SR20–10–00

ROSCOE T.M.

User Guide

APPLIED DATA RESEARCH

THE SOFTWARE BUILDERS®

Ref. Manual Price: $15.00

TABLE OF CONTENTS

TABLE OF EXAMPLES

I. ROSCOE FACILITIES

1.1 INTRODUCTION

ROSCOE is a remote terminal system which facilitates use of the 360/370 Operating System, by means of *remote job entry* (RJE) which permits *scheduling* of OS Job execution in batch, and *retrieval* at the terminal of job output.

Job streams (consisting of JCL, source programs, input data cards, etc.) are created at the terminal through ROSCOE's extensive data entry and editing capabilities, and maintained in the ROSCOE library. The data which ROSCOE handles are *card images*, which may actually be documents (text), source modules, JCL, or any arbitrary data which can be placed on card images.

Access to ROSCOE is through terminals connected to a central computer, and by means of a special language, the ROSCOE commands. These commands provide the user with a full working environment.

1.2 THE ROSCOE ENVIRONMENT

The ROSCOE environment consists of:

a. An area in which data are created, edited, and otherwise manipulated. This area is named the Active Work Space (always abbreviated as AWS). The AWS normally has a capacity of 1000 lines (=card images). Note that this capacity is established by the installation manager at any number of lines suitable for users. A job stream may be submitted to OS for execution directly from the AWS. There is also two-way traffic between the AWS and the ROSCOE libraries.

b. Permanent storage facilities; these are the ROSCOE line libraries in which the user may preserve data created in the AWS, and from which data may be transferred into the AWS. There are also facilities permitting transfer of data into the libraries from batch. Data are stored on the libraries under a name assigned by the user, and are, therefore, called "named data sets." The storage capacity of the ROSCOE libraries is practically unlimited.

c. Two-way communication between the user's terminal and the central system operator.

d. Remote job entry and retrieval of data from batch.

e. Programming aids (the Syntax Checkers).

f. ROSCOE self-running interactive procedures (called "ROSPROCS").

g. ROSCOE MONITOR routines, which are optional extensions of the system written by the site programming staff or provided by ADR.

1.3 TERMINAL FUNCTIONS

There are four terminal-dependent functions – RETURN, ATTN, BACKSPACE, and TAB – which are activated differently for the different terminal devices which may be attached to ROSCOE. Device-dependent information for the various terminal types supported by ROSCOE (2741, TTY, 2260, 3270) is contained in Appendices A, B, C, D, and F, where the specific manner of establishing contact with ROSCOE and activating the terminal functions is described. Except in the appendices, all discussion in these pages will be on the logical and device-independent level. The four terminal functions are as follows:

a. RETURN is used to send a line of text or data from the terminal to ROSCOE or to transmit a command to ROSCOE.

b. ATTN is used to interrupt or cancel the current processing in ROSCOE.

c. BACKSPACE is used to delete characters from a line before sending the line to ROSCOE.

d. TAB is used to format text on a terminal line, as for a normal typewriter.

1.4 ROSCOE COMMANDS

A ROSCOE command has the normal form:

```
COMMAND OP1,OP2,OP3....,OP12
```

The presence and number of operands are determined by the nature of the specific command. In the following explanations, the symbols below are used to describe the operands:

dsn	data set name (name of data set on a ROSCOE library)
m	beginning line number
n	ending line number
i	increment value
/	any special character (neither a letter nor a digit)
string	any sequence of characters which can be typed on the terminal keyboard.

NOTE: In commands using m and n to represent a range of sequence numbers in the AWS or a library data set, neither m nor n need exist in the referenced data. Thus, if the AWS consists of lines 10,20,30, the command

```
LIST 1,35
```

is effectively the same as

```
LIST 10,30
```

and

```
LIST 9,11
```

is the same as

```
LIST 10
```

Normally, if the value 0 (zero) is used for m, n, or i, it is an error. However, 0 is used to address the ROSCOE counter (see Section VIII), which may be set to any desired value, so that 0 will be replaced in the command by the current value of the counter.

When a command is entered and its format is found incorrect, ROSCOE will display:

```
COMMAND INVALID
```

If the operands of a valid command are in error, the incorrect operand will also be identified. If the entry is not a recognized ROSCOE command, ROSCOE will display:

```
ENTRY INVALID
```

Certain ROSCOE commands (mainly those which reference the AWS) have a long form (e.g., LIST) and a short form, which is always the first letter of the command (e.g., L). When the long form is used, a space between the command and the first operand is optional (it is disregarded), but a comma may not appear between them. A short form must have either a space or a comma between them. Examples of valid commands are:

```
LIST10,20

LIST 10,20

L 10,20

L,10,20
```

Examples of invalid commands are:

```
LIST,10,20

L10,20
```

Whenever an abnormal condition occurs, such as an I/O error over the phone line connecting the terminal to the computer, ROSCOE will display on the terminal a self-explanatory message and recover back to the point where the error occurred. ROSCOE informs a terminal user of the completion of command execution either by a message, or more normally, by a *prompt* (an underscore, "_", written on the left margin of the paper), to indicate that it is ready to receive the next command from the terminal.

1.5 DATA SET NAMING CONVENTION

The user must be aware of the naming convention used for library data sets.

The full data set name actually consists of the 2-character user-prefix, assigned to him by his installation manager, followed by the 8-character name assigned by the user. Thus, if my prefix is AX and I enter the command SAVE NEWSET, the full name of the data set is AXNEWSET. But a terminal user never needs to prefix the name with his own prefix. It is possible to save a data set with someone else's prefix. Thus, if your prefix is MN, I can place a data set in your library sets by the command:

```
SAVE  MN.NEWSET         or      S,MN.NEWSET

SAVE  .MNNEWSET         or      S,.MNNEWSET
```

where the dot before or after the prefix indicates to ROSCOE that explicit prefixing is in use. This facility should be used carefully, for only the user who has saved the data can delete it. This is so because when a data set is saved, the user's protect code is placed in the data set. (The protect code is unique to every user and is assigned by the installation manager.) No one can delete a data set unless his protect code is the same as that on the data set. This affords write protection. By using the explicit prefixing, you can also LIST, FETCH, DO, SUBMIT, etc., data sets created by another user.

NOTE: Prefixing is indicated by a point (".") placed just before or just after the prefix. For example: Prefix is RO; dsn is SNEWS; to access the data, the command is:

```
DO  .ROSNEWS            or      DO  RO.SNEWS

L,.ROSNEWS              or      L,RO.SNEWS
```

II. ESTABLISHING CONTACT

The following list outlines the ROSCOE log-on and log-off procedures.

1. When the terminal has established contact with ROSCOE, a log-on request will be displayed at the terminal:

   ```
   Enter ROSCOE Key mm/dd/yy
   ```

   The mm/dd/yy is the current date. The terminal user then types in his key and hits RETURN. The "key" is one that has been assigned to the user by his installation manager and is unique to him. ROSCOE then attempts to find this key in the user-profile data set. If the key cannot be found, the log-on request is repeated four more times. If a valid key has not been entered by then, the terminal will be disconnected by ROSCOE after display of a termination message:

   ```
   (Terminal has been Signed Off)
   ```

   It is also possible when replying to the log-on request to type OFF, which will disconnect the terminal at once. If ROSCOE finds that a user is already active in the system using the key, a message will be displayed:

   ```
   ACCOUNT ALREADY SIGNED ON. TRY LATER
   ```

   If ROSCOE finds that it cannot allocate space to handle an additional user, a message will be displayed:

   ```
   ROSCOE UNABLE TO HANDLE YOUR TERMINAL. TRY LATER.
   ```

   In either of the last two cases, the termination message will be displayed and the terminal disconnected.

2. If none of the above cases occurs, ROSCOE will then display the message:

   ```
   LINE #nn TIME IN hh.mm.ss
   ```

   where nn is the identifying line number assigned for the session to the user, and hh.mm.ss is the current time in hours, minutes, and seconds. Simultaneously, a session report record will record that the user has signed on, and (at installation option) the operator at the systems console will be informed of the fact. The user should remember his line number in case the system terminates abnormally during his session, for the contents of his AWS will be saved in the ROSCOE library under the name .ROSAWSnn. This data set can then be recalled into the AWS and the interrupted session resumed when the system has been brought up again.

3. At log-on time, additional activity may be automatically activated at the option of the installation manager or the terminal user.

   - A system message, originating from the manager or system programmer, may be displayed on the terminal before the status message of item 2 above. The contents of this message may be a notice to all terminal users, and will always be a self-explanatory text.

   - If the user has established a password to protect his data sets from unauthorized use, ROSCOE will display the password request:

```
ENTER PASSWORD ACDEGGHKLKOWM
```

where the user will reply over the masked area. If the correct password is not entered, ROSCOE will display the request two more times. If then the correct password has not been entered, the terminal will be disconnected.

- If the user has established a sign-on procedure, it will be executed immediately after the status message of 2 above. (A sign-on procedure is a ROSPROC which is automatically performed during sign-on, see Section VIII.)

4. To establish a password, the terminal user at any time during his session enters the command:

```
PASSWORD aaaaaaaaaa
```

where aaaaaaaaaa is a new password required for logging on under his key. The password is from 1 to 10 characters in length. To cancel an existing password without substituting a new one, use the command:

```
PASSWORD
```

with no operands. If you should forget your password, the installation manager has a universal password (known only to him and his delegates) which he can use to open your account for a session. NOTE: At the discretion of system management, certain keys are not permitted to modify or delete the password. For such accounts, any such invalid attempt will elicit the warning message:

```
ACCOUNT PROTECTION CHECK – NO ACTION TAKEN.
```

5. To establish a sign-on procedure, use the command:

```
SIGNON dsn
```

where dsn is the name of the procedure you wish performed whenever you sign on for a session. The dsn may be in your own library or in the library of another ROSCOE user, in which case the fully qualified name in the form pp.dsn must be entered (pp. is the prefix of the other user and the dot indicates that the prefix is not your own). To cancel an existing sign-on procedure without substituting a new one, use the command:

```
SIGNON
```

without operands. NOTE: At the discretion of system management, certain keys are not permitted to modify or delete the sign-on procedure. For such accounts, any such invalid attempt will elicit the same warning message as for the password:

```
ACCOUNT PROTECTION CHECK – NO ACTION TAKEN.
```

NOTE: For restrictions on the name of a sign-on procedure, refer to the SAVE command (see Section 4.1).

6. To terminate a ROSCOE session, use the command:

```
OFF
```

Do not simply leave the terminal or turn it off, for it will still be connected to ROSCOE and your account will still be active on it. When OFF has been entered, ROSCOE will display the two final log-off messages:

```
LINE #nn TIME IN hh.mm.ss, ELAPSED hh.mm.ss

(Terminal has been Signed Off)
```

At the same time, a log-off record will be written to the session report, and the central console operator will be notified.

NOTES:

- Users of local and remote 3270's are especially cautioned against turning their units off after signing off, or as a means to sign off.
- The OFF command cannot be issued from a ROSPROC unless for a restricted user (i.e., a user who is not permitted to exit from procedure state).
- At the option of site management, a confirmation of the OFF request will be issued if the AWS is not empty at sign-off time. The following message will be written to the terminal:

  ```
  AWS NOT EMPTY – ENTER "OFF" TO SIGN OFF.
  ```

  If OFF is entered from the terminal, sign-off processing will continue. Otherwise, the reply from the terminal will be taken as a new command (e.g., DELETE) which will be executed. In this case, the OFF command must be reentered.

7. For dial-up lines, where the user wishes to terminate his session under one account and initiate one under another, the command:

```
OFFON
```

may be used; normal log-off activities will take place as outlined above, and then the log-on request will be displayed immediately. This command, therefore, permits telephone dialing to be bypassed when a phone connection has already been made.

8. If a line failure occurs such that the terminal is disabled during a session, any data active in the AWS will be saved by ROSCOE under the name xx.SAVAWS, where xx is the user's prefix.

III. AWS COMMANDS

3.1 ENTERING DATA FROM THE TERMINAL

Any line of text preceded by a sequence number and a space, and terminated by a RETURN will be placed into the AWS. The sequence number is from 1 to 4 characters in length, and between the values 1 and 9999 inclusive. A sequence number 0 is valid only when associated with the ROSCOE counter (see Section VIII). The text consists of a maximum of 80 characters (blanks included). If an error is made in typing, the BACKSPACE function may be used to rub out incorrect characters. An entire line may be cancelled by using ATTN instead of RETURN. The lines are maintained in sequential order at all times. A line typed in with the same sequence number as an existing line will replace the existing line. A line with a sequence number less than the highest existing number and not replacing an existing line will be inserted into its correct place.

3.2 THE NUMBER COMMAND

The user can request ROSCOE to display line numbers to serve as prompters by entering *Autoline*, or *number*, mode. The command is:

```
NUMBER m,i

N,m,i
```

where m is the value of the first sequence number and i the increment. If a starting value of 10 and increment of 10 are desired, use:

```
NUMBER

N
```

with no operands.

To reset only the increment, use:

```
NUMBER i

N,i
```

where i is the new increment value.

The prompter numbers will be displayed line-by-line on the left of the carriage. (On the 2260, they will be displayed on the right, a screen-full at a time: 12 lines.) The user enters his text after the number (on the 2260 preceding it), following the text by a RETURN. The next number will then be displayed. To exit from Autoline mode, a null line (no data) followed by RETURN, is used. ROSCOE then displays the prompter and is ready for the next command. If the next command is a line preceded by a sequence number (e.g., an insert), ROSCOE will accept it and then resume Autoline mode immediately, unless the terminal is a 2260 or 3270. Otherwise, Autoline is suspended. To resume from the number where suspension occurred, use:

```
NUMBER

N
```

with no operands.

NOTE: For the format of 3270 Autoline, see Appendix F.

3.3 DELETING LINES FROM THE AWS

To delete the entire contents of the AWS, use the command:

```
DELETE
```

with no operands. The AWS is cleared of data, and the Autoline increment and starting value are reset to 10,10.

To delete a single line, use:

```
DELETE m
```

where m is the sequence number of the line to be deleted. Autoline values are not affected.

To delete several lines, use:

```
DELETE m,n
```

where m is the first and n the last of the range of lines to be deleted.

NOTE: DELETE without operands is executed immediately. DELETE with operands is deferred until the next LIST, SAVE, EDIT, or SUBMIT command. Therefore, the reply to the AWS inquiry may be incorrect at times.

This command does not have a short form.

3.4 THE TAB COMMAND

To format a line of text by using the tab setting of the terminal, first set the actual typewriter tabs to the desired columns, then enter the command:

```
TAB  t1,t2,t3,t4

T,t1,t2,t3,t4
```

where t1, etc., are the columns of the desired tab setting. To reset the tab positions, TAB t1, etc., can be entered again with new settings. Example: TAB 10,16,34,71.

To cancel all tab settings, use the command:

```
TAB

T
```

with no operands.

WARNING: When all tabs have been so cancelled, any use of the tab on the typewriter will be taken as a character of data, and cause the carriage to skip when being listed.

A 2260, which has no hardware tab feature, performs a tab by using a special character as the tab character. This tab marker is preset to the "not" sign (upper case I), but may be reset to any special character convenient to the terminal user by the command:

```
TAB  /

T,/
```

where / represents the special character to be used as a tab. This last form of command is recognized only for 2260's, although it will not be considered an error when entered on other terminal types; it will be disregarded. 2260 users are cautioned not to use "&" as a tab character without resetting the ROSPROC packed command delimiter (see Section 8.15).

If the special tab character defined by this command is a colon, ROSCOE recognizes a request for hardware tabbing (a special option on the 2260). In this case, ROSCOE will permit use of the hardware tab feature when in the Autoline mode by placing the colon one position to the left of the desired tab positions in the line. In addition, a colon will be placed in column 75 (just before the Autoline sequence prompter) to stop run-away tabbing. (The hardware tab on the 2260 will cause the cursor to wraparound from one line to the next until the next colon is found.) The colons can be overwritten; ROSCOE will replace those colons remaining in the tab positions with blanks. The colon can be used when entering data explicitly (i.e., not in Autoline mode) as a software tab.

NOTE: For 3270 tab handling, see Appendix F.

3.5 THE LIST COMMAND

To list all of the lines in the AWS, use the command:

```
LIST

L
```

without operands. To interrupt and cancel the listing, the ATTN function is used.

To list a single line in the AWS, use:

```
LIST  m

L  m
```

where m is the line number desired.

To list a range of lines in the AWS, use the form:

```
LIST  m,n

L  m,n
```

where m and n are respectively the first and last line numbers of the set to be listed.

NOTE: When the listing is completed, ROSCOE will display the prompter (underscore). If a LIST command is entered but there is no data in the AWS, ROSCOE will display the message:

```
AWS EMPTY – NO ACTION TAKEN
```

followed by the prompter.

If no lines are found within the m,n range, ROSCOE will inform the user with the diagnostic:

```
NO LINES FOUND IN RANGE.
```

(This diagnostic will be typed for any AWS operation to which it can apply.)

3.6 THE RENUMBER COMMAND

To resequence all lines in the AWS, use the command:

```
RENUMBER m,i

R m,i
```

where m is the starting sequence number and i is the increment.

To renumber the entire AWS starting with 10 and incrementing by 10, use the form:

```
RENUMBER

R
```

with no operands.

NOTE: Before performing the actual resequencing, ROSCOE will verify whether the requested renumbering will result in sequence number overflow. If it will, no action is taken, and ROSCOE displays the overflow message:

```
AWS LINE LIMIT EXCEEDED – SAVE, DELETE, or RENUMBER.
```

When in suspended Autoline mode, the current increment can be reset by the command:

```
RENUMBER i

R i
```

where i is the new increment value. Autoline will not be reentered immediately, as for N i, but the increment will be reset for the next use. This command is used mainly in preparation for the APPEND command.

3.7 THE APPEND COMMAND

To add lines at the end of the current AWS, use the command:

```
APPEND

A
```

When the command is used with no operands, the complete AWS is appended to itself, using the current increment value as increment, and the sequence number +i of the last line in the AWS as the starting line number.

To append a single line from the AWS to the end, use the form:

```
APPEND m

A m
```

To append a contiguous sequence of lines, use the form:

```
APPEND m,n

A m,n
```

The use of APPEND with library data sets is described in Section IV.

3.8 THE INSERT COMMAND

To move lines anywhere in the AWS from anywhere else in the AWS, use the command:

```
INSERT m,n,o

I m,n,o
```

where lines m through n will be inserted after line o in the AWS, using a temporary increment of 1, so that the new lines will becomes line o+1, o+2, etc. The lines m through n are actually removed from their original location and moved into the new position, so that they will no longer exist under their original line numbers. If the insertion of lines causes duplicate line numbers with the lines already in the AWS, as many lines of text following the last line inserted will be renumbered (by 1) as necessary to retain the data in its original sequencing.

To insert a single line at any location, use the form:

```
INSERT m,o

I m,o
```

The use of INSERT without any operands is not permitted. The use of INSERT referencing a library data set is discussed in Section IV. To reproduce the same line or lines at more than one point in the AWS, the library INSERT may be used.

NOTE: If the value of o is equal to or greater than the current last sequence number, the command is in error and will be rejected.

If no data records are found in the range m,o, no action is taken, and ROSCOE displays the notification message:

```
NO LINES FOUND IN RANGE
```

3.9 THE EDIT COMMAND

To scan (search) the AWS for the occurrence of any string (series) of characters, use the command:

```
EDIT /string/

E,/string/
```

where / represents any special character used as a delimiter for the string (within which the delimiter itself cannot appear). The string may be of any length, up to 80 characters, as long as the entire command can fit into a single line at the terminal. ROSCOE will scan the AWS for the string, and if a match is found, the line will be listed at the terminal preceded by the sequence number.

To limit the search to a single line or a range of lines, use the form:

```
EDIT /string/m,n        or     EDIT /string/m

E,/string/m,n           or     E,/string/m
```

To replace in the AWS any desired string, use the form:

```
EDIT /string1/string2/

E,/string1/string2/
```

where string1 is the string to be replaced (wherever it occurs) by string2. The two strings do not have to be the same size. If string1 is replaced by a shorter string, all data to the right of the original string will be shifted over to the left, and vice versa. When data is shifted to the left, the right-hand end of the record will be padded with blanks. If a data shift to the right would result in characters going beyond the last column of data, such characters will be lost.

Example — Original line:

```
ABCDEFGHIJKL
```

Command:

```
EDIT /ABC/Z/
```

Result:

```
ZDEFGHIJKL
```

Example — Original line:

```
ABCDEFGHIJKL
```

Command:

```
EDIT /A/123456789 /
```

Result:

```
123456789 BCDEFGHIJKL
```

Note that a space is a significant character within a string.

To replace a string within a single line or range of lines, use the forms:

```
EDIT /string1/string2/m        or    EDIT /string1/string1/m,n

E,/string1/string2/m           or    E,/string1/string2/m,n
```

To delete every occurrence of a given string (throughout the AWS or by line) use the form:

```
EDIT /string1//     or    EDIT /string1//m    or    EDIT /string1//m,n

E,/string1//        or    E,/string1//m       or    E,/string1//m,n
```

To replace an entire line or range of lines in the AWS, use the forms:

```
EDIT //new line/    or    EDIT //new line/m    or    EDIT //new line/m,n

E,//new line/       or    E,//new line/m       or    E,//new line/m,n
```

Users should select this command to replace single lines in the AWS in preference to typing an entire new line preceded by its sequence number, for the use of EDIT is much more rapid and does not involve a subsequent reorganization of the AWS.

When performing replacements on the entire AWS, it is often useful to see the lines being selected for editing as they are updated. To do this, use either:

```
TRACE EDIT
    or
TRACE E
```

which will cause every line being updated to be displayed at the terminal immediately *after* it has been edited and replaced in the AWS. Use of trace-display will permit the user to cancel further editing by using the ATTN function. This is useful when errors have been made in selecting search argument or functions. (For further information, see Section 5.4.)

It is often useful to be able to limit EDIT operations to a range of columns within the 80-column card image. To define start and stop columns for EDITing, use the form:

```
EDIT m,n

E,m,n
```

where m is the column in the line where EDIT operations are to start, and n the column where they are to stop (inclusive column numbers).

Example:

```
EDIT 32,48
```

Result:

No scan or replacement will take place in columns 1-31 and 49-80.

To set only the stop columns, use the form:

```
E,n
```

which is equivalent to EDIT 1,n.

Start and stop definitions are set to 1,80 at sign-on time.

NOTE: The first column of a line is 1, the last is 80. The m and n may be equal (this will limit EDITing to a single position in the line); otherwise, n must be greater than m and not higher than 80.

If EDIT cannot find a match during the search process, ROSCOE will inform the user with the diagnostic:

```
NO MATCH FOUND IN RANGE.
```

3.10 THE EDIT LIST COMMAND

To display on the 3270 screen a set of lines from the AWS for on-screen modification, use the command forms:

```
EDIT LIST

EDIT LIST,m

EDIT LIST,m,n
```

"LIST" is a required operand; m and n, designating a specific line or range of lines, are used when required. The AWS records requested will be displayed a page at a time on the display screen in the following format:

> Each record will be displayed on two consecutive lines. The first line will contain the sequence number and is formatted as a protected field. The second line will contain a full 80-character data record from the AWS, and is formatted as an unprotected field. A page consists of 12 such records (24 lines), unless SCALE is ON, in which case a scale line will be displayed in the top and bottom lines of the screen, with only 11 records (22 lines) displayed in the intervening lines. The cursor will be positioned at the beginning of the first unprotected field on the screen. As more than one write to the screen may be required before the page is completely displayed, the keyboard will remain locked until the entire page is on display.

The AWS records may be modified in any manner desired by the user. When editing operations on the page are complete, it is returned to ROSCOE by the ENTER key (any other program attention key is interpreted as an ATTN, and will terminate the command and cause any data on the screen at the time to be ignored). ROSCOE receives, along with the page returned, an indication of which data records were modified by the user, and will replace them in the AWS in their modified form.

NOTE: Since the scale lines and the sequence numbers are protected fields, they cannot be modified in any manner by user action. To advance the cursor from one data line to the next, the NEWLINE or TAB keys are used. To modify the records displayed, any facility of the device may be freely used (INSERT, DELETE). The use of ERASE EOF may, however, cause any modifications to that record to be ignored.

NOTE: The command is EDIT, and can therefore be abbreviated normally to E.

NOTE: This command is valid only when entered from a 3270 display station. If entered from a different device, it will generate an error message and be ignored.

3.11 THE FILL COMMAND

To place an arbitrary string of characters into specific columns of an AWS line or lines, regardless of the existing contents of those columns, use the command:

```
FILL /string/m,n
```

If m,n are omitted, the string will be forced into every line of the AWS, otherwise only into the range named.

NOTE: FILL is a special form of EDIT, where the argument search is always bypassed. The column limits are set by the EDIT m,n command (FILL m,n is also valid: please observe that the FILL limit settings are the same as the EDIT limit settings). If the string to be forced into the card image is too long to fit, then ROSCOE will terminate the command with the diagnostic

```
TOO MANY FILL CHARACTERS.
```

If the string is shorter than the area into which it is to go, it will be duplicated as often as needed to fit, and if necessary the last duplication will be truncated.

Example – Limit columns are 1,10, and original line:

```
0010 1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

Command:

```
FILL /TEST/
```

Result:

```
0010 TESTTESTTEABCDEFGHIJKLMNOPQRSTUVWXYZ
```

3.12 THE OMIT COMMAND

To perform an exclusive scan, use the command:

```
OMIT /string/

O,/string/
```

where line numbers in the form m or m,n may be placed after the second delimiter. This command acts as a NOT-EDIT, and will display at the terminal only those lines of the AWS in which the string defined between the special delimiters was NOT found. No replacement function is permitted in an OMIT command. The column-search limits are set by the normal command EDIT m,n or OMIT m,n. The column limits for EDIT, FILL and OMIT are the same counters, not separate counters.

Example – AWS contains two lines, as follows:

```
0010 1234567890ABCDEFGHIJ
0020 ABCDEFGHIJ1234567890
```

EDIT column limits are 11,80.

Command:

```
OMIT /1234567890/
```

will cause a display of line 0010 only, since the string was NOT found in that line between columns 11 and 80. Line 0020 will be rejected, since the string did occur in the selected columns.

3.13 THE SCALE COMMAND

The format and effect of this command depends on the terminal type.

A. For typewriter devices (2741, TTY), the command format:

```
SCALE
```

without operands will cause an immediate type-out of a calibration line:

```
.........1.........2.........3.........4.........5.........6.........7.........8
```

This line will begin in the 6th position on the page from the left margin; therefore, the positions correspond to those taken by data listed by ROSCOE or entered in Autoline mode. Any operands added to the command will be ignored.

B. For screen (display) devices, the command:

```
SCALE ON
```

will cause ROSCOE to print a calibration line (as shown above) as the first line of the screen when displaying a screen of sequence prompters in Autoline mode. The calibrator will also be displayed as line 1 of the screen when in the EDIT LIST command. The command:

```
SCALE OFF
```

will discontinue the use of the calibrator.

NOTE: The SCALE is OFF at sign-on time. The use of SCALE with no operands is equal to SCALE ON.

3.14 THE SEQ AND NOSEQ MODES

A physical line of text in the AWS consists at all times of 80 characters of data, with a sequence number associated with the record but not part of it. Normally, however, ROSCOE considers that a user is in *sequence mode*; that is, that a logical line in the AWS consists of 76 characters of data followed by a 4-character sequence number. Thus a listing of the AWS will normally display only card images, with a sequence number taken from columns 77 through 80. SEQuence mode is set at log-on, but can be altered at any time during the session by the command:

```
NOSEQ
```

which places the user into a new mode, where 80 characters (if there are that many) will be displayed as a result of list or edit-trace. Likewise, when in NOSEQuence mode, a job submitted from the AWS will go into the job stream as a series of card images with 80 characters of data and no sequence number. The two modes can be switched at any time by entering:

```
SEQ
```

or

```
NOSEQ
```

as desired. The effect of these modes on the library data sets is noted in Section IV.

3.15 THE AWS COMMAND

To determine at any moment how many lines of data are in the AWS, use the command:

```
AWS
```

which will cause ROSCOE to display the AWS status message:

```
LAST LINE mmmm, TOTAL LINES nnnn, INCR pp
```

The last line number is that of the last line actually entered into the AWS, and the total of lines is the true total at the moment. Thus, if reorganization is pending, the AWS status message may appear to be inaccurate.

Example — The user first enters into the AWS the following:

```
0010 line one

0020 line two

0030 line three
```

He then inserts a line:

```
0015 line one and a half
```

and then replaces a line:

```
0010 new line one
```

The AWS status message at that point will appear as:

```
LAST LINE 0010, TOTAL LINES 0050, etc.
```

But if the AWS is LISTed, it will be listed correctly as:

```
0010 new line one

0015 line one and a half

0020 line two

0030 line three
```

and the AWS status message at that point will be correct:

```
LAST LINE 0030, TOTAL LINES 0004, etc.
```

3.16 THE EXTENDED AND BASIC MODES (X, B)

Two character modes are available through ROSCOE. These are the *basic* mode (only upper case alphabetic, plus all available numerals and special characters), and *extended* mode (upper and lower case alphabetic, plus all available numerals and special characters). ROSCOE normally works in basic mode, so that lower case alphabetics entered at the terminal are automatically translated to upper case and will be so retained in the AWS. To use the extended character set, the command is:

```
X
```

which then places the user in extended mode. To return to basic mode, the command is:

```
B
```

NOTE: These commands are disregarded on a 2260 and TTY which have only upper case alphabetics available. The one significant difference between X and B is that in the former, a BACKSPACE in upper case (that is, shift key depressed while backspacing) will not function as a rub-out, but be incorporated in the text. This feature is useful mainly for underscoring (e.g., THIS IS IMPORTANT). Regardless of the mode a terminal is in at a given moment, text being listed will be in extended or basic sets depending on the mode operative when it was entered.

3.17 ERROR CONDITIONS

Although the terminal user is normally unaware of the mechanics of the AWS, he must be aware that actually the AWS resides on a disk, and is therefore subject to permanent I/O errors. When such an error occurs, ROSCOE displays the message:

```
I/O ERROR TO THE AWS
```

and resets all AWS pointers (that is, all current data in the AWS is lost). In this event, the operator at the central systems console also receives a warning message, and the matter will be brought to the attention of the systems management, who may wish to shut down ROSCOE and reformat the AWS. This situation is extremely rare, and always represents a 'hard' error on disk. There is no recovery from a hard error.

IV. THE LIBRARIES AND LIBRARY COMMANDS

The ROSCOE line libraries are shared by all terminal users although each user is unaware that such sharing is taking place. A library data set or *named data set* (dsn) is in a special compressed format which is incompatible with normal OS usage (that is, cannot be used as data input by any programs except ROSCOE and the special ROSCOE utilities ROSCOPY and ROSDATA, described in Section IX).

The number of physical line libraries varies from site to site at the option of site management. A terminal user is logically connected to a library when he enters a library command; the connection is severed at termination of command execution.

Most library commands effect a transfer of data between the libraries and the AWS. As with the AWS, a permanent I/O error while performing a library function is always possible. No data is lost on the library when this occurs, unless the error is due to a defective track or physical record which cannot be read. In the latter case, the user data set being accessed has probably been damaged and cannot any longer be retrieved; it should be deleted, an operation which should be performed by the site system programmer off-line.

A more serious condition arises when all space is exhausted on the libraries. This condition should never occur, but if it does, the user is sent a diagnostic and the central console operator is informed. Normally, the system will be shut down and site management will allocate more space to the ROSCOE libraries.

4.1 THE SAVE COMMAND

To save text in the library, the command is:

```
SAVE dsn

S,dsn
```

where dsn is the name of the new data set. The dsn is from 1 to 8 characters in length. The first character of the name should be alphabetic, but ROSCOE makes no test of the name except for length. A SAVEd dsn should not begin with national characters "#", "$", or "@". Where dsn already exists, ROSCOE displays the error message:

```
DUPLICATE DATA SET NAME – NO ACTION TAKEN.
```

A SAVE command does not alter the state of the AWS.

When a data set is SAVEd, ROSCOE notes whether the user is in SEQ or NOSEQ mode, and flags the data set accordingly. This flag is examined only by SUBMIT and ROSCOPY (see Sections VI and IX).

NOTE: At the discretion of system management, a maximum limit may be set to restrict the total number of records which a user may SAVE in the library. If the maximum is exceeded, a SAVE or UPDATE operation will be rejected, and the following warning message will be displayed at the terminal:

```
LIBRARY MAXIMUM EXCEEDED – NO ACTION TAKEN.
```

When this occurs, the user should examine all his library data sets and purge those no longer needed in order to lower his current line count. Whenever a data set is SAVEd in the library, the current line count is incremented; whenever a data set is DELETEd from the library, the current line count is decremented. The line count is not incremented when data sets are added using ROSDATA, the post-processors, or the ROSCOE Writer Facility. The STATUS command will list the current line count and the maximum line count for users who are restricted in their SAVEs. The facility to maintain the current line count is operational only in the on-line portion of ROSCOE, and any data sets to be DELETEd from the library should accordingly be removed using the ROSCOE DELETE command.

NOTE: If the data set being SAVEd is to be named in a SIGNON command (see Section II, 5.), the maximum length of the portion of the name assigned by the user in the SAVE command is 6 characters, not 8.

4.2 THE FETCH COMMAND

To bring a data set on a library into the AWS, use:

```
FETCH dsn

F,dsn
```

where, again, the dsn may be prefixed if belonging to another user.

If no such data set exists, ROSCOE displays the message:

```
dsn DATA SET NOT FOUND
```

(This message is displayed whenever a command requests the use of a data set which does not exist.) In systems which have the ROSCOE read-protect feature, an attempt to access a data set protected from you (the protection arrangement is defined by the installation manager) will cause display of the message:

```
dsn DATA SET PROTECTED
```

The entire data set is transferred from the library to the AWS (which is cleared previously) with the sequence numbers originally SAVEd with it.

When a library data set is FETCHed, its name is stored by ROSCOE. The name of the last data set FETCHed is available through the STATUS command. The name is kept by ROSCOE until any of the following occurs:

- Another data set is FETCHed;
- The AWS is completely deleted; or
- The contents of the AWS are replaced by a MERGE or CHAIN.

A special form of the UPDATE command permits automatic referencing of the stored name (see Section 4.4).

4.3 THE DELETE COMMAND

To delete a library data set, the command is:

```
DELETE dsn
```

If the data set named is protected from you, the protect message will be displayed. If the data set named does not exist, the 'not found' message will be displayed.

NOTES:

- This command does not have a short form.
- At the option of site management, a confirmation of a library DELETE request will be issued before executing the request. The following message will be written to the terminal:

  ```
  DELETE xx.yyyyyyyyy:  ENTER "DELETE" TO CONFIRM.
  ```

  If DELETE is entered from the terminal, the DELETE request will be executed. Otherwise, the reply from the terminal will be taken as a new command which will be executed.

  If, however, the DELETE dsn command is issued from a ROSPROC, no confirmation message will be issued, and the request will be executed immediately.
- An attempt to DELETE a ROSPROC belonging to the terminal user and which he is currently executing will be rejected with the following message:

  ```
  DATA SET PROTECTION CHECK:  NO ACTION TAKEN.
  ```

  See Section E.7 for further details.

4.4 THE UPDATE COMMAND

To replace a data set with an updated version, use the command:

```
UPDATE dsn

U,dsn
```

If the dsn does not exist, the 'not found' message will be displayed, and no action will be taken.

To update the latest data set fetched from the library, use the form:

```
UPDATE *

U,*
```

where the asterisk will be replaced in the command by the name of the last data set fetched. Refer to Section 4.2 for a discussion of the conditions under which the data set name is stored. Note that an asterisk is not a reserved character, and it is possible to SAVE a data set named *.

NOTES:

- If UPDATE is entered while in PROC mode, it will cause a SAVE under all circumstances. That is, while in PROC mode, UPDATE acts either as SAVE or UPDATE. If, during an UPDATE operation, an I/O error occurs so that the operation is incomplete, the old version of the data set being updated will be retained. (For additional information, see Section 4.1.)

- An attempt to UPDATE a ROSPROC belonging to the terminal user and currently being executed will be rejected via the following message:

  ```
  DATA SET PROTECTION CHECK:  NO ACTION TAKEN.
  ```

  See Section E.7 for further details.

4.4A THE RENAME COMMAND

To rename a library data set without modifying the data in any manner, use the command:

```
RENAME oldname,newname
```

where oldname gives the name of the existing data set, and newname gives the name to be assigned to the data set.

NOTE: This command will fail for either of the following reasons (which will be diagnosed by a terminal error message):

- oldname does not exist or does not belong to the terminal user; or
- newname already exists (duplicate name condition). In this case, the rename must be preceded by a DELETE command for the data set with the duplicate name.
- An attempt to RENAME a ROSPROC belonging to the terminal user and currently being executed will be rejected via the following message:

  ```
  DATA SET PROTECTION CHECK: NO ACTION TAKEN.
  ```

  See Section E.7 for further details.

4.5 THE LIST COMMAND

To list a data set in the library without fetching it into the AWS, the command is:

```
LIST dsn

L,dsn
```

which will list the data set in its entirety, or:

```
LIST dsn,m          or    LIST dsn,m,n

L,dsn,m             or    L,dsn,m,n
```

to list a single line or a range of lines. The data will be listed without sequence numbers, as pure text.

NOTE: If no lines are found within the range m,n, ROSCOE will so inform the user with the diagnostic:

```
NO LINES FOUND IN RANGE.
```

This diagnostic will be typed for any library operation to which it can apply.

4.6 THE MERGE COMMAND

To merge library data sets into the AWS, the command is:

```
MERGE dsn1,dsn2,....dsn12
```

(If only one dsn is coded, the command becomes a FETCH.)

The data sets named will be merged into the AWS by line number.

Example – Command is:

```
MERGE ABC,DEF,GHI
```

The data sets contain data such that:

```
         ABC             DEF               GHI

9000 xxxxx      0001 aaaaa      0003 @@@@@@@@@
9010 yyyyy      0002 bbbbb      9020 zzzzzzzzz
                0003 ccccc
```

The AWS after the merge will contain:

```
0001 aaaaaa
0002 bbbbbb
0003 @@@@@@@@@
9000 xxxxx
9010 yyyyy
9020 zzzzzzzzz
```

If one of the dsn's is not found, the 'not found' message will be displayed, and the execution of the command will be terminated. Any data already brought into the AWS will remain there for use.

4.7 THE CHAIN COMMAND

To bring several data sets serially into the AWS without merging them, the command is:

```
CHAIN dsn1,dsn1,....dsn12

C,dsn1,dsn2,....dsn12
```

The data sets named are brought into the AWS one-by-one, the original sequencing is disregarded, and the lines are renumbered from 1 by increments of 1.

4.8 THE APPEND COMMAND

To append data from a library data set to the current contents of the AWS, the command is:

```
APPEND  dsn

A,dsn
```

which will append the entire dsn to the end of the AWS, sequencing in the manner described in the AWS APPEND (see Section III).

To append by line number, use:

```
APPEND  dsn,m        or      APPEND  dsn,m,n

A,dsn,m              or      A,dsn,m,n
```

Note that it is possible to append into an empty AWS, so that a fetch by line number is possible.

4.9 THE INSERT COMMAND

To insert lines from a library data set into the AWS, the command is:

```
INSERT dsn,m,n,o

I,dsn,m,n,o
```

which inserts lines m through n of dsn into the AWS following line o;

```
INSERT dsn,m,o

I,dsn,m,o
```

which inserts line m of dsn into the AWS following line o;

```
INSERT dsn,o

I,dsn,o
```

which inserts the entire data set following line o.

Attempts to insert into an empty AWS are treated as errors. It is also an error if the value of "o" is not less than the actual last line number in the AWS.

NOTE: If a line or lines is to be INSERTed into the AWS at more than one point, the AWS INSERT cannot be used. The procedure is to SAVE the desired line or lines as a named data set, and INSERT from the library.

4.10 THE LIBRARY COMMAND

To list all data sets belonging to a user, the command is:

```
LIBRARY
```

The contents of the AWS will be replaced with a list of the user's data sets. When the list is complete, ROSCOE will immediately begin to list it from the AWS to the terminal.

To retrieve the list of library data sets without listing it at the end of retrievel, the command is:

```
LIBRARY N
```

where N stands for NOLIST.

To list only library data sets which have been added to the system through the batch interfaces; i.e., ROSDATA, post-processors, or SMB's (writer data sets), the command is:

```
LIBRARY B
```

where B stands for BATCH.

B and N may be used together as operands, if coded with a comma between them. The order of the two operands is arbitrary; i.e., the two following forms are equivalent:

```
LIBRARY B,N
LIBRARY N,B
```

If no data sets exist for the signed-on key, the following message will be displayed:

```
NO DATA SETS FOR THIS ACCOUNT.
```

The information available through the LIBRARY command on-line is available also from the batch report ROSMAILS, which is run at regular intervals at the option of site management and distributed to the owners of the data sets at the discretion of site management.

The list of library data sets is in alphabetic order. Each entry has the following format:

```
xx.dddddd f c-date u-date cnt
```

where:

xx	is the user's prefix.
dddddd	is the assigned name.
f	is an asterisk (*) if the data set is output of the writer (SMB data set); otherwise, it is blank.
c-date	is the creation date of the data set.
u-date	is the date the data set was last updated. (This will be a blank if the data set has never been updated.)
cnt	is the record count of the data set.

A sample listing from the LIBRARY command is shown in Example 1.

NOTE: The owner of a library data set is determined by protect code (i.e., an internal code not known to the terminal user) rather than prefix. However, because the protect code and prefix of a library data set normally are associated with the same ROSCOE sign-on key, the listing produced by the LIBRARY command is selected by prefix rather than protect code. If a user desires to see a listing of his data sets selected by protect code rather than prefix, the operand C may be used, as follows:

```
LIBRARY C
```

This form could be used when a user has, for some reason, saved data sets using a prefix not his own.

```
0003   RO.APARS      01/12/75   04/10/75   00021
0004   RO.APRIL1     05/23/71   04/10/75   00011
0005   RO.APT        08/26/71   04/10/75   00001
0006   RO.AUG29      08/30/72              00005
0007   RO.BACKUP     01/12/75   04/10/75   00030
0008   RO.BLA        01/14/75              00008
0009   RO.BST        08/26/71              00002
0010   RO.BYONES     03/29/74              00002
0011   RO.CHA        09/13/71              00015
0012   RO.CRT        10/29/71              00007
0013   RO.DMBASK     01/14/75              00009
0014   RO.DUMP       01/12/75              00516
0015   RO.FEB8       02/08/73              00008
0016   RO.HELLO      08/11/69              00014
```

EXAMPLE 1. FORMATTED LIBRARY LISTING

4.11 THE FIND COMMAND

To obtain for a single library data set the same information provided for all data sets by the LIBRARY command, use:

```
FIND dsn
```

where dsn is the name of the data set whose index entry is desired. The entry for the data set, if found, is displayed at the terminal. (It is not placed in the AWS.)

4.12 THE SPACE COMMAND

To display during ROSCOE execution how many free blocks remain in the on-line library(s), use the command:

```
SPACE
```

The response will be the following message:

```
LIBRARY BLOCKS AVAILABLE:  nnnnn
```

where nnnnn is the number of free blocks remaining.

NOTE: This command is primarily for the use of ROSCOE site management.

V. INTERROGATIVE AND STATUS COMMANDS

5.1 THE MESSAGE COMMAND

To send a message at any time to the central console operators, the command is:

```
MESSAGE string
```

where the string (which does not need to be enclosed in quotes) is the message to be sent, up to a length not exceeding 130 characters. This message will be typed at the operator's typewriter immediately, unless the installation manager has disabled the message function.

The operator can also communicate from his typewriter with the ROSCOE terminals, either broadcast or by Telecommunications line number. Messages originating with the operator are queued at a terminal until a terminal command in execution has completed and a RETURN has occurred.

A message cannot be typed while a terminal is idle. There is no provision in ROSCOE for switching messages between user terminals.

NOTE: A null string (i.e., a message command followed by all spaces) will be suppressed.

5.2 THE TIME COMMAND

To find out at any moment the current time and date, use:

```
TIME
```

to which ROSCOE responds with the message:

```
          hh.mm.ss                mm/dd/yy

(hours.minutes.seconds)    (month/day/year)
```

5.3 THE STATUS COMMAND

To find out details of the current status of the user's resources and certain environment information, the command is:

```
STATUS
```

ROSCOE will then print a 2- to 3-line message with the following information:

a. Name of last dsn data set fetched from library;
b. Mode – SEQ or NOSEQ;
c. Special tab character used on display screen;
d. Current tab columns;
e. SCALE – ON or OFF on display screen;
f. Current BREAK value;
g. Character mode – X or B;
h. TRACE – (E, P);
i. Current EDIT column settings;
j. Special character used to separate packed commands in a ROSPROC;
k. Current value of ROSCOE counter; and
l. Number of records SAVEd on library and maximum number that may be SAVEd (applies only to users restricted in the number of lines they may SAVE on the ROSCOE library).

Any information not applicable is omitted; e.g., if tabs are not set, no tab information is given. Following this information, ROSCOE will print a final status line:

```
LINE #nn TIME IN hh.mm.ss, ELAPSED hh.mm.ss –oo USERS
```

which gives the user's line number (nn), the time he logged-on (hh.mm.ss), time elapsed since then (the second hh.mm.ss), and the number of users currently using ROSCOE resources (oo).

5.4 THE TRACE AND NOTRACE COMMANDS

These commands permit certain ROSCOE activities to be TRACEd at the terminal during execution, or to terminate the TRACE. The formats are as follows:

- For the EDIT command:

  ```
  TRACE EDIT
       or
  TRACE E

  NOTRACE EDIT
       or
  NOTRACE E
  ```

  For a full explanation, see Section 3.9.

- For ROSPROC execution:

  ```
  TRACE PROC
       or
  TRACE P

  NOTRACE PROC
       or
  NOTRACE P
  ```

  For a full explanation, see Section 8.3.

- For both TRACEs, the following formats are valid:

  ```
  TRACE ALL
       or
  TRACE (no operand)

  NOTRACE ALL
       or
  NOTRACE (no operand)
  ```

VI. REMOTE JOB ENTRY: SUBMIT

The RJE facility of ROSCOE is handled by the SUBMIT command which has two forms:

```
SUBMIT
```

which places the contents of the AWS into the OS job stream; and:

```
SUBMIT dsn1,dsn2,....dsn12
```

which takes the 1 to 12 library data sets in the order named and places them into the OS job stream. The following paragraphs should be carefully noted.

6.1 EFFECTS OF SEQ AND NOSEQ MODES

When submitting from the AWS, the mode (SEQ or NOSEQ) currently active determines at SUBMIT-time whether the 4-character ROSCOE sequence number will be placed in columns 77-80 of every card image. Data sets submitted directly from the library will have a sequence number or not, depending on the SEQ/NOSEQ flag created when the data set was placed in the library. This latter feature enables a job stream to contain parts with sequence numbers (such as source programs, where the sequence number will aid in debugging) and parts without (such as data used as input to a problem program, where the presence of a sequence number would be an intrusion on the format of the data).

6.2 VALIDATING THE JOB CARD

ROSCOE will test the first card image of the job stream for a valid job card; that is, the first record must contain in columns 1 and 2 "//", followed immediately by a valid job name and at least one space, followed by the word "JOB" and one space. If an invalid job card is found, the job is rejected, and ROSCOE will display the message:

```
INVALID JOB CARD  j o b  c a r d  i m a g e
```

6.3 ERROR RECOVERY

If an I/O error occurs at any time while a job stream is transferred to the batch set, the job will be deleted, and a warning message displayed at the terminal. An I/O error occurring on the batch set itself is considered a system error, and the operator also is sent a warning message. The SUBMIT function will then be discontinued for the rest of the ROSCOE session.

6.4 INVALID DATA SET NAMES

If while submitting from the library one of the data sets named is not found or is protected, the job stream will be deleted and the terminal user sent a warning message. In a HASP system, ROSCOE will cancel the job from HASP by writing a "/*DEL" command to the internal reader.

6.5 COMPOSITION OF JOB STREAM

ROSCOE does not check whether the job stream submitted consists of one job or multiple jobs. ROSCOE automatically appends a /* to the end of the job stream. If HASP is used, ROSCOE automatically closes the internal reader at the end of the job stream by appending a /*EOF card.

6.6 USER JOB STREAM ANALYZER

Your installation may have its own job stream analyzer in operation, which can inspect a job stream submitted from a terminal as it is being written out. It may command ROSCOE to reject the job stream, in which case ROSCOE will display a message at the terminal that such is the case. The message will originate either from the installation routine or from ROSCOE itself. If you are unsure of the reason for the rejection, see your installation manager.

6.7 SUCCESSFUL SUBMISSION

If a job stream has been successfully submitted to OS for execution, ROSCOE will display the message:

```
jobname SUBMITTED AT hh.mm.ss
```

where jobname is taken from the job card, and hh.mm.ss is the time submitted. The system operator, at installation option, will also receive notice of the submission. Once a job has been submitted, it is subject to the queueing and control of OS and the system operator.

To recover batch output at the terminal, the post-processors are used, as described in Section IX. To precheck jobs before submitting them, the language and JCL Syntax Checkers are used as described in Section VII.

6.8 STANDBY

The SUBMIT function is one of the three ROSCOE functions which are enqueued; that is, only one user at a time can use it. If another terminal user is in the process of submitting a job, you will receive the STANDBY warning. You can then use the ATTN function to dequeue yourself from SUBMIT, thus allowing you to accomplish another terminal task while deferring the SUBMIT. Otherwise, you can simply wait for SUBMIT to become available.

6.9 INSUFFICIENT SPACE FOR USER JOB

If there is not enough space on the ROSCOE batch data set to write out a job, the terminal user will be notified by the message:

```
JOB NOT SUBMITTED DUE TO INSUFFICIENT SPACE. PLEASE TRY LATER.
```

When this happens, the console operator is notified, and ROSCOE attempts to remedy the situation by starting an OS Reader. Under normal circumstances, the user can successfully re-submit his job within 5 minutes. This message cannot appear in a HASP system.

6.10 SUBMIT DISABLED

If the central console operator has disabled the use of the SUBMIT command (due to job queue overflow), the terminal user will be notified by the message:

```
SUBMIT UNAVAILABLE AT OPERATOR REQUEST.
```

The central console operator has the facility to inform all users when the SUBMIT command is re-enabled.

VII. ROSCOE SYNTAX CHECKERS AND EXTERNAL ROUTINES: VERIFY AND RUN COMMANDS

7.1 SCOPE

The Syntax Checkers are a ROSCOE option which may be included in your system. The function of a Syntax Checker is to determine whether the input statements passed to it by ROSCOE are constructed according to the rules governing the formation of statements in the source language. If they are, they are accepted without comment; otherwise, an error diagnostic is displayed at the terminal. No meaning is assigned to the input statements by the Syntax Checker; that is, if the statement:

```
MOVE A TO B
```

is formally correct in a given language, it will be passed, without attempting to verify that A and B have been properly defined elsewhere in the same source program. Likewise, as a further example, if the JCL Checker finds the formally correct statement:

```
// DSN=SYS3.LIBA,DISP=OLD
```

no attempt will be made to verify the existence of a data set named SYS3.LIBA.

7.2 THE VERIFY COMMAND

The command used to perform syntax checking is VERIFY (or V) in the forms:

```
VERIFY lang              or      V,lang
VERIFY lang,m            or      V,lang,m
VERIFY lang,m,n          or      V,lang,m,n
VERIFY lang,dsn          or      V,lang,dsn
VERIFY lang,dsn,m        or      V,lang,dsn,m
VERIFY lang,dsn,m,n      or      V,lang,dsn,m,n
```

where lang is the name of the language being checked. The current languages checked are FORTRAN, COBOL, and JCL.

7.3 COBOL SYNTAX CHECKER

For COBOL syntax checking, lang appears as:

COBE	(COBOL E)
COBF	(COBOL F)
COBA	(ANS COBOL)
COBD	(COBOL D, for programmers maintaining DOS versions)

When these forms are coded, the input is expected to be a complete COBOL program, beginning with the IDENTIFICATION DIVISION.

But independent segments of a program may be checked as well, by adding:

```
–I   or   –E   or   –D   or   –P
```

to the operand, signifying that checking begins respectively with the IDENTIFICATION, ENVIRONMENT, DATA, or PROCEDURE DIVISIONs. Examples of this include:

```
VERIFY COBA
V,COBF–P,COBP,200,800
V,COBE–E,230,450
```

Diagnostic messages consist of the source string which is unacceptable to the Checker, the sequence number of the line in which it occurs, and a brief self-explanatory diagnostic. There may be multiple diagnostics on a single line. Occasionally, a COBOL syntax error cannot be detected until the line in error has been scanned past; in this case, the terminal diagnostic message will carry the sequence number of the *next* line.

The sequence number is taken from columns 77-80, the standard ROSCOE line number.

7.4 FORTRAN SYNTAX CHECKER

For FORTRAN syntax checking, lang appears as FORT.

The language checked is FORTRAN IV in its official definition. It is, therefore, possible that statements flagged by the Syntax Checker may be accepted by a particular implementation (compiler), and vice versa. Errors are flagged by listing first the source line, with sequence number, and then a line with a single dollar sign ($) under the invalid character.

If the error sign ($) is placed all the way to the left (under the source statement sequence number), the entire line is invalid and unacceptable to FORTRAN.

7.5 JCL SYNTAX CHECKER

For JCL syntax checking, lang appears as JCL if the input statements are to be checked without regard to their sequence as a job stream, or as JCL–J if verification is to be made that the statements represent a valid job stream. In the latter use, the first statement must be a JOB card, followed by a JOBLIB or EXEC card, etc. Diagnostics consist of the sequence number of the line in error, followed by a self-explanatory diagnostic, followed by that portion of the line which was found to be invalid. Multiple errors in a line are not flagged.

NOTE: Symbolic replacement parameters passed to a catalogued procedure on the EXEC card will be flagged with the message:

```
    ASSUMED SYMBOLIC PARAMETER
```

It is possible that a misspelled key word on such an EXEC statement will also be flagged with the same message. Thus, the statement:

```
    // EXEC ROSDATA,KEY='40021.JONES'
```

will be flagged, though correct (KEY is a symbolic parameter). The statement:

```
    // EXEC ASMFC,PRM='NODECK'
```

will be flagged with the same message (PRM is a misspelling of PARM).

The JCL Syntax Checker will terminate with a list of DSNAMES encountered during verification, for reference. It will bypass all non-JCL statements encountered; that is, lines not beginning with /, or data found between DD * or DD DATA and /* statements.

7.6 OBSERVATIONS

- COBOL and FORTRAN Checkers will also bypass any JCL statements encountered during verification.
- The Syntax Checkers are also enqueueable routines; that is, the STANDBY message may be received when requesting a Checker. See the description under SUBMIT for exiting from STANDBY.
- A Checker may be cancelled during execution by the use of the ATTN function while a diagnostic is being typed out.

7.7 THE RUN COMMAND

The syntax of the RUN command is identical to that of VERIFY (see Section 7.2). RUN is used to initiate execution of a ROSCOE External Routine (RER). Such routines are usually site-dependent programs written by the system programmer. (Information concerning the routine executions and functions should be distributed by site management). These routines are executed under ROSCOE Monitor control, which finds the routine desired and then allocates it to the terminal user. These routines are able to converse with the terminal user, read from the AWS or the Library, and create a new AWS.

A PAUSE state can be entered during execution of such routines. This condition occurs when a routine has entered a compute (CPU) bound loop and has remained there for a predetermined time limit. (This limit is set by the site manager and may vary according to the needs of site RER's). When this state is entered, ROSCOE will interrupt execution of the routine and display the inquiry:

```
PAUSE –– TYPE STOP TO TERMINATE
```

If the terminal user replies "STOP", ROSCOE will terminate execution of the routine. Any other response (including the word "stop" in lower case) will take the routine out of PAUSE state and resume execution.

Routines can be cancelled during execution if they are writing messages to the terminal by use of ATTN.

If a site RER terminates due to an unscheduled interruption ("abends"), ROSCOE will cancel use of that routine for the remainder of the day's execution, and will attempt to write out an indicative dump for debugging purposes. Normally, this will not occur when such RER's have been released for programmer use by site system programmers.

7.8 COPY

Applied Data Research offers a COPY routine to copy OS data sets into the AWS under terminal direction. Since this routine may have been modified by site management, no directions for its use are provided in this manual. Site management will distribute all information needed for RUNning COPY.

VIII. ROSCOE CONVERSATIONAL PROCEDURES

8.1 DEFINITION

A ROSCOE conversational procedure (ROSPROC) is a series of ROSCOE commands prepared by the user and stored in the ROSCOE library. Procedures are generally created for data manipulation functions that can be pre-planned, and that are repetitive and utilitarian in nature. Once a procedure has been created, it can be called and executed by entering a single ROSCOE command at a remote terminal.

A ROSCOE procedure can contain any of the commands, explicit or implicit, in the ROSCOE repertoire; in addition, a selection of specialized commands is provided to enable a user to build his "ROSPROC" conversational and decision-making characteristics. A well planned and implemented ROSCOE procedure, while performing its intended data manipulation function, can "converse" with the user, allow the user to monitor its progress, modify itself dynamically, and alter its sequence of activity based on variables introduced from the terminal. Thus, the ROSCOE procedure facility can be considered analogous to a high-level programming language, where AWS and library commands represent the data processing category of statements, and conversational procedure commands represent the control category of statements.

When execution of ROSPROC has been initiated by a user from his terminal, he relinquishes control of his terminal to that ROSPROC. Whereas ordinarily a terminal user can enter commands and data as he wishes, a procedure, once given control, governs all terminal activity. This condition, called terminal "procedure state," can be suspended to regain control and resume normal terminal activity. When a ROSPROC is terminated, the user receives a message stating "Procedure Concluded", and the terminal is released from the procedure state.

The paragraphs on the following pages describe the ROSPROC commands. Appendix E discusses the command usages and presents examples.

8.2 THE DO COMMAND

The DO command has the following formats:

DO dsn	Execute the ROSPROC named dsn.
DO dsn,m	Execute the ROSPROC named dsn, beginning at line m.
DO dsn,m,n	Execute the ROSPROC named dsn, beginning at line m and ending at line n.

The DO command initiates execution of the ROSPROC named dsn. The terminal is placed in the procedure state. In its first form, execution begins with the first line of the ROSPROC and continues to the last line, or until execution is suspended or terminated by a command within the procedure. The second form directs ROSCOE to begin execution of the ROSPROC beginning with line m; the third form directs ROSCOE to execute the ROSPROC beginning with line m and ending with line n. At the completion of the named ROSPROC, the terminal will be released from the procedure state.

The DO command can also be used within a procedure. In this context, it is used to alter the sequential execution of the ROSPROC by providing an unconditional transfer capability. Data set name, dsn, can refer to the ROSPROC in execution or to another ROSPROC.

To any of the three formats of the command shown above, an in-line reply may be appended. The text of the reply cannot be greater than 80 characters, must be delimited by a special character (as for the REPLY command, Section 8.8), and must be separated from the preceding operand by a single comma.

Example:

```
DO PROCA,*    DSN=SYS1.MACLIB,DISP=SHR*
```

The results of this format of the command is first to move the in-line reply into the reply buffer, and then initiate execution of the ROSPROC named "PROCA".

NOTE: The reply may be a null string, indicated by two consecutive delimiters (e.g., **). The effect of this format is to clear the reply buffer and reset the length of the buffer to 0.

8.3 THE TRACE AND NOTRACE COMMANDS

The TRACE and NOTRACE commands have the following formats:

TRACE PROC TRACE P	Each procedure command will be displayed at the terminal as it is executed.
NOTRACE PROC NOTRACE P	Turns off the TRACE facility.

The TRACE command provides a facility for assisting the user in checking out his ROSPROC. When the TRACE option is activated, each command in a procedure is displayed at the terminal just prior to its execution. TRACE and NOTRACE can be entered from a terminal at any time; the terminal need not be in the procedure state. These commands can also be included within a procedure; the effects are the same as when entered from a terminal.

NOTRACE is preset at sign-on time.

NOTE: If during the TRACE listing of a command prior to execution, ATTN is struck, the effect will be to place the procedure in the PAUSE mode. Similarly, if a LIST command is being executed in a procedure, an ATTN will place the procedure in PAUSE.

See also Section 5.4.

8.4 THE PAUSE COMMAND

The PAUSE command has the following format:

PAUSE	Suspends execution of a procedure (no operands).

The PAUSE command halts execution of a ROSPROC and places the terminal in suspended procedure state, thus taking control of the terminal away from the ROSPROC and restoring it to the user. The user can enter any of the AWS or user library commands; he can enter a GO command and resume execution of his ROSPROC at the line following PAUSE; he can enter an EXIT command and terminate his ROSPROC, releasing the terminal from the procedure state. The user can also enter a DO command, which effectively terminates the current procedure and invokes another procedure. The PAUSE command cannot be entered from the terminal.

8.5 THE GO COMMAND

The GO command has the following format:

GO	Resume execution of a ROSPROC (no operands).

The GO command is used to return control to a ROSPROC that has temporarily suspended execution. Execution of the procedure is resumed at the line following the command that suspended execution. The GO command will be accepted only when the terminal is in suspended procedure state (see Paragraph 8.4, the PAUSE command). It cannot be used within a ROSPROC.

8.6 THE EXIT COMMAND

The EXIT command has the following format:

EXIT	Terminate a ROSPROC (no operands).

The EXIT command terminates execution of a ROSPROC and releases the terminal from the procedure state. This command can be entered at a terminal only when a ROSPROC has suspended execution and placed the terminal in suspended procedure state. ROSCOE acknowledges the EXIT command with the response, "Procedure Concluded". EXIT can also be used within a ROSPROC to terminate execution of a first-level procedure, or to exit from a second-level procedure. (See Paragraph 8.12, the CALL command, for a discussion of procedure levels.)

8.7 THE TYPE COMMAND

The TYPE command has the following format:

TYPE string	Enables a ROSPROC to communicate with the user.

The TYPE command in a ROSPROC causes the message string to be displayed at the user's terminal. The message can be up to 80 characters in length. TYPE is commonly used preceding a PAUSE or REPLY command.

8.8 THE REPLY COMMAND

The REPLY command has the following formats:

REPLY	Enables the user to communicate with a ROSPROC.
REPLY n	Enables data in the AWS to be made available to a ROSPROC.
REPLY /string/	Makes a literal string available to a ROSPROC.
REPLY KEY	Place current user's ROSCOE key in the reply buffer.
REPLY PREFIX	Place current user's ROSCOE prefix in the reply buffer.
REPLY TIME	Place time and date into the reply buffer in the format: hh.mm.ss mm/dd/yy.

The REPLY command in a ROSPROC (in the first form above) unlocks the user's terminal, allowing data requested by the procedure to be entered. Data entered can be upper or lower case alphanumeric and is accumulated in an 80-character reply buffer, which is made available to subsequent ROSPROC commands.

The REPLY n form of the command causes up to 80 characters of data to be taken from line n of the AWS and placed in the reply buffer; no terminal activity will occur. A ROSPROC using this form of REPLY permits a user to enter all variable data into his AWS prior to invoking the procedure.

The third form of REPLY places the character string appearing between delimiters in the reply buffer. The length of a reply is accessible by the SET command.

Substring replacement within the reply buffer is performed by using the command format:

```
REPLY /string/m
```

The string will overlay the existing contents of the reply buffer beginning at column m for the length of the characters between the delimiters. An explicit length may be declared using the format:

```
REPLY /string/m,n
```

where n is the length of the character string to overlay existing reply characters.

Example:

```
REPLY /ABC/6,4
```

This format results in the insertion of a 4-character string ("ABC ") into the existing contents of the reply buffer starting at column 6. The string between delimiters is expanded by padding with blanks to the right in order to attain the declared length. If the declared length is less than the string, the string is truncated to the declared length. If characters are added to the reply buffer in this manner (so that the reply is now longer than before the insert), the length of the reply is recalculated. If the reply is made shorter (by placing trailing spaces in it), the length is also recalculated.

Example:

Reply buffer contains the string:

```
ABCDEF123456
```

The PROC command is:

```
REPLY /XYZ/4,5
```

The new reply buffer contents are:

```
ABCXYZ  3456
```

Five characters beginning at column 4 have been replaced by the new substring.

A special facility has been provided in conjunction with the REPLY command to allow the user to terminate a ROSPROC. If the data provided in response to a REPLY command is a single question mark (?), the ROSPROC terminates, the terminal is released from procedure state, and the "Procedure Concluded" message is displayed.

8.9 THE COMPARE COMMAND

The COMPARE command has the following formats:

COMPARE /string/	Compares string data against reply buffer and sets a condition indicator.

The COMPARE command compares the data within delimiters to the data in the 80-character reply buffer. The comparison is performed left to right with each character assuming its normal collating value (numerics are high, alphabetics low). If the two fields are of unequal length, the shorter one is space filled on the right. Any special character can be used as a delimiter.

The relationship between the two fields sets a condition indicator as follows:

		Condition Indicator
/string/	greater than reply buffer	High
/string/	less than reply buffer	Low
/string/	equal to reply buffer	Equal
/string/	not equal to reply buffer	Unequal

The condition indicator remains set until a subsequent COMPARE command alters it. The COMPARE command is valid only within a ROSPROC.

Substring comparison is possible with the form:

```
COMPARE /string/m
```

where m is the starting column within the reply buffer (1 through 80). The length of the comparison is determined by the number of characters between the special character delimiters. An explicit length may be declared using the form:

```
COMPARE /string/m,n
```

where m is the starting column within the reply buffer, and n is the length of the comparison. If n, the declared length, is longer than the string given, the string is expanded on the right by spaces. The comparison is always performed only for the declared length. The rule for the values of m and n is that their sum less 1 cannot be greater than 80. Example F-4 (included in Appendix E) shows how to scan through the reply buffer for a given string.

To determine if the contents of the reply buffer are numeric, use the form:

```
COMPARE N
```

where N stands for numeric and is a reserved keyword. If the string is numeric, the condition indicator is set to Equal; if not, it is set to Low (Unequal). The comparison is performed by taking the actual length of the current reply, unless the following substring notation is used:

```
COMPARE N,m
```

The above format will perform a numeric test for the contents of the reply buffer starting at position m. The length of the comparison is the length of the actual reply minus (m-1); i.e., the number of positions bypassed. An explicit length may be given in the format:

```
COMPARE N,m,n
```

which will test the contents of the reply for numerics starting at position m for a length of n.

NOTE: The COMPARE N command will result in an ENTRY INVALID diagnostic for any of the following reasons:

- There is no reply in the buffer.
- A substring request starts to the right of the last character of the reply.
- An explicit length definition will cause the comparison to go past the last character in the buffer.

The current value of the ROSPROC counter can be tested by the format:

COMPARE v

where v is the value to be compared to the counter. See Section 8.14 for further details on this format.

8.10 THE IF COMMAND

The IF command has the following formats:

IF condition,dsn	If condition is true, execute ROSPROC named dsn; if not, continue in sequence.
IF condition,dsn,m	If condition is true, execute ROSPROC named dsn beginning with line m; if not, continue in sequence.
IF condition,dsn,m,n	If condition is true, execute ROSPROC named dsn beginning with line m and ending with line n; if not, continue in sequence.

The IF command provides a conditional transfer facility within a single ROSPROC or among multiple ROSPROCs. The condition field is written as HIGH, LOW, EQUAL, or UNEQUAL, and corresponds to the condition set by a preceding COMPARE command. The destination procedure will be entered at line m, if m is specified, and terminated at line n, if n is specified. The terminal is released from the procedure state after line n is performed. In all three forms of this command, when the condition specified is not true, the ROSPROC "drops through," and the next sequential command is executed. The command IF is valid only within a ROSPROC.

To any of the three formats of the command shown above, an in-line reply may be appended. The rules for the entry of the reply are the same as explained in Section 8.2 for the DO command.

8.11 THE SKIP COMMAND

The SKIP command has the following formats:

SKIP n	Skips n lines in the current ROSPROC.
SKIP	No operation; the next sequential command is executed.
SKIP n,condition	Skip n lines in the current ROSPROC if the condition specified is true; if not, continue in sequence.

The SKIP command is used to conditionally or unconditionally alter the sequential execution of commands in a ROSPROC. The value n is a relative line count. The command "SKIP 3" means to bypass the next three lines in the procedure and resume execution. The SKIP command can be used to effect a conditional transfer of control by specifying HIGH, LOW, EQUAL, or UNEQUAL in the condition field. The condition indicator is tested and if the condition is true, n lines are skipped; if it is not true, the next sequential command in the ROSPROC is executed. The SKIP command is valid only within a ROSPROC.

8.12 THE CALL COMMAND

The CALL command has the following formats:

CALL dsn	Enables one ROSPROC to link to another ROSPROC.
CALL dsn,m	Enables one ROSPROC to link to another, beginning execution at line m.
CALL dsn,m,n	Enables one ROSPROC to link to another, beginning at line m and ending at line n.

The CALL command permits one ROSCOE procedure to call another; the calling procedure is termed a first-level ROSPROC, and the called procedure is termed a second-level ROSPROC. When a CALL command is performed, control is transferred to the first line, or line m of the ROSPROC named dsn. The called procedure is executed until the last command or line n is reached, or until an EXIT command is executed; control then passes back to the first-level ROSPROC at the line following the CALL command. The CALL command is valid only within a ROSPROC.

A CALLed procedure may issue DO or IF commands; the effect is to re-enter procedure state at the first level. A CALLed procedure may CALL another ROSPROC, which in turn may CALL a third ROSPROC. Control at EXIT or end-of-procedure on the CALL level will return to the original first-level procedure which was entered via a DO or an IF.

An in-line reply may be appended to any of the three formats of the command shown above. The rules for the entry of the reply are the same as explained in Section 8.2 for the DO command.

8.13 DYNAMIC MODIFICATION OF ROSPROCs

A ROSPROC command can be altered, just before it is executed, by data in the reply buffer. This facility enables a user to prepare a generalized procedure which, in effect, asks for and inserts its own variables at execution time.

To use this feature, the user must determine the commands and/or operands in his ROSPROC that are to be provided with variables during execution. The lines affected contain two delimiters; the first delimiter (any special character) must be in data position one to indicate dynamic modification, while the second delimiter marks the position at which variable data entered from a terminal is to be placed within the line. When ROSCOE interprets a ROSPROC line beginning with a delimiter, it takes the contents of the reply buffer (see REPLY) and places it in the command at the point designated by the second delimiter. Interpretation and execution of the command then proceed normally. This activity takes place within ROSCOE's interpretation phase, and the ROSPROC in the user library remains unchanged.

For example, assume a procedure is prepared to manipulate a data set, and the data set name is to be supplied at procedure execution time. The following commands (as they appear in the user library) might be included:

```
TYPE ENTER DATA SET NAME

REPLY

*FETCH*
```

"ENTER DATA SET NAME" is displayed at the terminal and the terminal unlocks, awaiting input. If the user enters "JOE", the ROSPROC resumes and executes the command "FETCH JOE", and the data set named JOE is loaded into the AWS.

Another example is:

```
REPLY

*TYPE MY* IS YELLOW

*DELETE*
```

If the response at the terminal is BASKET, the message "MY BASKET IS YELLOW" appears at the terminal and the data set named BASKET is deleted from the user library. Examples F1—F3 in Appendix E illustrate symbolic replacement.

8.14 THE ROSCOE COUNTER

Commands in a ROSPROC can reference an arithmetic counter to facilitate iteration and simple computations. The COUNTER has a value range of 0000 to 9999. It can be set to an initial value, increased, decreased, and tested. The counter is modulo-9999; that is, if incremented beyond 9999 or below 0, it will wrap around.

The COUNTER is indirectly addressable in a command by referring to it as 0. The command "LIST 0" directs ROSCOE to take the current value of COUNTER and use it as the operand field of the LIST command. If the COUNTER contained "10", line 10 in the AWS would be displayed at the terminal. Similarly, the command "REPLY 0" will result in line 10 of the AWS being placed in the reply buffer.

The value of COUNTER can be tested using the COMPARE command. The operand field of the COMPARE command must be numeric without delimiters. The command "COMPARE 10" compares the value of 10 to the current value of COUNTER. The condition indicator is set to HIGH, LOW, EQUAL, or UNEQUAL, as the case may be, and the standard IF or SKIP commands are used to effect a conditional transfer of control. The COUNTER is RESET at sign-on time, and no reference to it is valid before it is explicitly SET by the user. The COUNTER may be stored in a location called the counter store, restored from that location, and values may be interchanged between the two locations. The counter store can also be used as an index value for incrementing and decrementing the COUNTER.

The counter commands are SET, INCR, DECR, STORE, LOAD, and SWAP.

8.14.1 THE SET COMMAND

The SET command has the following formats:

SET n	Sets COUNTER to the value n.
SET	Places the current value of COUNTER in the reply buffer. The value is converted to a four-digit field and placed left-justified in the reply buffer.
SET LENGTH	Places length of current reply buffer contents into the ROSCOE counter. If the reply buffer is empty, a length of 0 is returned.

8.14.2 THE INCR COMMAND

The INCR command has the following format:

INCR n	Adds the value n to the value in COUNTER incrementing COUNTER. If n=0, the command is a no-op.
INCR	Add to COUNTER the current value of the counter store, incrementing COUNTER.

8.14.3 THE DECR COMMAND

The DECR command has the following format:

DECR n	Subtracts the value n from the value in COUNTER decrementing COUNTER. If n=0, the command is a no-op.
DECR	Subtract the value in counter store from COUNTER, decrementing COUNTER.

8.14.4 THE STORE COMMAND

The STORE command has the following format:

STORE	The current value of the COUNTER is stored in the counter store.

8.14.5 THE LOAD COMMAND

The LOAD command has the following format:

LOAD	The current value in the counter store is restored to the COUNTER.

8.14.6 THE SWAP COMMAND

The SWAP command has the following format:

SWAP	The values of the COUNTER and the counter store are interchanged.

8.15 PACKING COMMANDS IN A ROSPROC

Normally, a ROSPROC has as many lines as commands. Each line is a record stored on disk, so that some space and time is wasted with such organization. It is possible to pack more than one command as a single line (as long as the number of characters does not exceed 78) by separating commands with a special delimiter. The delimiter is preset to "&", but may be changed at any time by the command:

```
SET /
```

where / is any special character (neither alphabetic nor numeric) which will serve to delimit packed commands. Care must be exercised that the command delimiter does not duplicate special characters used as delimiters *within* commands (EDIT, REPLY /string/, symbolic replacement indicators).

NOTE: The SKIP command does not bypass commands in a line, but lines in a procedure.

8.16 ERRORS AND FORCED SUSPENSION

If an error of any sort is found while executing a ROSPROC, the procedure is terminated by ROSCOE with a diagnostic message, as follows:

```
ENTRY INVALID
AT LINE nnn, IN PROCEDURE dsn
(Procedure Concluded)
```

If the error is due to an I/O error reading the procedure from the library, ROSCOE will display the message:

```
I/O ERROR READING FROM PROCEDURE
(Procedure Concluded)
```

ROSCOE attempts to guard against endless loops caused by logic errors in the ROSPROC by suspending the procedure when more than 255 lines of a procedure or a set of procedures has been read without addressing the terminal. ROSCOE in this case displays the message:

```
COMMAND LIMIT EXCEEDED IN PROCEDURE – GO OR EXIT
```

The meanings of GO and of EXIT have been explained previously.

Certain possible errors in a ROSPROC are termed user errors, occurring when, for example, a ROSPROC user has entered invalid data, such as an incorrect data set name. In such cases, it is possible to trap the error within the procedure and continue execution through recovery procedures. This facility is described in Sections 8.17 and 8.18.

8.17 THE TRAP COMMAND

The TRAP command has the formats:

TRAP	Enables the TRAP facility
TRAP ON	Enables the TRAP facility
TRAP OFF	Disables the TRAP facility.

The TRAP facility is disabled at sign-on time. When it is set ON, the value of the TRAP code is reset to zero. Any "trappable" error occurring in ROSPROC from this point will place a corresponding value into the TRAP code.

When the TRAP facility is disabled, the value of the TRAP code remains unchanged. Only a TRAP ON command can reset the value in this state. The values of the TRAP code are explained in Section 8.18. At procedure termination, TRAP is automatically reset to OFF.

8.18 THE TEST COMMAND

The TEST command has the format:

```
TEST v
```

where v is a numeric value to be compared with the current value of the TRAP code. The order and result of comparison is identical to the usage of COMPARE, described in Section 8.14. After the ROSCOE condition code has been set by the TEST command, skips and jumps are permitted through the use of the appropriate forms of the IF and SKIP commands. It is the responsibility of the author of the ROSPROC that enabled the TRAP facility to TEST the result of any commands which might generate one of the TRAP codes listed below. The TRAP facility will prevent the normal diagnostic message and termination of procedure; the ROSPROC will continue after the error with no other indication of error than the code setting. (In a case where no TESTs are made, it is likely that the procedure will terminate due to a subsequent error.)

The following list indicates all possible values of the TRAP code:

VALUE	MEANING
0	No errors
1	Either no lines found in range, or no match found in range. NOTE: No error condition is generated if no line is found in range in a ROSPROC being CALLed or DOne.
2	Syntax checker requested has been disabled.
3	Fill string is too long.
4	Library data set not found (unless the dsn is being DOne, CALLed, IFed, or SUBMITted). NOTE: In systems with read protection, trap code 4 can also indicate this data set is protected on read.
5	Duplicate library name (on SAVE).
6	AWS empty; no action taken.
7	AWS full or sequence number overflow in AWS.
8	User library full or nearly full; SAVE or UPDATE cannot complete.
9	Data set protection check (caused by attempting to update or delete a data set belonging to a different user, or attempting to delete, update, or rename a ROSCOE procedure belonging to the terminal user and currently being executed by him).
10	JOB xxxx not found (DISPLAY, GET, CANCEL).
11	Account protection check (user attempted to change password or sign-on procedure when not permitted to do so).
12	Library maximum count exceeded; SAVE or UPDATE not executed.

IX. STORING AND RETRIEVING LIBRARY DATA SETS OFF-LINE

To store data sets on the line libraries outside ROSCOE, the utility ROSDATA is used. To retrieve library data sets outside ROSCOE, the utility ROSCOPY is used. *No other utility should (or can) be used to perform these functions.*

9.1 ROSDATA

The procedure ROSDATA is catalogued in the system procedure library. ROSDATA creates on the ROSCOE Line Libraries one or more ROSCOE data sets from input data taken either from the SYSIN data set (normally 80-column cards) or from any number of auxiliary data files (FROM data sets) defined by the user. Except as noted below in the discussion of the FROM control statement, all input data must consist of 80-character records, blocked to any size, and written in fixed format. However, 81-character records will be accepted from SYSIN and FROM data sets. The first character will be ignored.

9.1.1 ROSDATA OPTIONS

Input data (including FROM data) may be edited in various manners (FILL/CLEAR options); sequence numbers may be taken from the input data (USE option), or new sequence numbers generated by ROSDATA using a user-defined starting sequence number and increment (SEQ1 and INCR options); the output members in ROSLIB01 may be flagged as having SEQ, NOSEQ, or COBOL attributes (SEQ options); auxiliary (secondary) input data sets may be used as input, and (optional) site routines may be named which are to perform preliminary I/O on such data sets (FROM options); multiple input sets may be concatenated to form a single output set (ADD options), or multiple, or a single large, input set(s) may be broken up into a set of separate output sets (MAXSIZE option). ROSCOE site management (system programmers) have a special option whereby they may direct ROSDATA to create output library sets for more than one ROSCOE key within a single execution of ROSDATA (KEY option).

9.1.2 SOURCES OF ROSDATA OPTIONS: PARM FIELD AND CONTROL STATEMENTS

The PARM field of the EXEC card for ROSDATA must contain the user key. For example:

```
//RSD EXEC ROSDATA,PARM='37924.PARTRIDGE'
```

All other control information for ROSDATA is supplied in the SYSIN data.

The SYSIN control statements have the general format:

```
$control operands
```

where $ (in the U.K., a £ sign) must appear in column 1 of the input record, the control word must follow it without any spaces, and one or more operands are coded following one or more spaces. Continuations, where permitted, are indicated by placing a comma after the last operand on the card, and continuing the operands on the following cards preceded by at least one space. Operands may be coded on a card through column 71; end of operands on a card is indicated by one space or more, and comments may be placed to the right of that space ad lib. The dollar sign ($) is reserved for control statements and no input data may have the sign in its first column. (Data input from FROM data sets is not subject to this restriction, as indicated below.)

There are four controls: ADD, FROM, FILL, and CLEAR. The ADD control is required; the other three are optional. The syntax, meaning, and effects of these controls are described below:

A. $\text{\$ADD LIST=}\left[\begin{array}{l}\underline{\text{ALL}}\\\text{NONE}\\\text{CONTROL}\end{array}\right]\text{,KEY=userkey,}\left[\begin{array}{l}\text{MEMBER=name}\\\text{NAME=name}\end{array}\right]\text{,MAXSIZE=}\left[\begin{array}{l}\underline{\text{1000}}\\\text{defval}\\\text{OFF}\\\text{EOF}\end{array}\right]$

$\text{SEQ1=}\left[\begin{array}{l}\underline{\text{10}}\\\text{val}\end{array}\right]\text{,INCR=}\left[\begin{array}{l}\underline{\text{10}}\\\text{ival}\end{array}\right]\text{,COBOL,}\left[\begin{array}{l}\underline{\text{SEQ}}\\\text{NOSEQ}\end{array}\right]\text{,USE=start,length}$

When an $ADD card is encountered, all data processed to that point will be saved on ROSLIB01 according to the options in effect; all variable options will then be reset to their default values (indicated by underlinings), to be altered as indicated by the operands of the $ADD control. The default values shown here are actually site variables, and may differ at installations; check with site management for the actual defaults in effect at the installation. The operands are:

- LIST=
  - ALL: All input, both data and control statements, will be listed on the high-speed printer. The data records will be listed after editing specified in the SEQ, CLEAR, and FILL options has been performed.
  - NONE: No input data or control statements will be listed. However, error diagnostics will be listed, if generated.
  - CONTROL: Only input control statements and possible diagnostics will be listed.

  The listing is the only way you have to verify ROSDATA processing before terminal listing. If sequence numbers have been generated by ROSDATA, it is the only way to discover the sequence numbers before terminal listing.

- KEY= This option is available only to system programmers. The method of bypassing the protection features and using it are described in the *System Reference Manual.* If this option is encountered where the protections have not been disabled, it is illegal and will cause termination of ROSDATA processing.

- MEMBER=name
  NAME=name NAME and MEMBER are synonyms, provided for user convenience. The name supplied will be prefixed by the user's ROSCOE prefix to form the member name of the data set placed by ROSDATA on the ROSCOE library; therefore, the name as supplied here may not be longer than 8 characters, and must not begin with a numeric character (0 through 9). If input segmentation is in effect (see below, MAXSIZE), the first 6 characters of the name are used as a base to be suffixed with the segmentation index, as described below.

NOTE: The name SAVAWS is a reserved name, as it is used for on-line data-save forced by terminal errors, and will be rejected if coded here.

• MAXSIZE=	1000	The default value is normally either the capacity of the on-line AWS, or half that capacity, as determined by site management. If the default value is chosen, MAXSIZE may be omitted.
	defval	This is an unsigned number to be used as the segmentation break.
	OFF	No segmentation at all is to be performed. If a sequence number overflow occurs (sequence number greater than 9999 is generated), it is considered an error, and ROSDATA will terminate.
	EOF	Segmentation is to occur at every end-of-file condition on the auxiliary input data sets (FROM data sets).

NOTE: When the segmentation break is reached, all records read to that point are saved, and ROSDATA begins creating a new output set. The first sequence number of the second (third, etc.) set will be set back to the SEQ1 option, unless the USE option is taken.

• SEQ=10	10 is the standard default option. Otherwise 'val' (an unsigned number) may be provided: ROSDATA will begin generation of sequence numbers starting with 0010 or 'val'.
• INCR=10	10 is the standard default option. ROSDATA when generating sequence numbers will use 10 or the user-provided 'ival' as the increment.
• SEQ	The output data saved on ROSLIB01 will be assigned the data attribute of SEQ. SEQ and NOSEQ are mutually exclusive parameters.
• NOSEQ	The output data saved on ROSLIB01 will be assigned the data attribute of NOSEQ. SEQ and NOSEQ are mutually exclusive parameters.
• COBOL	The output data saved on ROSLIB04 will be flagged as being a COBOL source program. This feature is at present unsupported in ROSCOE itself, and is provided for planning ahead. Please note that if COBOL is coded here, no assumption whatsoever is made as to the presence or location of sequence numbers in the input data records.
• USE=	ROSDATA is not to generate new sequence numbers while saving the data, but to accept and use sequence numbers already contained in the input data starting at column 'start' for the 'length' defined (normally, this will be "73,8"). ROSDATA will perform a check of the old sequence numbers: the check will fail and cause termination of processing if the field defined does not contain a valid numeric field; if the old sequence numbers are not in ascending order; or if a sequence number higher than 9999 is encountered. The USE subparameter is mutually exclusive with SEQ1 and INCR, and also with any segmentation except MAXSIZE=OFF or MAXSIZE=EOF.

NOTE: When segmentation is enabled, the names of the output data saved on ROSLIB01 will be formed as follows: the name provided by the user is prefixed with his prefix; the first 6 characters of the provided name (or less if the name is less than 6 characters) are then taken as the base name. To this base name are suffixed successive indices to form the member name on ROSLIB01. The indices are numeric segmentation identifications, from 00 through 36. For example, if the segmentation break is at 100 lines (output records), base name is STREAM, and the user prefix is HO, an input set of 550 records would be saved as 6 output members, named HOSTREA00, HOSTREA01, HOSTREA02, HOSTREA03, HOSTREA04, HOSTREA05. Each set would contain exactly 100 lines, except the last which would contain 50. If segmentation requests cause saving of more than 36 output sets from the same base name, ROSDATA will terminate.

B. $CLEAR s1,x1,s2,x2,....sn,xn

Up to 10 fields in the input data may be specified in a single CLEAR control statement. The list may be enclosed in parentheses, ad lib. The fields which start at column s in the input record, will be cleared to all spaces for a length of x character. s+x may not be greater than 81. No check is made for overlapping fields. CLEAR operations are performed immediately after sequencing operations, and immediately before FILL operations (see below). It is customary when specifying retention of old sequence numbers (the USE option in the ADD statement) to CLEAR the old sequence field; the sequence numbers in ROSCOE library or AWS data are not maintained within the 80-column record, as explained elsewhere in this manual. Therefore, old sequence number fields would be treated by ROSCOE simply as data fields.

The use of a CLEAR statement with no operands will disable all CLEAR definitions in effect up to then. The default is no CLEAR operations. As many CLEAR statements as needed may be inserted within the SYSIN stream. Each such statement will disable the effects of any prior statement and will place into effect new CLEAR definitions for all data following the control statement. No comments may be placed on a $CLEAR which has no operands.

C. $FILL *string*,s

The string between the special character delimiters ("*") will be forced into every input record beginning at column s of the record. The length of the string between special characters defines the length of the fill operation. See the description of the EDIT terminal command elsewhere in this manual for an explanation of the use of the special character delimiter. The string may not spill over column 80 of the source record. (The length of string plus starting location, may not exceed 81.) Any number of FILL fields may be defined by using continuation cards for every subsequent string. No more than 1 string may be defined per card. A FILL control statement with no operands will disable all prior FILL definitions, as for CLEAR above. No comments may be placed on a $FILL with no operands. The following is an example of continuation of a FILL statement to define more than one field:

```
$FILL &JAM YESTERDAY, &,12,                                100
      ¢JAM TOMORROW, ¢,28,                                 200
      @BUT NEVER JAM TODAY          @,47                   300
```

Statement 100 initiates new FILL definitions; statements 200 and 300 are continuations of the same FILL control statement. Three fields are defined (starting at input columns 12, 28, and 47), into which the strings between the special character delimiters will be forced.

D. $FROM $\begin{bmatrix}\text{ddname} \\ \text{ddname (member)}\end{bmatrix}$,usermod

ROSDATA is instructed to suspend input operations from the SYSIN (RSDIN) input stream, and to begin reading input records from the data set defined in the ROSDATA JCL by the DDNAME declared here. If the data set referenced is a partitioned data set, the member name to be retrieved must be supplied within parentheses on the FROM control statement (not on the DD card). If the data set referenced is in a format which cannot be read by ROSDATA (e.g., a LIBRARIAN master file), the name of an I/O routine written by your site's system programmer may be supplied. Conventions for using this optional usermod are given in the *System Reference Manual,* and users will be informed of their availability by site management. When the end-of-file is reached in the FROM data set, ROSDATA will resume reading the SYSIN stream for the next records.

NOTE: ROSDATA control statements encountered in the FROM data will be ignored and treated as input data records.

9.1.3 ERROR CONDITIONS FROM ROSDATA

If errors are encountered during execution of ROSDATA, a diagnostic will be written to the printer listing, and ROSDATA will terminate. The diagnostic is a self-explanatory one-line explanation of the error. If the error is caused by invalid control statements or missing JCL, the user may correct and resubmit. If the error was an I/O error on any of the data sets accessed by ROSDATA, the job may be retried, but if the error condition persists, give all listings to the site management (system programmer). Certain error conditions which are not considered serious but which cause termination of ROSDATA permit creation on ROSLIB01 of a member under the name assigned by the user, containing not the user data but the same diagnostic which appears on the printer listing.

9.1.4 SAMPLE JCL AND CONTROL STATEMENTS FOR ROSDATA

A. STEPLIB defines the ROSCOE executing library from which ROSDATA itself is fetched. If a user I/O processor is used, and it is on neither the ROSCOE executing library nor SYS1.LINKLIB, a DD statement defining its library may be concatenated to the STEPLIB. For example, SITE.LOADLIB contains such a processor. In that case, the user places in his JCL the following override:

```
//STEPLIB  DD  DSN=ROSCOE.ROSLIB,DISP=SHR
//         DD  DSN=SITE.LOADLIB,DISP=SHR
```

B. The ROSCOE Line Libraries are designated by DD names in the form ROSLIBnn.

C. SYSACCDS defines the user account file on which the user ROSCOE key is to be found.

D. SYSPRINT defines the print file used to list the new data sets, control cards, and diagnostics. The format of this file is variable length records, maximum record length 120, using machine control characters for the printer. A blocksize of 3350 is set by ROSDATA if no block size has been coded on the DD statement. Otherwise, the user may block this file ad lib. The block size must be at least 4 bytes greater than the record size (viz., 124). An example of coding a block size is as follows:

```
//SYSPRINT DD SYSOUT=A,DCB=(RECFM=VBM,BLKSIZE=804)
```

E. The main input data is identified in the JCL by a SYSIN card in the form:

```
//SYSIN  DD  (data identification)
```

The data identification follows the normal JCL conventions of OS; that is, DD * or DD DATA for card input, or fully identifying parameters for tape or disk sets.

Example 1: Mixed input, multiple output with editing

```
//STEPB EXEC ROSDATA,KEY='999C.CASPY.A',REGION=64K          0100
//FROMDD1   DD   DSN=CASPY.SRCELIB,DISP=SHR                 0110
//FROMDD2   DD   UNIT=TAPE,VOL=SER=TAPEIN,DISP=OLD,         0120
//               LABEL=(2,SL),DSN=CASPY.BACKUP.SOURCE       0130
//MASTER    DD   DSN=LIBR.MASTER,DISP=OLD                   0140
//SYSIN     DD   DATA                                       0150
$ADD NAME=PROG1,MAXSIZE=OFF                                 0160
.........source statements for PROG1                        0170
...............                                             0180
.......................                                     0190
$ADD MEMBER=SOURCE,USE=73,8,MAXSIZE=EOF                     0200
$CLEAR 73,8                                                 0210
$FROM   FROMDD1(IEHVTOC)                                    0220
$FROM   FROMDD2                                             0230
$FROM   MASTER(IAJSORT),ROSDFAIR                            0240
$ADD MEMBER=ZZ23A,SEQ1=100,INCR=5,MAXSIZE=500,NOSEQ         0250
$FILL    *ZZ23A *,1                                         0260
......input cards......                                     0270
$FROM   FROMDD1                                             0500
.....more input cards                                       0510
/*                                                          1000
```

Explanation of Example 1 by line number:

LINES	EFFECT
100	ROSDATA is invoked, and a user key is provided; a REGION size is also defined for MVT.
110	A FROM data set is defined; it is a partitioned data set which contains source programs to be transferred to ROSCOE, and it is catalogued.
120	Another FROM data set is defined; it is a tape set, sequential, and is the second file on the tape.
140	A third FROM data set is defined; it is a LIBRARIAN master file, and will be read by an installation I/O routine.
150	The input stream follows this card.
160	The first $ADD card is presented to ROSDATA, with instructions to disable segmentation, and to name the output data PROG1. The default sequencing options (SEQ1=10,INCR=10) are chosen, and the output data set will be flagged as SEQ by default.
170-190	The data records for PROG1 are inserted in the SYSIN stream.
200	The new $ADD control causes all the records read in to this point to be saved on Line Library 0 under the name PROG1 (the prefix is not shown), and all options to be reset. New options are now defined; a base name of SOURC is given, old sequence number in columns 73 through 80 are to be used, and segmentation is to be performed when end-of-file is reached on every FROM data set defined.

210	Columns 73 through 80 on every input record are to be cleared with spaces (thus deleting the old sequence numbers after they have been used by ROSDATA).
220	A member named IEHVTOC is to be taken from FROMDD1 (card 110); it will be read in, edited in accord with the defined options, and saved at the end-of-file as SOURC1.
230	The file defined in card 120 (FROMDD2) will be read in, edited, and saved at the end-of-file as SOURC2.
240	The member IAJSORT will be taken from the master file defined in card 140 (MASTER), using the ROSCOE/LIBRARIAN interface to obtain the records and present them to ROSDATA for editing and output. The I/O routine is named ROSDFAIR (refer to Section 9.1.5).
250	A new ADD control causes termination of all prior processing and resetting of all options to defaults. A base name of ZZ23A is defined, and initial starting sequence number of 100 is requested, with an increment of 5, all output sets are to be flagged as NOSEQ, and segmentation breaks are forced after every 500th record of the input.
260	The characters ZZ23A will be forced into columns 1 through 6 of every output record.
270-510	The input records are taken partly from the SYSIN stream, partly from an auxiliary data set.
1000	The end of the input to ROSDATA is signalled to OS by the end-of-file card.

9.1.5 ROSCOE/LIBRARIAN INTERFACE

For ADR customers who have The LIBRARIAN, the module ROSDFAIR is provided to retrieve modules from a LIBRARIAN disk master (tape masters are not handled by this routine). The ddname for The LIBRARIAN disk master file must be MASTER. An example of its use is shown in the preceding section.

The following notes should be carefully studied:

- History records, hash counts, and update indicators are not copied along with the actual source module records. If the module is to be replaced on The LIBRARIAN master file after modifications through ROSCOE, it will be, in effect, an entirely new module with no relation to a prior version.
- Sequence numbers should be cleared by ROSDATA before the records are placed in ROSLIB01, for two reasons: the ROSCOE sequence numbers are now the only valid identification field; and carrying along the superfluous sequence numbers is extremely inefficient in terms of disk storage and terminal time when listing.
- Where a source module is placed in ROSCOE to undergo very extensive modification and editing, the transfer between LIBRARIAN and ROSCOE is justified in terms of convenience and speed in modification. But where the modifications are to be made to a very large source module (number of statements greater than the AWS capacity) or are straightforward deletes, inserts, and replacements, it is extremely inefficient. The proper technique (whether the source

library is LIBRARIAN or another) is to take advantage of both source master and ROSCOE capabilities by making and maintaining in ROSCOE only the changes applied to the source (the transactions upon the base), and applying them to the source on a temporary basis (TEMP option) without updating the source. When the changes have been tested and approved, they can then be applied permanently to the source. This has the advantage of not duplicating data in two places, and of keeping the ROSCOE library set small and easy to handle. Using this technique, a typical job stream submitted from ROSCOE would consist of a LIBRARIAN step to apply temporary changes to the module and to place the new version on the OSJOB file (a PDS), followed by the compiler and linkage steps using that file as input.

9.2 ROSCOPY

The ROSCOPY procedure is catalogued in the procedure library. ROSCOPY is used to retrieve data sets from the line libraries in batch.

Three sets of information are required to execute ROSCOPY.

A. Parm field of EXEC card – your ROSCOE key.

B. A definition of your output (SYSOUT). All information required to create a new set or add to or reuse an existing set, must be coded on the SYSOUT DD card. LRECL always = 80; if BLKSIZE is not indicated on the DD statement or the DSCB of an existing data set, ROSCOPY will force a default of 80. The output may be a sequential data set or a member of a partitioned data set. In the last case, the member name must be given in the JCL. For example:

    DSN=ADR.LIB1(NEWMEMB)

C. A SYSIN data set in card image format contains all control information as to which data sets on the ROSCOE library are to be retrieved, which resequencing options are possible, and which auxiliary data is to be inserted before and after the library input. Control cards have the format:

name	command	operands and subcommands	id or seq field
1			72

The name field is optional, and may be any length. It is followed by at least one space. The command field contains either the COPY or MEMBER command (see below). The operands are separated by at least one space from the command field, and may extend through column 71 of the card image. Continuations from one card image to another are indicated by the comma ending one subcommand on the first card, and the next command starting in column 16 of the continuation card. There may be as many continuation cards as required. Continuations may be indicated between one operand and the next, but not within an operand. The contents of columns 72-80 is ignored. The format of the auxiliary data cards differs from the command cards, and is described below.

- The COPY command is required as the first control card. There may be only one COPY statement. The operands, which may be coded in any order, are:

MAXNAME=n	This operand is the only required operand. n is a number not less than the maximum number of data set names indicated on the MEMBER statement.

RENUMBER=(s,i)	All SEQ and NOSEQ attributes of library data sets will be ignored. Instead, all data sets retrieved will be renumbered sequentially, using s as the starting value and i as the increment. Both s and i cannot be greater than 4 digits.
FIELD=(p,l)	The sequence number field is normally columns 77-80 for library data sets with the SEQ attribute, or for output data being RENUMBERed. The FIELD operand defines a different sequence number field, where p is the starting column number (with 1 as the first character on the output record), and l as the length of the field (maximum of 8 positions, minimum of 2). NOTE: FIELDS is accepted by ROSCOPY as an alternative for FIELD.
NOSEQ	All sequence number operations are suppressed. If NOSEQ is indicated, neither RENUMBER nor FIELD may be.
HASPJOB	For HASP systems only. The SYSOUT data set may be assigned to the HASP internal reader. ROSCOPY will close the internal reader by writing a /*EOF record following all user data. If an error in processing occurs, the job stream will be cancelled by writing a /*DEL record.

- The MEMBER statement (only one per execution) may have any number of continuation cards:

```
MEMBER NAME=(dsn1,etc.......)
```

The name list must be enclosed by parentheses even if there is only one name. The names are fully prefixed library names (no "." between the prefix and the rest of the name). If you need continuation cards, the last member named on the card must be followed by a comma (",") and the next name begins in column 16 of the next card.

- Supplementary input data may be written to the SYSOUT data set from the SYSIN data either preceding the library input or following it. Such data is placed in the SYSIN control set after the MAXNAME command. Data cards to be placed before library data are preceded by the control statement –FRONT (must begin in column 1). Data cards to be placed after library data are preceded by the control statement –END (must begin in column 1). The data cards are transcribed directly to the SYSOUT data set. If a RENUMBER operation has been indicated, the data cards are renumbered also. Data cards which follow a –FRONT or –END control statement and have a dash (–) in column 1 but are not control statements, will be accepted by ROSCOPY as user data. All data cards will be listed on the SYSPRINT message file as they are read from SYSIN.

Sample JCL for ROSCOPY:

```
//COPY JOB (ACCT–)
//A      EXEC ROSCOPY,PARM='1234.JONES.J'
//SYSOUT DD DSN=NAME,VOL=SER=WRK004,
//       UNIT=2314,DISP=(PASS),SPACE=(TRK,(2,1)),
//       DCB=BLKSIZE=400
//SYSIN DD *
         COPY MAXNAME=8,RENUMBER=(100,10),FIELD=(1,8)
         MEMBER NAME=(ALDATA1,D2PGM2,XYZ)
/*
//B      EXEC COBFC
//COB.SYSIN DD DSN=*.A.SYSOUT,DISP=OLD
//
```

Sample JCL for ROSCOPY to create a HASP job stream:

```
//SUBMIT   JOB (accnt),CLASS=E
//MAKEJOB EXEC ROSCOPY,PARM='USER.KEY',REGION=28K
//SYSOUT   DD  UNIT=INTRDR
//SYSIN    DD  DATA
COPYCMND  COPY  HASPJOB,MAXNAME=3
NAMECMND  MEMBER   NAME=(UKCOB1,UKCOB2,UKCOB3)
–FRONT
//COBTEST JOB (accnt),CLASS=E,REGION=128K
//STEP1    EXEC  COBUCLG
//COB.SYSIN DD *
–END
//GO.FILEA  DD  SYSOUT=A
//GO.FILEB  DD  DSN=COBOL.TESTLIB,DISP=SHR
/*
```

The above execution of ROSCOPY will write a job stream out to the HASP internal reader consisting of a JOB card, an EXEC statement for the COBUCLG procedure, a SYSIN card, followed by three members from the ROSCOE library (these members are a complete COBOL source program). The source program is followed by two DD statements for the GO step of the COBOL procedure.

X. DATA RETRIEVAL FROM BATCH: THE POST-PROCESSORS

10.1 DEFINITION

The post-processors (hereafter abbreviated as PP) are ROSCOE's mechanism for bringing back to the terminal user the output of jobs run in batch and listings of data sets existing at his installation. All PP's are *selective* functions, designed to retrieve only information considered relevant by the terminal user. The PP's are programs which run in batch steps in your job stream, and which retrieve output files or old data sets, and pass them to a special form of ROSDATA for storage in ROSCOE.

There are three types of PP's: the language (compiler) PP's; the linkage-edit PP; and the General PP. (The system message retrieval system is described in Section XI.) The language PP's are designed to first examine the output of a compilation or assembly for any diagnostics created by the compiler, and then to format these along with the source lines they reference so that they can be inspected at a terminal. The linkage-edit PP merely transcribes all or part of the printer output of the linkage editor. The General PP accepts arbitrary data for formatting and storage in ROSCOE libraries.

There can be any number of post-processor steps in a job stream. For example, it is possible to run a job consisting of a COBOL compile, to post-process the compilation, perform a linkage edit, post-process that, follow with a GO (execute) step of the module so created, and to post-process (or sample) any output data sets written by the problem program. To facilitate usage of the PP's, a set of PP procedures has been installed in your library. New procedures can be created as desired with a minimum of effort. (Such new procedures should be requested from the installation systems programmer so that the JCL follows the standards of the installation.) The overhead in time and space (region) requirements is very small; each PP is designed to fit in the region or partition used by "its" compiler, etc. The ROSCOE PP-catalogued procedures for languages are modified versions of the standard IBM procedures for these languages. That is, the JCL for a COBOL compile and post-process is precisely the same as the IBM procedure for compiling a COBOL source module, with the insertion of one or more additional steps to handle post-processor functions. The JCL of IBM procedures may be found in PART IV of IBM manual GC28-6703, *Job Control Language User's Guide.* The reference material following in this section of the *ROSCOE User Guide* discusses only the JCL requirements of the inserted PP steps. The JCL requirements for the other steps are unchanged, and can be found discussed in detail in the various IBM *User Guides* for the source languages.

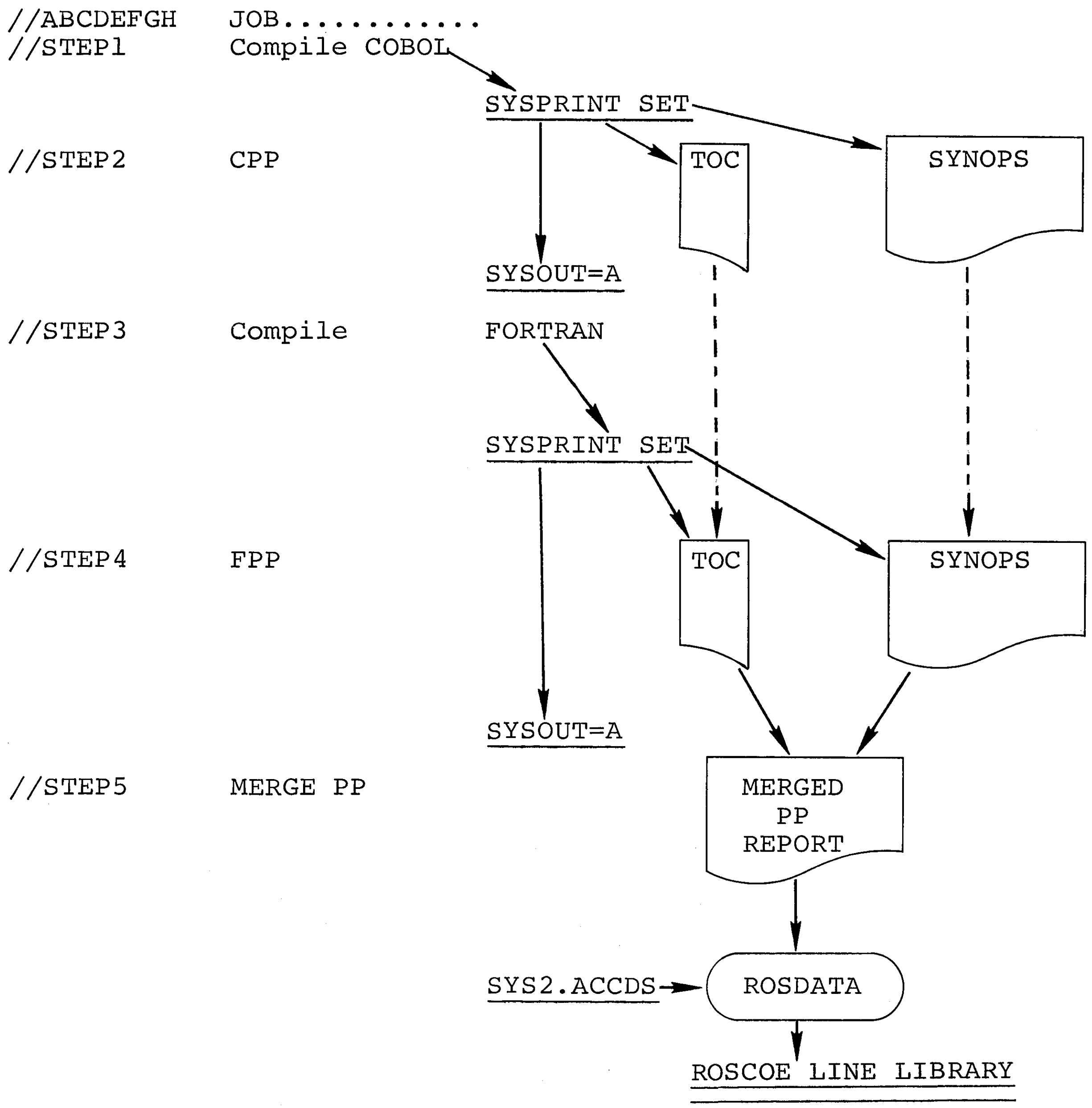
To place the data retrieved by the PP's into the ROSCOE libraries (Line Library 0), a special PP function is required, the PP MERGE, which is a program to be placed at the end of any PP job stream as the final step in the stream. Certain PP-catalogued procedures include a PP MERGE step to reduce the amount of JCL coding required. A PP job stream may have one and only one PP MERGE step in it.

If hard copy of the PP output is desired on the high-speed printer, the utility EDITSIM may be used (the catalogued procedure is also called EDITSIM). It is placed in the job stream after the PP MERGE step.

Example 2 is a logic diagram of a typical PP job stream, showing the steps executed and the flow of the data. As PP's are problem programs, a PP job stream can be submitted either through batch or through ROSCOE RJE.

10.2 PP OUTPUT FORMAT

Before describing the functions and uses of the various PP's, the format of the PP output as placed



EXAMPLE 2. TYPICAL POST-PROCESSOR STREAM

in the ROSCOE library must be described.

A. PP output is placed in the ROSCOE libraries as a named data set. Its name consists of the user's prefix (as for all other named data sets) and the JOBNAME. If a previous PP data set with the same name is encountered on ROSLIB01, when the PP output set is being stored, the new set will automatically replace the old set.

B. The PP data set consists of two parts: A TABLE OF CONTENTS (TOC), and the PP data proper (sometimes referred to as SYNOPS, the synoptic set). The TOC consists of a header line:

```
***POST PROCESSOR TABLE OF CONTENTS***
```

followed by a number of two-line entries. Each of these entries corresponds to a single PP step in the PP job stream, and will contain the job name, the PP step-id, the time and date of run, the line numbers used to list the PP data, and variable information, depending on the PP being used. The variable information will be described below in the discussion of each PP. The second line of each TOC entry is usually reserved for PP diagnostic messages which are recorded when an unusual event occurs during PP processing (I/O errors, etc.), but may contain other information as well.

C. The TOC is displayed on the terminal by using the LIST command in the form:

```
LIST dsn
```

where "dsn" is the name of the PP data set as described just above. When the TOC has been listed, the user can then list the data portion as detailed by the TOC via the command:

```
LIST dsn,m,n
```

D. PP data sets can be FETCHed into the AWS. When they are FETCHed, the TOC is stripped off, all records are truncated to 80 characters of data, and sequence numbers (from 1 in increments of 1) are appended.

The PP data sets can also be deleted with the command:

```
DELETE dsn
```

No other ROSCOE command can address post-processor data. If another command (such as APPEND or CHAIN or SUBMIT) addresses a PP data set, ROSCOE will terminate the command after display of the error message:

```
I/O ERROR READING FROM PROCEDURE
```

Finally, the PP user should be aware that retrieving batch data by the PP's will not cause any loss of data; i.e., the output of the COBOL compiler will be retrieved, but will also be listed on the printer as in a non-PP job stream. There is no limit to the number of such jobs that can be submitted.

10.3 CATALOGUED PROCEDURES FOR THE PP's SUPPLIED BY ADR

The following list identifies all of the ADR-supplied post-processor procedures.

NAME OF PROC	FUNCTIONS
EDITSIM	Produce printer listing of PP data.
PMERG	Store PP data into ROSCOE ROSLIB01.
PFGC	FORTRAN (G) compile and post-process.
PMFGC	Same as preceding, adding the PP MERGE function.
PFGCLG	FORTRAN (G) compile, post-process compilation, link-edit, and go.
PMFGCLG	Same as preceding, adding the PP MERGE function.
PCOBF	COBOL (F) compile and post-process.
PMCOBF	Same as preceding, adding the PP MERGE function.
PCOBU	COBOL ANS compile and post-process.
PMCOBU	Same as preceding, adding the PP MERGE function.
PCOBE	COBOL (E) compile and post-process.
PMCOBE	Same as preceding, adding the PP MERGE function.
PASMFC	Assembler (F) compile and post-process.
PMASMFC	Same as preceding, adding the PP MERGE function.
PASMFCL	Assembler (F) compile post-process, and link-edit.
PPL1LFC	PL/1 compile (with object module output) and post-process.
PMPL1LFC	Same as preceding, adding the PP MERGE function.
PPL1FCLG	PL/1 compile, post-process compilation, link-edit, and execute.
PLINKF	Linkage-editor (F) and post-process.
PGEN	General post-processor.
PMGEN	General post-processor plus PP MERGE step.

10.4 DESCRIPTION OF THE POST-PROCESSOR CATALOGUED PROCEDURES

10.4.1 EDITSIM

Function: Print post-processor output on the high-speed printer.

Input defined by programmer: None

JCL required from programmer: An EXEC card to invoke the program.

Example:

```
//stepname  EXEC  EDITSIM
```

This card must be at the end of the job stream, or following the JCL for a PP MERGE function.

10.4.2 PMERG

Function: Combine and store all post-processor output created during JOB execution in ROSCOE Line Library 0.

Input defined by programmer: On the EXEC card, the programmer's ROSCOE KEY, as a PARM field.

JCL required from programmer: A single EXEC card with a PARM field.

Example:

```
//stepname  EXEC  PMERG,PARM='1234.BONES.B'
```

The step name of the merge step in the procedure is MERGE. This card must be located in the job stream following the last post-processor executing step.

10.4.3 PFGC

Function: Compile a FORTRAN (G) module and search output of the FORTRAN (G) compiler for possible diagnostic messages and/or MAPS; format these for terminal retrieval.

Input defined by programmer: Optional PARM field entries on EXEC card to retrieve maps and assign a user-ID in the TOC. The PARM field entries are:

STEP=stepid

an arbitrary identifier, no longer than 8 characters, which will be placed in the TOC entry for this step. The default step-id is FORTRAN.

MAP

which will cause the post-processor to retrieve and list all compiler-generated maps. The total memory requirements and name of each main routine or subroutine are always retrieved.

NOMAP

No maps will be retrieved. This is the default option.

JCL required from programmer: An EXEC card with PARM fields and a DD statement for input.

Example:

```
//COMPILE EXEC PFGC,PARM.FORT=MAP,PARM.FGPP='MAP,STEP=APPLE'
//FORT.SYSIN DD *
......source deck for the compilation......
/*
//ENDUP EXEC PMERG,PARM.MERGE='99923.NEWTON.I'
//
```

The name of the FORTRAN PP step is FGPP. Each diagnostic found will be listed as a card image of the line in error, followed by a line with a "$" under the error followed by a line or lines with the diagnostic messages.

See Example 3.

```
 ***   POST PROCESSOR TABLE OF CONTENTS   ***

RSFORTWU.FORT (G): LINES 0001-0036:      5 ERRS. 10.40.40   05/04/71

 *** FORTRAN (G) POST PROCESSOR ***

          DIMENSON NAME(10)                                                 0200
           $
** 01)   IEY013I SYNTAX**********************************************************

 2        READ(5,3,END=9999) (NAME(I),I=1,10),RATE                          0210
                                $
** 01)   IEY011I UNDIMENSIONED***************************************************

         WRITE(6,5)(NAME(I),I=1,10),RATE,WRATE,BMRATE,ARATE                 0280
                   $
** 01)   IEY011I UNDIMENSIONED***************************************************

 5       FORMAT(1H ,10A1,4X,'$',F10.2,4X,'$',F12.2,4X,'$',                  0290
                                                  $             $
** 01)   IEY013I SYNTAX****************02)  IEY015I NO END CARD*****************

** IEY022I     UNDEFINED LABEL
   9999

SUBPROGRAMS CALLED
SYMBOL    LOCATION     SYMBOL    LOCATION      SYMBOL    LOCATION
IBCOM#       90        IBERH#       94

SCALAR MAP
SYMBOL    LOCATION     SYMBOL    LOCATION      SYMBOL    LOCATION
WRATE        98        RATE         9C         BMRATE       A0
ARATE        A4

FORMAT STATEMENT MAP
SYMBOL    LOCATION     SYMBOL    LOCATION      SYMBOL    LOCATION
    3        A8

****END OF MODULE  MAIN
```

EXAMPLE 3. FORTRAN POST-PROCESSOR OUTPUT

10.4.4 PMFGC

Function: Compile and post-process a FORTRAN module and store the results of the post-processor in Line Library 0.

Input defined by programmer: Same as preceding (PFGC) procedure with the addition of a PARM field for the MERGE step.

JCL required from programmer: EXEC statement with PARM fields and a DD statement for input.

Example:

```
//COMPILE EXEC PMFGC,PARM.FORT=,PARM.FGPP=,PARM.MERGE='22.VV.BB'
//FORT.SYSIN DD ,*
......source deck......
/*
```

10.4.5 PFGCLG

Function: Compile a FORTRAN module, post-process the compilation, link-edit the object module, execute the program. The use of the procedure is the same as the IBM-supplied procedure FORTGCLG, with the addition of the post-processor step. The step names are the same as for the IBM procedure, and the post-process step is FGPP.

10.4.6 PMFGCLG

Function: Compile a FORTRAN module, post-process the compilation, link-edit the object module, execute the program and store the post-processor output in ROSCOE.

Example:

```
//COMPILE EXEC PMFGCLG,PARM.FORT=MAP,PARM.FGPP='MAP,STEP=COGO'.
//     PARM.MERGE='1113.JONES.J',PARM.LKED=(XREF,MAP)
//FORT.SYSIN DD *
......source deck......
/*
//GO.FT05F001 DD SYSOUT=A
```

10.4.7 PCOBF

Function: Compile a COBOL (F) module and post-process the compilation.

Input defined by programmer: COBOL source deck for compilation, and optional PARM field entries for the PP on the EXEC card.

The entries are:

STEP=stepid

as explained for the PFGC procedure. The default step is

ERRCNT=n

where "n" is the maximum number of compiler-generated diagnostics which are to be retrieved and formatted. This value is preset to 30, and has a maximum user-defined value of 50. Where "n" is exceeded, the TOC will give the tally of all errors encountered, but only the first "n" errors will be listed in the SYNOPS set.

CONT

The context of every source line flagged by the compiler will be listed; that is, the source lines immediately preceding and following the line in error will be listed along with the actual line in error. The diagnostics for these lines will then follow.

Diagnostics for each line in error will appear in ascending order of severity code.

NOCONT

turns off the CONT option. NOCONT is the default option. The source line is reproduced exactly as found, and the diagnostic is placed immediately below it. The number to the extreme left of the diagnostic is the compiler-generated source statement reference. Global diagnostics (reference number blank or zero) are re-numbered as 9999 and placed at the *end* of the entire listing.

NOTE: In certain rare cases (for example, the programmer has omitted the IDENTIFICATION DIVISION statement), the post-processor will be unable to retrieve the source lines in error, and will format only the compiler-generated diagnostics.

JCL required from programmer: An EXEC card with PARM entries and a DD for input.

Example:

```
//COBPGM  EXEC  PCOBF,PARM.COBPP='CONT,ERRCNT=35'
//COB.SYSIN  DD  *
......COBOL source deck......
/*
```

See Example 4.

```
list rscobc
  ***   POST PROCESSOR TABLE OF CONTENTS    ***

 RSCOBCWW.CONT    : LINES 0001-0020;0006 ERRORS. 00.26.08   04/15/71

list rscobc,1,20

  ***   COBOL (F) COMPILER: FIRST   6 DIAGNOSTICS.   ***

       ENVIRONMENT DIVISION.                                              0070
       CONFIGGURATION SECTION.                                            0080
       DATA DIVISION.                                                     0090
** 0004 IEQ1004I E  INVALID WORD CONFIGGURATION. SKIPPING TO NEXT RECOGNIZABLE
                    WORD.
** 0004 IEQ1087I W  'CONFIGGURATION' SHOULD NOT BEGIN A-MARGIN.

       PROCEDURE DIVISION.                                                0100
           IF NONAME LESS THAN NOCOMPARE,                                 0110
               GO TO NOPROC                                               0120
** 0007 IEQ3001I E  NONAME NOT DEFINED. TEST DISCARDED.
** 0007 IEQ3001I E  NOCOMPARE NOT DEFINED.

** 0008 IEQ3001I E  NOPROC NOT DEFINED. STATEMENT DISCARDED.

           ELSE GO TO BEGIN.                                              0130
           STOP RUN.                                                      0140
** 0009 IEQ3001I E  BEGIN NOT DEFINED. STATEMENT DISCARDED.
-
```

EXAMPLE 4A. COBOL POST-PROCESSOR OUTPUT USING "CONT" OPTION

```
list rscobf

  ***   POST PROCESSOR TABLE OF CONTENTS    ***

 RSCOBFWW.COBMOD: LINES 0001-0016;0006 ERRORS. 14.34.13   04/15/71

list rscobf,1,16

  ***   COBOL (F) COMPILER: FIRST   6 DIAGNOSTICS.   ***

       CONFIGGURATION SECTION.
** 0004 IEQ1004I E  INVALID WORD CONFIGGURATION. SKIPPING TO NEXT RECOGNIZABLE
                    WORD.
** 0004 IEQ1087I W  'CONFIGGURATION' SHOULD NOT BEGIN A-MARGIN.

           IF NONAME LESS THAN NOCOMPARE,
** 0007 IEQ3001I E  NONAME NOT DEFINED. TEST DISCARDED.
** 0007 IEQ3001I E  NOCOMPARE NOT DEFINED.
               GO TO NOPROC
** 0008 IEQ3001I E  NOPROC NOT DEFINED. STATEMENT DISCARDED.

           ELSE GO TO BEGIN.
** 0009 IEQ3001I E  BEGIN NOT DEFINED. STATEMENT DISCARDED.
-
```

EXAMPLE 4B. COBOL POST-PROCESSOR OUTPUT WITHOUT "CONT" OPTION

10.4.8 PMCOBF

Function: Compile a COBOL source module and post-process the compilation; store the results in ROSCOE.

Input defined by programmer: Same as preceding procedure with addition of a PARM field for the MERGE step.

JCL required from programmer: EXEC statement and DD for input.

Example:

```
//COMPILE EXEC PMCOBF,PARM.MERGE='1234.USER03'
//COB.SYSIN DD *
......source deck......
/*
```

10.4.9 PCOBU/PMCOBU

These procedures are the same as for PCOBF and PMCOBF, but are used for the ANS-level of the COBOL compiler.

NOTE: The post-processor cannot handle batched (i.e., end-to-end) compilations; if you require batched compilations, follow the procedure described below in Section 10.5.

10.4.10 PCOBE/PMCOBE

These procedures are the same as for PCOBF and PMCOBF, but are to be used for the E-level of the COBOL compiler.

10.4.11 PASMFC

Function: Assemble (level F) an ALP program and post-process the assembly. NOTE: The assembler post-processor works for both F and G levels of the assembler, and also for the VS assembler.

Input defined by programmer: Optional PARM field entries on the EXEC card.

JCL required from programmer: Optional PARM field entry on the EXEC card. The PARM field is:

```
STEP=stepid
```

as for all other post-processors. The name of the post-processor step in this procedure is ALPP. Note that when an error in a macro expansion is flagged, the assembler PP will list the original macro call, and the statement of the expansion flagged.

Example:

```
//ASMBLR EXEC PASMFC,PARM.ALPP='STEP=MOD1'
//ASM.SYSIN DD *
......source deck......
/*
```

See Example 5.

```
list rsasmb

  ***   POST PROCESSOR TABLE OF CONTENTS   ***

 RSASMBWW.ASM     : LINES 0001-0016:   4 ERRORS. 12.13.37   05/04/71

list rsasmb,1,16
   *** ASSEMBLER POST PROCESSOR ***

         NOP                                                          0080
** IEU039      NEAR OPERAND COLUMN   1--INVALID DELIMITER

         PUT    SYSPRINT                                              0090
** IEU024      NEAR OPERAND COLUMN   3--UNDEFINED SYMBOL

         LC     4,3                NON EXISTENT OP CODE               0100
** IEU088      UNDEFINED OPERATION CODE

$        THE DOLLAR SIGN SHOULD HAVE BEEN A STAR (*)                  0110
** IEU088      UNDEFINED OPERATION CODE

    4 STATEMENTS FLAGGED IN THIS ASSEMBLY
   12 WAS HIGHEST SEVERITY CODE

_
```

EXAMPLE 5. ASSEMBLER POST-PROCESSOR OUTPUT

10.4.12 PASMFCL

Function: Same as preceding, with a linkage-edit step.

10.4.13 PMASMFC

Function: Same as PASMFC, with addition of the MERGE step.

Input defined by programmer: Same as PASMFC with addition of a PARM field for MERGE step.

JCL required from programmer: EXEC statement and DD for input.

Example:

```
//ASMBL  EXEC  PMASMFC,PARM.MERGE='1234.USER02'
//ASM.SYSIN  DD  *
......source deck......
/*
```

10.4.14 PPL1LFC

Function: Compile a PL/1 module, with object module output; post-process the compilation.

Input defined by programmer: Optional PARM field entries on the EXEC card. The PARM field can contain:

    STEP=stepid

as for all post-processors.

    WARN

All compiler-generated WARNing messages will be retrieved. (WARN is the default option.)

    NOWARN

Compiler-generated WARNing messages will be bypassed.

    ERRCNT=n

where n is the maximum of compiler-generated diagnostics the post-processor is to retrieve and format. The value of this limit is preset to 50.

JCL required from programmer: EXEC statement and DD for input.

Example:

```
//XYZ   JOB   (ACCT...)
//ONE   EXEC PPL1LFC,PARM.PL1PP=NOWARN
//PL1L.SYSIN  DD  *
        /***SOURCE DECK***/
/*
```

An error in the PARM field will not terminate the run. Preset options will be forced.

The output of the post-processor is the same as for the other language post-processors: an error tally in the table of contents entry, and in the data portion the compiler diagnostics, preceded by the statement in error. Warning diagnostics which do not reference a specific statement will be listed together at the end. Diagnostics for each statement are listed in order of severity. Where a PL/1 statement extends over a single line (80-column card image), the *first* and *last* lines of the source statement will be listed to define the extent of the statement as defined by the compiler. It is, therefore, possible that the error flagged is somewhere in the middle of the statement and therefore not listed.

See Example 6.

```
list rspllw
  ***   POST PROCESSOR TABLE OF CONTENTS   ***

 RSPL1WW .  PL/I  : LINES 0001-0033:    13 ERRS. 16.21.47   05/04/71

list rspliw,1,33

   RSPLIW DATA SET NOT FOUND

-


list rspllw,1,33


         ON ENDFILE (PAYMAST) GO TO EOJ;                                      0650

**IEM0687I GO TO EOJ TRANSFERS CONTROL ILLEGALLY TO ANOTHER BLOCK OR GROUP.
           EXECUTION ERRORS MAY OCCUR.

         PUT FILE (SYSPRINT) EDIT (IDATE,(25)'X')                             0790
                                                    END;                      0800

**IEM0185I OPTION IN GET/PUT IS INVALID AND HAS BEEN DELETED.
**IEM0163I FORMAT LIST MISSING, (A) INSERTED
**IEM0182I TEXT BEGINNING 'END' SKIPPED

         PUT FILE (SYSPRINT) EDIT                                             0960
        ***********
         PUT FILE (SYSPRINT) SKIP(3);                                         0990

**IEM0185I OPTION IN GET/PUT IS INVALID AND HAS BEEN DELETED.
**IEM0182I TEXT BEGINNING 'PUT ' SKIPPED
**IEM1835I INVALID FILE OPTION IGNORED

               FILE (SYSPRINT);                                               1090

**IEM0725I HAS BEEN DELETED DUE TO A SEVERE ERROR NOTED ELSEWHERE.
**IEM0673I INVALID USE OF FUNCTION NAME ON LEFT HAND SIDE OF EQUAL SYMBOL, OR IN
           REPLY KEYTO OR STRING OPTION
**IEM0031I OPERAND MISSING .   DUMMY OPERAND INSERTED.
**IEM0080I EQUAL SYMBOL HAS BEEN INSERTED IN ASSIGNMENT

**IEM0764I ONE OR MORE FIXED BINARY ITEMS OF PRECISION 15 OR LESS HAVE BEEN
           GIVEN HALFWORD STORAGE.

**IEM1790I DATA CONVERSIONS WILL BE DONE BY SUBROUTINE CALL
-
```

EXAMPLE 6. PL/1 POST-PROCESSOR OUTPUT

Restrictions on the Use of the PL/1 Post-Processor

1. End-to-end compilations (multiple compilations) cannot be processed in the current version.
2. The procedure options label (MAIN) must be at least two characters in length.
3. Beware of using the compiler option SOURCE=NO.
4. The PL/1 post-processor described in this section operates only for PL/1 F. Other levels can be handled by a program named UMPOSTP, which site management can install in the ROSCOE system.

10.4.15 PMPL1LFC

Function: Same as preceding, with addition of MERGE step.

Example:

```
//XYZ   JOB   (ACCT....)
//TWO    EXEC PMPL1LFC,PARM.PL1PP='STEP=COMPILE',
//       PARM.MERGE='YOUR.ROSCOE.KEY'
//PL1L.SYSIN DD *
     –
     –
     –
/*
```

10.4.16 PPL1FCLG

Function: Compile a PL/1 module, post-process the compilation, link-edit the object module, execute.

Example:

```
//XYZ  JOB  (ACCT...)
//THREE EXEC PPL1FCLG,
//PARM.PL1PP='NOWARN,ERRCNT=10,STEP=PL1L392',
//  PARM.LKED='LIST,NCAL,XREF'
//PL1L.SYSIN DD  *
     –
     –
/*
//LKED.SYSLMOD DD DSN=USER.LOADLIB(PL1L392),
//     DISP=OLD
//GO.SYSUDUMP DD SYSOUT=A
//GO.INFILE      DD ----------
//GO.OUTFILE     DD ---------
//* the following step merges the post-processor
//* results
//ENDUP EXEC PMERG,PARM='USER.ROSCOE.KEY'
//
```

10.4.17 PLINKF

Function: Link-edit, post-process linkage-editor listing.

Input defined by programmer: Link-edit step (step-id LKED): SYSLIN control cards; SYSLIB information; user LIBRARY information; PARM field for linkage-editor (set to LIST,MAP).

Post-processor step: Optional PARM field. Stepname of PP step is LKEDPP. The optional PARM field is:

STEP=stepid

as for other PP's.

MAP

All maps and cross-reference listings generated by the linkage-editor will be retrieved for listing at the terminal.

NOMAP

No maps, etc., will be retrieved; only the diagnostic messages will be retrieved. Map is the preset option.

JCL required from programmer: (LKED-step): DD cards for SYSLIN, SYSLIB, user LIBRARY, if needed.

Example:

```
//LINK EXEC PLINKF,PARM.LKED='OL,XREF,LIST,MAP,NCAL',
//     PARM.LKEDPP='MAP,STEP=stepid'
//LKED.SYSLIB DD DSN=SYS1.FORTLIB,DISP=SHR
//LKED.LIBA DD DSN=USER.MODLIB,DISP=SHR
//LKED.SYSLMOD DD DSN=USER.EXECLIB,DISP=OLD
//LKED.SYSLIN DD *
  INCLUDE LIBA(MODA,MODB,MODC)
  ENTRY MAINX
  ALIAS ACCTRUN
  NAME ACCTSCO2(R)
/*
//ENDUP EXEC PMERG,PARM='2235.CASEY.J'
//
```

NOTE: To insert the linkage-editor post-processor into larger routines, such as compiles and links, request a new procedure from your system programmer. The PLINKF procedure is useful for a linkage step which links the output of previous steps in the job with already existing modules.

10.4.18 DESCRIPTION OF THE GENERAL POST-PROCESSOR

Due to the extensive capabilities and complexity of this post-processor, we are first presenting a description of its use and abilities before the procedures for using it. The General post-processor can retrieve and format for terminal display any kind of data set (subject to restrictions described below) existing in the computer system. The input data set can be either a pre-existing set (so that the General post-processor can constitute a job stream in itself), or a data set created by a preceding execution step within the job stream. Any number of data sets can be processed; there will then be as many General post-processor steps as there are data sets to be retrieved. The GPP performs two kinds of selection and editing: automatic formatting, and user-directed formatting and selection. The automatic formatting consists of removing all non-printable characters from the data, so that transmission of the data to the terminal will not be interrupted by illegal binary codes: line-length formatting, the most convenient line length for the data to be displayed, is chosen so that data consisting of records from 1 to 120 characters long will be displayed full width of the terminal; print-records with a record length of 132 will also be displayed full length; records of length 81, 121, and 133 will have the first character truncated, on the assumption that the first character is a punch or printer control character, and therefore not part of the data itself. Records which do not fall into the categories named above in respect to length will be printed at the terminal as a series of 120-byte records. (NOTE: On devices which do not permit listing full length, the records will be broken at the end of the screen or carriage and continued on the next line.)

Finally, the General post-processor acts as a selective file sampler; it does not retrieve the entire input data set (except under certain conditions), but *samples* the file by retrieving a pre-defined number of records from the beginning of the file, and as many again from the end of the file. The first and last records will always be retrieved. In this way, it allows verification of very large files which do not permit complete visual inspection because of length.

The user-directed formatting and selection is controlled completely by the PARM field entries, which are:

STEP=stepid

as for all other post-processors, to assign a user id to the TOC entry for listing.

NREC=nn

where "nn" is the total number of records in the input set which the General post-processor is to format for display at the terminal. NREC is preset to 100, and has a maximum value of 1000. If a value greater than 1000 is coded, the preset option will be used. If NREC is greater than the total records existing in the input, the entire file will be retrieved. Otherwise, the first nn/2 records from the beginning of the file, and the last nn/2 records at the end of the file will be retrieved.

SKIP=mm

where "mm" represents the number of records in the input data set which are to be skipped or bypassed between every record which is accepted for processing. The combination of this and the preceding parameter permits random file-sampling. The value of SKIP is preset to 0 (skip no records).

NOPRINT

This is the preset option; every record retrieved by the post-processor will be prefixed (on a line preceding the data) by its identification, in the form of a relative record number. If the input set consists of variable-length records, the actual data length of the record will also appear on the identification line.

PRINT

This option suppresses the identification line and permits listing of print-oriented data in print format.

DUMP

The data retrieved will be displayed in standard IBM dump format: 32 bytes of data per line, divided into groups of 4 hex-bytes, and followed to the right by a character format listing. The hex dump line is prefixed on the line itself with the relative column number of the first byte in the line (see the following examples). This format is used mainly for displaying binary data.

The General post-processor accepts as input data residing on any input medium which can be processed by QSAM. It accepts fixed length, variable or variable spanned length records, blocked or unblocked, created by BSAM or QSAM. On disk data sets, it will accept records created with track overflow, and records with hardware keys (which will, however, be stripped off the record). ISAM and BDAM records are not accepted, nor are files consisting of undefined records (RECFM=U). Members of a partitioned data set are accepted if named in the DD statement.

10.4.19 PGEN

Function: Retrieve any user-defined data set format for terminal display.

Input defined by programmer: PARM field entries to govern selection and editing of data; the data be processed.

JCL required from programmer: An EXEC card with PARM field entries, and a DD card defining the data set to be processed.

(Refer to Example 7.)

10.4.20 PMGEN

Function: Same as PGEN, with MERGE step added.

Sample JCL:

```
//GENPOST   EXEC   PMGEN,
//        PARM.GENPP='DUMP,STEP=SAMPLE,NREC=50',
//        PARM.MERGE='MY.ROSCOE.KEY'
//GENPP.INPUT DD DSNAME=MY.TEST.DATA,DISP=(OLD,KEEP),
//  UNIT=2314,VOL=SER=USER05
```

(Refer to Example 7.)

```
  List
  0010 //RSTESTWW JOB 70018,WOOD,CLASS=C,MSGCLASS=B
  0020 //CREATE EXEC PGM=IEBGENER
  0030 //SYSPRINT  DD  SYSOUT=A
  0040 //SYSIN      DD   DUMMY
  0050 //SYSUT2     DD   UNIT=2314,DSN=ROSCOEGP,DISP=(,PASS),
  0060 //    SPACE=(TRK,(2,2)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=1600)
  0070 //SYSUT1     DD      *
  0080 THIS IS A SERIES OF TEST DATA RECORDS TO DISPLAY THE GENERAL POST PROCESSOR
  0090 THIS      IS    THE                   SECOND
  0100           AND THIS LITTLE PIGGY WENT TO MARKET
  0110 THIS LITTLE PIGGY STAYED HOME
  0120 IBM MANUALS ARE OFTEN LONG AND WEARYSOME
  0130                PROPHETS ARE NEVER HONORED IN THEIR OWN COUNTRY
  0140 ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789,./;'½=-?.,":¼+_)(*&¢%$#@± 555555555555
  0150           DOCTOR FAUSTUS WAS A GOOD MAN
  0160           HE WHIPPED HIS SCHOLARS NOW AND THEN.
  0170           WHEN HE WHIPPED HE MADE THEM DANCE
  0180           OUT OF ENGLAND INTO FRANCE
  0190           OUT OF FRANCE INTO SPAIN
  0200           AND THEN HE WHIPPED THEM BACK AGAIN.
  0210 ===========================================================================
  0220 I AM HIS HIGHNESS'S DOG AT KEW.
  0230 PRAY TELL ME, SIR, WHOSE DOG ARE YOU?
  0240 ===========================================================================
  0250 WE        ARE   COMING                TO THE END OF THE DATA
  0260 H E R E  I S   THE  LAST  RECORD...........
  0270 LLLLLLLLLLLLLLLLL
  0280 /*
A { 0290 //PP1 EXEC PGEN
    { 0300 //GENPP.INPUT DD  DSN=*.CREATE.SYSUT2,DISP=(OLD,PASS)
    { 0310 //PP2 EXEC PGEN,
B { 0320 //  PARM.GENPP='STEP=DUMPITUP,NREC=12,DUMP,SKIP=2'
    { 0330 //GENPP.INPUT DD  DSN=*.CREATE.SYSUT2,DISP=(OLD,PASS)
    { 0340 //PP3 EXEC PGEN,
C { 0350 //  PARM.GENPP='STEP=PRINTLST,PRINT'
    { 0360 //GENPP.INPUT DD  DSN=*.CREATE.SYSUT2,DISP=(OLD,PASS)
    { 0370 //PP4 EXEC PGEN,
D { 0380 //  PARM.GENPP='BADPARM'
    { 0390 //GENPP.INPUT DD  DSN=*.CREATE.SYSUT2,DISP=(OLD,PASS)
    { 0400 //PPLAST EXEC PMGEN,
E { 0410 //  PARM.GENPP='STEP=NOINPUT',PARM.MERGE='70018.WOOD.H'
    { 0420 //* NOTICE THERE IS NO INPUT CARD-- AN ERROR
  0430 //ENDUP   EXEC    EDITSIM
  _
```

NOTES ON EXAMPLE:

To illustrate the various options of the General post-processor, the above set of data cards were placed in a temporary disk set (lines 20-280), and then used as input to five executions of the General post-processor (lines 290-420). Each of the five PP steps differs only in the PARM field options chosen; the effects are shown in the figures on the opposite page. The following items correspond to the letters appearing in the figures .

A. In this step (source lines 290-300), default values only are requested for all options. Therefore, the stepid is "GENERAL", every record is preceded by an id-line with relative record number, and every record in the set was retrieved.

B. In this step (source lines 310-330), a stepid of "DUMPITUP" was assigned, the DUMP format was requested, every third record of the input was processed (that is, 2 records were skipped after every record processed), and a limit of 12 records was requested (due to the small size of the input, the limit was never reached).

C. In this step (source lines 340-360), the stepid "PRINTLST" was assigned, and the PRINT option was taken: every record of input was retrieved and reproduced without a preceding id-line.

D. In this step (source lines 370-390), an error was made in the PARM field, and the post-processor could not complete. The error notification is placed in the second line of the TOC entry.

E. This step (source lines 400-420) invoked the PMGEN procedure (post-process and merge), since it is the last post-processor step in the job stream. A stepid of "NOINPUT" was assigned, and a DD-statement to define the input data set was omitted. The post-processor was unable to complete, and placed a diagnostic in line two of the TOC entry.

EXAMPLE 7.

```
list rstest

 ***   POST PROCESSOR TABLE OF CONTENTS   ***

RSTESTWW.GENERAL : LINES 0001-0062;              13.43.22   12/10/70          }
ROSCOEGP                                    HAS    20 RECORDS.  20 PROCESSED. } A
RSTESTWW.DUMPITUP: LINES 0063-0106;              13.44.08   12/10/70          }
ROSCOEGP                                    HAS    20 RECORDS.   8 PROCESSED. } B
RSTESTWW.PRINTLST: LINES 0107-0128;              13.44.40   12/10/70          }
ROSCOEGP                                    HAS    20 RECORDS.  20 PROCESSED. } C
RSTESTWW.GENERAL :  NO OUTPUT   :;             13.45.06   12/10/70            }
POST PROCESSOR COULD NOT COMPLETE DUE TO ERROR IN PARM FIELD                  } D
RSTESTWW.NOINPUT :  NO OUTPUT   :;             13.45.25   12/10/70             }
POST PROCESSOR COULD NOT COMPLETE DUE TO UNACCEPTABLE RECORD OR DATA SET FORMAT} E
```

```
list rstest,1,15

1........10........20........30........40........50........60........70........*

 RECORD 000001:
THIS IS A SERIES OF TEST DATA RECORDS TO DISPLAY THE GENERAL POST PROCESSOR

 RECORD 000002:
THIS      IS    THE                SECOND

 RECORD 000003:
         AND THIS LITTLE PIGGY WENT TO MARKET

 RECORD 000004:
THIS LITTLE PIGGY STAYED HOME

-
```

A

```
list rstest,63,75

***   DUMP FORMAT REQUESTED ON INPUT SET   ***

 RECORD 000001:
::::1     E3C8C9E2 40C9E240 C140E2C5 D9C9C5E2    40D6C640 E3C5E2E3 40C4C1E3 C140D9C5   *THIS IS A SERIES OF TEST DATA RE*
:::33     C3D6D9C4 E240E3D6 40C4C9E2 D7D3C1E8    40E3C8C5 40C7C5D5 C5D9C1D3 40D7D6E2   *CORDS TO DISPLAY THE GENERAL POS*
:::65     E340D7D9 D6C3C5E2 E2D6D940 40404040                                          *T PROCESSOR     ................*

 RECORD 000004:
::::1     E3C8C9E2 40D3C9E3 E3D3C540 D7C9C7C7    E840E2E3 C1E8C5C4 40C8D6D4 C5404040   *THIS LITTLE PIGGY STAYED HOME   *
:::33     40404040 40404040 40404040 40404040    40404040 40404040 40404040 40404040   *                                *
:::65     40404040 40404040 40404040 40404040                                          *                ................*
```

B

```
list rstest,107 128

1........10........20........30........40........50........60........70........*
THIS IS A SERIES OF TEST DATA RECORDS TO DISPLAY THE GENERAL POST PROCESSOR
THIS      IS    THE                SECOND
         AND THIS LITTLE PIGGY WENT TO MARKET
THIS LITTLE PIGGY STAYED HOME
IBM MANUALS ARE OFTEN LONG AND WEARYSOME
                PROPHETS ARE NEVER HONORED IN THEIR OWN COUNTRY
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789,./;'½=-?.,":¼+_)(*&¢%$#@± 5555555555555555
         DOCTOR FAUSTUS WAS A GOOD MAN
```

C

GENERAL POST-PROCESSOR OUTPUT

10.5 USING COMPILER SYSTEM OUTPUT

Beginning in OS Release 19, many of the standard IBM compilers and assemblers have added a SYSTERM output file. Likewise, most of the new IBM program product compilers (such as the PL/1 Optimizing Compiler) have a SYSTERM output file. SYSTERM files are designed for terminal retrieval of compiler diagnostics, like the ROSCOE post-processors. Where the SYSTERM option is available for your site's compilers, and especially for the IBM program product compilers, which may be handled incorrectly by the ROSCOE post-processors, we recommend that the post-processing step be omitted, and the SYSTERM file be passed as input to the General post-processor. Sample JCL for obtaining the SYSTERM output is shown below, using the VS Assembler as an example. For other compilers, the appropriate IBM *Programmer's Guide* should be consulted for PARM-field options and JCL. (The JCL for the VS Assembler is only for the sake of the example, since the standard Assembly post-processor handles VS output.)

```
//ASMBL       EXEC      ASMFC,PARM.ASM='LOAD,NODECK,TERM'
//SYSTERM     DD        UNIT=SYSDA,DSN=&&ASMTERM,DISP=(,PASS),
//                      SPACE=(TRK,(2,2)),DCB=(LRECL=121,BLKSIZE=2178,RECFM=FBA)
//SYSIN       DD        *
****** INPUT TO THE COMPILER ******
/*
//GETBACK EXEC           PGEN,PARM.GENPP=PRINT
//GENPP.INPUT DD DSN=&&ASMTERM,DISP=(OLD,DELETE)
```

NOTE: At the option of site management, additional post-processor facilities may be made available to the user community. These facilities may include variations of the catalogued procedures described in the preceding sections, the use of the UMPOSTP post-processor for ASMG, LINKEDIT-F and higher levels of the PL/1 compiler, and implementations of the customer post-processor interface (CUSP). Site management will publish a complete description of such options for the use of the ROSCOE user community.

XI. SYSTEM MESSAGES RETRIEVAL

11.1 DEFINITION

Batch retrieval exists at two levels: retrieval of program output (the post-processors) and retrieval of system messages. The system messages consist of JCL (user JCL and additional JCL created when OS expands catalogued procedures in-line) and system messages proper (error diagnostics, allocation messages, etc.; these messages begin with the IBM prefix IEF). Retrieval of system messages is highly desirable (especially if the JOB was not run due to JCL errors). To get back system messages from JOBs submitted through ROSCOE, a special message class is used. The name of this message class is defined by your installation manager, but usually it will be class 0 (zero). To request retrieval of system messages, you need only code in your JOB statement:

```
MSGCLASS=0
```

ROSCOE will recognize the request and insert into your job stream (immediately following the JOB statement) a comment card with your ROSCOE account key. If you desire to submit a job through batch instead of ROSCOE, but retrieve through ROSCOE the system messages, you must code not only the MSGCLASS parameter on the JOB card, but also insert the comment card following it. The comment card must be exactly as follows:

columns 1-3:	//*
column 4:	blank
column 5 and following:	your ROSCOE account key
columns 50-71:	optional comment.

11.2 DISPLAY

To list the system messages, use the command:

```
DISPLAY jobname,m,n
```

where m and n are optional line numbers (the messages are numbered from 1 by increments of 1), and jobname has the special form used also for post-processor jobnames: the first 6 characters of the jobname are prefixed by your ROSCOE prefix. For example, if you submit a job called HMPROGW5, you will ask for a display of HMPROG (or of XX.MHPROG, if XX is your prefix). The system messages will then be listed exactly as found by ROSCOE. If the job cannot be found, ROSCOE will assume that it has not run, and will display a message:

```
jobname JOB NOT FOUND
```

11.3 CANCEL

To delete the system messages data set, use the command:

```
CANCEL  jobname
```

Note that the CANCEL command does *not* cancel a job waiting in the OS queues; it only deletes from the ROSCOE library the system messages set.

11.4 GET

To fetch system messages data sets into the AWS, use the command:

```
GET jobname

G,jobname
```

11.5 MISCELLANEOUS

LIST jobname will cause a listing of post-processor output (if any) from the same job.

NOTE: Message class 0 may also be used as a SYSOUT class for small volumes of data (such as the SYSPRINT sets of IBM utilities). This is done by coding SYSOUT=0 in the appropriate JCL statement. Users should not attempt to retrieve large amounts of data in this manner. Use for this purpose the post-processors. Any SYSOUT data sets directed to class 0 will be truncated or padded, if necessary, to 80 characters. The option of using the message class designation for SYSOUT sets is not in effect at a HASP site. Consult site management for specific details.

NOTE: The ROSCOE system message retrieval facility, as described in this section, does not operate in ASP systems. Site management at ASP installations will publish a complete description of the system message retrieval facility under ASP for the benefit of the ROSCOE user community.

APPENDICES

NOTE: Terminal-dependent information is presented in Appendices A, B, C, D, and F. The actual operation of the terminal (switching on and off, location of keys and switches) is not described, since the information is readily available in manufacturers' publications. The following Appendices describe only the ROSCOE usage of the hardware, and should be carefully studied by users.

APPENDIX A

2741 TERMINAL

APPENDIX A: 2741 TERMINAL

1. Establishing the connection with ROSCOE.

   A. Terminals on leased lines ("hard-wired terminals"). A 2741 can be connected directly to the terminal control unit by a leased line; that is, it is not necessary to dial a special phone number to make connection. Contact is established by turning the terminal ON (if on already, it must be turned OFF and then ON again). The RETURN key is hit; after a pause, a slight click is heard. RETURN is again hit until ROSCOE replies with the log-on request.

   B. Otherwise, a terminal is connected to the control unit by means of ordinary telephone lines, and the computer must be dialed up on a telephone which is placed by the terminal. The number to be dialed will be made known to users by the installation management. When the number is dialed, the computer 'answers' with a high-pitched tone. The hand set is then placed in the coupler (also located by, or in some units, as part of, the terminal), which can be turned on then, or already have been turned on. The green ready light will then light up on the terminal, and the user hits RETURN one or more times, as for hard-wired terminals.

2. Operations on a 2741 are in half-duplex mode; that is, the terminal can be either sending or receiving, but not both at once. When ROSCOE is reading data from the terminal, the terminal will be locked. It will be unlocked when ROSCOE is either writing to the terminal or waiting for data from the terminal. The user must wait for the terminal to unlock before attempting to type on it. ROSCOE normally notifies the user that the terminal is unlocked and open for input by typing the prompter (an underscore, "_").

3. ROSCOE has no way of sensing whether paper is present on the terminal. When paper is exhausted, ROSCOE will keep typing out, so the user must be sure he has paper enough for his session.

4. Different manufacturers' models of the 2741 have different character sets; that is, transmission codes (this has nothing to do with the characters found on the type-ball). ROSCOE determines the character set of the terminal at log-on time by trying out different character sets until the right one is found. Therefore, the first log-on request from ROSCOE will often appear as a line of meaningless characters. The user should then hit RETURN again, whereupon ROSCOE will select another character set. When the right set appears on the terminal, the user should begin his log-on entry.

5. The RETURN function is handled on the 2741 by hitting the RETURN key, thereby causing a line feed and carriage return. The line feed and return are interpreted by ROSCOE as the termination of the current command or entry, and are not transmitted as actual characters. The normal way of entering a command or other entry into ROSCOE is to type it on the terminal and hit RETURN.

6. Tabs are set on the 2741 as for a normal typewriter. A tab will be recognized as such by ROSCOE only if the TAB command has been used to inform ROSCOE of the desired tab positions. Otherwise, the use of TAB will cause the actual tab character to enter as data. ROSCOE will usually flag this intrusive tab as an error in commands. When it is part of a line entered into the AWS, it will go in as data, and when the line is listed, the carriage will skip. This is to be avoided, as an error will usually occur when ROSCOE attempts to return the carriage to the beginning of the next line.

7. The BACKSPACE function is filled by using the backspace key, going back over the characters to be rubbed out. The exception is when the user is in extended mode (X command): an uppercase backspace (that is, backspacing while the shift key is held) will be accepted as actual data.

8. The ATTN function is filled by hitting the ATTN key. This key can be hit only when the terminal is typing out. If used when the terminal is not typing out (but unlocked), it will be taken as a RETURN, without causing either return or line feed. If ATTN is used when the carriage is returning (that is, between lines), a time-out of 24 seconds (more or less) will be caused, during which the terminal is locked, and the user should not attempt to strike any key. ATTN will stop a listing, returning the user to the ROSCOE dispatcher to await the next command. In Autoline mode, a line typed in with too many errors to correct can be cancelled by ATTN; ROSCOE will then type out the line number again as a prompter. A final use of ATTN should be noted. The three enqueued routines in ROSCOE, LIBRARY, SUBMIT, and VERIFY will occasionally display the message STANDBY, indicating that the routine is in use and the user must wait. The terminal is then unlocked for a period of 2 or more seconds, during which the user can hit ATTN to exit from the queue.

9. The 2741 terminals are very sensitive to noise on the wires. If a line sent from the terminal to ROSCOE could not be received due to such errors, ROSCOE will display the message:

```
TERMINAL I/O ERROR: DATA LOST
```

The line can then be typed in again and resent. However, if such an error occurs while the terminal is in Proc mode, the procedure will be concluded. If errors occur while ROSCOE is attempting to send data to the terminal, retry procedures will be entered. In the rare case that errors are found during the retry procedures, indicating severe communications problems, ROSCOE will sign the terminal off (disconnect) automatically. The user can then attempt to reestablish connections on another line.

10. The user must be very careful to log-off correctly. The procedure is to enter the command OFF. (The use of OFFON is described in Section II.) ROSCOE will then display two log-off acknowledgments:

```
LINE #nn –TIME ON hh.mm.ss –ELAPSED hh.mm.ss –mmmLIBRARY
RECORDS AVAILABLE
```

(Terminal has been signed off.)

A hard-wired terminal can then be turned off. On a dial-up line, the red CHECK light will then light up and remain on for a moment. When the light goes out (accompanied usually by a slight jump of the carriage), the user may turn the unit off, hang up the phone, and turn off the coupler.

Log-off procedures can be forced by ROSCOE in response to any of three conditions:

A. Central console operator has requested ROSCOE to sign the user off;

B. The user has turned off the power or hung up the phone;

C. Irrecoverable error conditions have occurred on the phone lines, such that the line became inoperable.

When any of these conditions occur, ROSCOE will sign the user off, and any data in the user's

AWS will be SAVEd under the reserved name xx.SAVAWS, where "xx" is the user's prefix. The contents of the AWS will not be saved automatically if the user enters the OFF or OFFON command. Where the central console operator has shut down ROSCOE while users are still active, the AWS will be saved as described in Section 2.

11. When the red check light goes on, due to some line error, it can be reset by hitting the RESET key.

12. Terminals which do not have an ATTN key, or else lack the BREAK feature (a hardware option on the terminal controller) may use the ROSCOE BREAK command to control output to the terminal. The form of the command is:

    ```
    BREAK n
    ```

    where n is the number of lines to be typed to the terminal before pausing. At the end of the nth line, ROSCOE will pause. The user then hits RETURN to continue output, or enters any data to terminate the current function. (The data will not be scanned by ROSCOE for a valid command; it is used only to signal an ATTN.) To reset the BREAK limit without substituting a new n value, use:

    ```
    BREAK
    ```

    with no operands.

    NOTE: A BREAK value of 15 is preset at sign-on time.

APPENDIX B

TELETYPES

APPENDIX B: TELETYPES

1. Establishing contact with ROSCOE, on a leased line, see the terminal is in half-duplex mode and turn it on. On a switched line, follow dial-up procedures as outlined in Appendix A, Item 1B.

2. Operations on a TTY are in half-duplex mode; that is, the terminal can be either sending or receiving, but not both at once. When ROSCOE is reading data from the terminal, the terminal will be locked. It will be unlocked when ROSCOE is either writing to the terminal or waiting for data from the terminal. The user must wait for the terminal to unlock before attempting to type on it. ROSCOE normally notifies the user that the terminal is unlocked and open for input by typing the prompter (a back-arrow "←").

3. ROSCOE has no way of sensing whether paper is present on the terminal. When paper is exhausted, ROSCOE will keep typing out, so the user must be sure he has paper enough for his session.

4. Because alphabetic characters on teletypes are only uppercase, the X (extended characters set) command is ignored when received from a TTY.

5. The RETURN function is provided by X-OFF or X-ON. If the EOT is not enabled on the unit, it too, may be used as a RETURN; if it is enabled, it will result in a true end of transmission and disconnection from ROSCOE. The TTY has normally a carriage 72 characters wide. Therefore, lines being typed by ROSCOE to the terminal which are longer than 72 characters will be automatically broken and typed as two lines, the second line beginning at column 73. On input, up to 130 characters may be typed in. The control sequences return/linefeed and linefeed/return may be used to aid in entry, and are removed by ROSCOE. But isolated linefeed characters are kept.

6. The TAB function is provided by the normal use of the TAB key. Note that the Model 33 teletype does not move the carriage when TAB is struck; the Model 35 does. Whether 33 or 35, ROSCOE will recognize the TAB character provided that the ROSCOE TAB command has been used.

7. The BACKSPACE (rub-out) function is provided by use of the backslash character (uppercase L).

8. The ATTN function is provided by the CTRL/WRU key, which is interpreted as a cancel/retry. The interrupt ATTN is provided by CNTRL/BREAK but the key must be held down for some slight length of time, and may sometimes be ineffective.

9. The ROSCOE prompt appears on the TTY as a back-arrow, not an underscore.

10. TTY terminals are sensitive to noise on the wires. If a line sent from the terminal to ROSCOE could not be received due to such errors, ROSCOE will display the message:

    ```
    TERMINAL I/O ERROR: DATA LOST
    ```

    The line can then be typed in again and resent. However, if such an error occurs while the terminal is in Proc mode, the procedure will be concluded. If errors occur while ROSCOE is attempting to send data to the terminal, retry procedures will be entered. In the rare case that errors are found during the retry procedures, indicating severe communications problems, ROSCOE will sign the terminal off (disconnect) automatically. The user can then attempt to reestablish connections on another line.

11. The user must be very careful to log-off correctly. The procedure is to enter the command OFF. (The use of OFFON is described in Section II.) ROSCOE will then display two log-off acknowledgments:

```
LINE #nn –TIME ON hh.mm.ss –ELAPSED hh.mm.ss –mmmmLIBRARY
RECORDS AVAILABLE
```

(Terminal has been signed off.)

A hard-wired terminal can then be turned off.

Log-off procedures can be forced by ROSCOE in response to any of three conditions:

A. Central console operator has requested ROSCOE to sign the user off;

B. The user has turned off the power or hung up the phone;

C. Irrecoverable error conditions have occurred on the phone lines, such that the line became inoperable.

When any of these conditions occur, ROSCOE will sign the user off, and any data in the user's AWS will be SAVEd under the reserved name xx.SAVAWS, where "xx" is the user's prefix. The contents of the AWS will not be saved automatically if the user enters the OFF or OFFON command. Where the central console operator has shut down ROSCOE while users are still active, the AWS will be saved as described in Section 2.2.

12. The TTY user may request ROSCOE to pause automatically after n lines of output by using the command:

```
BREAK n
```

where "n" is the number of lines. The command:

```
BREAK
```

with no operands disables the BREAK function. ROSCOE will pause at the end of the nth line. To continue printing, enter X-OFF. Otherwise, output is terminated.

NOTE: The value of n=15 is set at sign-on time.

13. If the physical width of the TTY carriage is greater than 72 positions, ROSCOE can type data up to the maximum width. The physical width of a carriage is defined to ROSCOE in the command:

```
WIDTH n
W,n
```

where "n" is a value between 72 and 132, inclusive. If the operand is omitted, the default width of 72 positions is forced. The default width is also preset at log-on time. Lines longer than the physical width as defined to ROSCOE will be broken and typed as two consecutive lines.

14. Because of the hardware differences between standard TWX TTY 33/35 terminals and the various TTY-compatible terminals (which may operate at different speeds of 10, 20, and 30 characters per second), ROSCOE may not set up a sufficient number of Pad characters in an output file. Therefore, characters may occasionally be lost at either end of the line or printed while the carrier is on the fly returning to the beginning of the next line of the paper. In order to handle correctly all types of TTY terminals, a special table of Pad characteristics is set up at every site, with entries corresponding to whatever terminals are used by ROSCOE programmers. This table is set up by the site system programmer, who will inform users which kinds of terminals are employed. To indicate to ROSCOE that a different number of Pad characters should be utilized at the user terminal, use the command:

```
TTY  n
```

where "n" is the site code for the specific user terminal type. If the operand is omitted, ROSCOE will use the site default value in the table. This command is used both when the terminal is printing too fast (that is, data characters are being lost), and when it is too slow (that is, when there is too long an interval between the time the carrier returns to the left margin of the paper and the first character of the line is printed). In all cases, the numeric value "n" may not be higher than the highest number entry in the site table established by the system programmer.

APPENDIX C

2260 LOCAL STATION

APPENDIX C: 2260 LOCAL STATION

1. The RETURN function is provided by the SHIFT/ENTER combination.

2. The ATTN function is provided by moving the cursor away from the top left-hand corner of the screen and pressing SHIFT/ENTER. This is the output ATTN to exit from a command which is listing to the screen. To exit from Autoline mode, a line with an EOM in the first position will suspend Autoline (EOM is generated by the combination SHIFT/ENTER).

3. As 2260's have only uppercase alphabetical characters, the command X (extended mode) is ignored when received from a 2260.

4. BACKSPACE is provided by the BACKSPACE key. DOWN, UP, and SPACE can also be used for these respective functions.

5. The TAB function is provided by designating a special character (not alphabetic nor numeric nor a control character) to act as a TAB mark. The TAB mark is preset to the NOT-sign (uppercase I) but may be re-defined at any time by the user by the command:

   ```
   TAB/
   ```

   where / is any special character which, when used, will be translated by ROSCOE into a tab.

6. The BREAK command is ignored when entered from a 2260. ROSCOE automatically suspends output when 12 lines of the screen have been filled. To continue, use RETURN. To stop, use ATTN.

7. As the width of a 2260 screen is 80 characters, a line longer than 80 characters will be displayed as two lines, broken at column 80. When listing the AWS in NOSEQ mode, the line number will appear by itself on the first of two lines, and the data on the second.

8. It is at present impossible to enter a full 80 characters of data on the screen (79 is the limit).

9. The cursor serves as the ROSCOE prompt, and will be positioned to the left of the next line from which ROSCOE expects input.

10. When the terminal is in Autoline mode, ROSCOE will display the next 12 sequence numbers on the right of the screen as prompts. The user may write in his data line-by-line, terminating input after any line by RETURN. To write multiple lines on a screen, use shift/newline at the end of each line. The user can overwrite the ROSCOE prompt numbers, which are regenerated when the lines are received by ROSCOE.

    When entering data into the AWS using explicit (user-generated) sequence numbers, the sequence number is entered beginning in the left-most position of a line. A single space must be left between the last digit of the sequence number and the first character of the input data.

11. A line on the screen corresponds to a ROSCOE command; that is, no command, whether a normal command or an AWS text-entry, may be broken over two lines.

12. Up to 12 ROSCOE commands may be entered at once from the screen, one to a line, each line terminating with shift/newline, and the 12th line with shift/enter. But a command that causes an output to the screen will terminate the current command sequence. For example:

```
FETCH MYDATA
LIST
APPEND DATA2
```

will cause ROSCOE to execute the FETCH and LIST, but not the APPEND.

13. A useful feature of 2260 ROSCOE is that a line of data from the AWS can be edited directly on the screen by listing the line or lines desired, moving the cursor to the desired location, and editing in place by over-writing. The sequence number can also be edited, *UNLESS THE USER IS IN* NOSEQ mode, where the in-place editing is illegal. Though convenient, this facility takes more time than the normal use of the EDIT command, unless the AWS is of moderate size (not more than ¼ to ½ of the AWS maximum). In absolute terms, it should be avoided where the AWS is over 500 lines in length. Any line edited on the screen must be terminated by a shift/newline or shift/enter symbol.

14. The user is cautioned to remember that ROSCOE suspends operations after the 12th line has been displayed on the screen. Therefore, he must be attentive to RETURNing during listings, especially when receiving a listing in response to the LIBRARY command. This command is not usable by other users while he has it in use, or when he is receiving output from a syntax checker, which may be enqueued, depending on installation parameters.

15. Even if the unit is equipped with a 1053 printer, the PRINT key cannot be used. ROSCOE makes no check for its use, and users are warned that use will normally cause a serious system interlock.

16. The user is also cautioned not to become too impatient at possible slow response. Continuing to hit RETURN before response to the previous RETURN will result in unpredictable loss of data and command. Turning the unit off, unless after log-off procedure, will also result in loss of data and possible system interlock.

APPENDIX D

REMOTE 3270 TERMINALS

APPENDIX D

REMOTE 3270 TERMINALS

Remote 3270 terminals are treated in every respect as local 3270 terminals (please refer to Appendix G of this manual for operating procedures). The following special considerations should be studied for features unique to remote 3270's:

- When the ROSCOE System is put into execution, it will display on all remote 3270 screens the word "ROSCOE" to identify the terminal to remote users.

- If a display station is off when ROSCOE begins execution, the identification will not be written when it is turned on. It is good practice, therefore, to be sure the station is on when ROSCOE begins, and to leave it on after the session for the next user. Recommended also is turning the brightness control down as far as possible consistent with legibility; this prevents the screen from burning out.

- Unlike the other terminal types handled by ROSCOE, a remote 3270 is a polled device; i.e., rather than an instantaneous response to an ENTER, the screen is read at the end of a polling interval (usually not more than one second).

- The possibility of I/O errors on a remote 3270 line is far greater than that for any other type of line. If an error occurs local to a single terminal, recovery is very rapid. If, however, the error is one affecting the entire line (a line controls several terminals), recovery is achieved after waiting a predetermined interval of time, then addressing the terminal again. (During this interval, it will appear that the system has stopped.) If at the end of several minutes terminal activity has not resumed, it is possible that the line has been disabled due to a massive series of errors. (Note that such occurrences are most common during periods of electrical disturbance, such as a thunder storm.) In this case, your active data in the AWS will be saved under the name "SAVAWS" (preceded by your prefix) so that when the line is restarted, you can resume your session. The operations staff is responsible for informing users at a remote site that the line they are working on is no longer active, and also for telling them when the line is restarted. NOTE: An occasional hardware malfunction on remote 3277's will cause all display stations to be locked out. This situation may be remedied in some cases by turning the control unit (not the display station) off, then on again at once. This operation should not be attempted by terminal users, who should notify the system manager or their supervisor of the condition and request that the latter reset the control unit.

- Due to the hardware design of the stand-alone 3275 display station, use of the ENTER or other PA/PF keys causes the display screen to be cleared for a brief interval and then refilled with the screen image. This feature should not be of concern to the user.

- Remote 3270 support permits use of a printer to produce hard copy. Users should be aware of the comparatively slow speed of the printer and not attempt to list large volumes of data on it. Handling of the printer differs slightly depending on whether the user is at a 3277 or 3275 display station. Where such differences in handling are pertinent, they are noted in the following description.

– The terminal command to cause a print operation is:

```
PRINT
  or
P
```

The operands of this command are identical to those of the LIST (L) command (see Sections 3.5 and 4.5). This command is valid only if entered at a remote 3270 terminal and there is a printer unit (or units) attached within the same cluster as the display station which issues the command. For 3277's, the printer will normally be in the same area as the display station. For remote 3275's, the printer is part of the 3270 complex at which the user is seated. If there are no printers meeting the above requirements, the PRINT command will be rejected and the following message displayed on the display screen:

```
PRINT REQUEST REJECTED
```

If there is a printer on the 3275, the print operation will begin immediately. With 3277's, it may happen that all printers are in use at the moment; in this case, the user will receive the following message on the display screen:

```
ALL PRINTERS BUSY
STANDBY
```

The user may then wait for a printer to become available, or may exit from the printer queue in the same manner as exiting from any other ROSCOE standby condition (see Section 6.8). When the print request can be honored, the following message is written on the display screen (3277 only):

```
PRINT REQUEST ACCEPTED
```

The print operation is then performed, and when done, the following message is sent to the display screen:

```
PRINT COMPLETE
```

The user is then back in the normal ROSCOE command mode.

When ROSCOE data is listed to the printer, it is preceded by ten blank lines to separate the data from previous listings. After these lines, the words:

```
START PRINT
```

are printed, followed by the user's key on the next line and two more blank lines. The data is then printed.

– If an I/O error occurs during the print operation, it will be terminated with an explanatory message written to the display station:

  I/O ERROR ON PRINTER

  A printer running out of paper or the lid of the printer being raised will also be treated as an I/O error.

– If a user wishes to interrupt a print operation to stop it, the action differs depending on the display station being used:

  On a 3277, the terminal user may use the ENTER key or any PF/PA key to terminate the printing. If commands are placed on the screen before operating ENTER, or if the PA/PK keys have been designated as having special significance, such commands will be ignored. Only the fact that an action was taken by the terminal operator is recorded, so that all actions are an unconditional request to terminate printing. The printer will terminate when the last line of the current printer buffer has finished printing. That is, after ENTER on the display station has been operated, printing may continue for some time before ROSCOE is able to address the display station again in normal terminal command mode.

  On a 3275, a print operation cannot be terminated by display station action. If it is necessary to terminate a print operation, the lid of the printer should be raised. This will create a condition which ROSCOE will handle as an I/O error, thus forcing the operation to halt. NOTE: Under no circumstances should the power of the print unit or display station be turned off. This will create an error condition with unpredictable results.

  On both 3277's and 3275's a user request to terminate the print operation will be acknowledged by the following message on the display screen:

  PRINT REQUEST TERMINATED

– Special note for 3275 print: Due to the hardware design of the 3275 unit (i.e., the same buffer is used for both display screen and printer), the data being printed will also appear on the display screen. As the data is formatted for the printer, not for display, it will appear unformatted on the display screen. The cursor will move along the display screen as the print proceeds, indicating the character then being printed.

APPENDIX E

NOTES ON THE PROCEDURE LANGUAGE

APPENDIX E

NOTES ON THE PROCEDURE LANGUAGE

The following notes are miscellaneous observations concerning the use of the ROSCOE command language in ROSPROCs. These notes are intended to supplement and illustrate information presented in the text portion of this manual.

1. REPLY COMMAND/REPLY BUFFER

- The reply buffer is 80 characters in length (not 76).
- Replies placed into the buffer are left-adjusted and padded with trailing spaces.
- If a reply from the terminal consists of either all blanks or a RETURN, the reply buffer is effectively cleared, and its length is 0. The length of the contents of the reply buffer is always known and can be tested within a ROSPROC by using the SET LENGTH command.
- Replies can be loaded as follows:
  - —— From a terminal response;
  - —— From a line of the AWS;
  - —— In-line from an operand string in REPLY (e.g., REPLY ¢abc¢);
  - —— With the value of the ROSCOE counter by means of SET.
- When a literal is compared to a reply, it is expanded to the length of the contents of the reply buffer (or a substring within the buffer) by padding with spaces to the right of the last non-blank character.
- Since the condition code set by COMPARE is determined by the EBCDIC collating sequence, it is possible to test by class (not by character) for numeric, alphabetic, alphanumeric, and special characters (as well as the negatives or any combination of the preceding), so long as the comparand string can be made exactly equal in length of significant characters to the reply string or substring.
- A common error resulting from disregard of the rule governing the special separator character for packing commands on a line is described below.

  Assume a procedure with the sequence:

  ```
  0010  REPLY

  0020  #TYPE#
  ```

  and assume a reply of:

  ```
  123ABC&$50.00
  ```

The result will be that the echo command (#TYPE#) will type out only 123ABC, and a command error (or random error) will occur when ROSCOE attempts to execute the next command of the procedure. The reasons for this error are as follows:

1. When executing a ROSPROC, ROSCOE first moves an 80-character line from the procedure read-in buffer into the command buffer.

2. Detecting the special characters (#), ROSCOE then inserts the contents of the reply buffer into the command as indicated.

3. ROSCOE then scans the expanded command buffer from left to right to unpack commands – that is, until either the special separator character (&) is detected or the end of command buffer is reached.

4. A pointer is left at the end of the scan, marking either start of the next command in the buffer or end of buffer. The unpacked command is then dispatched to the command interpreter, which will in turn dispatch it to the command executer.

5. When the command execution is completed, the pointer from step 4 above is tested. If it points to buffer end, the cycle is reinitiated from step 1; otherwise, the next set of characters in the command buffer is taken as the next command. In the example given, the next set of characters is $50.00, which will be rejected by the command interpreter and force termination of the procedure.

When a terminal reply is solicited, it is advisable to warn the procedure user that the reply must not contain the special command delimiter.

- The PAUSE command is useful when a simple YES/NO reply is to be made (GO/EXIT).
- On terminals where X and B modes apply (e.g., 2741), B mode should be forced before pausing for a reply, unless lower case characters are valid in the expected response.
- Instructions listed to the procedure user by means of the TYPE command should be as complete as possible, and will look and read most naturally and easily if in normal upper and lower case.

2. SKIP

SKIP can be used to form a jump table. Assume that the user of a procedure has four possible options which will result in four different actions. In the example which follows, the action consists of creating an in-line reply which is the name of a data set on the library which will be SUBMITted at the end of the procedure. The sample procedure creates JCL for a job stream. It presupposes that the terminal user has saved a source program under the name SYSIN, and that two other library sets have also been created (JOBCRD and MORJCL) with a JOB card and more JCL.

```
0010 TYPE Enter your compiler option:
0020 TYPE 1. ANS COBOL
0030 TYPE 2. FORTRAN G
0040 TYPE 3. PL/1-V
0050 TYPE 4. ALGOL
0060 TYPE
0070 REPLY
0080 COMPARE #1#
0090 IF HIGH, ERROR
0100 COMPARE #4#
0110 IF LOW, ERROR
0120 #SET#
0130 DECR 1
0140 SKIP 0
0150 REPLY #COBJCL# &SKIP 3
0160 REPLY #FORJCL# &SKIP 2
0170 REPLY #PL1JCL# &SKIP 1
0180 REPLY #ALGJCL# &SKIP
0190 #SUBMIT JOBCRD,#,SYSIN,MORJCL
0200 TYPE your compilation has been submitted for execution.
```

Lines 10-70 of the procedure list the possible compiler options for the terminal user, and accept his choice. Lines 80-110 test for a valid response (between 1 and 4, inclusive). If the response is invalid, another procedure named ERROR is entered to take whatever recovery action is desired. If the response is valid, the numeric value is placed into the ROSCOE counter by line 120, decremented by one for the jump table, and then picked up as a replacement for the operand "0" in line 140. The line SKIPped to as a result sets the desired data set name and, in turn, SKIPs to line 190, which will pick up the compiler option in the submit stream.

3. CREATING AN AWS

When creating a small AWS (for example, to make up initial JCL with JOB card and EXEC cards preceding a SYSIN deck), it is more convenient and rapid to make it in-line, rather than to fetch a set of dummy JCL. For example:

```
0010 DELETE

0020 TYPE What is your JOB-name?

0030 REPLY

0040 #0010  //#  JOB  (¢¢¢),CLASS=B

0050 TYPE What account number is to be used on the JOB-card?

0060 REPLY

0070 #EDIT/¢¢¢/#/10

0080 (continues)
```

If the reply to line 30 is "COPYPACK", line 40 will place a line in the AWS as follows:

```
0010 //COPYPACK  JOB  (¢¢¢),CLASS=B
```

If the account number received by line 60 is "345-982", then the EDIT in line 70 will alter line 10 of the AWS to:

```
0010 //COPYPACK  JOB  (345-982),CLASS=B
```

Further JOB card key words can be inserted in the same fashion, either by editing dummy strings in a card image, or by placing replies into continuation cards of the JOB statement (see examples F—1, F—2 and F—3 at the end of this Appendix).

4. SUBMIT

If a procedure builds a job stream for the terminal user, it is recommended that he be permitted to list the job for a final review before submitting it to OS. Passing the jobstream through the JCL syntax checker and (if applicable) through the language checkers is also advisable.

5. RECURSION/UPDATING THE ROSPROC BEING EXECUTED

The ROSPROC being executed may be updated by itself. When this occurs, the original ROSPROC continues to be executed until the procedure is terminated (control returns to the terminal level). If the ROSPROC is CALLed during execution, however, the updated version will be executed (return will be to the original version).

Apparent recursion of a procedure is possible, as in the following example of a ROSPROC named XPROC:

```
0010 REPLY

0020 COMPARE *BREAK*

0030 SKIP 1,EQUAL

0040 DO XPROC,10

0050 (next action within the procedure)
```

If the terminal reply received by line 10 is the word "BREAK", the procedure will continue at line 50; otherwise, the procedure XPROC will be reentered beginning at line 10 and will continue with a repeated sequence of command. The recursion is only apparent – that is, it is not nested in a stack. The overhead for such recursion is negligible, since ROSCOE stores the name and disk location of every procedure entered via a DO or IF command, and can reenter that procedure without a preliminary search for it in the Line Libraries. When a procedure terminates so that control returns to the terminal, the associated name and disk location pointers are reset (that is, cleared). Note that if a command has been encountered within the above procedure to CALL XPROC at a given starting line number (in-line subroutine), the CALL would not use the pointers, but would search for XPROC as though for an unknown procedure.

6. PROMPTING

The conversational and decision-making capabilities of the procedure language can be effectively utilized in language prompters. An example of a JCL prompter is shown in Example F–1 (building JCL for IEBPTPCH). More elaborate prompters should be prepared for those users who are inexperienced in the language being used, or who do not understand (or desire to understand) ROSCOE facilities. In such cases, a ROSPROC can turn ROSCOE into a virtual computer of any sort.

A COBOL prompter is used as the example in this section. This prompter should offer the following functions to be optionally elicited by terminal replies:

- Instruction in the elements of the language;
- Instruction in the use of any given reserved word or other feature of the language;
- Use of abbreviations which the procedure will expand;
- Invocation of the COBOL Syntax Checker;
- Generation of JCL;
- Storage of the source program being created;
- Alignment of source statements;
- Generation of standard coding sequences, ranging from copying blocks of common code to coding division headers.
- Any other functions required by the individual site.

All such prompters should be stored with the RO prefix, so that they can be accessed by all users in a system. Prompters should use the TRAP/TEST facility where necessary to prevent user errors from terminating the procedure. The prompters should also inform the user where he can reenter the procedure in case errors occur which terminate the procedure prematurely (e.g., irrecoverable errors which cannot be TRAPped).

The prompter is entered by the command:

```
DO  RO.COBOL
```

The sample procedures which follow can be used as the basis for creating any COBOL prompter.

Initial Procedure: RO.COBOL

```
0010 NOTRACE &TRAP ON &B &TYPE

0020 TYPE ANS COBOL PROMPTER: &TYPE Enter function desired:

0030 TYPE 1. EXPLANATION OF COBOL SYNTAX;

0040 TYPE 2. FORMATTING OF SOURCE PROGRAM;

0050 TYPE 3. SYNTAX CHECK OF SOURCE PROGRAM;

0060 TYPE 4. JCL GENERATION AND SUBMISSION;

0070 TYPE ? To terminate this procedure:

0080 TYPE &REPLY &COMPARE *4* &SKIP 1,LOW

0090 COMPARE *0* &SKIP 2,LOW

0100 TYPE You have entered an invalid code: please try again.

0110 DO RO.COBOL,80

0120 SET &DECR 1 &SKIP 0

0130 DO RO.EXPLN

0140 DO RO.FORMAT

0150 DO RO.SYNCHK

0160 DO RO.COBJCL

0170 EXIT
```

Line 10 turns off the TRACE, sets the TRAP ON to trap user errors, places the terminal in the Basic character mode, and spaces a line before starting the dialogue. The dialogue initiated in line 20 offers various options. Lines 80-110 validate the response and permit retry if invalid. Otherwise, beginning at line 120, the appropriate procedure is selected (by using the function 1, 2, 3, or 4 within the ROSCOE counter as a skip factor) and entered. Line 170 can never be executed, but is coded to show the end of the initial procedure.

Second-Level Procedure: RO.EXPLN (Instruction in COBOL Syntax and Usage)

```
0010 TYPE Enter one of the following codes for type of explanation:

0020 TYPE 1. basic concepts;

0030 TYPE 2. by division;

0040 TYPE 3. by COBOL reserved word or feature;

0050 TYPE ? to terminate this procedure.

0060 TYPE &REPLY

0070 (verification of the reply)

............

0110 *CALL RO.EXPLN*

0120 TYPE If you desire further explanation, enter Y, else enter N.

0130 TYPE &REPLY &COMPARE *Y* &IF EQUAL,RO.EXPLN

0140 COMPARE *N* &IF EQUAL,RO.COBOL,30

0150 TYPE Your reply was incorrect: retype it, please:

0160 DO.ROEXPLN,130

0170 EXIT
```

If the user desires explanations of certain COBOL features, this procedure is entered. It includes multiple options which the user selects by entering the appropriate code. If the entered code is valid, the reply buffer will contain a 1, 2, or 3. Line 110 suffixes this digit to the procedure name RO.EXPLN, thereby creating a command CALL RO.EXPLN1 or CALL RO.EXPLN2 or CALL RO.EXPLN3. These names correspond to other procedures in the ROSCOE library, which contain appropriate instruction in the language feature requested. The CALLed procedures may also contain CALLs on other procedures; however, they will always be terminated by an EXIT statement, which will return procedure control to line 120 of RO.EXPLN. The user is then asked whether he needs further instruction; if not, the initial procedure, RO.COBOL, is reentered to allow the user to begin the next stage of composing a COBOL program.

Second-Level Procedure: RO.FORMAT (Formatting a COBOL Source Program)

```
0010 TYPE Enter one of the following codes for DIVISION generation:
0020 TYPE 100 IDENTIFICATION &TYPE 200 ENVIRONMENT &TYPE 300 DATA
0030 TYPE 400 PROCEDURE &TYPE ? to terminate the procedure &TYPE
0040 REPLY
0050 (verification of the reply)
............
0090 SET &DO.ROFORMAT,0
0100 DELETE
0101 0001 ƀƀƀƀƀƀƀIDENTIFICATION DIVISION.
0102 TYPE Enter your program ID (no quotes or periods, please): &REPLY
0103 *0002 ƀƀƀƀƀƀƀPROGRAM-ID. '*'.
0104 0003 ƀƀƀƀƀƀƀENVIRONMENT DIVISION.
............
0110 0008 ƀƀƀƀƀƀƀFILE-CONTROL. &SET 9
0111 TYPE Enter name of next FILE to select, or DONE if no more:
0112 TYPE &REPLY &COMPARE *DONE* &SKIP 3,EQUAL
0113 *0 ƀƀƀƀƀƀƀƀƀƀƀSELECT * ASSIGN %%%.
0114 TYPE Enter assignment for that file: &REPLY
0115 *EDIT /%%%/*/0 &INCR 1 &DO RO.FORMAT,111
0116 INCR 1 &TYPE You have terminated the IDENTIFICATION DIVISION.
0117 TYPE Do you wish to SAVE what you have written so far?
............
0200 TYPE You are now beginning the DATA DIVISION.
............
```

As usual, the user is presented with a set of options for generating the different COBOL DIVISIONs. The options are digits corresponding to line numbers within this procedure. The option selected is placed in the counter, and addressed in line 90 to transfer control to the appropriate line number of the procedure. Assuming that the user begins with the IDENTIFICATION DIVISION, line 100 of the procedure clears the AWS, lines 101-109 generate the standard division header statements, and lines 111-115 generate the SELECT statements with file-id's and assignments. These lines use the counter to contain the current line number of the source program being written into the AWS, and use symbolic replacement and the EDIT command to generate the SELECT statements properly. In line 117, the user is given the option to SAVE the source program so far. If this option is taken, a name will be solicited, the current contents of the AWS will be SAVEd under that name, and the AWS will be cleared. Generation of the next DIVISION begins with line 200.

A variation of the above usage is to permit recognition of the word HELP whenever a REPLY is solicited. Each HELP causes a CALL to an appropriate instruction procedure which will explain syntax and usage of the statement type being generated.

Other portions of the prompter follow the same pattern and should be as elaborate and complete as possible, to simplify and expedite mechanical coding procedures for the user. The following two recommendations should be observed:

- All prompting messages to the user should be as self-explanatory and unambiguous as possible, even at the cost of extra terminal time.
- The packing facility (multiple commands on a single line) should be used as much as possible, to save disk space and execution (I/O) time in fetching the next command to be executed.

7. INTERRUPTING A ROSPROC

A procedure interrupted by a terminal ATTN immediately enters PAUSE mode, and may be resumed by a GO command. Execution will resume at the procedure line <u>following</u> that in which the ATTN occurred.

It is forbidden to DELETE, RENAME, or UPDATE a ROSPROC while it is being executed. A ROSPROC is considered in execution either while it is in control of a terminal session, or when it is in suspended execution (pause state). A ROSPROC enters pause state when either a PAUSE command is issued by the procedure itself, or if the terminal user has interrupted execution of the procedure by use of the terminal ATTN. For example, if a LIST command is issued by a ROSPROC, and the terminal user interrupts the listing, then not only is the LIST processing terminated, but the issuing procedure is placed in pause state.

If, during execution or pause of a ROSPROC, a DELETE, RENAME, or UPDATE is issued naming the current procedure, the command will be rejected with the terminal diagnostic "DATA SET PROTECTION CHECK: NO ACTION TAKEN". The command can be reissued if first the EXIT command is entered at the terminal to exit from the procedure, (the user will receive the message "PROCEDURE CONCLUDED"), and then the rejected command reissued.

8. TRAP/TEST

Serious system errors (such as an I/O error or an error during SUBMIT processing) will not be TRAPped. Invalid commands and entries are also not TRAPped, since the use of a ROSPROC with the TRAP ON presupposes that the procedure has been thoroughly tested beforehand. If the TRAPped error is not TESTed, unpredictable results will occur.

When writing a ROSPROC for generalized use, it is recommended that the user be informed at appropriate points during execution how he can restart the procedure from the latest entry point should a serious error occur. Such information is also helpful if the procedure itself is lengthy, and the nature of the processing presupposes prior actions within the procedure. For example, if procedures A, B, and C have been successfully executed, it may be appropriate upon entering procedure D to type out a message informing the user that, in case of premature termination of procedure D, he may restart his operations by DOing procedure DRST.

The TRAP facility has another use in addition to TRAPping possible user errors. Since the results of certain I/O operations can be TRAPped, it is possible to set up conditional FIND and SCAN routines — for example, if a given library data set is present/not present in the AWS, take a certain action; if a given character string is present/not present in the AWS, take a certain action; etc. Since both the reply buffer and the contents of the AWS can, in effect, be subjected to substring scans and replacements, a ROSPROC can provide very powerful string handling capabilities. Thus, a data entry verification routine might be written as a ROSPROC, with those portions that cannot be performed by ROSCOE (such as hash totals, conversion, external file I/O, etc.) handled by a specially written MONITOR routine.

```
f mc.ptpch
list

0010 TYPEThis interactive procedure demonstrates in one simple case the use of
0020 TYPEROSCOE in prompting the user and aiding him in using OS facilities.
0030 TYPE
0040 TYPEThis procedure helps the user to print and/or punch his data sets.
0050 FETCH .MCPTPCH1
0060 TYPEDo you wish to PRINT or PUNCH ?
0070 REPLY
0080 COMPARE 'PRINT'
0090 IF EQUAL,.MCPTPCH,140
0100 COMPARE 'PUNCH'
0110 IF UNEQUAL,.MCPTPCH,60
0120 0006 //SYSUT2 DD SYSOUT=P
0130 0008  PUNCH MAXFLDS=1,TYPORG=PO
0140 TYPEPlease enter the data set name
0150 REPLY
0160 '0005 // DSN='
0170 TYPENow enter your JOB card
0180 REPLY
0190 '0001 '
0200 TYPEDo you wish to see your job stream before it is submitted ?
0210 REPLY
0220 COMPARE 'YES'
0230 SKIP 1,UNEQUAL
0240 LIST
0250 SUBMIT
0260 TYPEIf you wish to use this procedure again, DO.MCPTPCH
0270 TYPE AU REVOIR

fetch mc.ptpch1
list

0002 // EXEC PGM=IEBPTPCH
0003 //SYSPRINT DD SYSOUT=A
0004 //SYSUT1 DD DISP=SHR,
0005 //
0006 //SYSUT2   DD SYSOUT=A
0007 //SYSIN    DD *
0008  PRINT MAXFLDS=1,TYPORG=PO
0009  RECORD FIELD=(80)
0010 /*

_
```

EXAMPLE E-1. BUILDING JCL FOR IEBPTPCH

```
fetch sum
list
0010 TYPEENTER FIRST VALUE
0200 REPLY
0030 #SET #
0040 TYPEENTER THE SECOND VALUE
0050 REPLY
0060 #INCR #
0061 SET
0070 #TYPETHE SUM OF THE TWO VALUES = #
0080 DOSUM,10

_



do sum
ENTER FIRST VALUE
34
ENTER THE SECOND VALUE
67
THE SUM OF THE TWO VALUES = 0101
ENTER FIRST VALUE
?
(Procedure Concluded)

_
```

EXAMPLE E–2. SIMULATED DESK CALCULATOR

Simulated desk calculator with execution shown.

```
fetch ai.*

list

0001 b
0010 TYPEDO YOU WANT TO SET TABS? &REPLY &COMPARE "NO" &SKIP 3,EQUAL
0020 TYPE ENTER YOUR TAB SETTINGS &REPLY &"TAB "
0030 'TYPE TABS SET AT '
0031 TYPESET YOUR TYPEWRITER TABS. WHEN READY, TYPE GO&PAUSE
0040 B &TYPEYOU ARE NOW USING ONLY UPPER CASE LETTERS.
0050 TYPEDO YOU WANT TO USE BOTH UPPER AND LOWER CASE? &REPLY
0060 COMPARE¢NO¢&SKIP1,EQUAL&COMPARE¢no¢&SKIP1,equal
0070 X &TYPEYou can now use both UPPER and lower case.
0080 D &SET 0 &TYPE BEGIN ENTERING DATA, TYPE STOP TO STOP
0090 INCR1&REPLY&COMPARE"STOP"&SKIP1,EQUAL&COMPARE"stop"&SKIP1,equal&'0 '&DO.AI*,90
0100 PAUSE
0110 COMPARE 1 &SKIP 1,UNEQUAL &TYPE NO DATA ENTERED YET &DO.AI*,80
0120 PAUSE
0130 TYPEDO YOU WANT TO SAVE THE DATA? &REPLY &COMPARE 'NO'
0140 skip1,equal&compare¢no¢&skip1,equal&typeenter 6 letter name&reply
0150 'UPDATE ' &D &'10 LIST ' &U $$$$$$ &SKIP 4
0160 TYPELIST?&REPLY&COMPARE"NO"&IFEQUAL,.AI*,100,100&COMPARE"no"
0161 ifequal,AI.*,100,100
0170 CALL .AI*,210,210
0180 LIST &SKIP 6
0190 TYPELIST THE DATA? &REPLY &COMPARE "YES" &SKIP 1,EQUAL
0200 SKIP 3
0210 DECR1 &SET &'TYPE THERE ARE ' LINES &TYPE
0220 CALL $$$$$$
0230 TYPEYOUR DATA IS SAVED. TO CREATE ANOTHER SET, DO.AI*,80
0240 PAUSE
0250 TYPETO SAVE THE DATA DO .AI*,130. &PAUSE

off
LINE #02 - LIBRARY SIZE 08235 - TIME IN 10.31.19, ELAPSED 00.05.52
(Terminal has been Signed Off)
_
```

EXAMPLE E–3. AUTOMATIC DATA ENTRY.

This ROSPROC packs multiple commands on a line, and uses almost every command in the ROSCOE repetoire.

```
0010 TYPE This ROSPROC is called RO.SCAN &B
0020 TYPE Enter your response, please &REPLY &SET 1
0030 INCR 3 &COMPARE 78  &SKIP 1,LOW
0040 TYPE The argument was not found at all.  &EXIT
0050 DECR 3 &COMPARE /ABC/0,3
0060 SKIP 1,EQUAL
0070 INCR 1  &DO RO.SCAN,30
0080 REPLY /XYZ/0,3
0090 #10 #
0100 SET
0110 #TYPE Match found in column # of your response
0120 REPLY 10
0130 TYPE And was replaced by XYZ as follows:
0140 #TYPE#
0150 PAUSE
```

EXAMPLE E–4. SUBSTRING SCAN THROUGH REPLY BUFFER

APPENDIX F

3270 TERMINALS

APPENDIX F

3270 LOCAL DISPLAY SYSTEM

This appendix discusses ROSCOE's special handling of the 3270 display station. Full details on how to operate the display station and its special editing functions are contained in the IBM publication, *Operator's Guide for IBM 3270 Information Display Systems.*

1. At any moment during the ROSCOE session, the screen is formatted with two types of fields – *protected* and *unprotected.* A *protected* field consists of a whole line or series of lines on the screen which cannot in any way be modified by operator action. An *unprotected* field consists of a line or series of lines on the screen which can be modified in any manner desired by the operator. Modification consists of entering characters into blank locations, or changing characters already on display. There are various keys on the 3270 keyboard that are used to facilitate the entry of data. These keys are UP, DOWN, RIGHT, LEFT, BACKSPACE, BACKTAB, TAB, NEWLINE, DELETE, and INSERT. However, note that the DUP (duplicate) key cannot be used, and the special character generated by it will be changed by ROSCOE to a space.

2. There are two standard formats on the screen: normal read-write and special Autoline/EDIT-LIST. The latter format is used only when the NUMBER or EDIT-LIST command is being executed. At all other times, the first format is used.

3. NORMAL READ-WRITE. All output from ROSCOE will be displayed as a protected field occupying lines 7 through 24 on the screen. All operator input will be taken from lines 1 through 6, which will be formatted as 6 separate unprotected fields. At the end of every write operation, the cursor will be positioned to the top left-hand corner of the screen. The first position of every input line is occupied by a special non-displayable attribute character, which cannot be overwritten (attempts to do so will result in keyboard lockout). Likewise, the last (rightmost) position of line 6 is occupied by an attribute character. All command and data entry will therefore begin in the second position of each line and extend through position 80 of each line, except for line 6, where data can extend only through position 79. When ROSCOE is not ready to accept input from the screen, the keyboard will be disabled. When it is ready to accept the next input, ROSCOE will unlock the keyboard. Keyboard disable is marked by the INPUT INHIBITED indicator on the right of the screen.

4. AUTOLINE/EDIT-LIST. If SCALE is ON, the scale line will be displayed on lines 1 and 24 as protected fields. The Autoline prompting numbers, or the EDIT-LIST sequence numbers, will be displayed as protected fields in the first position of alternate lines. The intervening lines are unprotected fields which are completely blank for Autoline entry, or contain data from the AWS for EDIT-LIST modifications. Please note a peculiarity of the 3270 display station: if an unprotected field has not been modified by the operator, it is not read in. Therefore, since Autoline prompts are regenerated when data is read in, an unmodified field (a line on the screen) is ignored and will not generate an AWS data line of all spaces.

5. To move the cursor to the first position of the next unprotected field on the screen, use the NEWLINE key. If there are no more available fields, the cursor will wrap around to the first position of the first line. ROSCOE ignores the cursor location when reading data from the screen. The TAB key may also be used to position the cursor in the next field.

6. When the operator has filled the screen and wishes to transmit it to ROSCOE, the ENTER key is used. Likewise, when ROSCOE is waiting at the end of a page of output for the signal to display the next page, the ENTER key will cause the next page to be displayed.

7. The ATTN function is indicated in either of two manners. It can be signaled by using any

Program Attention key except ENTER. (The Program Attention keys are CLEAR, CNCL, TEST REQ, all Program Function (PF) keys and Program Access (PA) keys.) ROSCOE will terminate the current command and unlock the keyboard again to accept the next input. This ATTN is the only means to exit from the EDIT-LIST command or Autoline mode. The other way to indicate ATTN, used when ROSCOE is pausing at the end of one page for the signal to display the next page, is by entering the next input (commands) in lines 1 through 6 of the screen and using the ENTER key. ROSCOE will terminate the current command, and begin execution of the next data entered. Please note that the use of the light pen is inhibited at all times.

8. The BACKSPACE function is provided by the normal BACKSPACE, BACKTAB, etc., keys which move the cursor back to the characters to be changed.

9. Extended (X) and Basic (B) character modes apply to a 3270 screen, with the hardware peculiarity that the display station is capable of displaying only upper case alphabetic characters. However, both upper and lower case alphabetic characters can be entered at the keyboard. Users are cautioned to be aware at all times which mode they are using (B is preset at log-on time). ROSCOE automatically translates all commands, library data set names, and required alphabetic operands to upper case before interpreting them.

10. The TAB function is provided by designating any special character as a TAB mark. The mark is preset to a NOT-sign, but may be redefined at any time by the command:

    ```
    TAB /
    ```

    where / is a special character to be recognized as the TAB mark.

11. A line on the screen corresponds to a ROSCOE entry; that is, no command may be broken over two lines.

12. Up to 6 commands in normal read-write mode can be entered at one time. ROSCOE will execute them in the order presented, top to bottom, but any command which causes output to the screen will terminate the current command sequence.

13. The user is cautioned that ROSCOE suspends operations after the 18th line has been displayed on the screen. Therefore, he must be attentive to RETURNing during listings, especially when receiving output from one of the syntax checkers which are normally unavailable to other users of the system when already in use by a given user.

14. To log on, use the ENTER key. To erase invalid entries before transmitting the screen, the ERASE EOF and ERASE INPUT keys may be used. The former clears an unprotected field from the cursor to the rightmost position in the line. The latter clears (resets) all unprotected fields to nulls and moves the cursor to the top left of the screen. When an attempt is made to overwrite a protected field or attribute character, a keyboard lockout results. Use the RESET key to unlock the keyboard.

APPLIED DATA RESEARCH, INC.

Route 206 Center, CN-8 • Princeton, New Jersey 08540 • (609) 924-9100

U.S. OFFICES

BOSTON, MASS., 26 Princess Street, Wakefield, Mass. 01880 (617) 245-9540
CHICAGO, ILL., Suite 109, 1550 Northwest Hwy., Park Ridge, Ill. 60068 (312) 775-9855
CLEVELAND, OHIO, 18615 Detroit Avenue 44107 (216) 228-0880
HOUSTON, TEX., Suite 538, 3801 Kirby Dr. 77006 (713) 526-3188
LOS ANGELES, CALIF., Suite 800, 11661 San Vicente Blvd. 90049 (213) 826-5527
NEW YORK, N.Y., 360 Lexington Ave. 10017 (212) 986-4050
PRINCETON, N.J., Route 206 Center 08540 (609) 924-9100
WASHINGTON, D.C., 800 Follin Lane, Vienna, Virginia 22180 (703) 281-2011

FOREIGN OFFICES (Sales Representatives)

AUSTRALIA, MELBOURNE (Carlton, Victoria), International Programming Pty. Ltd.
SYDNEY (Crows Nest, N.S.W.), International Programming Pty. Ltd.
AUSTRIA, VIENNA, CPP Austria
BELGIUM, BRUSSELS, CPP Belgium
BRAZIL, RIO DE JANEIRO, Control Data Do Brasil LTDA
SAO PAULO, Control Data Do Brasil LTDA
CANADA, MONTREAL, Multiple Access Limited
TORONTO, Multiple Access Limited
DENMARK, VALBY, Scan Data A/S
ENGLAND, LONDON, CAP England
FINLAND, HELSINKI, Finnsystems OY
FRANCE, PARIS, CAP/SoGETI
GERMANY, DUSSELDORF, CPP Germany
MUNICH, CPP Germany
HOLLAND, AMSTERDAM, CPP Holland
ISRAEL, TEL-AVIV, Advanced Technology, Ltd.
ITALY, MILANO, Syntax, S.p.A.
ROMA, Syntax, S.p.A.
JAPAN, OSAKA, Data Electronics Kaisha, Ltd.
TOKYO, Data Electronics Kaisha, Ltd.
KOREA, SEOUL, Korea Business Service Co., Ltd.
MEXICO, MEXICO CITY, Equipos Cientificos y de Computacion, S.A.
NEW ZEALAND, LOWER HUTT, Systems and Programs (NZ) Limited
NORWAY, OSLO, Effektiv Databehandling A/S
PHILIPPINES, MANILA (Makati, Rizal), Decision Systems Corporation
PUERTO RICO, SANTURCE, Management Data, Inc.
REPUBLIC OF SOUTH AFRICA, JOHANNESBURG (Sandton, Transvaal), Systems Programming (Pty.) Ltd.
SINGAPORE, Singapore Computer Systems Services (Pte.) Ltd.
SPAIN, MADRID, Entel Ibermatica
SWEDEN, STOCKHOLM, Systematik AB
SWITZERLAND, ZURICH, CPP Switzerland
TAIWAN, TAIPEI, International Data Applications, Inc.
THAILAND, BANGKOK, DATAMAT, LTD.

YOUR ADR® REPRESENTATIVE IS: