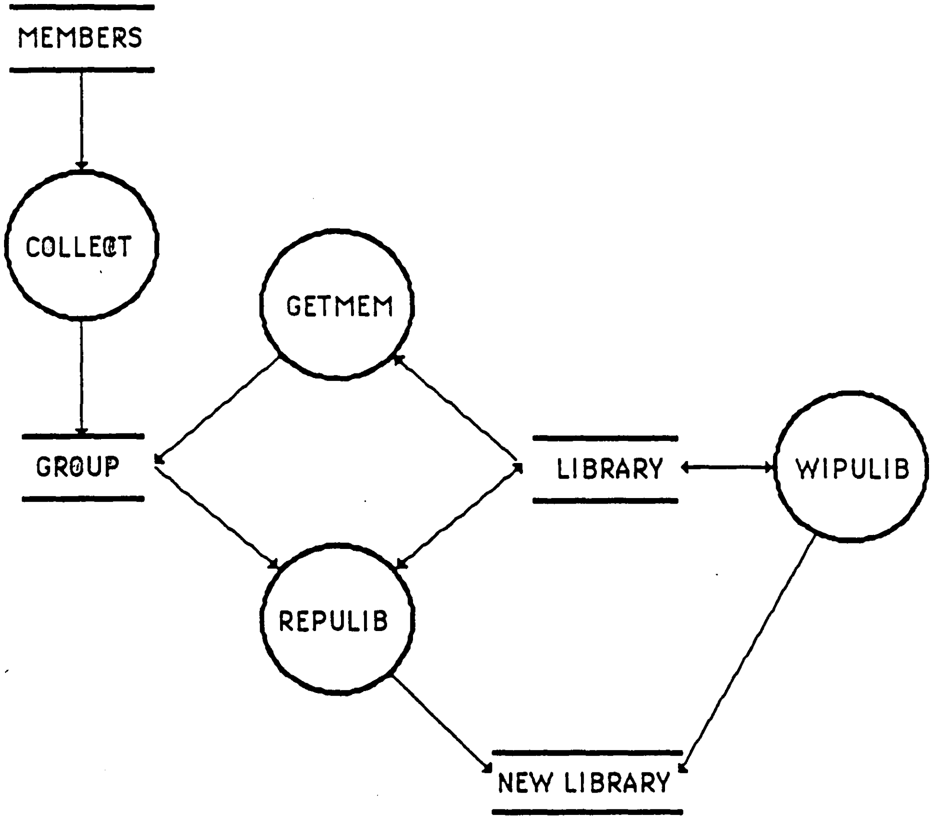
SES. REPULIB - BUILD/REBUILD LIBRARY
GROUP → BASE

SES. WIPULIB - DELETE MEMBERS

SES. GETMEM - LIBRARY → GROUP

SES. CATLIB — CATALOG LIBRARY
CONTENTS

# OBJECT LIBRARIES



— DELETE RECORDS

- A MEMBER is a single record on a library or group file. Usually members are relocatable or absolute binary.

- A GROUP is a collection of members. Each member occupies one record.

- A LIBRARY is a multi-record file. Generally, libraries contain binary records. The CYBER loader is capable of extracting records from a user library to satisfy externals.

# FEATURES

- Procedures

| | |
|---|---|
| SES.GETMEM | Library --> Group |
| SES.REPULIB | Group --> User Library |
| SES.REPMEM | |
| SES.COLLECT | Miscellaneous files --> Group |
| SES.CATLIB | Short list of library contents |
| SES.LISTMEM | List library contents |
| SES.SRTULIB | Sort a user library |
| SES.SORTMEM | |
| SES.WIPULIB | Delete members from a user library |
| SES.WIPEMEM | |
| SES.LIBEDIT | Use LIBEDIT directly |

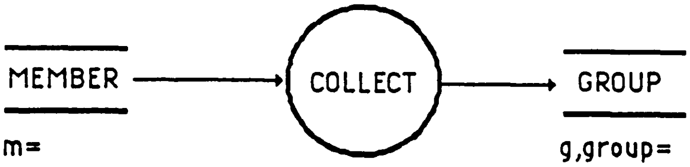
- Profile Variables

| | | |
|---|---|---|
| \LIB | = | Library Name |
| \LIBOWN | = | User Name |
| \NEWLIB | = | New Library Name |
| \GROUP | = | Group File Name |
| \SESTMPL | = | Temporary Library |
| \INTRLOK | = | Interlock File Name |
| \LOKMODE | = | LOCK, NOLOCK |
| \NX | = | Suppress cross-reference list |

- Common Parameters

l,b: library name<br>
m: member<br>
g,group: group file name<br>
nl,nb: new library name<br>
un: user name<br>
lock,nolock: lock control<br>
msg,nomsg: informative message control<br>
status,sts: retrieve NOS status for own use

# MAKE A GROUP



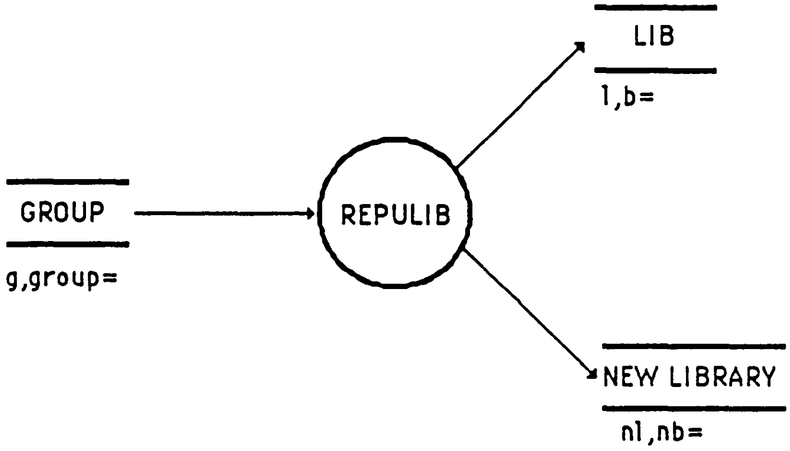
SES.COLLECT

```
m,mem,mems,member,members=
   (member_files)
c,com,oplc
r,reg,opl
g,group=group_file  [*,GROUP]
KEY: msg,nomsg
```

*  =  profile  variable

If the m-parameter is used, COLLECT does not add a name.

# REPLACE MEMBERS



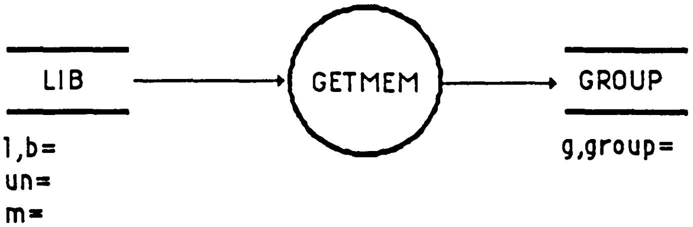
SES.REPULIB

```
    m,mem,member,members=(members)  [all]
    g,group=group_filename [*, GROUP]
    l,b=library_name  [*,  LIB]
    nl,nb=new_library_name  [*,  l]
    un=user_name
    KEY: lock,nolock
    Batch Job Parameters
    KEY:  nx
          rep,add,repadd,addrep [addrep]
          msg,nomsg

    *  =  profile  variables
```

- The CYBER loader is able to selectively load binaries from user libraries only.

# GET MEMBERS



SES.GETMEM

m=(members or ranges of members); KEY: all
g,group=output_file [*,GROUP]
l,b=input_library [*, LIB]
un=user_name [*, current]
KEY: status,sts

* = profile variable

KEY: text, opl, oplc, opld, rel, abs, oul, pp, ppu, ulib, cap, proc [TEXT]

# LIST LIBRARY CONTENTS

SES.CATLIB

```
l,b=library_name [*, LIB]
un=usr_name [*, current]
o=output_file [OUTPUT]
KEY: short,long
```

SES.LISTMEM

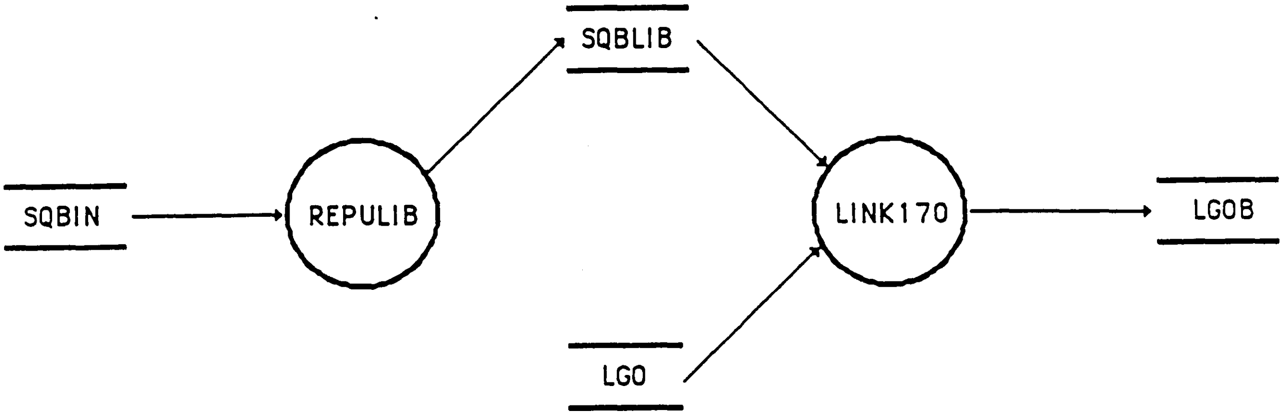
```
l,b=library_name [*, LIB]
o=output_file [fabricated name]
un=user_name [*, current]
Batch Job Parameters
print=(PRINT procedure parameters)
KEY: short
```

# DELETE MEMBERS

SES.WIPULIB

```
l,b=library_name [*, LIB]
nl,nb=new_library_name [*, l]
un=user_name
mem,m,rel=(relocatable binary members)
KEY: lock,nolock
Batch Job Parameters
KEY: msg,nomsg
KEY: nx [*, nx]
```

## USER LIBRARY



## BUILD A LIBRARY

```
/ses.catbase sqbase
 COMCAT...OPLC     SQA......OPL      SQB......OPL      OPL......OPLD
REVERT.   END CATBASE


/ses.gencomp m=sqb b=sqbase
*    GENERATING COMPILE FILE  COMPILE
REVERT.    END GENCOMP    COMPILE <- SQBASE
/ses.ftn5 b=sqbin
*    COMPILING  COMPILE
REVERT.   END FTN5


/ses.repulib g=sqbin l=sqblib
*    CREATING NEW LIBRARY SQBLIB
*    NEW LIBRARY ON SESTMPL
*    NEW LIBRARY NOW ON SQBLIB
*    SESTMPL PURGED
REVERT.   END REPULIB
```

# USER LIBRARY (CONT.)

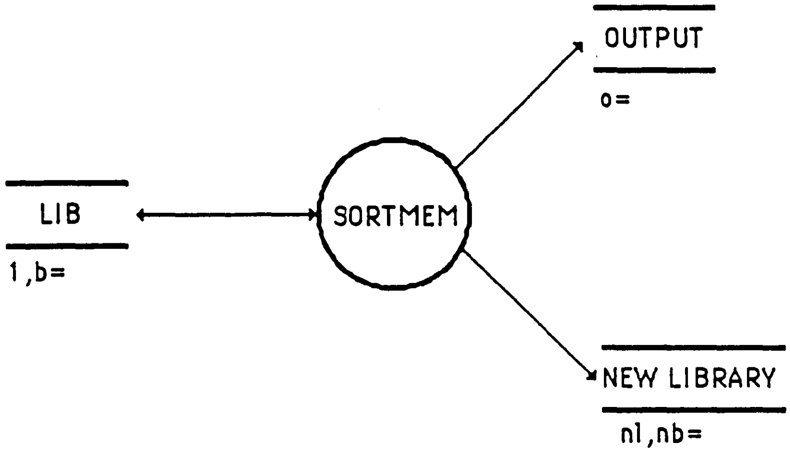
## USE THE LIBRARY

```
/ses.gencomp m=sqa b=sqbase
*    GENERATING COMPILE FILE  COMPILE
REVERT.    END GENCOMP    COMPILE <- SQBASE
/ses.ftn5 b=sqabin
*    COMPILING  COMPILE
REVERT.   END FTN5


/ses.catlib sqblib
 SQBLIB...ULIB     SQUARER..REL      SQBLIB...OPLD
REVERT.   END CATLIB


/ses.link170 f=sqabin p=sqblib l=output
REVERT.    END LINK170    LGOB
/lgob
 INITIAL ARRAY
    1
    2
    3
    4
    5
 SQUARED ARRAY
    1
    4
    9
   16
   25
```

# SORT LIBRARY



SES.SORTMEM

l,b=library_file [*, LIB]
nl,nb=new_library_name [*, l]
un=user_name [*, current]
o=output_file [none]
edtsort=See SES User's Handbook.
KEY: lock,nolock
Batch Job Parameters
KEY: build,nobuild,dir,nodir [dir]
msg,nomsg

* = profile variable

# RUN LIBEDIT UTILITY

```
SES.LIBEDIT

   g,group,lgo=replace/insert_file  [*,  GROUP]
   l,b=library_file  [*,  LIB]
   nl,nb=new_library_file  [*,  l]
   un=user_name  [*,  current]
   i,input=(directives)  or  directive_file  [INPUT]
   o=output_file  [OUTPUT]
   KEY:  lock,nolock
         msg,nomsg

   *  =  profile  variable
```

# CYBIL INTERACTIVE DEBUGGER

- Variation of CID

- References CYBIL
  - Variable names
  - Line numbers
  - Modules
  - Procedures

- SCL-like command structure

- Capabilities
  - Breakpoints
  - Traps
  - Debugger procedures
  - Variable control
  - Memory and register control
  - Traceback

# LESSON 7

# DOCUMENT FORMATTING SYSTEM

# LESSON 7

## DOCUMENT FORMATTING SYSTEM

Lesson Preview:

This lesson discusses TXTCODE documents, creating documents and revising documents.

Objectives:

After completing this lesson, the student will be able to:

- Invoke the various formatting utilities using SES procedures.
- Explain the codes used by TXTCODE, and format a complete document using them.
- Use the SES procedures to produce revision packages and revision bars for existing TXTCODE documents.

References:

- SES User's Handbook, Chapter 4
- TXTCODE ERS
- TXTFORM ERS

## DOCUMENT FORMATTING SYSTEM

- Capabilities
  - Interprets TXTCODE codes or TXTFORM codes or both.
  - Prints document or source or both.
  - Generates revision bars and revision packages.
  - Runs in batch or local mode.
  - Checks for spelling errors.
  - Formats diagrams.
  - Provides a template for preparing memos.
  - Computes the reading difficulty of documents.
- Major procedures

```
SES.FORMAT
SES.FORMREV
SES.SPELL i, o, dict, dictun, lo
SES.MEMO i, o, to, from, when, subject
SES.DIAGRAM o, C1=(shapes), C2=(shapes), C3=(shapes)
SES.TWOPAGE i, o, width, seq
SES.GRADLVL i, o, incset, outcset, hical
```

- Supplementary procedures

```
SES.TXTCODE
SES.TXTFORM
SES.TXTHEAD
SES.GENREVB
SES.GENREVP
```

# DOCUMENT FORMATTING SYSTEM

- Capabilities

  - Interprets TXTCODE codes or TXTFORM codes or both.
  - Prints document or source or both.
  - Generates revision bars and revision packages.
  - Runs in BATCH or LOCAL mode.
  - Checks for spelling errors.
  - Formats diagrams.
  - Provides a template for preparing memos.
  - Computes the reading difficulty of documents.

- Major procedures

```
SES.FORMAT
SES.FORMREV
SES.SPELL i, o, dict, dictun, lo
SES.MEMO i, o, to, from, when, subject
SES.DIAGRAM o, C1=(shapes), C2=(shapes), C3=(shapes)
SES.TWOPAGE i, o, width, seq
SES.GRADLVL i, o, incset, outcset, hical
```

- Supplementary procedures

```
SES.TXTCODE
SES.TXTFORM
SES.TXTHEAD
SES.GENREVB
SES.GENREVP
```

# DIAGRAM

```
/ses.diagram dlo c1=(box,line,stack)..
..?              c2=(in,conn,out)..
..?              c3=(cube,cond,table)
REVERT.    END DIAGRAM  DLO
/list,f=dlo
        |                \     /                 |
        |                 \   /             +----+------+
+-------+-------+          \ /             /     |     /|
| xxxxxxxxxxxxx |           |             +-----------+ |
| xxxxxxxxxxxxx |           |             | xxxxxxxxx | |
| xxxxxxxxxxxxx |           |             | xxxxxxxxx | +
+-------+-------+           |             | xxxxxxxxx |/
        |                   |             +------+-----+
        |                   -                    |
        |                 /   \                  |
        |                1  x  1                / \
        |                 \   /          +-----/   \-----+
        |                   -            | xxxxxxxxxxxxx |
        |                   |            | xxxxxxxxxxxxx |
        |                   |            | xxxxxxxxxxxxx |
        |                   |            +-----\ ? /-----+
        |                   |                   \ /
        |                   |                    |
        |                  / \                   |
    +---+-------+         /   \          +-------+-------+
    |           |        /     \         |     TABLE     |
  +-+---------+ |                        |---------------|
  |           |-+                        | xxxxxxxxxxxxx |
+-+---------+ |                          | xxxxxxxxxxxxx |
| xxxxxxxxx |-+                          | xxxxxxxxxxxxx |
+-------+---+                            +-------+-------+
        |                                        |
 EOI ENCOUNTERED
```

# MEMO

```
es.memo i=memtxt o=mem..
?  to ('Blaise Pascal','Pere LaChaise')..
?  from ('Niklaus Wirth','New Prague')..
   subject='semicolons'
ERT.    END MEMO  MEMTXT -> MEM
es.format mem d=output·local
   PROCESSING TXTCODE ON MEM
   FORMATTED DOCUMENT ON FILE LISTING
   CREATING DISPLAYABLE DOCUMENT OUTPUT
```

```
                                                                          1




                         MM    MM  EEEEEEE  MM    MM   OOOOO
                         MM    MM  EEEEEEE  MM    MM   OOOOO
                 C       M M M M   E        M M M M  O     O        C
                 C       M M M M   E        M M M M  O     O        C
                 D       M  M  M   EEEEE    M  M  M  O     O        D
                 D       M  M  M   EEEEE    M  M  M  O     O        D
                 C       M     M   E        M     M  O     O        C
                 C       M     M   E        M     M  O     O        C
                         M     M   EEEEEEE  M     M   OOOOO
                         M     M   EEEEEEE  M     M   OOOOO



           DATE :  January dd, yyyy

             TO : Blaise Pascal                   LOCATION : Pere
         LaChaise

           FROM : Niklaus Wirth                   LOCATION : New
         Prague

         SUBJECT : semicolons
                   __________



         We are experiencing extreme difficulties due to  the  lack  of
         semicolons.   We   need  your  help.   We  see  two  possible
         solutions:
            1. Make semicolons optional.
            2. Provide a facility for cutting colons in half.
         We would appreciate your immediate attention.

         Sincerely,


         H.  C.  Libyc
```

# TXTCODE FEATURES

- High level formatter
  - Preprocessor of TXTFORM.
  - Common structures are easy to create.
  - One command can replace several TXTFORM commands.
  - Command and text are sometimes on one line.
  - Command operates over large portions of text.

- Types of commands
  - Input file control
  - Structuring (mode) control
  - Page designation - titles, table of contents, and so forth
  - Section numbering
  - Feature selection - paragraphing, line numbering, double spacing, underlining, and so forth.
  - Text formatting - margins, centering, page eject, boxes, and so,forth.
  - Escape characters for underlining, bold facing, and so forth.

# PRELIMINARY EXAMPLE

## SOURCE

```
\block
\setu¬
\center=¬TXTCODE¬
\skip2
 TXTCODE is a 'high-level' formatter. It attempts to
predict the most common structures, provides for them to
be set up with single commands and then controls the
generation of the structure for the writer. These
commands are called MODE commands. A list of the MODE
commands follows:

\table:10,20
 :ASIS:Leave the source as it is.
 :BLOCK:Make nice paragraphs.
 :COMMENT:Embed comments.
 :TABLE:Produce tables.
 :FLOWTAB:Allows for a flowing column of text in a table.
 :ITEM:Make nicely indented lists.

\block
 MODE commands are the heart of the TXTCODE formattter.
```

## DOCUMENT

```
/ses.format tcpre local d=output
*    PROCESSING TXTCODE ON TCPRE
*    FORMATTED DOCUMENT ON FILE LISTING
*    CREATING DISPLAYABLE DOCUMENT OUTPUT
1                                                                   1
                                                             yy/mm/dd

                                      TXTCODE
+
                                      -------


          TXTCODE is a 'high-level' formatter.  It attempts  to  predict
          the  most  common  structures,  provides for them to be set up
          with single commands and then controls the generation  of  the
          structure  for  the  writer.   These  commands are called MODE
          commands.  A list of the MODE commands follows:

                   ASIS      Leave the source as it is.
                   BLOCK     Make nice paragraphs.
                   COMMENT   Embed comments.
                   TABLE     Produce tables.
                   FLOWTAB   Allows for a flowing column of text in a
          table.
                   ITEM      Make nicely indented lists.

          MODE commands are the heart of the TXTCODE formattter.
```

## MODE COMMANDS

- \ASIS

  - All lines are passed to output as they are.

- \BLOCK, Jnn, Inn, Hnn

  - The BLOCK command controls the building of paragraphs. If a text bloc^k^ has a blank in column one, a paragraph is built. If the PARA option has been selected, the paragraph will be indented three spaces. If the text block has two or more blanks in the first columns, the whole block will be indented. J indicates lines to skip; I is another way to obtain indentation; H is the indenting of subsequent text.

- \COMMENT

  - Comments can be embedded in the text. Commands continue to be processed, and only MODE commands cause escape from COMMENT mode.

- \TABLExtt,...,tt

  - The TABLE command arranges data in columns. The x (often ;) moves the tabulator index to the next tab (tt). A blank in column one causes a new output line.

- \FLOWTABxtt,...,tt,ff

  - FLOWTAB combines some of the features of BLOCK with the features of TABLE. ff is the last tab position. Paragraphs will 'flow' between ff and the right margin.

- \ITEM, Jnn, Inn, Snn, Gnn, Hnn,Zxx, C=3, O

  - Itemized lists can be built. ITEM controls the indenting. I is the indenting which precedes the item; S is the size of the item; G is the gap which follows the item and precedes the description of the item. H is the indentation of text on subsequent lines of description; it defaults to I+S+G. J is the number of lines to skip, Z, X, and O control automatic label generation.

# BLOCK AND ASIS

## SOURCE

```
\block
\para
 This text block will start a paragraph since the first column
is a blank. It is indented because the PARA command is
specified.

          This text block starts with ten blanks so the whole
block is indented ten spaces.
\asis

                          +-------+-------+
                          |    TXTCODE    |
                          |      IS       |
                          |  WONDERFULL   |
                          +-------+-------+

\block
```

## OUTPUT

```
1                                                                             1
                                                                       yy/mm/dd

               This text block will start  a  paragraph  since  the  first
            column is a blank.  It is indented because the PARA command is
            specified.

                     This text block starts with ten blanks so  the  whole
                     block is indented ten spaces.

                                    +-------+-------+
                                    |    TXTCODE    |
                                    |      IS       |
                                    |  WONDERFULL   |
                                    +-------+-------+
```

# TABLE AND COMMENT

## SOURCE

```
\comment
This table must be updated each time this manual is changed
to reflect current status. ************************
\table;20,30,40
 Project Name; Budget; Cost; Completion Date
 _______ ____; ______; ____; __________ ____
 Compilers ; 571; 568; 1/2/86
 Tools     ; 118; 155; 2/3/87
 Simulator ; 304; 158; 3/4/88
 Operating System;1444; 489; 4/5/89
 ;____;____
 ;2437;1370
\block
```

## OUTPUT

```
1                                                                   1
                                                             yy/mm/dd

          Project Name         Budget     Cost       Completion Date

          _______ ____         ______     ____       __________ ____
          Compilers            571        568        1/2/86
          Tools                118        155        2/3/87
          Simulator            304        158        3/4/88
          Operating System     1444       489        4/5/89

                               ____       ____
                               2437       1370
```

# FLOWTAB

## SOURCE

```
\flowtab;5,25,37
\setu¬
 ;¬Project Name¬;¬Comp.Date¬;¬Comments¬
 ;Compilers ;1/2/86 ;FORTRAN is complete. LACSAP is still
waiting for a shipment of semi-colons.
 ;Tools ;2/3/87 ;The project is still waiting for
approval of the Design Requirements Document.
 ;Simulator ;3/4/88 ;On schedule.
 ;Operating System ;4/5/89;On schedule.
\block
```

## OUTPUT

```
1                                                         1
                                                   yy/mm/dd

         Project Name      Comp.Date  Comments

         ____________      _________  ________
         Compilers         1/2/86     FORTRAN    is   complete.
                                      LACSAP  is  still  waiting
                                      for   a   shipment    of
                                      semi-colons.
         Tools             2/3/87     The  project   is   still
                                      waiting  for  approval  of
                                      the  Design  Requirements
                                      Document.
         Simulator         3/4/88     On schedule.
         Operating System  4/5/89     On schedule.
```

# ITEM

## SOURCE

```
\block
 The following features of TXTCODE make it easy to use
and easy to maintain:
\item,i3,s2,gl
 a. Mode commands replace several, more primitive, TXTFORM
commands
 b. The meaning of the commands and therefore of the source
document is more obvious because of the choice of mnemonic
words and because of the straight forward syntax.
 c. Some commands are on
the same line as the text
they control.
\asis

                 +--------+--------+
                 |     TXTCODE     |
                 |       IS        |
                 |   WONDERFULL    |
                 +--------+--------+

\block
```

## OUTPUT

```
1                                                               1
                                                         yy/mm/dd

          The following features of TXTCODE make it easy to use and easy
          to maintain:
             a. Mode  commands  replace several, more primitive, TXTFORM
                commands
             b. The meaning of the commands and therefore of the  source
                document  is  more  obvious  because  of  the  choice of
                mnemonic words  and  because  of  the  straight  forward
                syntax.
             c. Some  commands  are  on  the  same line as the text they
                control.

                           +--------+--------+
                           |     TXTCODE     |
                           |       IS        |
                           |   WONDERFULL    |
                           +--------+--------+
```

# MODE NESTING

- Use

  - Nested modes are used to make outlines or lists with sublists. Usually, they are used in conjunction with ITEM.

- Commands

| | |
|---|---|
| \+mode | Advance nesting level |
| \-mode | Retract nesting level |
| *mode | Change mode at current level |
| \+ | Advance level |
| \ - | Retract level |
| \EXITNEST,C | Exit the nest |
| \CLEARNEST | Destroy the nest |
| \NESTIND | Indent from last nested level |
| \NONESTIND | Indent from margin |

# MODE NESTING

## OUTLINE

```
\+item,R.
 . INTRO
\+item,L
 . SES Procedures
 . Syntax
\-
 . GETTING INFO
\+item,L
 . Documents
 . File Information


\exitnest,c
```

```
  I. INTRO
        A  SES Procedures
        B  Syntax
 II. GETTING INFO
        A  Documents
        B  File Information
```

## SUBLIST

```
\seth!
\+item,h5
 MAP_OPTIONS!or!MO
\+item
 NONE - No map
 FULL - Full map
 PART - No entry points
\-
```

```
MAP_OPTIONS or MO
      NONE - No map
      FULL - Full map
      PART - No entry points
MAP_NAME or MN
```

# DESIGNATION CONTROL

- \TITLEn=title
  - Up to six titles will be formatted on the top of each page of output.

- \DATE=date
  - Any date can be selected. It will replace the current date, which is chosen by default.

- \FOLIO=text
  - The text chosen will be right justified on the bottom of each page. FOLIO is nice for privacy information.

- \TABLCON
  - TABLCON should appear at the beginning of the source if a table of contents is desired.

- \AUTOSEC and NOAUTOSEC
  - Section numbers will be assigned if the AUTOSEC command is exercised. This works with the section numbering commands. The NOAUTOSEC command turns the option off.

- \SETAPP=a_title
  - SETAPP sets up the appendix identifier and title for the page header and the table of contents.

The titles, date and folio information on this page were formed by the following commands:

```
\title1= TXTCODE
\title2= TITLE, DATE, FOLIO Examples
\date= MM/DD/YY
\folio= CDC Private
```

## SECTION NUMBERS

- \n.n.n...text

  - In this form, the writer can specify the appropriate section number (\1.2 title). If automatic section numbering has been selected, then "n" can be used instead of numbers, and TXTCODE will provide the number. The blanks after the \ and before the title are necessary. HEADA, HEADB, and HEADC can be used to select layout for the various levels.

- \n text

  - "n" can have values 1 to 7 indicating the level of this section title. "n" can also have values 11 to 17 indicating the level, but in that case, the number itself would not be part of the output. The blanks, before and after "n" are necessary. HEADA, HEADB, and HEADC can be used to select layout for the various levels.

- \text

  - If this form is used, an underlined title will be produced. There is no section number associated with it.

# SECTION NUMBERING

## SOURCE

```
Option 1                  Option 2                 Option 3


                                                   \autosec
\ 1.0 Intro               \ 1 Intro                \ n.0 Intro
\ 1.1 First               \ 2 First                \ n.n First
\ 1.2 Second              \ 2 Second               \ n.n Second
\ 2.0 Next                \ 1 Next                 \ n.0 Next
```

## OUTPUT

```
\comment
 Option 1 output below   Option 2 output is like  Option 3 output is like
                         option 1 output minus    option 1 output plus
                         the numbers              start of page sec titles

1                                                                        1
                                                                  yy/mm/dd

           1.0 INTRO
+          _________

           1.1 FIRST
+          _________

           1.2 SECOND
+          __________


     ****************************************************************

1                                                                        2
                                                                  88/01/21

           2.0 NEXT
+          ________
```

# BOX CONTROL

- Procedures

  \BOX, A, B, R, NN, -NN<br>
  \MODBOX, A, B, R, NN, -NN<br>
  \BOXLINE<br>
  \NOBOX

## INPUT

```
\box,1,6,19,60
\flowtab;4,8,21
\length-5
 ;A ;ABSOLUTE ;This parameter causes the following values to
refer to the ABSOLUTE margin.
\boxline
 ;B ;BASE ;This parameter causes the following values to
refer to the BASE margin.
\boxline
 ;R ;RELATIVE ;This parameter causes the following values to
refer to the RELATIVE margin.
\nobox
\length
```

## OUTPUT

```
+----+------------+--------------------------------------+
|  A | ABSOLUTE   | This parameter causes  the  following |
|    |            | values   to  refer  to  the  ABSOLUTE |
|    |            | margin.                               |
+----+------------+--------------------------------------+
|  B | BASE       | This parameter causes  the  following |
|    |            | values to refer to the BASE margin.   |
+----+------------+--------------------------------------+
|  R | RELATIVE   | This  parameter  causes the following |
|    |            | values  to  refer  to  the   RELATIVE |
|    |            | margin.                               |
+----+------------+--------------------------------------+
```

## FEATURE SELECTION

\PARA & \NOPARA

Controls the indenting of paragraphs.

\UNDER & NOUNDER

Everything between UNDER and NOUNDER will be underlined.

\JUST & NOJUST

Right justification is default. NOJUST will stop TXTCODE efforts to align on the right margin.

\BOLD & NOBOLD

Everything between BOLD and NOBOLD will be printed bold-faced.

\SEQ & \NOSEQ

Sequence numbers will be provided five spaces to the right of the right margin if SEQ is selected.

\KEEPnn & \NOKEEP

KEEP, keeps the next nn lines on one page. NOKEEP documents the end of the kept lines.

\HEADA, \HEADB, \HEADC

Controls the style or layout of section names; for example, capitalizing, underlining, and spacing.

\MARK

Insert marks to indicate margin notes. Footnotes can be added using \FOLIO.

\SINGLE & \DOUBLE

Allows single and double spacing of text. Single spacing is the default.

\CODES

Produces the TXTFORM statistics and error summary.

\ \

Direct TXTFORM command.

## BODY TEXT COMMANDS

- Text format control

| | |
|---|---|
| \MARGINnn | Set the left margin to column nn. |
| \MARGIN | Set the left margin to the default. (10) |
| \MARGIN+nn | Increase the margin by nn. |
| \MARGIN-nn | Decrease the margin by nn. |
| \LENGTHnn | Set the number of columns per line (62). |
| \SKIPnn | Skip nn lines or to next page. |
| \PAGE | Page eject. |
| \PAGEnn | Page eject. New page number is nn. |
| \BLANKnn | Insert nn blanks. |
| \CENTER=t | Center the text (t) |
| \CENTER,B,U=t | Center, bold-face, and underline the text. |
| \CENTER,B=t | Center and bold-face the text. |
| \CENTER,U=t | Center and underline the text. |

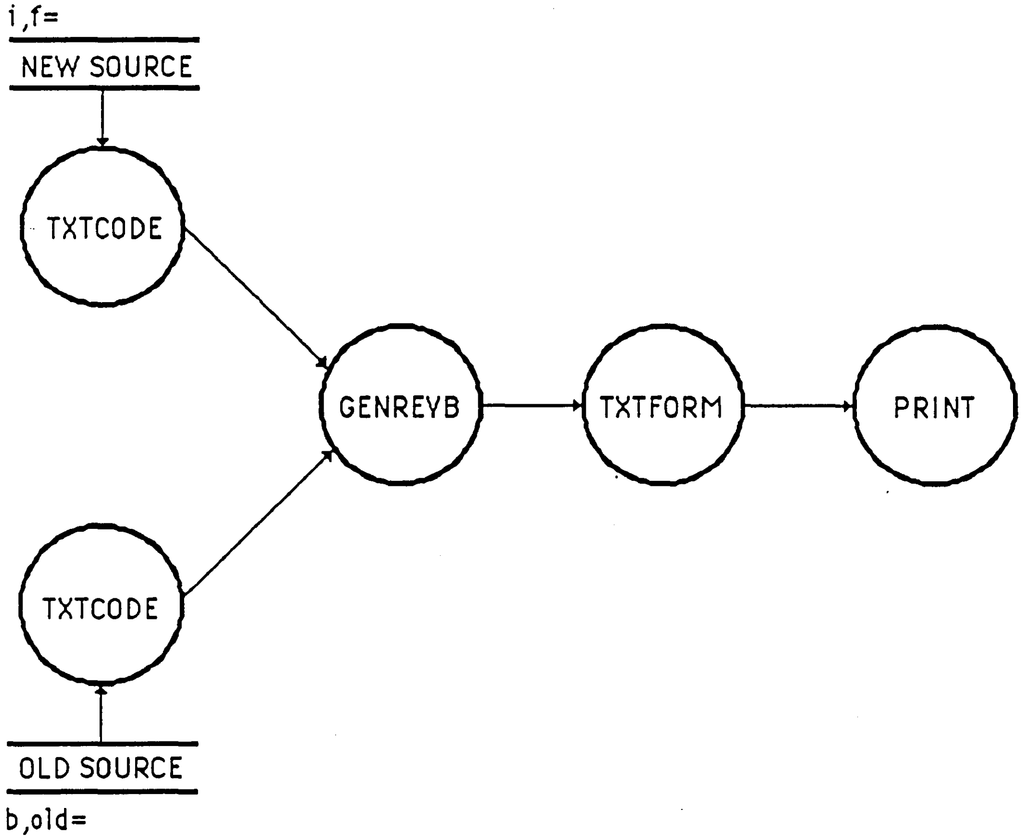
- Escape characters

| | |
|---|---|
| \SETEx | Set master control character to x. (\) |
| \SETUx | Everything enclosed by the escape character (x) is to be underlined. (~ and _ are recommended). |
| \SETBx | Everything enclosed by the escape character (x) is to be bold-faced. (% is recommended). |
| \SETHx | The escape character (x) can be used to force a blank. |

# SES.FORMAT

SES.FORMAT

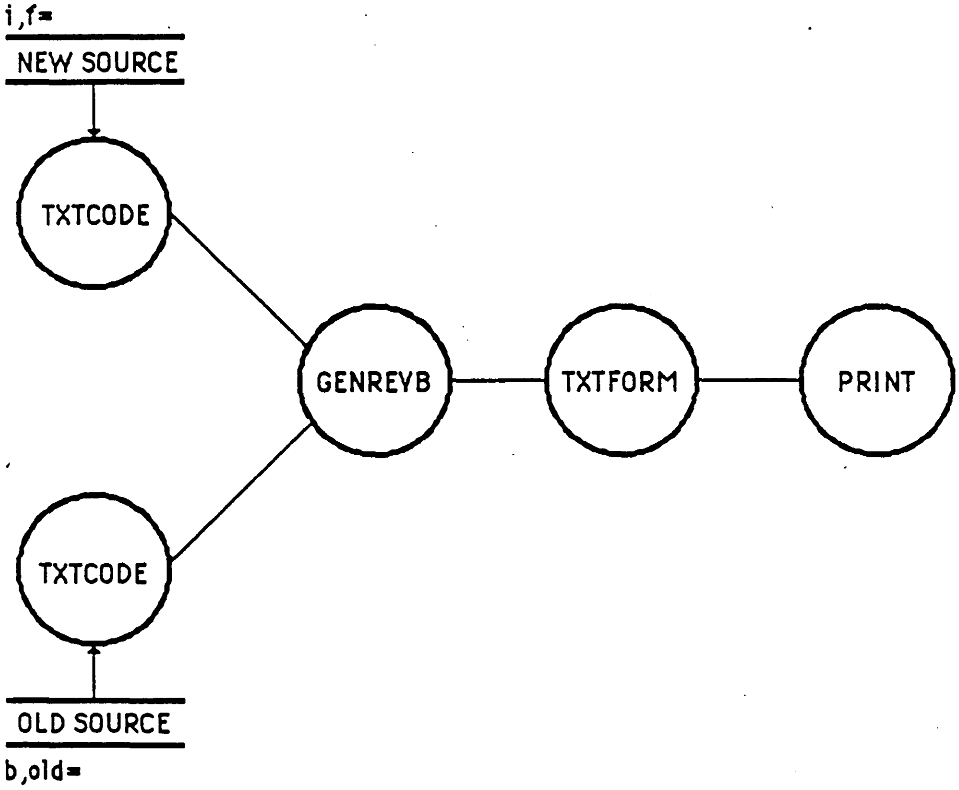
```
    i,f=input_file
    b,old=old_source
    l,listing=printable_document  [listing]
    d,display=displayable_document  [none]
    Batch job parameters [batch]
    Print=(parameters)
    KEY:  s,source
    k,keepout=document_file_name_to_define  [none]
    KEY:  code,form,txtcode,txtform  [TXTCODE]
    KEY:  head,txthead
    Twopage=(parameters)
    KEY:  msg,nomsg
          backup
```



# SES.FORMAT

```
SES.FORMAT

   i,f=input_file
   b,old=old_source
   l,listing=printable_document  [listing]
   d,display=displayable_document  [none]
   Batch job parameters [batch]
   Print=(parameters)
   KEY:  s,source
   k,keepout=document_file_name_to_define  [none]
   KEY:  code,form,txtcode,txtform  [TXTCODE]
   KEY:  head,txthead
   Twopage=(parameters)
   KEY:  msg,nomsg
         backup
```



# SES.FORMAT

```
/ses.format usdocu
 16.19.00.  SUBMIT COMPLETE.  JSN IS AIUA.
REVERT.    JOB  FORMAT  SUBMITTED


/ses.format i=usdocu s=docs k=docl
 16.20.25.  SUBMIT COMPLETE.  JSN IS AIUE.
REVERT.    JOB  FORMAT  SUBMITTED


/ses.format usdocu d=output local
*    PROCESSING TXTCODE ON USDOCU
*    FORMATED DOCUMENT ON FILE LISTING
*    CREATING DISPLAYABLE DOCUMENT OUTPUT
1                                                                    1-1
          SAMPLE DOCUMENT
                                                                yy/mm/dd
          Status Report
         --------------------------------------------------------------------
         1.0 OVERVIEW

         --------------------------------------------------------------------

            1.0 OVERVIEW
+               ________


 *TERMINATED*

ses.format i=docnews old=usdocu l=docnewl local
*    GENERATING REVISION BARS DOCNEWS : USDOCU
*    PROCESSING TXTCODE ON USDOCU
*    FORMATED DOCUMENT ON FILE LISTING
*    DOCNEWS : USDOCU -> DOCNEWL
REVERT.  END FORMAT
```

# DOCUMENT SOURCE 1

```
\length60
\tablcon
\autosec
\para
\title1= SAMPLE DOCUMENT
\title2= Status Report
\folio= CDC Private
\ n.0 Overview
 This section overviews the progress of the major projects
with respect to time and budget.
\ n.n Schedules
 The table below lists the current projects and their proposed
completion date.

\setu¬
\flowtab;5,25,37
 ;¬Project Name¬;¬Comp.Date¬;¬Comments¬
 ;Compilers ;1/2/86 ;FORTRAN is complete. LACSAP is still
waiting for a shipment of semi-colons.
 ;Tools ;2/3/87 ;The project is still waiting for
approval of the Design Requirements Document.
 ;Simulator ;3/4/88 ;On schedule.
 ;Operating System ;4/5/89;On schedule.
\block
\ n.n Budget
 This table compares the budgetted cost with the
actual cost to date.

\seth%
\table;5,25,37
 ;¬Project Name¬;¬Budget¬;¬Cost¬
 ;Compilers ;%571;%568
 ;Tools ;%118;%155
 ;Simulator ;%304;%158
 ;Operating System;1444;%489
 ;;____;____
 ;;2437;1370
\block
```

# DOCUMENT SOURCE 2

```
\ n.0 The Compilers Project
 The compilers project has responsibility for all assemblers
and compilers.

\ n.n Assemblers Subproject
 The assemblers were completed on  1/2/86. No bugs have been
found.

\ n.n Implementation Language
 The semicolon problem persists. The last shipment contained
only colons. These were accepted because it is easier to cut
colons in half than to make semicolons from scratch.

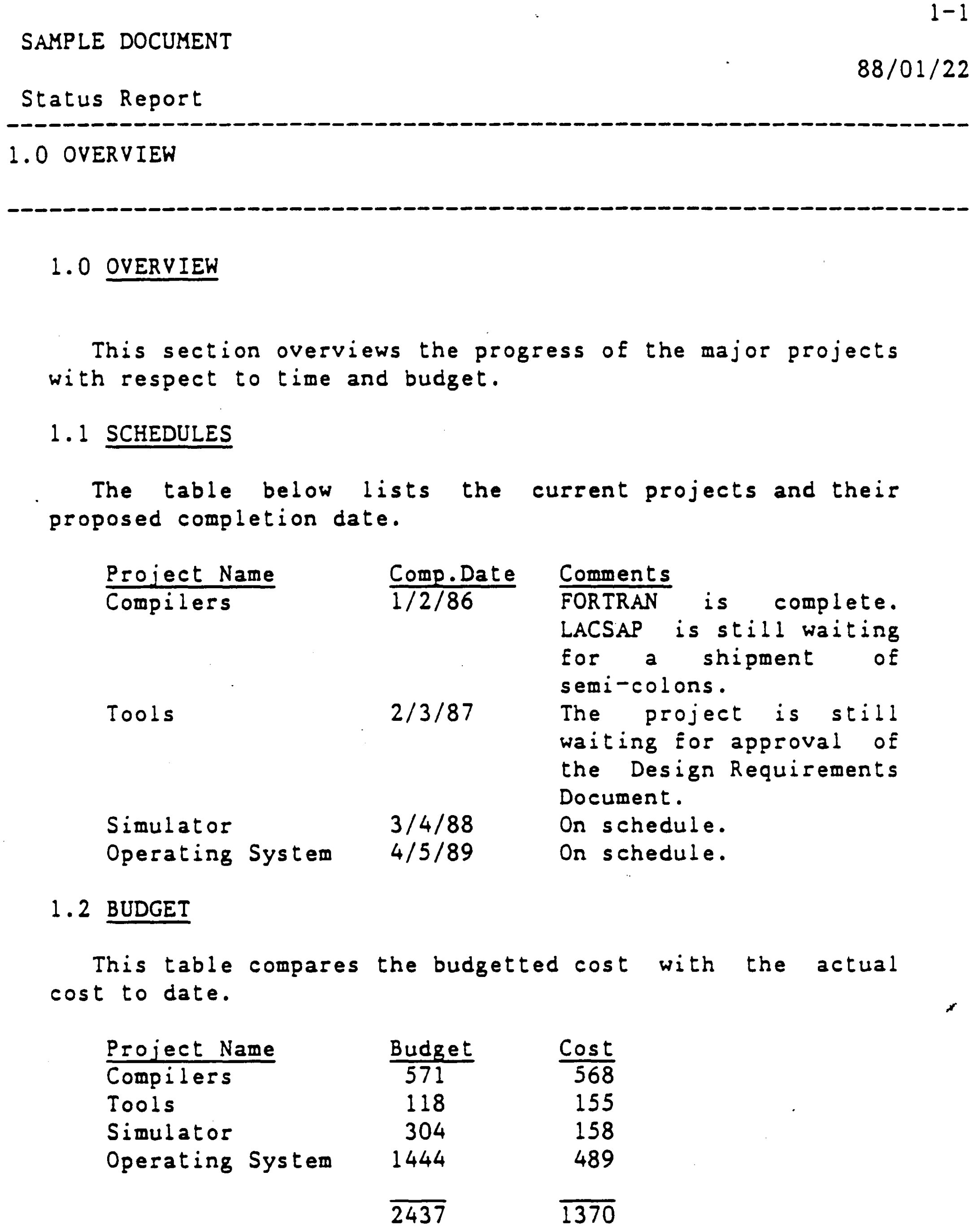
\ n.n FORTRAN
 The Fortran compiler was completed on 7/7/86. However, it was
refused by the maintenance group. The compilers project continues
to maintain it until the maintenance group is able to get someone
who understands it.

\ n.n UNIBOL
 The has been a proposal to write a UNIBOL compiler. This
proposal is being reviewd.

\keep8
\ n.n Summary
\table;5,25

 ;ASSEMBLER;Complete
 ;LACSAP;Waiting for semicolons
 ;FORTRAN;Not accepted by maintenance
 ;UNIBOL;Under consideration
\block
```

# SAMPLE DOCUMENT



```
                                                        1-1
SAMPLE DOCUMENT
                                                   88/01/22
Status Report
-----------------------------------------------------------
1.0 OVERVIEW

-----------------------------------------------------------
```

## 1.0 OVERVIEW

This section overviews the progress of the major projects with respect to time and budget.

### 1.1 SCHEDULES

The table below lists the current projects and their proposed completion date.

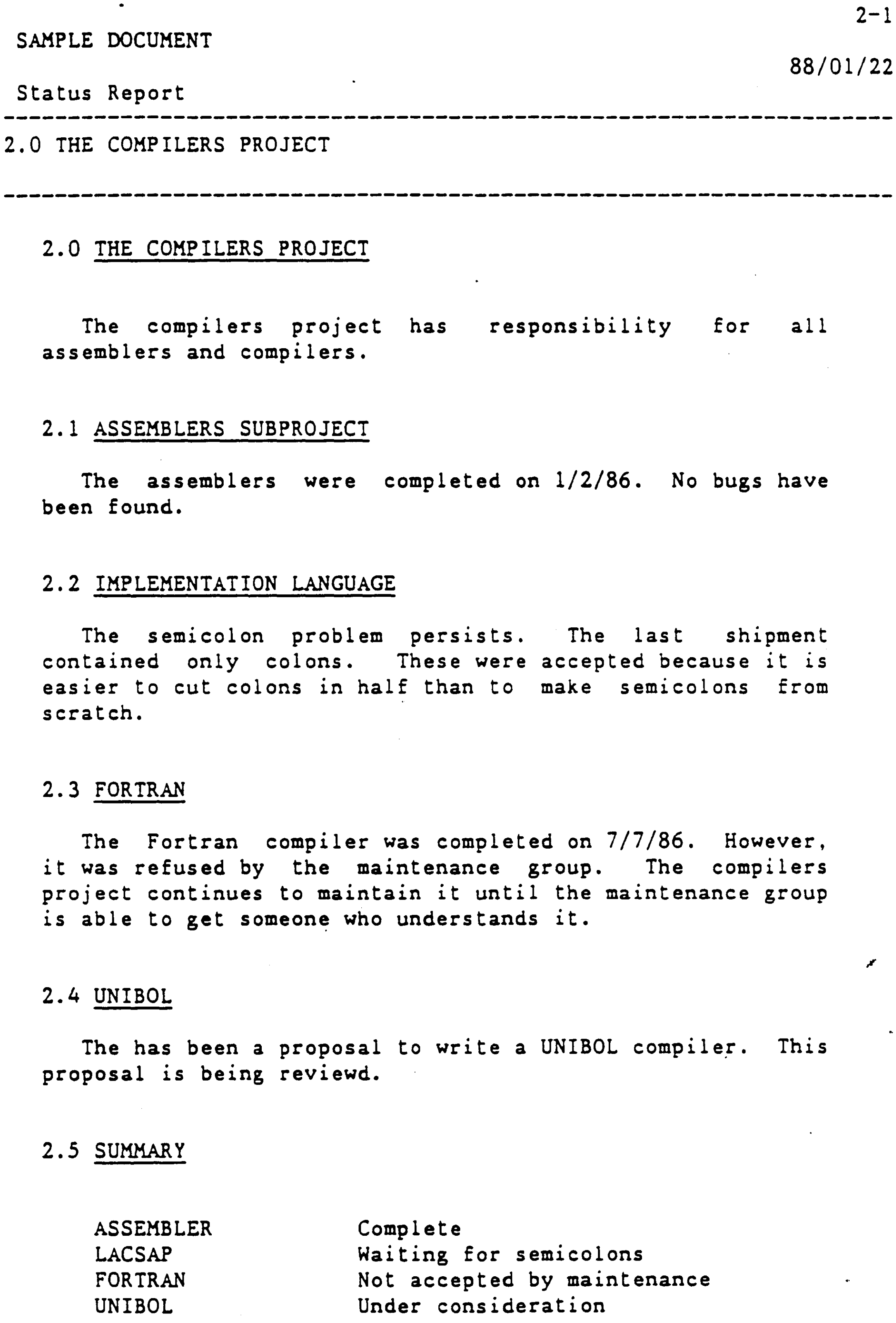
| Project Name | Comp.Date | Comments |
|---|---|---|
| Compilers | 1/2/86 | FORTRAN is complete. LACSAP is still waiting for a shipment of semi-colons. |
| Tools | 2/3/87 | The project is still waiting for approval of the Design Requirements Document. |
| Simulator | 3/4/88 | On schedule. |
| Operating System | 4/5/89 | On schedule. |

### 1.2 BUDGET

This table compares the budgetted cost with the actual cost to date.

| Project Name | Budget | Cost |
|---|---|---|
| Compilers | 571 | 568 |
| Tools | 118 | 155 |
| Simulator | 304 | 158 |
| Operating System | 1444 | 489 |
| | 2437 | 1370 |

# SAMPLE DOCUMENT



SAMPLE DOCUMENT 2-1

Status Report 88/01/22

2.0 THE COMPILERS PROJECT

## 2.0 THE COMPILERS PROJECT

The compilers project has responsibility for all assemblers and compilers.

### 2.1 ASSEMBLERS SUBPROJECT

The assemblers were completed on 1/2/86. No bugs have been found.

### 2.2 IMPLEMENTATION LANGUAGE

The semicolon problem persists. The last shipment contained only colons. These were accepted because it is easier to cut colons in half than to make semicolons from scratch.

### 2.3 FORTRAN

The Fortran compiler was completed on 7/7/86. However, it was refused by the maintenance group. The compilers project continues to maintain it until the maintenance group is able to get someone who understands it.

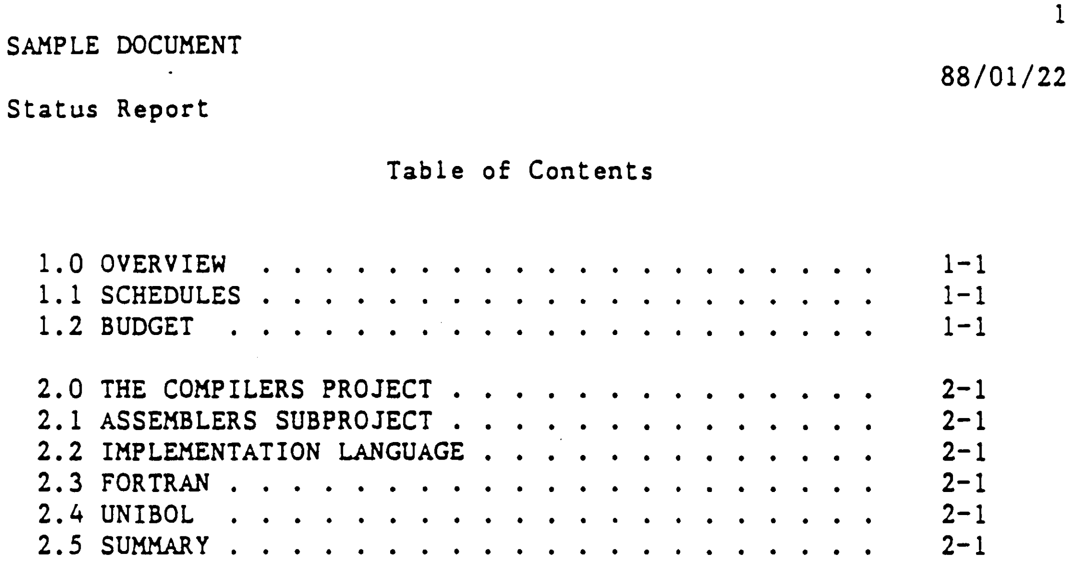
### 2.4 UNIBOL

The has been a proposal to write a UNIBOL compiler. This proposal is being reviewd.

### 2.5 SUMMARY

| | |
|---|---|
| ASSEMBLER | Complete |
| LACSAP | Waiting for semicolons |
| FORTRAN | Not accepted by maintenance |
| UNIBOL | Under consideration |

# SAMPLE DOCUMENT



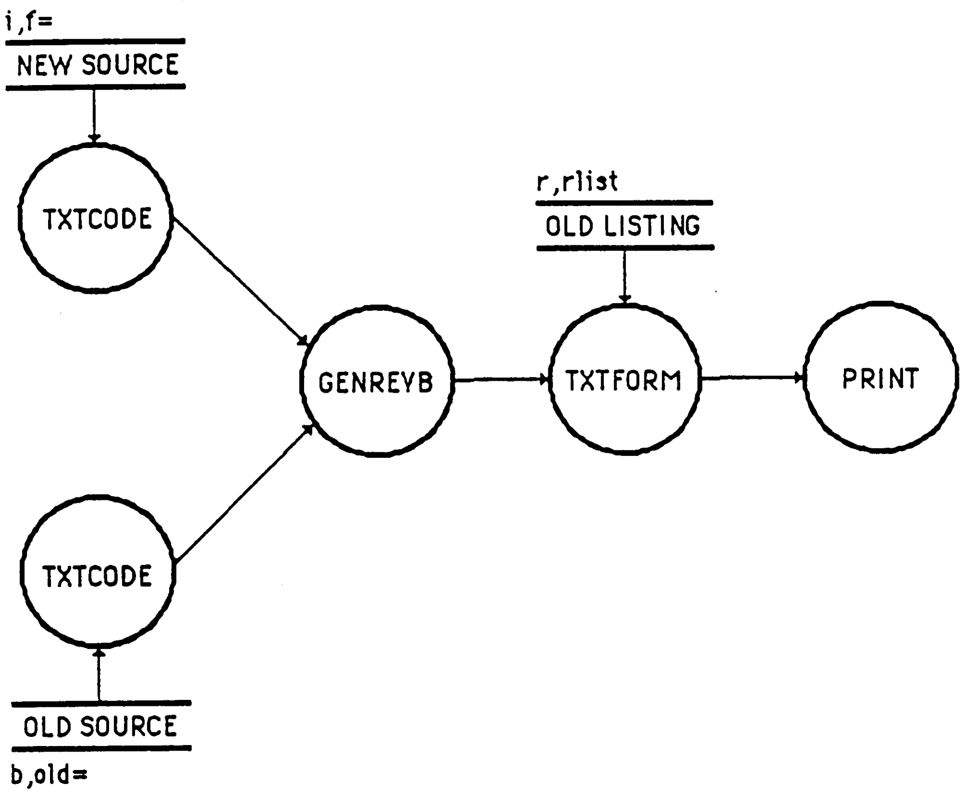
```
                                                                    1
SAMPLE DOCUMENT
                                                             88/01/22
Status Report

                         Table of Contents


  1.0 OVERVIEW  . . . . . . . . . . . . . . . . . . . . .    1-1
  1.1 SCHEDULES . . . . . . . . . . . . . . . . . . . . .    1-1
  1.2 BUDGET  . . . . . . . . . . . . . . . . . . . . . .    1-1

  2.0 THE COMPILERS PROJECT . . . . . . . . . . . . . . .    2-1
  2.1 ASSEMBLERS SUBPROJECT . . . . . . . . . . . . . . .    2-1
  2.2 IMPLEMENTATION LANGUAGE . . . . . . . . . . . . . .    2-1
  2.3 FORTRAN . . . . . . . . . . . . . . . . . . . . . .    2-1
  2.4 UNIBOL  . . . . . . . . . . . . . . . . . . . . . .    2-1
  2.5 SUMMARY . . . . . . . . . . . . . . . . . . . . . .    2-1
```

# SES.FORMREV

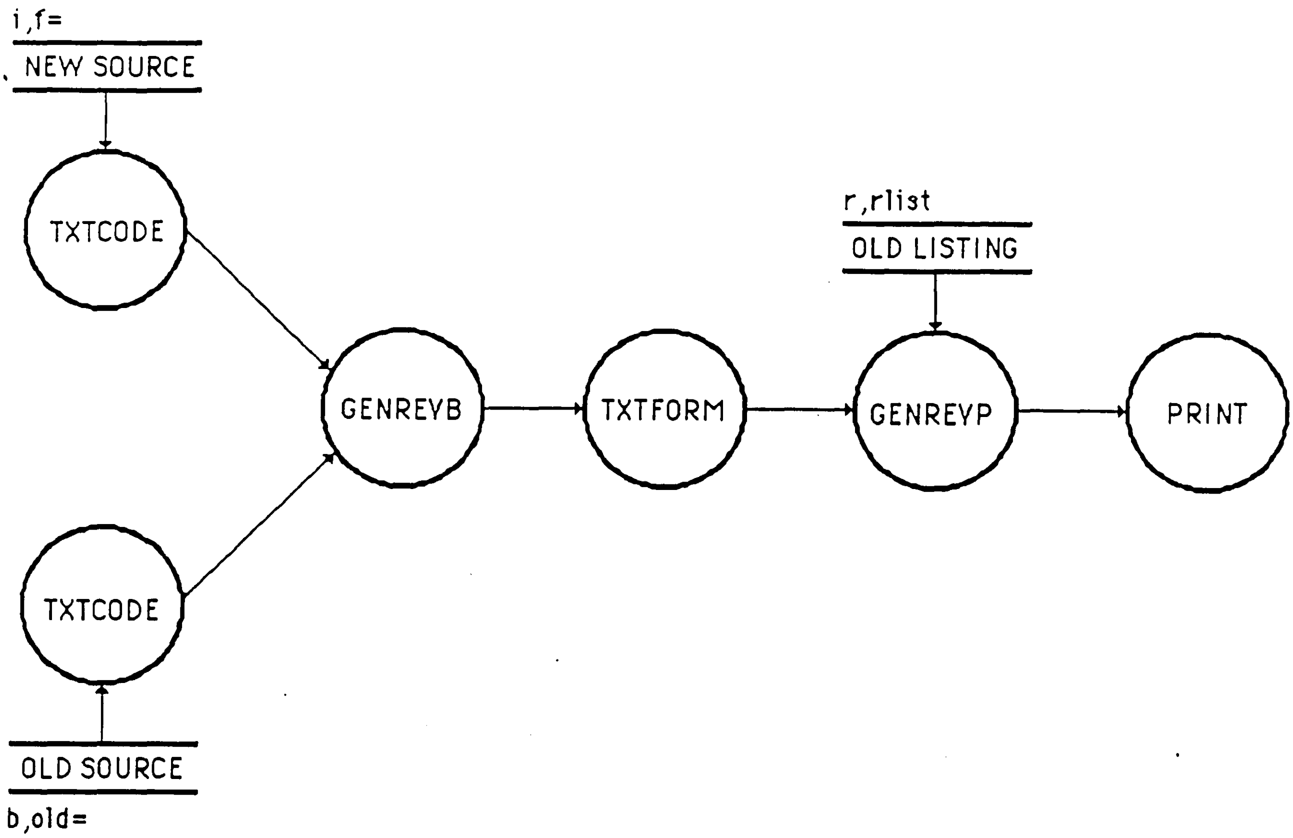


SES.FORMREV

i,f,new=new_source
b,olb=old_source
r,rlist=old_document_listing
l,listing=output=changes [LISTING]
s,summary=change_summary [SUMMARY]
o,output=message_file [* or OUTPUT]
Batch job parameters [batch]
Print=(parameters)
Format,Format1=(parameters)
Format2=(parameters)
Genrevp=(parameters)
KEY: msg,nomsg

* = Profile variable

# SES.FORMREV



SES.FORMREV

i,f,new=new_source
b,olb=old_source
r,rlist=old_document_listing
l,listing=output=changes [LISTING]
s,summary=change_summary [SUMMARY]
o,output=message_file [* or OUTPUT]
Batch job parameters [batch]
Print=(parameters)
Format,Format1=(parameters)
Format2=(parameters)
Genrevp=(parameters)
KEY: msg,nomsg

* = Profile variable

# SES.FORMREV

```
/ses.formrev i=docnews old=usdocu r=docl
 16.47.42.  SUBMIT COMPLETE.  JSN IS AIVX.
REVERT.     JOB  FORMREV  SUBMITTED


/ses.formrev i=docnews old=usdocu r=docl local
***  FORMAT NEW DOCUMENT
*    GENERATING REVISION BARS DOCNEWS : USDOCU
*    PROCESSING TXTCODE ON DOCNEWS
*    FORMATTED DOCUMENT ON FILE ZQNOI8S
*    DOCNEWS : USDOCU -> ZQNOI8S
***  COMPARE DOCUMENTS
 IGNORING LINES   2 TO   2  COLUMNS  60 TO  79    PAGE NUMBER
 IGNORING LINES   3 TO   4  COLUMNS  60 TO  79
 IGNORING LINES   1 TO ***  COLUMNS  80 TO ***
 NUMBER OF IDENTICAL PAGES =    3
 NUMBER OF ALTERED PAGES   =    1
***  DOCNEWS,DOCL --> LISTING,SUMMARY,OUTPUT
REVERT.    END FORMREV


/copy,summary
 Replace page      1-1
 EOI ENCOUNTERED.


/ses.spell docl
  REFS    LINE  WORD

   1        39  BUDGETTED
   3        65  CDC
   2        94  COLONS
   1        23  COMP.DATE
   2        26  LACSAP
   1        14  OVERVIEWS
   1       112  REVIEWD
   1        28  SEMI-COLONS
   2        83  SUBPROJECT
   4       108  UNIBOL

REVERT.   END SPELL
```

# LESSON 8

# CONVERSION UTILITIES

# LESSON 8
## CONVERSION UTILITIES

Lesson Preview:

This lesson covers editors, editing multi-record files, manipulating ASCII files, and character set conversions.

Objectives:

After completing this lesson, the student will be able to:

- Invoke the SES-supported editors using SES procedures.
- Edit multi-record files.
- Use SES procedures to copy, sort, merge, and demerge ASCII files.
- Perform character set conversions using SES procedures.

References:

- SES User's Handbook, Chapter 15
- SES User's Handbook, Appendix F, H, L
- Text Editor V1 Reference Manual

# CONVERSION UTILITIES

- Editing

| | |
|---|---|
| SES.EDT | NOS's EDIT with extra features. |
| SES.MULTED | Facilitates editing multi-record and multi-file files. |
| SES.PACK | Logical - physical EOR and EOF. |
| SES.UNPACK | Physical - logical EOR and EOF. |

- Manipulation

| | |
|---|---|
| SES.ASORT | Sort ASCII coded text. |
| SES.COPYACR | Copy ASCII coded records. |
| SES.DEMERGE | Split a file by columns. |
| SES.MERGE | Merge a file by columns. |
| SES.UNIQUE | Eliminate repeated lines. |
| SES.COMPARE | Compare ASCII files. |
| SES.COUNT | Count things in a file. |
| SES.SELECT | Select lines. |

- Character set conversion

| | |
|---|---|
| SES.LOWTOUP | Convert lower case to upper case. |
| SES.UPTOLOW | Convert upper case to lower case. |
| SES.CONV | Convert one character set to another. |
| SES.XLIT | Transliterate characters. |
| SES.FIND | Find patterns. |
| SES.CHANGE | Change one pattern to another. |

# EDT

```
SES.EDT
   i,f=input_file [*, GROUP]
   ec,input=EDT_commands [INPUT]
   eo,output=EDT_output [OUTPUT]
   t,tabs=tabs_name  [*]
   KEY: save,rep
*  = profile variable
```

- EDT Features

1. Column search control.

   COL: /c1,c2/

2. Column control on EDT commands.

   X:  Δstring1Δ,Δstring2Δ;/c1,c2/;n

   F: / 123 / ; / 2,6 /

3. Reverse find.

   F;-n

4. predefined tab stops.  See Handbook appendix.
5. List tab character and stops.

   LT

6. Termination of list output.

   | | |
   |---|---|
   | BREAKOFF | End listing, but not editor. |
   | BREAKON | End editor; do not overwrite local file. |
   | ABORT | Exit to KCL procedure. |

7. Multiple commands per line.

   S;*.S;-1. RS:/end/ ,/end;/

8. Put commands on EDT call.

   EDT,conv2,n=0. s;1. rs: /1/ ,/2/.l;*

9. Switch between full and partial ASCII.

   | | |
   |---|---|
   | UP | partial |
   | LOW | full |

10. List column numbers.

    LC;n

11. String must be delimited by special chars.

    RS: /The/;d

12. Tabs can be used in text-mode.

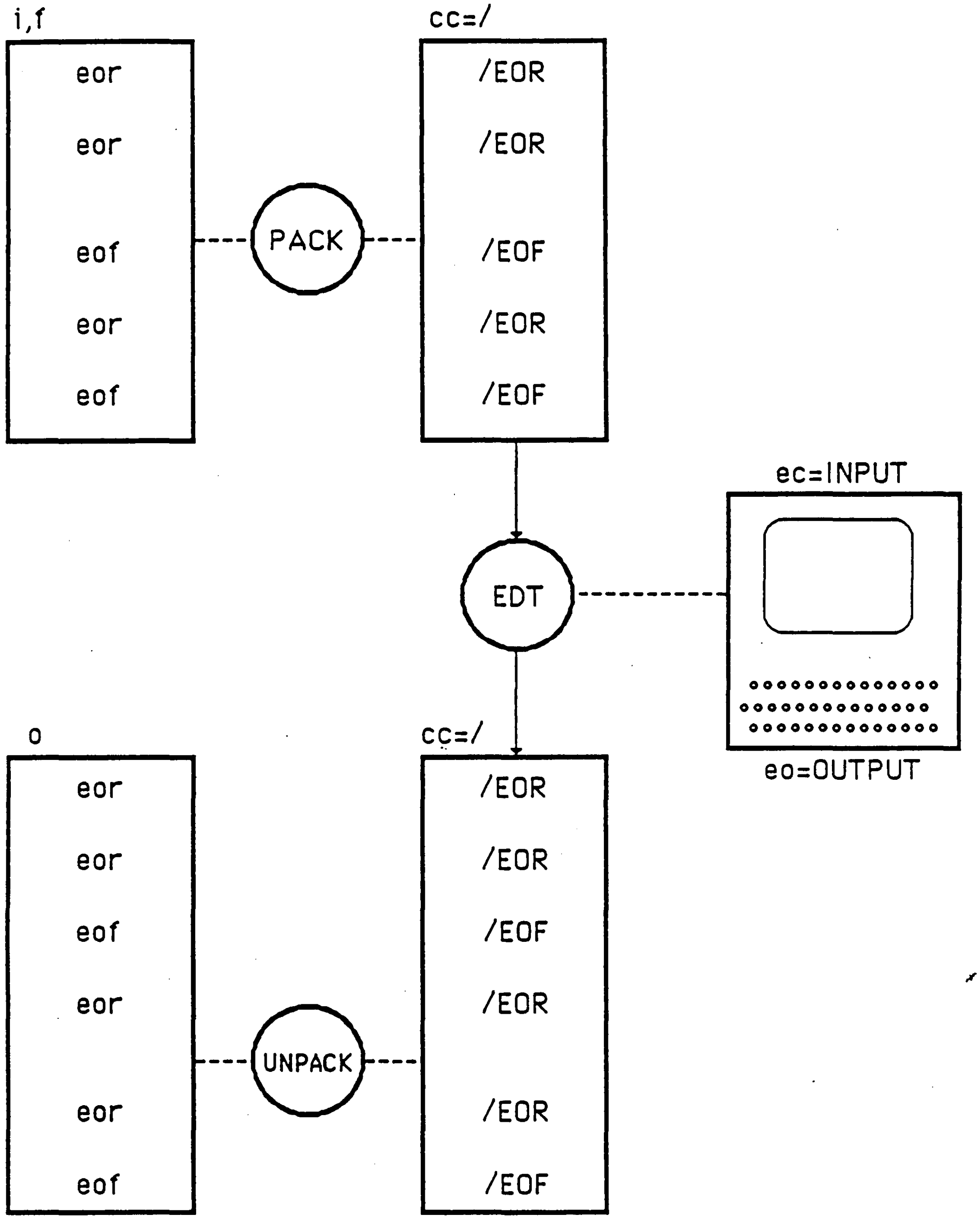
    DT: /;/

    T:  /10,20,30/

13. Comma can replace colon.

    RS:  /string1/  or  RS,/string1/

## MULTI-RECORD AND MULTI-FILE FILES



# PROCEDURES

SES.MULTED

```
i,f=input_file
o=output_file  [i,f]
ec,input=edit_command_file  [INPUT]
eo,output=edt_output  [OUTPUT]
t,tabs=(edt_tabs)  [none]
cc=control_character  [/]
KEY:  save,rep
```

SES.PACK and SES.UNPACK

```
i,f=input_file [*,  GROUP]
o=output_file  [i,f]
cc=control_character  [/]
incsest,ic=input_char_set  [*,  cs612]
outcset,oc=output_char_set  [*,  cs612]
```

- Profile variables

```
/GROUP filename
/TABS n1,n2,...
```

# SES.MULTED

```
/ses.getmods m=(sqa,sqb) b=sqbase
REVERT.    END GETMODS  DLO <- SQBASE


/ses.multed
 BEGIN TEXT EDITING.
? f:./.
/EOR
? s;-1
? 1;5
      END
/EOR
SQB
      SUBROUTINE SQUARER
*CALLC COMCAT
? end
 END TEXT EDITING.
REVERT.    END MULTED  DLO


/ses.edt
 BEGIN TEXT EDITING.
? f:./.
 PHRASE NOT FOUND.
? s;*
? f:-1
      END
? end
 END TEXT EDITING.
REVERT.    END EDT  DLO
```

# MANIPULATION PROCEDURES

## SES.COPYACR

```
i,f=input_file
o=output_file  [i,f]
cols,c=low_column. .high_column  [full width]
incset,ic=input_char_set  [*, cs612]
outcset,oc=output_char_set  [*, cs612]
```

## SES.ASORT

```
i,f=input_file
o,f=output_file  [i,f]
keys=((start, length, a or d)) [none,1,a]
KEY: retain
```

## SES.MERGE

```
o=output_file
m,merge=(column_lengths)
i,f=(input_files)  [TAPE1,...,TAPE5]
```

## SES.DEMERGE

```
i,f=input_file
d,demerge=(column_lengths)
o=(output_files)  [TAPE1,...,TAPE5]
```

# MANIPULATIONS

## ORIGINAL FILE

```
/get,usrost
/list,f=usrost
  J.H.Wick         jerry    etcpes 642-3106
  R.D.Palm         bob      arh263 482-2605
  A.R.Rothstein    ron      hqb02m 853-7075
  R.E.Jorgensen    ray      arh274 482-2853
  W.L.Reynolds     wayne    etcpes 642-3377
  H.  McGilton     henry    sv1100 826-8640
  J.J.Krautbauer   jim      arh254 482-2632
  E.  Huntley      erwin    sv1160 826-7187
```

## SORT

```
/ses.asort usrost o=o1 keys=((7,7,a))
REVERT.    END ASORT USROST -> O1
/copy,o1
  E.  Huntley      erwin    sv1160 826-7187
  R.E.Jorgensen    ray      arh274 482-2853
  J.J.Krautbauer   jim      arh254 482-2632
  H.  McGilton     henry    sv1100 826-8640
  R.D.Palm         bob      arh263 482-2605
  W.L.Reynolds     wayne    etcpes 642-3377
  A.R.Rothstein    ron      hqb02m 853-7075
  J.H.Wick         jerry    etcpes 642-3106
 EOI ENCOUNERED.
```

## COPY ASCII CODED RECORD

```
/ses.copyacr usrost output c=19..26
jerry
bob
ron
ray
wayne
henry
jim
erwin
REVERT.    END COPYACR  USROST -> OUTPUT
```

# MANIPULATIONS (CONT.)

## ORIGINAL FILE

```
/copy,usrost
  J.H.Wick        jerry   etcpes 642-3106
  R.D.Palm        bob     arh263 482-2605
  A.R.Rothstein   ron     hqb02m 853-7075
  R.E.Jorgensen   ray     arh274 482-2853
  W.L.Reynolds    wayne   etcpes 642-3377
  H.  McGilton    henry   sv1100 826-8640
  J.J.Krautbauer  jim     arh254 482-2632
  E.  Huntley     erwin   sv1160 826-7187
 EOI ENCOUNTERED.
```

## DEMERGE

```
/ses.demerge usrost d=(18,8,7)
REVERT.    END DEMERGE  USROST
/ses.merge mailist m(18,7) i=(tape1,tape3)
REVERT.    END MERGE  TAPE1,TAPE3 -> MAILIST
/copy,mailist
  J.H.Wick        etcpes
  R.D.Palm        arh263
  A.R.Rothstein   hqb02m
  R.E.Jorgensen   arh274
  W.L.Reynolds    etcpes
  H.  McGilton    sv1100
  J.J.Krautbauer  arh254
  E.  Huntley     sv1160
 EOI ENCOUNTERED.
```

## MERGE

```
/ses.merge output m=(26,26,26) i=(mailist,mailist,mailist)
  J.H.Wick        etcpes    J.H.Wick        etcpes    J.H.Wick        etcpes
  R.D.Palm        arh263    R.D.Palm        arh263    R.D.Palm        arh263
  A.R.Rothstein   hqb02m    A.R.Rothstein   hqb02m    A.R.Rothstein   hqb02m
  R.E.Jorgensen   arh274    R.E.Jorgensen   arh274    R.E.Jorgensen   arh274
  W.L.Reynolds    etcpes    W.L.Reynolds    etcpes    W.L.Reynolds    etcpes
  H.  McGilton    sv1100    H.  McGilton    sv1100    H.  McGilton    sv1100
  J.J.Krautbauer  arh254    J.J.Krautbauer  arh254    J.J.Krautbauer  arh254
  E.  Huntley     sv1160    E.  Huntley     sv1160    E.  Huntley     sv1160
REVERT.    END MERGE  MAILIST,MAILIST,MAILIST ->
```

# MANIPULATIONS (CONT.)

SES.UNIQUE

```
i,f=input_file
o=output_file  [i]
```

SES.COMPARE

```
new=new_text_file  [NEW]
old=old_text_file  [OLD]
o=output_file [OUTPUT]
newcset=cs612,cs64,cs812  [cs612]
oldcset=  cs612,cs64,cs812  [cs612]
outcset=  cs612,cs64,cs812  [cs612]
KEY: is, ignorls [LS]
```

SES.COUNT

```
KEY:   c,chars,l,lines,w,words
i,f,in:   input_file
o,to:  output_file [OUTPUT]
```

SES.SELECT

```
line,lines=(numbers  or  ranges)
i,f,of   =input_file
o,to=output_file  [OUTPUT]
```

## MANIPULATIONS (CONT.)

```
/copy,newrost
  J.H.Wick         jerry   etcpes 642-3106
  R.D.Palm         bob     arh263 482-2605
  A.R.Rothstein    ron     hqw05i 853-7075
  R.E.Jorgensen    ray     arh274 482-2853
  W.L.Reynolds     wayne   etcpes 642-3377
  H.  McGilton     henry   sv1100 826-8640
  J.J.Krautbauer   jim     arh254 482-2632
  E.  Huntley      erwin   sv1160 826-7187
 EOI ENCOUNTERED.
```

### COMPARE

```
/ses.compare newrost usrost
 AFTER LINE 2 WHICH CONTAINS
       R.D.Palm         bob     arh263 482-2605
 REPLACED
       A.R.Rothstein    ron     hqb02m 853-7075
 WITH
       A.R.Rothstein    ron     hqw05i 853-7075
REVERT.    END COMPARE  NEWROST : USROST -> OUT
```

### COUNT

```
/ses.count chars in newrost
336
REVERT.    END COUNT  NEWROST
/ses.count lines in newrost
8
REVERT.    END COUNT  NEWROST
```

### SELECT

```
/ses.select (1..4,7) of newrost
  J.H.Wick         jerry   etcpes 642-3106
  R.D.Palm         bob     arh263 482-2605
  A.R.Rothstein    ron     hqw05i 853-7075
  R.E.Jorgensen    ray     arh274 482-2853
  J.J.Krautbauer   jim     arh254 482-2632
REVERT.    END SELECT  NEWROST -> OUTPUT
```

# CONVERSION PROCEDURES

## SES.LOWTOUP and SES.UPTOLOW

```
i,f=input_file
o=output_file  [i,f]
```

## SES.CONV

```
i,f=input_file
o=output_file  [i,f]
incset,ic=input_char_set
outcset,oc=output_char_set
```

- Character sets

| CHARACTER SET | NOS | NOS/BE | N/WORD | N | COMMENTS |
|---|---|---|---|---|---|
| n63 | X | | 10 | 63 | |
| n64 | X | | 10 | 64 | cs64 |
| n612 | X | | 05 | 256 | cs612 |
| n612u | X | | 05 | 64 | upper case |
| n6121 | X | | 05 | 64 | lower case |
| nbe63 | | X | 10 | 63 | |
| nbe64 | | X | 10 | 64 | |
| nbe63a | | X | 05 | 63 | ascii subset |
| nbe64a | | X | 05 | 64 | ascii subset |
| ascii4 | X | X | 04 | 256 | |
| ASCII5 | X | X | 05 | 256 | cs812 |

# CONVERSIONS

## ORIGINAL FILE

```
/copy,newrost
  J.H.Wick         jerry   etcpes 642-3106
  R.D.Palm         bob     arh263 482-2605
  A.R.Rothstein    ron     hqw05i 853-7075
  R.E.Jorgensen    ray     arh274 482-2853
  W.L.Reynolds     wayne   etcpes 642-3377
  H.  McGilton     henry   svl100 826-8640
  J.J.Krautbauer   jim     arh254 482-2632
  E.  Huntley      erwin   svl160 826-7187
 EOI ENCOUNTERED.
```

## LOWER TO UPPER CASE

```
/ses.copyacr newrost lll c=19..26
REVERT.    END COPYACR  NEWROST -> LLL
/ses.lowtoup lll
REVERT.   END LOWTOUP  LLL
/copy,lll
JERRY
BOB
RON
RAY
WAYNE
HENRY
JIM
ERWIN
 EOI ENCOUNTERED.
```

## CHARACTER SET CONVERSION

```
/ses.conv newrost output ic=n612 oc=n612u
  J.H.WICK         JERRY   ETCPES 642-3106
  R.D.PALM         BOB     ARH263 482-2605
  A.R.ROTHSTEIN    RON     HQW05I 853-7075
  R.E.JORGENSEN    RAY     ARH274 482-2853
  W.L.REYNOLDS     WAYNE   ETCPES 642-3377
  H.  MCGILTON     HENRY   SVL100 826-8640
  J.J.KRAUTBAUER   JIM     ARH254 482-2632
  E.  HUNTLEY      ERWIN   SVL160 826-7187
REVERT.    END CONV  N612/NEWROST -> N612U/OUTPU
```

# CHANGING PATTERNS

SES.XLIT TRANSLITERATE

```
    i,f=file_to_be_changed
    from='old pattern'
    to='new pattern'
    with,using=pattern_file
    onto,o=output_file  [i]
```

SES.FIND

```
    p,pattern='search pattern' or
    with,using=pattern_file  [INPUT]
    i,in,f=file_to_search
    o,onto=output_file  [i]
```

SES.CHANGE

```
    i,f=file_to_search
    from='old_pattern'
    to='new_pattern'
    with,using=pattern_file
    onto,o=change_file  [i]
```

# PATTERNS

- Types of patterns
  - Character strings
  - At beginning of line
  - At end of line
  - With characters between
  - Classes of characters
  - Repeated strings

- Examples

```
/copy,usrost
  J.H.Wick        jerry   etcpes 642-3106
  R.D.Palm        bob     arh263 482-2605
  A.R.Rothstein   ron     hqb02m 853-7075
  R.E.Jorgensen   ray     arh274 482-2853
  W.L.Reynolds    wayne   etcpes 642-3377
  H.  McGilton    henry   svl100 826-8640
  J.J.Krautbauer  jim     arh254 482-2632
  E.  Huntley     erwin   svl160 826-7187
 EOI ENCOUNTERED.
/ses.find i=usrost o=output
? svl
  H.  McGilton    henry   svl100 826-8640
  E.  Huntley     erwin   svl160 826-7187
REVERT.    END FIND  USROST -> OUTPUT
/ses.find i=usrost o=output
? ..J
  J.H.Wick        jerry   etcpes 642-3106
  R.E.Jorgensen   ray     arh274 482-2853
  J.J.Krautbauer  jim     arh254 482-2632
REVERT.    END FIND  USROST -> OUTPUT
/ses.find i=usrost o=output
? ↑..J
  J.H.Wick        jerry   etcpes 642-3106
  J.J.Krautbauer  jim     arh254 482-2632
REVERT.    END FIND  USROST -> OUTPUT
/ses.find i=usrost o=output
? J@.J
  J.J.Krautbauer  jim     arh254 482-2632
REVERT.    END FIND  USROST -> OUTPUT
```

## LESSON 9

# MISCELLANEOUS PROCEDURES

# LESSON 9

## MISCELLANEOUS PROCEDURES

Lesson Preview:

This lesson covers calculator, generating control statement, splitting and concatenating files and producing an interpreted hex dump. Setting IAF defaults and producing an object list are also discussed.

Objective:

After completing this lesson, the student should be able to use the procedures that do not fit into any particular category.

Reference:

- SES User's Handbook, Chapter 16

# LESSON 9

## MISCELLANEOUS PROCEDURES

Lesson Preview:

This lesson covers calculator, generating control statement, splitting and concatenating files and producing an interpreted hex dump. Setting IAF defaults and producing an object list are also discussed.

Objective:

After completing this lesson, the student should be able to use the procedures that do not fit into any particular category.

Reference:

- SES User's Handbook, Chapter 17

# PROCEDURES

- SES.IAF

  - Changes defaults that are established by NOS Interactive Facility (IAF). The control, line cancel, interrupt, and terminate characters can be changed. The page length and page width can be reset. See the User's Handbook for the current defaults and the parameters for changing these things for the type of terminal that you have. Also, SES.IAF will alert you about mail and current SES information.

## PROCEDURES (CONT.)

SES.CONCAT

```
i,f=(files_to_be_concatenated)
g,group=multi_record_file  [GROUP]
KEY:  msg,nomsg
```

SES.SCATTER

```
m,f=(files_to_receive_records)
g,group=multi_record_file  [GROUP]
KEY:  nr - don't rewind group file
```

# PROCEDURES (CONT.)

SES.DO

- Control statement generator

  cs=('control statements')  
  i=commands [INPUT]  
  l=library  
  un=user_name [current]  
  f,file,files=(local file names)  
  Batch Job Parameters [local]

SES.MATH

- Calculator

  =,+,-,/,//,*,** expression  
  name=expression

SES.BELL

- Ring bell

  n=number of rings

SES.BYE

- Logs you off

# DO AND MATH

## SES.MATH

```
/ses.math
VALUE = 0
? +41
VALUE = 41
? +9*3      27
VALUE = 68
? =(value+2)*3
VALUE = 210

? =0
VALUE = 0
? +10(16)
VALUE = 16
? -10(8)
VALUE = 8

? =hex(60+60)
VALUE = 78
? =hex(100+100)
VALUE = 0C8

? =hex(37(8)+5*0a0(16))
VALUE = 33F
? quit
 -EXPECTING EQUAL SIGN-
QUIT = 0
? bye
$REVERT.CCL
```

## SES.DO

```
ses.do batch ..
..? cs=('ses.gencomp all b=sqbase; ftn5', ..
..? 'lgo', ..
..? 'dayfile,d', ..
..? 'save,d')
 13.55.13. SUBMIT COMPLETE.  JSN IS AQBY
REVERT.    JOB  DO  SUBMITTED
```

# APPENDIX A

## PROBLEM SETS

# LAB FOR LESSON 1

1. Login to NOS in order to use SES. We described a login sequence in lesson one. Follow this sequence, substituting either the student account information provided by the instructor or your own account information at your own place of work.

2. Build a profile. The BLDPROF procedure is the recommended way to create a profile.

3. Add to your profile, a variable called BASE with the value of XXXHOME where XXX represents something unique like your initials. Any editor that you are familiar with can be used to accomplish this in a similar fashion to the example that added a variable called GROUP with its value.

4. Use the mail procedures to send, receive and read messages. If you used the BLDPROF procedure to accomplish step 2, then you not only have a permanent file called PROFILE but also another permanent file called MAILBOX. Create a message using any editor. If you are on a student account, then MAIL the message to another student, else a reasonable exercise is to send the message to yourself.

# LAB FOR LESSON 2

1. Use SES.TOOLDOC to see what is available via SES on this machine. Only in the case where you are NOT on a student account, should you print any document which you need.

2. Compare SES.CATLIST with the NOS command CATLIST.

3. List 10 lines of your dayfile starting at the last occurrence of the phrase 'CAT'. Try again, looking for the phrase 'S.CAT'.

4. Use SES.INFO to write information to a file called XYZ. LIST or COPY or edit the file XYZ in order to look at what it says.

5. Get status for some SES procedure. The file XYZ should suggest that you access the status of some procedure. If it does, then get status for that procedure. Remember that not all procedures have status available, and potentially, none will have status.

6. Compare SES.FILES with the NOS command ENQUIRE,F.

## LAB FOR LESSON 2

1. Use SES.TOOLDOC to see what is available via SES on this machine. Only in the case where you are NOT on a student account, should you print any document which you need.

2. Compare SES.CATLIST with the NOS command CATLIST.

3. List 10 lines of your dayfile starting at the last occurrence of the phrase 'CAT'. Try again, looking for the phrase 'S.CAT'.

4. Use SES.INFO to write information to a file called XYZ. LIST or COPY or edit the file XYZ in order to look at what it says.

5. Get status for some SES procedure. The file XYZ should suggest that you access the status of some procedure. If it does, then get status for that procedure. Remember that not all procedures have status available, and potentially, none will have status.

# LAB FOR LESSON 3 AND 4

1. Get help for SES.PRINT.

2. Test PRINTID and/or COPYSAF. Create an input file. Process it. Examine the output file.

3. Create a FORTRAN program or a CYBIL program, as trivial or as complex as you want.

4. Compile your program.

5. If necessary, edit again and compile again until there are no compilation errors.

6. Use a formatter to change the error free source program into a formatted error free source program on another file. Compare the unformatted program with the formatted version.

# LAB FOR LESSON 5

## CREATING A SOURCE LIBRARY

1. Create files having these names and contents:

| | | | |
|---|---|---|---|
| Filename = | RON | Contents = | R. Reagan<br>White House<br>*call wash |
| Filename = | BOB | Contents = | R. Price<br>Tower<br>*call bloom |
| Filename = | MIKE | Contents = | M. Conrad<br>Mod B<br>*call bloom |
| Filename = | WASH | Contents = | Washington<br>DC<br>*call USA |
| Filename = | USA | Contents = | United States of America |

2. Use SES.COLLECT to create a group file which contains all six files form step 1. Note: RON, BOB, and MIKE should be regular decks; WASH, BLOOM, and USA are common decks.

3. Create a source library called XXXHOME where XXX are 3 unique characters. The group file from step 2 is the input to SES.REPMOD. Hint: Does anything you did in the first lab save you typing a parameter.

4. List the decks on your source library using SES.CATBASE.

5. Generate cross reference information using SES.LISTMOD. Note: This can be done either BATCH or LOCAL.

6. Examine the cross reference information from step 5 to determine which decks call BLOOM.

7. Expand the decks on the library for everyone living in Bloomington using SES.GENCOMP. Hint: The M parameter can take a value list like M=(deck1,deck2).

# ALTERNATE LAB FOR LESSON 5 USING UPDATE

## CREATING A SOURCE LIBRARY

1. Create files having these names and contents:

| | | | | | |
|---|---|---|---|---|---|
| Filename | = | RON | Contents | = | *DECK RON<br>R. Reagan<br>*CALL WASH |
| Filename | = | BOB | Contents | = | *DECK BOB<br>R. Price<br>Tower<br>*CALL BLOOM |
| Filename | = | MIKE | Contents | = | *DECK MIKE<br>M. Conrad<br>Mod B<br>*CALL BLOOM |
| Filename | = | WASH | Contents | = | *COMDECK WASH<br>Washington<br>DC<br>*CALL USA |
| Filename | = | BLOOM | Contents | = | *COMDECK BLOOM<br>Bloomington<br>MN<br>*CALL USA |
| Filename | = | USA | Contents | = | *COMDECK USA<br>United States of America |

2. Use SES.COLLECT to create a group file which contains all six members from step 1. Note: All files should be given as members.

3. Create a source library called XXXSLIB where XXX are 3 unique characters. The group file from step 2 is the modification input to SES.UPDATE which causes a creation run. One or more profile variables might help.

4. List the decks on your source library. Create an input file, IN which has contents = *IDENT LOOK. Then ACQUIRE,XXXSLIB. And UPDATE, I=IN,L=F, P=XXXSLIB.

5. Skip.

6. Examine the input information from step 1 to determine which decks call BLOOM.

7. Expand the decks on the library for everyone living in Bloomington using SES.GENUPCF. Hint: The M parameter can take a value list like M=(deck1,deck2).

## LAB FOR LESSON 5 (CONT.)

### MANIPULATE DECKS

8. Add a new deck as follows:
   Filename = NAN        Contents = N. Reagan
                                    White House
                                    *call wash

   Hint: edit, collect, and repmod

9. Delete the deck MIKE permanently. Hint: "Wipe" it out.

10. Set up a text file as follows:
    Filename = DON        Contents = D. Miller
                                     Mod B
                                     *call bloom

11. Expand this text file using the common decks from XXXHOME; but without first adding it to the library. Hint: Check the SF parameter for GENCOMP.

### EDIT DECKS

12. Suppose R. and N. Reagan move from the White House to a Palace in Bloomington. Edit the decks RON and NAN to reflect the move. Hint: getmod, edit, gencors, and modify.

13. Demonstrate that everyone is now in their proper place. Hint: This can be done at least 2 ways, either gencomp for the compiler view or listmod for the library view.

# LAB FOR LESSON 7

Generate a three page document.

Page 1: Center the following title

COMPILE
from
SOURCE LIBRARY
and
EXECUTE

Page 2:

PROCESS

```
+--------+--------+
|    GENERATE     |
|    COMPILE      |
|      FILE       |
+--------+--------+
         |
         |
         |
+--------+--------+
|                 |
|     COMPILE     |
|                 |
+--------+--------+
         |
         |
        / \
+------/   \------+                      +--------+--------+
|       ABS       |  Y                   |       LINK      |
|      NEEDED     |----------------------|      OBJECT     |
|        ?        |                      |     MODULES     |
+------\   /------+                      +--------+--------+
        \ /                                       |
         |                                        |
         |  N                                     |
         |                                        |
+--------+--------+                      +--------+--------+
|     EXECUTE     |                      |     EXECUTE     |
|       REL       |                      |       ABS       |
|     BINARY      |                      |     BINARY      |
+--------+--------+                      +--------+--------+
```

# LAB FOR LESSON 7 (CONT.)

Page 3: Use item and box commands to produce the following:

## EXPLANATION

1. GENERATE COMPILE FILE
   Expand selected decks from source library.
   Use "SES.GENCOMP".
2. COMPILE
   Compiles source statements and produces a relocatable binary file.
   Use "SES.FTN5 or SES.CYBIL".
3. EXECUTE REL BINARY
   Use a file name call.
4. LINK OBJECT MODULES
   Links relocatable binary modules to produce an absolute binary file.
   Use "SES.LINK170".
5. EXECUTE ABS BINARY
   Use a file name call.

## EXAMPLES

```
+-------------------------------+
|  / SES.GENCOMP all b = base   |
|  / SES.FTN5                   |
|  / LGO                        |
+-------------------------------+
```

```
+-------------------------------+
|  / SES.GENCOMP all b = base   |
|  / SES.FTN5                   |
|  / SES.LINK170                |
|  / LGOB                       |
+-------------------------------+
```

SOFTWARE ENGINEERING SERVICES

# TXTCODE 2.0

Reference Manual

60460280 01

Received 04/05/88

# REVISION RECORD

| REVISION | | DESCRIPTION |
|---|---|---|
| 01 | (02-17-84) | Preliminary manual released. |

\ddress comments concerning this manual to:

Control Data Corporation
Software Engineering Services
4201 North Lexington Avenue
St. Paul, Minnesota 55112

i0460280 01

Table of Contents

```
        +---------+            TXTCODE
       /         /|           OVER-VIEW
      /  TERM   / |           ---------               +---------+
     /         /  |                                  /-- ---- -/|
    +---------+   |                                 +---------+||
    | -- - -  |   |                                 |o OUTPUTo|||
    |  --- -  |   +                                 |o LIST- o|||
    | -- ---  |  /                                  |o  ING  o||+
    |  -- --  | /                                   |o       o|/
    | ######  |/                                    +----A----+
    +--+------+                                          |
       |                                                 |
       V                                           +-----+-----+
    +--+------+                                    |  CVT096   |
    |  \ \STOP|                                    +-----A-----+
    |   \     +---------+                                |
    |    \    |     ____V___                       +-----+-----+
    | \GO \   |    (________)                      |           |
    +--+------+    | BIT    |                      | TXTFORM   |
       |           | BUCKET |                +---->+ FORMATTER |
       V           |__ ___ _|                |     |           |
    +--+------+       :   :                  |     +-----A-----+
    |  \ \TXTF|                              |           |
    |   \     +------------------------------+           +------+
    |    \    |                                                 |
    |\ENDT\   |                                                 |
    +--+-+----+                                                 |
       | |           /\\ OR \            +------------+         |
       | +------+-->/\REGULAR\---------->| COMMAND    +-------->|
       |        |   \ COMMAND/           | PROCESSING |         |
       |        |    \______/            +------------+         |
       |        |                                               |
       |        |                                               |
       |        |                                               |
       |        |                                               |
       |        |     ______                                    |
       |        |    /      \                                   |
       |        +-->/ \MODE  \                       MODE       |
       |           \ COMMAND/                    PROCESSING:    |
       |            \______/                      (ABCFIT)      |
       +---+             *       +--------+      +---------+    |
           |             *       |        +----->| ASIS    |--->|
           |             * * * * |        |      +---------+    |
         __V____                 |        +----->| BLOCK   |--->|
        /      \                 |        |      +---------+    |
       /  DATA  \--------------->|  MODE  +----->| COMMENT |--->|
       \  LINE  /                |        |      +---------+    |
        \______/                 | SWITCH +----->| FLOWTAB |--->|
                                 |        |      +---------+    |
                                 |        +----->| ITEM    |--->|
                                 |        |      +---------+    |
                                 |        +----->| TABLE   |----+
                                 +--------+      +---------+
```

## 1.0 INTRODUCTION

Thus far, automated document generation here at CDD has been accomplished using low level commands which, for the most part, only work on immediate and limited portions of text. Few commands exercise significant control over large portions of text. As a result, document preparation is hampered by the relatively large amount of work needed to prepare a text file for submission to a text formatter such as TEXTJAB or TXTFORM. In TXTFORM, lines containing commands often outnumber lines of text. Thus, the input file to the formatter is hard to read and work on.

TXTCODE is designed to allow high level formatting instructions, which correspond directly to common text structures, to be used to direct document generation. In this way, commands of great power and range are able to format a whole "text structure" using, in some cases, only an escape code and a word.

TXTCODE is designed so that the text being input by the user will help to trigger in his mind the appropriate TXTCODE command, or format. The entered text also suggests to the user the format of the original. Since relatively few codes are present the input file is normally very readable. Thus on-line editing is simpler too - both visually and from an EDITing standpoint. TXTCODE's input format, which is basically free-form, allows lines and text strings to be freely replaced, added and deleted.

## 2.0 MODE OF OPERATION

### Philosophy

The basic idea of TXTCODE is to have a command which corresponds to the text structure which is being input, so that the formatter "knows" what it's trying to construct. Thus it will interpret the input in the context of what it is expecting and fit the input into the expected pattern. To keep things synchronized, each input "mode" (context) has some simple conventions which trigger formatting actions at the correct time. These conventions consist of:

- blanks at the start of some lines
- trigger characters embedded in the text for some modes

### Definitions

BLANK LINE: a line which has no text data on it on input will normally generate one blank line at that point in the text on output.

COMMAND LINE: lines which start in position one with the master escape character (back slant \ by default, located to the right of the blue RETURN key on 713 terminals), are assumed to be TXTCODE command lines. Such lines are used to give various kinds of instructions to the formatter. In some cases, text is expected on the same line with the command.

DATA LINE: any line which is not a BLANK LINE or a COMMAND LINE is interpreted as a DATA LINE. This type of line is processed by the currently active mode (ASIS, BLOCK, COMMENT, FLOWTAB, ITEM, or TABLE) and is formatted into the corresponding text structure.

NOTE: Under certain circumstances, any or all of the above types of lines.may be ignored by the formatter. This happens in two cases: 1) a \STOP command has been issued and a \GO has not been encountered, 2) a \TXTF command was given and a \ENDT has not yet been found. The STOP/GO takes precedence over the TXTF/ENDT command.

Processing

The 6 mode commands and the section number commands are the heart of TXTCODE and with the occasional use of some of the other commands most text forms should be easily handled.

Before describing the commands and modes, an overview of TXTCODE's approach will be given.

At the highest level, \STOP and \GO have absolute control over the input file. All input text following a \STOP, until a \GO occurs will be utterly ignored. Thus sections of the input file may be "turned-off".

The next level down is TXTF/ENDT. Once \TXTF has been called, all input to TXTCODE (which reaches this level) is assumed to be for TXTFORM processing. Hence, the information is simply passed along to TXTFORM until a \ENDT is encountered which returns control to the current mode. TXTFORM would only be called though, if TXTCODE couldn't format a certain section of input text. With the addition of direct TXTFORM commands, there should be very few cases where \TXTF and \ENDT are actually needed. It should be used mainly to encapsulate existing TXTFORM input text.

Going down another level brings the input to TXTCODE proper. The input line is first checked to see if it starts with the escape code (usually \) and, if so, the mode is altered or some feature is set on/off etc. If the line does not start with the escape code, it is processed by the mode which is currently operational (i.e. interpreted according to the present context) with consideration given to any "feature" which is active at that time. (e.g. JUSTification ON in TABLE mode has no effect).

Normally a blank input line is used to generate a blank output line.

Input line length is 80 characters of information.

## 3.0 SUMMARY OF COMMANDS

All commands may be entered in either lowercase or uppercase.

### Input File Control Commands

```
\STOP
\GO
\TXTF
\ENDT
```

### Mode Commands

```
\ASIS
\BLOCK,Jnn,Inn,Hnn                 eg. \BLOCK,J1
\COMMENT
\FLOWTABxtt,tt,...tt,ff                \FLOWTAB;1,20
\ITEM,Jnn,Inn,Snn,Gnn,Hnn,Zxx,C=e,O   \ITEM,G2
\TABLExtt,tt,...tt                     \TABLE;10,20,30...
```

### Mode Nesting Commands

```
\+[...+]mode command   \+[...+]
\-[...-]mode command   \-[...-]
\*mode command
\NESTIND               \NONESTIND
\EXITNEST,C
\CLEARNEST
```

### Designation Text Control

```
\TITLEn=xxx...x           \TITLE=xxxx...x
\DATE=xxx...x
\FOLIO=xxx...x
\TABLCON
\AUTOSEC                  \NOAUTOSEC
```

### Text Body Commands

. Section numbers

```
\ n.n....n xxxxx...x   or  \ n xxx...x   or  \ xxx...x
\SETSECnn
```

. Feature Selection De-selection

```
\BOLD   \NOBOLD
```

```
\CODES
\HEADA   \HEADB    \HEADC
\JUST    \NOJUST
\KEEPn   \NOKEEP
\MARK
\PARA    \PARAnn   \NOPARA
\SEQ     \NOSEQ
\SINGLE  \DOUBLE
\UNDER   \NOUNDER
```

. Text Format Control

```
\MARGINnn         \MARGIN     \MARGIN+nn     \MARGIN-nn
\LENGTHnn         \LENGTH     \LENGTH+nn     \LENGTH-nn
\SKIPnn
\PAGE             \PAGEnnn
\CENTER=xxx...x   \CENTER,B,U=xxx...x
\BLANKnn
```

. Appendices

```
\SETAPP=a-xxx...x
```

. Box Generation Commands

```
\BOX      \BOX,[-]nn,A,nn,B,[-]nn,R,[-]nn
\MODBOX,[-]nn,A,nn,B,[-]nn,R,[-]nn
\BOXLINE
\NOBOX
```

. Escape Character Definition

```
\SETEx
\SETUx
\SETBx
\SETHx
```

Direct TXTFORM Commands

```
\\command
```

## 4.0 DESCRIPTION OF MODES

One mode of the 6 (ASIS, BLOCK, COMMENT, FLOWTAB, ITEM, or TABLE) is always active even if \STOP or \TXTF has been called. When the \GO or \ENDT is given, the mode resumes processing as if it had not been interrupted.

The initial mode is BLOCK. Transition to another mode is accomplished by simply calling the desired mode with or without any mode parameters. Initially the left margin is set at 10, and the right margin to 62 positions to the right of the left margin (i.e. LENGTH62).

Relevant features such as PARA or JUST are honored if they are applicable to the active mode.

All of these modes process commands issued within the mode and, unless specified otherwise, each mode processes any active trigger characters for underlining, boldfacing, and hard blanking.

## 4.1 ASIS MODE

\ASIS

Each data line encountered by the ASIS mode is copied as-is to output with no adjustment made to the line structure. The lines start at the margin and will be truncated if longer than 62 characters (default length). See the LENGTH command. A blank line produces a blank line on output.

PARA and JUST have no effect on as-is text. The underlining, boldfacing or hard-blank trigger characters are not recognized as trigger characters in ASIS mode. However the \BOLD command may be used to boldface one or more whole lines of as-is text.

Example:

```
SOME                  +------+
ASIS                  | ASIS |
TEXT                  +------+
```

## 4.0 DESCRIPTION OF MODES

## 4.1 ASIS MODE

Input:

```
\asis
SOME                  +-------+
ASIS                  | ASIS |
TEXT                  +-------+
\block
```

## 4.2 BLOCK MODE

\BLOCK,Jnn,Inn,Hnn

Block mode forms paragraphs from "text BLOCKs" passed to it. Note that a text block of one character is permissible.

A "text block" is a set of one or more lines of data which is started by a line of text with a blank in column one, and ended by either the next line with a blank in column one or by a command which places text in the output at that point.

Each text block is started on a new line and is normally right justified between the left and right margins. A blank line produces a blank line. A series of lines which start with a blank in column one would appear as a list at the left margin.

If the data for a text block starts in column two, the generated paragraph will begin at the left margin. But if x additional blanks are left after the text block start blank, the whole text block will be indented by x positions from the left margin.

If the PARAgraph feature is on, the first line in a text block will be indented by three spaces from the margin. See the PARA command.

J specifies the number of lines (nn) to JUMP at the beginning of each text block. This is in addition to any lines skipped by explicit TXTCODE commands. The default is J0.

I INDENTs the first line of the text block nn positions from the margin. The default is I0.

H HANGs (indents) the second and succeeding lines of the block text nn positions from the margin. The default is H0.

Example:

```
   This format, which is a BLOCK mode paragraph, is  generated
by the input shown below under Input.

   Indentation  like  this  is  accomplished  with  additional
   blanks on the first line of the text block.  The  PARAgraph
   feature is on throughout.
```

Input:

```
 This format, which is a BLOCK mode paragraph, is
generated by the input shown below under Input.

    Indentation like this is accomplished with
additional blanks on the first line of the text
block.  The PARAgraph feature is on throughout.
```

## 4.3 COMMENT MODE

\COMMENT

Comment mode ignores all non-command lines. Note that commands are still processed, including text on the command line. This is in contrast to \STOP which ignores everything until a subsequent \GO is received. Comment is not intended to be used to "turn-off" sections of the input file - use STOP/GO.

## 4.4 FLOWTAB MODE

\FLOWTABxtt,tt,...tt,ff          eg. \FLOWTAB;1,20

This mode allows text which is to "flow" between a left start position and the right margin to start on the same line as text which has been tabbed into position.

The first character x after the command, if non-blank, resets the tab character to the given character. The tab positions follow. (Defaults are ;1,20.) The last position given specifies the start position for the flowing text. Note that text may be tabbed left and right of the text which is to flow between the given start position and the right margin. When tabbing to the right, it is usually necessary to first adjust the right margin using the \LENGTH-nn command. The original right margin may be restored by using the \LENGTH command.

Data is entered in the order of the tabbing sequence used in the command. Tabbing is relative to the left margin, so that a tab set to 1 will cause the associated text to start immediately to the right of the left margin.

Each line starting with a blank forces a new line. An all blank line leaves, as usual, a blank line in the output.

A text block which starts with a blank but which does not have the tab character specified next, is assumed to be flowing text and is placed at the last position specified in the command. Any x additional blanks after the first one, cause the text block as a whole to be indented x positions from the beginning of the flowing text.

NOTE: Once the flowing text is reached, the tab character may be used freely in the flowing text.

See the SETHx command if a "hard blank" (ie. non-compressible) character would be useful in positioning the tabbed information.

The following shows how to set up some FLOWTAB type text:

Example:

```
COMMAND        PARAMETERS     ACTION

ADD            YES            Add the given files to the  tape
                              library   and   put  the  comment
                              information  on  the  last   file
                              given.

DAYFILE        NO             Leave  dayfile for batch job on a
                              permanent file when done.

                              This starts at the  flowing  text
                              position;  the  tab character may
                              be used normally here.

                                  This  is  indented  by   four
                                  positions    throughout    the
                                  paragraph.
```

Input: (\SETU specified at start of document)

```
\keep17
\FLOWTAB;1,15,30
 ;~COMMAND~ ;~PARAMETERS~ ;~ACTION~

 ;ADD ;YES ;Add the given files to the tape library
and put the comment information on the last file given.

 ;DAYFILE;NO ;Leave dayfile for batch job on a
permanent file when done.

 This starts at the flowing text position; the tab character
may be used normally here.

     This is indented by four positions throughout
the paragraph.
\block
```

## 4.5 ITEM MODE

\ITEM,Jnn,Inn,Snn,Gnn,Hnn,Zxx,C=e,O     eg. \ITEM

Item mode allows the user to generate a properly indented and aligned list of items where each item is normally preceded by some character string such as: -, *, ., 1., (a), A., or 1) etc. The above examples, plus any prefix character string, are handled automatically unless ITEM mode is called with its own settings. ITEM mode will set up default values for all parameters that are not supplied on the command.

Notice that the default value of one parameter can be automatically altered by specifying another parameter. E.g. giving G4 alters the default value of H.

(Preceeding the input data for the following six paragraphs with the command "\ITEM" would format that text as it appears below).

J specifies the number of lines (nn) to JUMP at the beginning of each new item. This is in addition to any lines skipped by explicit TXTCODE or TXTFORM commands. The default is J0.

I indicates the prefix characters are to be INDENTed from the margin by an amount nn. The default value for I depends upon the resulting value for S; see the table below.

S specifies the SIZE of the prefix string as nn. (nn may be greater than 3). The first actual item encountered will set the SIZE for the prefix string area, unless the S parameter is given on the ITEM command. Subsequent prefix strings will left adjust in the prefix string field if shorter than the first prefix string, while longer prefix strings will be right adjusted in the prefix string field. By specifically giving a large or small S value, the prefix string information can be forced to be left or right justified within its field.

G gives the GAP after the prefix string: nn columns. The default value for G is determined from the table below using the resulting value for S.

H HANGS the text which overflows line one under the first non-blank character following the prefix string or as requested. The default value for H is equal to I+S+G.

Z,C,O These are the parameters controlling automatic item label generation. A full description of these parameters is given in the section on Automatic Label Generation of Itemized Lists which follows shortly.

The following diagram shows the meanings of the various terms used:

```
+--- Hang Amount (e.g. 6 spaces)
|  +--- Indentation (e.g. 3 spaces)
|  |  +--- Prefix String (e.g. 2 characters)
|  |  | +--- Gap (e.g. 1 space)
|  |  | |  +--- Text Body
|  V  V V  V
|  ___1._XXXXX  XXX...XXX
+->______XXX XX.
```

Blank lines produce blank lines on output. Each line which starts with a blank forces a new line to be started.

The prefix string ends at the second blank character in the line. Hence, if positions one and two are blank a null prefix string is assumed and text starting in column three would be aligned with the text body of the previous item.

Any x additional blanks in columns 3,4,... cause the following text including overflow portions to be indented from the text body of the previous item by x positions throughout.

Finally, text which starts in position one of the line and which follows a blank line, will be treated as a non-item and aligned at the left margin.

The following table shows how ITEM will set up its I, S, G and H values for various sizes of prefix strings when the first item is encountered:

| SIZE | Inn | Snn | Gnn | Hnn (I+S+G) |
|---|---|---|---|---|
| 1 | 3 | 1 | 2 | 6 |
| 2 | 3 | 2 | 1 | 6 |
| 3 | 2 | 3 | 1 | 6 |
| 4 | 1 | 4 | 1 | 6 |
| 5+ | 0 | 5+ | 2 | 7+ |

Item Parameter Defaults Table

Example:

The input shown below would produce the format shown above, where the J, I, S, G, H, Z, C, and O parameters are explained:

Input:

\item
J specifies the number of lines (nn) to JUMP at the beginning of each new item. This is in addition to any lines skipped by explicit TXTCODE or TXTFORM commands. The default is J0.

I indicates the prefix characters are to be INDENTed from the margin by an amount nn. The default value for I depends upon the resulting value for S; see the table below.

S specifies the SIZE of the prefix string as nn. (nn may be greater than 3). The first actual item encountered will set the SIZE for the prefix string area, unless the S parameter is given on the ITEM command. Subsequent prefix strings will left adjust in the prefix string field if shorter than the first prefix string, while longer prefix strings will be right adjusted in the prefix string field. By specifically giving a large or small S value, the prefix string information can be forced to be left or right justified within its field.

G gives the GAP after the prefix string; nn columns. The default value for G is determined from the table below using the resulting value for S.

H HANGS the text which overflows line one under the first non-blank character following the prefix string or as requested. The default value for H is equal to I+S+G.

Z,C,O These are the parameters controlling automatic item label generation. A full description of these parameters is given in the section on Automatic Label Generation of Itemized Lists which follows shortly.
\block

## Automatic Label Generation on Itemized Lists

The following parameters exist on the item command to implement the automatic labelling of items:

```
,Z[x..x][a..a][=[b..b][c][d..d]] ,C=e ,O
```

Each of the parameters are optional and have the following meaning:

Z = L or l - auto label with letters: A, B, C,.. or a, b, c,..

```
    N or n - auto label with numbers: 1, 2, 3,..
    R or r - auto label with roman:   I, II, III, IV,.. or
                                      i, ii, iii, iv,..
```

In the absence of the x..x portion of the command, the case of the "Z" determines the case of the generated labels. Otherwise, x..x is used to specify both a non-default starting value and the case.

x..x a starting value for labelling, if it is present and is compatible with "Z"'s type. Otherwise it is treated as a..a text.

a..a suffix text to follow each generated label when no "=" option is used in the parameter.

"=" used for more complex types of auto generated label text where b..b text is to be specified.

b..b text which is to precede the auto generated string. It might be a "(" for an (a), (b) type sequence.

c a point of substitution single character for the auto generated sequences, which is "." by default. It may be redefined using the ",C=e" parameter, where "C" stands for "character" and "e" is the new editing substitution character to assume both in the command and in the following text data prefix strings.

d..d suffix text to follow each auto generated string. It might be a ")." such as for an (A)., (B)., type sequence.

C=e e is a single character which is to define the point of substitution position for the auto generated strings. This is for both the command itself and the text data which follows the command.

O Override parameter. When this parameter is specified on the command, the point of substitution character is not sought for in the prefix string region of the text data. Any non-null character string for a prefix region is completely removed and replaced with the auto generated sequence. It may or may not contain the actual substitution character.

## Functioning

The basic process that occurs is as follows:

The label type, starting value, and case is determined from the "Z" and any x..x. This material is the auto generated string and will be incremented for each substitution that occurs. If there is any a..a material, it is used to set up the area that supplies the suffix for each auto generated string. If there is an "=" in the command parameter, everything following the "=" (until the end of the parameter - "," or " " reached) until the point of substitution is found, becomes a prefix to the auto generated string.

When the substitution character is found, any text following this character until the end of the parameter is used to set up the suffix area material for the auto generated string.

At this point the command has been processed and the prefix, auto generated start string, and the suffix have been set up. These three regions as a whole, packed together, become the "item substitution text". This text, of course, changes each time it is used since it is being incremented for each use.

## Text Data Substitution

Now, each time a non-null prefix string is encountered in the text data, the prefix string is scanned for the point of substitution character (usually "."), and the item substitution text is used to replace the point of substitution character. At this point, the text data prefix string has been constructed, and it is now treated according to the settings of the parameters I, S, G and H, whether set specifically or by the item command.

So, it is possible to specify standard preceding and following material for the auto generated string, such as parentheses, but still specify special text for a particular item that is unique to that item - an "*" for example, for a footnote. I.e.

```
 (a).  xxxx
*(b).  yyy
 (c).  zzzz
```

```
   TO DO:    1.        A.       (a)          i.         A.
             2.        B.       (b)         ii.        *B.
             3.        C.       (c)        iii.         C.
   ISSUE:
          item,n.   item,L.  item,l=(.)   item,r.   item,L.
          ^.^----   ^.^----  ^.^------    ^.^-----  ^.^-----
          -------   -------  --------     ------    ------
          ^.^----   ^.^----  ^.^------    ^.^-----  ^*.^----
          -----     -----    ------       ------    -------
```

The "^" is used to represent a blank character.

## 4.6 TABLE MODE

\TABLExtt,tt,...tt \TABLE;10,20,30...

As the name implies, TABLE mode is for use in constructing tables or columns of tabbed information.

Unless redefined, the tab character x is ";" and the tabs are set to 10, 20, 30, 40, etc. If the tab character is not to be redefined, leave the position immediately after the mode name blank. The tabs are relative to the margin so that a tab set to 1 places text in the position right after the left margin.

Whenever a data line begins with a blank character, a new line is forced and the tabbing pointer is reset to tab number 1. (Data for a single line in a table may be input on several lines.) There are a maximum of 15 tabbing positions available.

An all blank line produces a blank line on output.

PARA and JUST are not applicable to this mode.

See the SETHx command if a "hard blank" (i.e. non-compressible) blank character would help in formatting the table.

Example: (\SETU specified at start of document.)

| SIZE | Inn | Snn | Gnn | Hnn (I+S+G) |
|---|---|---|---|---|
| 1 | 3 | 1 | 2 | 6 |
| 2 | 3 | 2 | 1 | 6 |
| 3 | 2 | 3 | 1 | 6 |
| 4 | 1 | 4 | 1 | 6 |
| 5+ | 0 | 5+ | 2 | 7+ |

Input:

```
\keep7
\table;7,17,26,35,44
 ;~SIZE~   ;~Inn~   ;~Snn~   ;~Gnn~   ;~Hnn~ (I+S+G)

\table;8,18,27,36,45
 ;1 ;3 ;1 ;2 ;6
 ;2 ;3 ;2 ;1 ;6
 ;3 ;2 ;3 ;1 ;6
 ;4 ;1 ;4 ;1 ;6
 ;5+;0 ;5+;2 ;7+
\block
```

## 5.0 MODE NESTING COMMANDS

This section describes the mode nesting facility of TXTCODE.

For example:

```
\+ITEM
 1. Level one in the nest.

\+ITEM,IO
 (a) Level two in the nest.

\-
 2. Level one in the nest again.
\-
```

The above example looks like this when formatted:

```
   1. Level one in the nest.

      (a) Level two in the nest.

   2. Level one in the nest again.
```

Mode nesting is a facility which allows an active mode to be temporarily interrupted so that another style of text may be processed, before returning to the interrupted mode and carrying on as though it had not been disturbed. The second mode can also be interrupted itself and so on. In addition, there is normally an automatic margin shift which causes the interrupting mode to be indented under the interrupted mode so that it aligns with the visual margin of the interrupted mode.

This provides a simple and powerful way of formatting complex text structures.

### \+mode *mode \-mode

Nesting is entered by the first occurrence of "\+mode" and, until the next "\mode" command, which signals the exit, mode commands of the form "\+...", "*..." and "\-..." control movement within the nest. Exit is also accomplished when a

\EXITNEST command is given or when a "\-mode" command causes the nest to decrease to the zero level. ("\-" may also do this.)

The nest has a maximum number of active 'parentheses' of 10. Nesting beyond this maximum level is processed using the ASIS mode. Dropping back below the upper limit causes the specified modes at each level to resume.

The mode to use for processing data at each level in the nest is set by the mode given on the mode command that most recently set a mode for that level. Each of "\+mode", "*mode" and "\-mode" set the mode at the level that they are causing a shift to in the nest. It is possible to move to a new level in the nest without specifying a mode, by giving "\+", "*" or "\-" with no mode. In this case, the mode most recently given for that level is used. Any level which never had an initial mode specified, uses the ASIS mode as a default.

Multiple levels may be shifted, if desired, by giving "\++..." or "\--..." with or without a mode name.

Moving to a lower level in the nest does not wipe out the modes specified for higher nest levels. Even exiting from the nest as a whole will not cause the nest mode data to be destroyed. It will remain until it is overwritten by a new mode at the appropriate level or until a "\CLEARNEST" command is given.

This allows the user to take advantage of repetitive data structures to simplify the mode calling process and also help ensure consistency of formats. Note that for ITEM mode especially, even the last used automatically generated prefix string is remembered and resumed when the correct level is again reached.

### \NESTIND

Another facility is provided when automatic indentation is on during nesting. This facility is on by default and is controlled using \NESTIND (nest indent) or \NONESTIND at any location in the document. With nest indent ON, whenever a "\+..." command is found, TXTCODE will cause the new mode to be right shifted to the "visual" margin of the previous mode. Actually, only the FLOWTAB and ITEM modes have visual margins which are usually shifted from the relative margin position. (BLOCK mode may also have a shift if the Hnn parameter is used.) It is the relative margin that is being moved to the position of the "visual" margin of the previous mode.

When coming back out of the nest, the nest indent indicator is ignored. Shifting back is done if shifting forward was done at this level.

## Level Shifting

The following is an example of some nested modes with an explanation of what is happening:

```
\block
\+item
\+FLOWTAB
\-
\+
\-
\+
\*asis
\-
\-
```

Initially, BLOCK mode is active. Then a nest is entered with the ITEM mode request. Then some FLOWTAB material is given under several particular items - all with the same FLOWTAB formats. After the last FLOWTAB section is some ASIS text which should have the same margin as the FLOWTAB mode had when it was called. Then ITEM mode is picked up and finished and finally the original mode (block) is resumed, which was active before the nest was entered.

"*mode" replaces the current mode at a particular level with a new mode. The new mode is not indented because it is at the same level as the one already there.

## Restored Modes

It should be noted that when a mode is restored, it is the mode that is restored and not all the associated conditions that may have been active during that mode.

All such features remain under the users direct control alone. For example, if a trigger character for underlining had been set but was later cleared, reentering the mode during nesting would not reestablish the underlining trigger character.

In addition, margin control is also still under the user's direct control. The changes in the relative margin are made during the shifting process in going from one mode to another in the nest (when nest indent is active) in order to fashion the indentation structure, but the user may give extra margin adjustments of his own. He should be careful to move the margin back when the structure is completed, and before he changes levels preferably. Otherwise, arbitrary margin changes will fracture the nest structure as a whole at that point.

Visual Margin

The visual margin is defined as the relative margin for ASIS, COMMENT and TABLE modes. For FLOWTAB mode, the visual margin is where the flowing text starts. For ITEM or BLOCK mode, the visual margin is where the body of the text falls when line overflow occurs. (Note that the default visual margin for BLOCK mode is the normal margin.)

Mode Switching

To go into the nest another level give:

```
\+mode
```

To change modes at the same level give:

```
\*mode
```

To drop out of the nest one level give:

```
\-mode
```

These three commands, containing the "+", "*" or "-" in front of the mode, control the movement within the nest, entrance into a nest (first "\+..."), and exit from the nest (a "\-" at level one). Any mode command without the plus, asterisk or dash is a regular non-nest mode command and will terminate a nest if one is active.

Explanation

Each time a deeper mode is entered, the visual margin value is saved for the previous mode - that is, the amount to shift the relative margin by is saved, if nest indentation is active. Next the relative margin value is adjusted if necessary. Then the new mode is set up and the actual mode data is processed.

## Nest Clearing

The entire mode nest may be cleared by using the \CLEARNEST (when outside a nest) or \EXITNEST,C commands. This destroys all memory of modes which have been defined in the nest structure.

## \EXITNEST,C

This command causes an immediate exit from any level of a nest. The most recent \mode command governs the active mode on exit. If the "C" parameter is specified, the whole nest is completely cleared of mode information.

## \CLEARNEST

The CLEARNEST command clears all nest levels beyond the current level. If nesting is inactive, it clears the whole nest - levels 1 to 10; if nesting is active, it clears levels n+1 through level 10. The current and lower levels are not affected and nesting is not disturbed.

## 6.0 DESCRIPTION OF COMMANDS

### 6.1 DESIGNATION TEXT COMMANDS

These commands apply to the title block (top of page title information), folio line (page footing information), and any supplemental lists generated such as a table of contents. (See \TABLCON command).

\TITLE

```
\TITLEn=xxxx...x        or        \TITLE=xxxx...x
```

The xxxx...x is the information which is to appear in the title position n (n=1,2...6) of subsequent pages. Up to six title lines may be in effect. Unused title lines will be squeezed out of the title block. The default title line is n=1, which appears between the page number and date lines. Other title lines follow the date title line.

\DATE

```
\DATE=xxxx...x
```

The current date (in YY/MM/DD format) will be replaced by xxxx...x at the upper right hand position of subsequent pages. It will appear on the current page too, if no text other than page header text has been given yet. E.g. \DATE=13 JUL 79

\FOLIO

```
\FOLIO=xxxx...x
```

At the bottom of the page on the folio line, xxxx...x will appear right justified on the current and following pages.

\TABLCON

If a table of contents is desired, this command should be issued once at the start of the input file. It should be given before any section numbers (\ n.n...n xxxxx...x) are given.

\AUTOSEC and \NOAUTOSEC

AUTOSEC turns on automatic section numbering. The level number is given by n on a section number command line. Any n.n....n forms have their level calculated from the n.n...n. That is, n.0 is level 1; n.n is level 2: n.n.n is level 3: etc. The first level 1 becomes 1.0; the second level 1 becomes 2.0 etc., while the first level 2 becomes 1.1 and so on.

Automatic section causes page numbers to change to the n-n format; it also causes a section header box to appear at the top of each page. See Section Numbers for additional detail.

## 6.2 BODY TEXT COMMANDS

These commands control the text which is in the middle of the page - between the title block information and the folio line.

### 6.2.1 SECTION NUMBERS

```
\ n.n...n xxxxx...x    or    \ n xxx...x    or    \ xxx...x
```

This command establishes a section number and title data in the text and in the table of contents. (See the \TABLCON command for generating a table of contents). Normally the first format is used with \NOAUTOSEC and the second form with \AUTOSEC. However, the first format can be used by \AUTOSEC and the second format will produce titles with no section numbers if used with \NOAUTOSEC.

Paging (n.0 levels), spacing, underlining, and capitalization are produced automatically and are governed by the level being processed. Standard TXTFORM format for section numbering is used. See the Heading Level Selection Table.

Note: the space after the back slant (\) and also after the section number. One space is sufficient.

In the second form, n is a level number from 1 to 7, or 11 to 17. (See \AUTOSEC)

In the third form, no section number appears and the text is underlined. It is equivalent to \ 14 xxx...x. (Note: the third format should not be used if the first word contains any periods "."; use format 2 with n=14. The same applies if the first part of the title text is numeric.)

NB: Individual section number lines which are too long may be omitted from a table of contents by TXTFORM.

HEADING LEVEL SELECTION TABLE

|  | NEW PAGE | ALL CAPS | LINES SPACED BEFORE | LINES SPACED AFTER | UNDER LINED |
|---|---|---|---|---|---|
| H1(H11) SECTION |  |  |  |  |  |
| \ 1 xxx...x |  |  |  |  |  |
| \ 11 xxx...x |  |  |  |  |  |
| \ n.0 xxx...x |  |  |  |  |  |
| LEVEL 1 | X | X |  | 5 | X |
| H2(H12) SECTION |  |  |  |  |  |
| \ 2 xxx...x |  |  |  |  |  |
| \ 12 xxx...x |  |  |  |  |  |
| \ n.n xxx...x |  |  |  |  |  |
| LEVEL 2 |  |  |  |  |  |
| \HEADA |  | X | 2 | 1 | X |
| \HEADB |  | X | 3 | 2 | X |
| \HEADC |  | X | 2 | 1 | X |
| H3(H13) SECTION |  |  |  |  |  |
| \ 3 xxx...x |  |  |  |  |  |
| \ 13 xxx...x |  |  |  |  |  |
| \ n.n.n xxx...x |  |  |  |  |  |
| LEVEL 3 |  |  |  |  |  |
| \HEADA |  | X | 2 | 1 |  |
| \HEADB |  | X | 3 | 2 |  |
| \HEADC |  | X | 2 | 1 |  |
| H4(H14) SECTION |  |  |  |  |  |
| \ 4 xxx...x |  |  |  |  |  |
| \ 14 xxx...x |  |  |  |  |  |
| \ n.n.n.n xxx...x |  |  |  |  |  |
| \ xxx...x |  |  |  |  |  |
| LEVEL 4 |  |  |  |  |  |
| \HEADA |  |  | 1 | 1 | X |
| \HEADB |  |  | 3 | 2 | X |
| \HEADC |  |  | 2 | 1 | X |

| | | | LINES | LINES | |
|---|---|---|---|---|---|
| | NEW | ALL | SPACED | SPACED | UNDER |
| | PAGE | CAPS | BEFORE | AFTER | LINED |
| H5(H15) SECTION | | | | | |
| \ 5 xxx...x | | | | | |
| \ 15 xxx...x | | | | | |
| \ n.n.n.n.n xxx...x | | | | | |
| LEVEL 5 | | | | | |
| \HEADA | | X | 1 | | |
| \HEADB | | X | 1 | | X |
| \HEADC | | X | 1 | | X |
| H6(H16) SECTION | | | | | |
| \ 6 xxx...x | | | | | |
| \ 16 xxx...x | | | | | |
| \ n.n.n.n.n.n xxx...x | | | | | |
| LEVEL 6 | | | | | |
| \HEADA | | | 1 | | X |
| \HEADB | | | | | X |
| \HEADC | | X | | | |
| H7(H17) SECTION | | | | | |
| \ 7 xxx...x | | | | | |
| \ 17 xxx...x | | | | | |
| \ n.n.n.n.n.n.n xxx...x | | | | | |
| LEVEL 7\HEADC | | | 1 | | X |

If AUTOSEC is off (default), the first form shown (\ n xxx..) is converted to the second form shown (\ 1n xxx..) and treated accordingly.

If AUTOSEC is on, (\AUTOSEC issued), page numbering changes from the simple sequential format to the n-n format. Also, a 4 line header box will be produced on each page which will contain the currently active major section (n.0 level) and the currently active heading level being processed, of whatever level. AUTOSEC doesn't produce numbering for (\ 1n xxx..) type headers, but it does replace any n.n...n forms, which effectively allows renumbering of old section numbers.

NOTE: that AUTOSEC effects do not occur until the first header command line (\ n xxx...x) has been encountered.

Also note that the form (\ xxx...x) is treated as the following form in all cases: (\ 14 xxx...x).

## 6.2.2 FEATURE SELECTION/DE-SELECTION

### \PARA

\PARA        \PARAnn        \NOPARA

With the PARAgraph feature selected, BLOCK mode paragraphs which begin at the left margin have their first line indented by the number of spaces specified for the \PARAnn command or by three spaces for the \PARA command.

### \UNDER and \NOUNDER

Specifying UNDERlining will cause the following lines of text to be underlined until a NOUNDERline command is given or until the mode is changed or a section number is encountered. This command is oriented toward underlining for which it is not convenient to use a trigger character for control. (See the SETUx command).

### \JUST and \NOJUST

Right JUSTification occurs as a default. Text which is being built up so that it overflows to the next line when the right margin is met is normally aligned with the right margin by adding extra spaces between words. This feature has no effect, even if selected, if it is not appropriate to the currently active mode. TABLE and ASIS mode ignore this feature. Note that JUSTification only allows backing up in the line for 30 characters. Therefore, if a blank, -, or / is not found, a 'LINE OVERFLOW' message will be issued since a place could not be found to break for JUSTification.

### \BOLD and \NOBOLD

The BOLDface command instructs the formatter to print each character which follows in such a way that the characters appear darker and thicker. BOLDfacing stops when the NOBOLDfacing command is given or when the mode is changed or when a section number is encountered. The command is oriented towards BOLDfacing text for which a trigger character in the text itself is not appropriate. (See the SETBx command).

### \SEQ and \NOSEQ

If it is desired to have each output line in the document followed by the number of the line, issuing the SEQ command will initiate this process. (Default is NOSEQuencing). SEQuencing numbers will appear right justified on the fifth character position to the right of the right margin or the length as established by the TXTFORM "FORMAT" command (default is 75), whichever is greater - absolute column 80 by default. SEQuence numbers start at 1 on each new page and appear only beside text in the body of the document. The feature may be turned on or off at any point in the document.

NOTE: The TXTFORM FORMAT command is used to give the margin values for heading and folio lines and to specify the total page depth. It may only be given once and must be before any text is input. Using the direct TXTFORM command capability, it may be specified as follows:

```
\\FORMATaa,bb,nn
```

where aa is the left margin (default is 7), bb is the absolute line length (default is 75), and nn is the total page depth (default is 60). These margin values do not apply to the main body of the text.

### \KEEPnn and \NOKEEP

Sometimes material on different lines must be kept on the same page to be readable. To make sure that the next nn lines of output appear on the same page specify KEEPnn. NOKEEP serves only to mark the end of the kept text and is not actually necessary.

### \SINGLE and \DOUBLE

It is often desirable to allow different standard line spacing on documents. For example, double spacing is quite useful for draft documents to which many changes are usually made. SINGLE and DOUBLE select single and double spacing respectively for the formatted output text.

### \HEADA, \HEADB, and \HEADC

These commands allow selection of the TXTFORM heading styles A, B, and C respectively. See the heading level selection table for a description of the effect of the different heading styles.

### \CODES

This is used to produce a TXTFORM command statistics and error summary list at the end of the formatted document.

\MARK

The MARK command is used to set a margin note in the line being formatted. An asterisk (*) is inserted at the current position in the line and a pointer (<) is placed in the right margin.

### 6.2.3 TEXT FORMAT CONTROL COMMANDS

\MARGIN

```
\MARGINnn        \MARGIN        \MARGIN+nn        \MARGIN-nn
```

This command determines where text should normally start at the left hand side of the page. MARGINnn redefines the BASE MARGIN (default is 10) to the new value nn. Normally adjustments are made to the margin by specifying the change relative to the old margin instead of specifying the absolute margin each time. The +nn and -nn form is used to accomplish this. Note that a \MARGIN+2 followed by \MARGIN+3 command is equivalent to a \MARGIN+5 command.

At any time, the BASE MARGIN, the one defined by the last MARGINnn command, or MARGIN10 by default, may be returned to by asking for MARGIN with no value. In this way the margin may be temporarily redefined and then returned to the old value without remembering that value.

The ABSOLUTE margin is the extreme lefthand side of the page; the BASE margin is at 10 unless redefined using \MARGINnn; and the RELATIVE margin is initially equal to the BASE margin. The RELATIVE margin is moved by \MARGIN+nn and \MARGIN-nn commands as well as by mode nesting commands and may be reset to the BASE margin by \MARGIN.

\LENGTH

```
\LENGTHnn        \LENGTH        \LENGTH+nn        \LENGTH-nn
```

This command specifies the LENGTH of the text line as measured from the BASE MARGIN (see above). The default value is 62. Hence, an ASIS line of up to 62 characters can be placed on each output line using the default value. (\MARGIN-nn and \MARGIN+nn will lengthen or shorten the effective line length.) Adjustments may be made relative to the old length in a similar manner to the MARGIN command above.

\SKIP

\SKIPnn

To write nn blank lines to output at any time, use the SKIPnn command. The nn defaults to 1. Note that if paging occurs in the process, the command does not leave the rest of the blank lines at the top of the next page. If an actual amount of blank paper is desired, say 2 inches, issue a KEEP12 (2 inches times 6 lines per inch = 12 lines) then a SKIP12.

\PAGE

\PAGE or \PAGEnnn

At any time, a PAGE eject may be forced using this command. Two consecutive page ejects do not, however, leave a blank page. (Entering the command BLANK between the two PAGE ejects will). If it is desired to cause the next PAGE number to be reset to a value nnn, include the value immediately after the command.

\BLANK

\BLANK or \BLANKnn

If this command is given, nnn blanks (default is 1) will be inserted in the text stream at the current position. See the SETHx command for generating "hard blanks" using a specially designated character instead of the BLANK command.

\CENTER

\CENTER=xxxx...x or \CENTER,B,U=xxx...x

The xxxx...x text will be CENTERed on a new line between the left and right margins. Consecutive embedded blanks are compressed to one blank. B and U parameters are for boldfacing and underlining xxx...x. Special trigger characters in the xxxx...x text will be recognized and processed, if set, if the first form of the command is used. See the SETU, SETB and SETH commands.

### 6.2.4 APPENDICES

#### \SETAPP

\SETAPP=a-xxx...x

This command is used to define appendix text. The character value of "a" defines the appendix identifier and "xxx...x" is the appendix title. This information appears in the table of contents (see the TABLCON command) and the user should set up his own appendix title page as desired as well as the body of the appendix text.

The appendix identifier "a" precedes both the section number and the page number wherever they appear. One method of setting up an appendix follows:

```
\SETSEC1                    - resets section numbers to 1
\SETAPP=A-APPENDIX A        - sets appendix character and title
\PAGE1                      - sets up title page as page 1
\BLANK
\SKIP20
\CENTER,B=* Appendix A *

\CENTER=Special Conditions
\PAGE                       - starts body of appendix text
\ 1 Introduction
  .
  .
  .
```

### 6.2.5 BOX GENERATION COMMANDS

The following commands are used to draw boxes but the vertical spacing in boxes is controlled by commands or text given while a box is active. Usually TABLE or FLOWTAB mode is used to supply the text for a box.

#### \BOX

\BOX or \BOX,[-]nn,A,nn,B,[-]nn,R,[-]nn

The BOX command is used to initiate the drawing of vertical lines which are all joined together by a horizontal line at the top. For example, the command \BOX,10,20,30 would be used to start the following box:

```
+---------+---------+
|         |         |
+---------+---------+
|                   |
+-------------------+
|                   |
+-------------------+
```

The two outside columns are at columns 10 and 30 with respect to the relative margin, while the center line (vertically) is at column 20. By default, the column positions are with respect to the relative margin. At any point in the command, the reference point may be changed to the BASE margin with the "B" parameter, to the ABSOLUTE margin using "A", or back to the RELATIVE margin with the "R" parameter. (See the MARGIN command for an explanation of margins.) The A, B, and R parameters may be used several times in the command if desired.

For the BASE and RELATIVE margins, it is possible to specify a column position to the left of the margin by preceding the column number with a minus sign "-". Therefore "-1" and "1" specify vertical lines which are adjacent to each other surrounding the margin ("0" is equivalent to "-1").

Up to 17 vertical lines may be in effect at once.

If \BOX is given with no parameters, the settings in effect for the previous box are used.

\MODBOX

```
\MODBOX,[-]nn,A,nn,B,[-]nn,R,[-]nn
```

This command is used to modify a box in progress. The parameters are the same as for the BOX command. For each column given, there are two possible actions. If that column already has a vertical line being drawn, the line is turned off. If it does not have a vertical line in effect, one is started. In addition, MODBOX joins the resulting outside verticals with a horizontal line. The above box had its center vertical terminated by the following command:

```
\MODBOX,20
```

\BOXLINE

This command draws a horizontal line between the inside and outside vertical lines of the box. The third horizontal in the box above was drawn this way.

\NOBOX

NOBOX is used to end a box currently in effect, drawing the final horizontal line terminating the box.

Example:

```
+------+------------+-----------------------------------+
|  A   | ABSOLUTE   | This  parameter causes the following |
|      |            | values  to  refer  to  the  ABSOLUTE |
|      |            | margin.                              |
+------+------------+-----------------------------------+
|  B   | BASE       | This  parameter causes the following |
|      |            | values to refer to the BASE  margin. |
+------+------------+-----------------------------------+
|  R   | RELATIVE   | Finally,  this  parameter causes the |
|      |            | following values  to  refer  to  the |
|      |            | RELATIVE margin.                     |
+------+------------+-----------------------------------+
```

Input:

```
\KEEP12
\BOX,-1,7,21,62
\FLOWTAB:3,10,24
\LENGTH-3
 ;A ;ABSOLUTE ;This parameter causes the following values to
refer to the ABSOLUTE margin.
\BOXLINE
 ;B ;BASE ;This parameter causes the following values to
refer to the BASE margin.
\BOXLINE
 ;R ;RELATIVE ;Finally, this parameter causes the following
values to refer to the RELATIVE margin.
\NOBOX
\LENGTH
```

## 6.2.6 ESCAPE CHARACTER DEFINITION

\SETEx

To reSET the master Escape character for commands to another character x issue this command. (Default is \)

\SETUx

To turn on a trigger character x to control the starting and stopping of Underlining give this command. The feature is turned off by giving a blank character for the x. (Default condition is no trigger character set). Also, note that the effect of a mode change or the occurrence of a section number command will turn off Underlining just as the second occurrence of the trigger character would. But the feature itself remains set. A suggested character to use for triggering underlining is _ or ~.

NOTE: If the underlining and boldfacing trigger characters are set to the same character at the same time, both functions will be controlled by the one trigger character.

\SETBx

This feature is exactly analogous to underlining (see above) except that it controls Boldfacing. A suggested character to use to trigger this feature on and off is @.

\SETHx

From time to time it is necessary to use special blanks in the text called "hard blanks", which will not be compacted into one (1) blank or expanded into more than one blank. The hard blank is also useful in forcing tabbed data into proper alignment if the left hand side is not straight. The BLANK command is not always convenient in such cases. This command, SETHx, allows the user to specify a character which is to be considered to be a hard blank until the feature is turned off. Each subsequent occurrence of the character forces a single blank to be shown in its place.

A suggested character to use is the circumflex (^) or the left apostrophe.

## 7.0 DIRECT TXTFORM COMMANDS

Any TXTFORM command may be given directly by prefixing that command by two master Escape characters rather than one. The preprocessor strips off the first master Escape character and passes the command directly to TXTFORM. No command validity checking is performed.

e.g. \\V10,20,30

* * *

Software Engineering Services TXTCODE 2.0
Reference Manual
60460280 01