DEC-10-YUFA-D

decsystem10

SOUP (SOFTWARE UPDATING PACKAGE) PROGRAMMER'S REFERENCE MANUAL

This manual reflects the software as of Version 3 of SOUP.

digital equipment corporation · maynard. massachusetts

1st Edition September 1970
Update Pages October 1973

Copyright © 1970, 1971, 1972, 1973 by Digital Equipment Corporation

The material in this manual is for informational purposes and is subject to change without notice.

Actual distribution of the software described in this manual will be subject to terms and conditions to be announced at some future date by Digital Equipment Corporation

DEC assumes no responsibility for the use or reliability of its software on equipment which is not supplied by DEC

The software described in this manual is furnished to purchaser under a license for use on a single computer system and can be copied (with inclusion of DEC's copyright notice) only for use in such system, except as may otherwise be provided in writing by DEC.

The following are trademarks of Digital Equipment Corporation, Maynard, Massachusetts:

DEC	PDP
FLIP CHIP	FOCAL
DIGITAL	COMPUTER LAB

CONTENTS

CONTENTS (Cont)

ILLUSTRATIONS

TABLES

FOREWORD

This manual describes the programs which constitute the Software Updating Package (SOUP). Chapter 1 introduces SOUP, describes the programs, and gives the equipment requirements. Chapter 2 describes the updating of DEC system software that has not been modified by the user. For users who have not made changes to the system software, Chapter 2 is all that is necessary to update their system files.

Chapters 3, 4, 5, and 6 describe the updating of user versions of system files. Chapter 3 gives the format of the correction file used in the updating process. Chapters 4 and 5 describe the various options of the SOUP programs, and Chapter 6 shows examples of updating files.

Two appendices have been included in this manual. Appendix A summarizes the switches that can be included in the command strings, and Appendix B lists the messages which may be typed during updating runs.

It is assumed that the reader has some knowledge of the PDP-10 computer system and its associated software.

ACKNOWLEDGEMENT

SOUP was originally developed at the Andrew R. Jennings Computer Center at Case Western Reserve University, Cleveland, Ohio. Digital Equipment Corporation gratefully acknowledges the work done at Case Western Reserve.

Chapter 1
Introduction

Software Updating Package (SOUP) is a set of programs that aids both Digital Equipment Corporation and the user in the updating of system source files.

1.1 PURPOSE

Software in the field is continually being updated to reflect additions and improvements made by the manufacturer. To make the updating process faster and easier for both DEC and the user of DEC equipment, SOUP has been implemented. SOUP removes the necessity of redistributing an entire software routine when changes to it, although significant in operation, represent a relatively small percentage of the code. When SOUP is employed, DEC can send to PDP-10 users a small file containing only the differences in a compressed editing format. The correction file is generated by one of the SOUP programs; another SOUP program edits the user's copy of the system file by making the changes specified in the correction file.

However, some PDP-10 users also maintain a separate set of system files that are modified to some degree to reflect the needs of their individual installation. These users, in addition to updating their DEC system files, should update their own versions of these files to include the revisions from DEC. SOUP is designed to update the user's system files as well as the DEC system files. Figure 1-1 illustrates the updating of both the system files and the user versions of the files.

Although the material in this manual is confined to a description of updating system files (both DEC and user), SOUP can be used to update any source file that has more than one version.

1.2 SOUP FILES

SOUP operates only on ASCII files that do not have line sequence numbers. SOUP numbers each line in the file for reference purposes, but these numbers are never added to the source file. A line of a

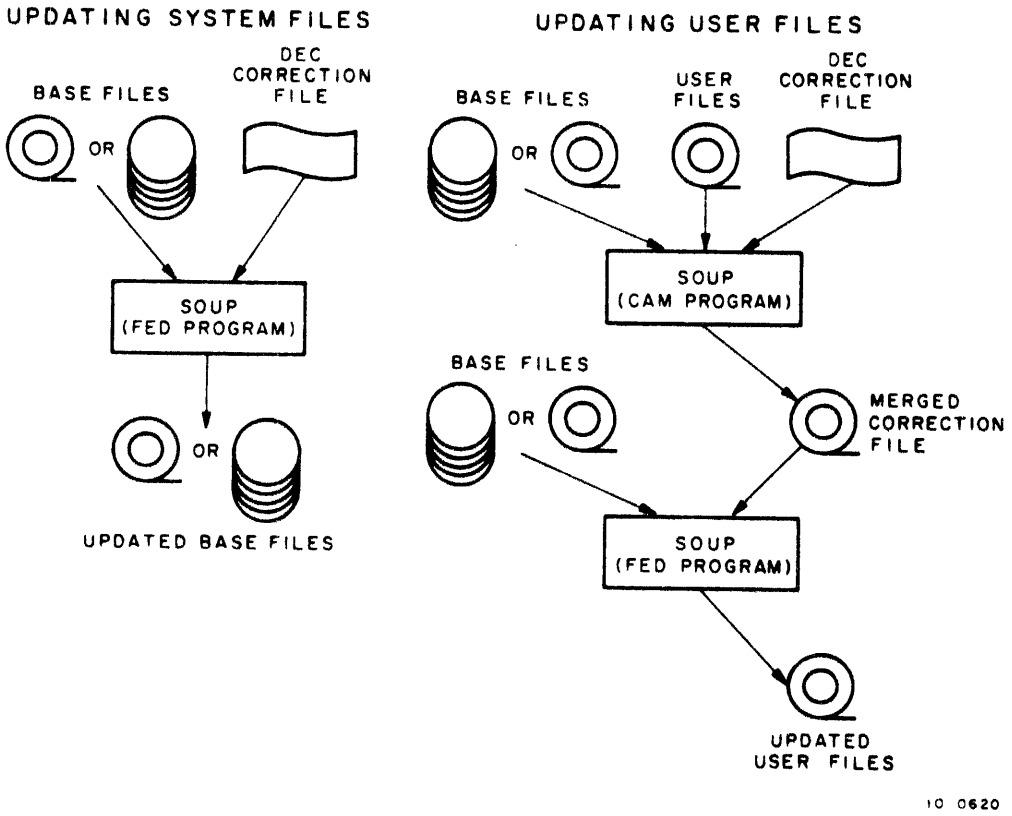

Figure 1-1 Updating Files Using SOUP

source file, as defined in a SOUP operation, is a string of ASCII characters terminated by a feed character. A feed character can be a line-feed, form-feed, or vertical-tab; SOUP treats them all as end-of-line characters.

The DEC system file to be updated is referred to as the base file; a group of base files is referred to as the base complex. The user's version of the system file is called the user file (or user-new file) and a group of user files is called the user complex. The correction file produced by comparing a user file with a base file is called the user correction file. The correction file sent by DEC is called the DEC (or manufacturer's) correction file. The final output of SOUP is referred to as the updated base file or the updated user file.

1.3 PROGRAMS CONTAINED IN SOUP

SOUP comprises three programs: CAM, FED, and COMP10.

1.3.1 CAM Program

The CAM program performs two functions. It compares the new version of the base file to the previous base file to produce a correction file (COMPARE option). The correction file contains a series of editing changes that reflect the differences between the old and new base files. CAM can also merge two correction files derived from the same base file to produce a single correction file (COMERGE option). If conflicts are detected between the two correction files, CAM notes them in the merged correction file and in a listing file. The user can manually edit the file to resolve the conflicts.

CAM can perform the COMPARE and COMERGE functions simultaneously by comparing base and user files and, at the same time, merging the user corrections with the DEC corrections. The output from CAM is always a correction file.

The CAM program can be run as COMPARE-only, COMERGE-only, or as COMPARE and COMERGE together. The option to be run is determined by the form of the command string to CAM. All three forms of CAM are initialized in the following manner.

```
.R CAM
* command string
```

1.3.2 COMP10 Program

Occasionally, the user file has been revised so drastically that the CAM program cannot compare it to the base file due to buffer overflow. The COMP10 program is used in lieu of CAM in this case because COMP10 has extremely large buffers. The output from COMP10 is also a correction file. The program is initialized as follows.

```
.R COMP10
* command string
```

1.3.3 FED Program

The FED program performs the final edit on the base file. It reads the correction file produced by CAM or COMP10 and makes the changes to the base file that were specified in the correction file, thus creating an updated file. FED can be used both to apply DEC's corrections to the base files and to apply the merged user/DEC corrections to the base files. Initialization of the FED program is performed in the following manner.

```
.R FED
* command string
```

1.4 COMMAND CONVENTIONS

The command strings for the SOUP programs are described with the applicable program, because they differ according to the program and the way in which it is run. However, certain conventions apply to the command strings for all the SOUP programs.

Switches can be inserted into a command string after filenames. They are indicated by a slash (/) followed by the switch name or by one or more switch names enclosed in parentheses. The latter method is used to include an identifier of up to three characters enclosed in quotation marks in the command string for CAM. This identifier is placed in the correction file on the lines which contain corrections from the user and in the listing of the updated file produced by FED. All the switches are summarized in Appendix A.

Tabs and spaces can be entered into the command string at any place for clarity and ease of reading. A semicolon can be used to continue a command on the next line. The semicolon causes the program to ignore the remainder of the line (including the carriage return) and to type an asterisk on the next line. The semicolon is also used to add comments to the command string. The program ignores these comments. A project-programmer number, enclosed in square brackets, can be written in the command string so that files on a disk area other than that of the current user can be accessed.

A command file containing any number of command strings can be written for either CAM or FED. When one of these programs is initialized, the command string consists of the device name and the name of the command file in the form:

```
*dev:file.ext@
```

The program reads each command string from the file and performs the designated operations until the command file is exhausted.

Errors in the command string cause a SOUP program to retype the command with a vertical arrow (↑) positioned below the approximate location of the error. Appendix B describes error conditions in detail.

1.5 EQUIPMENT REQUIREMENTS

The following core and input/output devices are necessary for SOUP operations.

a. Core Memory:

   (1) CAM - 9K of core (4K for high segment, 5K for low segment)
   (2) FED - 4K of core (3K for high segment, 1K for low segment)
   (3) COMP10 - 11K of core for low segment

b. Input/Output Devices

(1) Three input devices:

(a) one for base files (directory device)
(b) one for user files
(c) one for the correction file

(2) Three output devices:

(a) one for updated files
(b) one for listing files (optional)
(c) one for log files (normally the terminal)

Some devices are not required for certain SOUP options. The required devices are described for each option in the appropriate chapter.

Chapter 2
Updating System Files

DEC sends PDP-10 users correction files on paper tape or DECtape to update system source files. It is assumed that all installations have the most recent copy of each DEC system file to use as a base file.

Every user who wants to keep his copies of DEC system files up-to-date, whether or not he writes his own versions of system programs, must follow the procedures given in this chapter. Subsequent chapters describe the use of SOUP to update user versions of system files.

Updating the system files with DEC's correction file requires only the FED program. FED reads the correction file, which comprises one or more subfiles, and makes the specified editing changes to one or more base files to produce an updated base file or base file complex. Editing changes from the correction file can include insertion, deletion, and replacement of lines in each base file. In addition to making changes to files, FED can insert new files into the base complex and delete old files from the base complex according to the subfile headers in the correction file.

2.1 COMMAND STRING TO FED

After FED is initialized, the command string is written to indicate the input and output devices. The general form of the command string for FED is:

```
new-file dev:,list dev:,dev:log.ext←base dev:,dev:corr.ext.
```

The default conditions for the command string are as follows.

a. If the listing device is omitted, no listing is produced.
b. If the log device is omitted, the terminal is assumed.
c. If the name of the log file is omitted, it is assumed to be LOG.G01.
d. If the name of the correction file is omitted, DECCOR is assumed.

No other device or filename can be omitted from the command string. If the listing device is omitted and the log device is written, an additional comma must follow the new-file device in the command string.

Examples:

```
.R FED
*DTA6:,DSK:,TTY:LOG.G01 ← DTA4:,PTR:

.R FED
*DSK:,LPT: ← DSK:,DTA0:DECCOR
```

2.2 FED INPUT

Input to FED consists of the DEC correction file and a base file or base file complex.

The correction file contains one or more subfile headers: one for each file to be corrected, followed by the corrections; one for each file to be deleted; and one for each file to be inserted, followed by the insertion file. The correction file is normally read from paper tape or DECtape; however, it can be read from any input device.

The base file complex consists of one or more files that must reside on a directory device. The device can be a single project-programmer area on disk or a series of one or more DECtapes mounted successively on the same drive. The names of the base files need not be specified in the command string because FED reads them from the subfile headers in the correction file.

2.3 FED OUTPUT

The output from FED includes the updated base files, listing files, and a log file. The output files can be placed on any output devices.

The names of the new files are not specified in the command string, because the names are taken from the subfile headers in the correction file. The listing files are line-numbered listings of the updated base files. Those lines added to the base file are designated in the listing file by *NEW* in the right margin. Deletion of lines is indicated in the right margin by a decimal number preceded by a minus sign (e.g., -3 means that three lines were deleted from the old base file). The listing files have the same names as the new base files except that the second character of the extension is changed to L. For example, if the base file is named BASE.MAC, the listing of that file is named BASE.MLC.

The log file contains the messages from FED that are normally written at the terminal while FED is being executed.

2.4 MESSAGES FROM FED

During execution of FED, the messages described below are written to the log device, which is normally the terminal. When a base file has been edited, FED types the following message:

```
FED COMPLETED ON FILES -Inew-file.ext ←old-file.ext
```

If a file is inserted into the updated base complex, FED types the message:

```
FED COMPLETED ON FILES -Ifile.ext ←
```

and if FED deletes a file from the old base file complex, this message is typed:

```
FED COMPLETED ON FILES -I ←file.ext
```

When the base file complex is contained on more than one DECtape, the following message is typed when the input from each DECtape is exhausted.

```
***MOUNT NEXT INPUT DECTAPE ON UNIT DTAn
```

Control is returned to the monitor. If there is more input, the user must:

Step	Procedure
1	Mount the next input tape.
2	Type ASSIGN DTAn)
3	Type CONT) and when FED responds with an asterisk
4	Type D or DONE) to indicate that the tape is mounted and processing can continue.

If no more input exists, the user must:

Step	Procedure
1	Type CONT) and when FED responds with an asterisk,
2	Press the CARRIAGE RETURN key to terminate processing.

To signal that the end of the correction file has been reached, FED types the message:

```
***FED COMPLETED SUCCESSFULLY
```

The user must wait until FED types an asterisk after this message before returning control to the monitor because all output has not been completed until the asterisk is typed.

2.5 SAMPLE UPDATING RUN

Assume that DEC has sent a paper tape containing corrections to the source file PIP.30. The user's copy of PIP.30 is on DECtape and he wishes the updated version to be placed in his area on the disk. The log file will be written on the terminal, and the listing of the new file will be written on the line printer. The user should perform the following steps.

Step	Procedure
1	Mount the DECtape containing PIP.30 on drive 3.
2	Put the paper tape in the reader.
3	Call FED and type the command string in response to the asterisk.

```
.R FED
*DSK:,LPT:←DTA3:,PTR:
```

When the file has been edited, FED prints the messages:

```
FED COMPLETED ON FILES -!PIP.30 ← PIP.30
***FED COMPLETED SUCCESSFULLY
*
```

To return control to the monitor, the user types CTRL-C. Figure 2-1 illustrates the updating run on PIP.30.

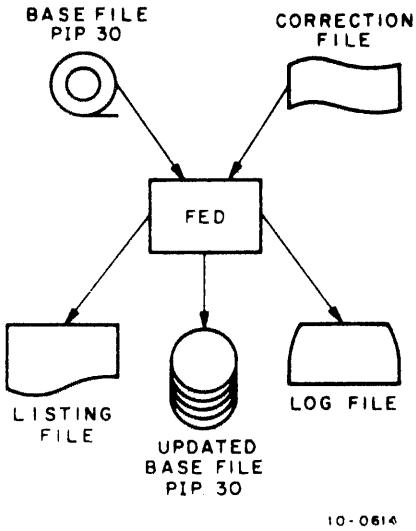

Figure 2-1 Sample Update of a System File

Chapter 3
Format of the Correction File

The correction file produced by CAM or COMP10 and used as input to FED is described in detail in this chapter. Any user who has his own versions of system files can update these files with the corrections from DEC by means of the SOUP programs. However, conflicts can arise between the changes made by the user and the changes made by DEC. The user must resolve these conflicts and edit the correction file accordingly.

Correction files consist mainly of command lines and insertion lines. Comments, messages, and SYNC lines can also be present in correction files. SYNC lines are used by FED to check that the base and correction files match. These lines are described below.

Command lines are distinguished from insertion lines by a beginning minus sign. The FED program interprets any line beginning with a minus sign as a command. Because an insertion line could contain a leading minus sign, CAM places an additional minus sign on any line from the source file which has a leading minus sign. The double minus sign indicates to FED that this is an insertion line. FED removes the extra minus sign before inserting the line into the updated file.

3.1 COMMAND LINES

Command lines contain instructions to the FED program to perform operations on the base file. Command lines can be subfile header lines, editing lines, message lines, or SYNC lines. A command line can contain an identifier that was placed on the line by the CAM program or by the user when he is editing the file.

3.1.1 Subfile Header Lines

A subfile header line indicates the names of the files being processed. One subfile header is present for each file. A header line is distinguished by an exclamation point (!) following the minus sign. Three kinds of subfile headers can be present: file correction headers, file insertion headers, and file deletion headers.

3.1.1.1 File Correction Header - This subfile header indicates that the named file is to be corrected or is to be transferred unchanged from the base device to the updated file device. The command line consists of a minus sign and exclamation point (-!) followed by the new file name, a back arrow, and the old file name.

Examples:

```
-!PIP.V11←PIP.V10
-!SOUP.MAC←SOUP.MAC
```

3.1.1.2 File Insertion Header - This subfile header indicates that a file not on the base device is to be transferred from the correction file to the updated file device. This command line is not produced by CAM or COMP10; the user must place the header and its associated file into the correction file by means of PIP or TECO. Care must be taken that each line in the new file that begins with a minus sign is given an additional minus sign. (TECO can be used to do this.) Also, if TECO is used to edit the file, commands that do not add form-feeds to the file should be used (e.g., N and EX).

The file insertion header consists of a minus sign and exclamation point followed by the new filename and a back arrow.

Examples:

```
-!NEW.00←
-!PATEN.MAC←
```

3.1.1.3 File Deletion Header - This subfile header indicates that a file currently on the base device is not to be copied to the updated file device. The file is thus deleted from the updated version of the base complex. A file deletion header consists of a minus sign and exclamation point followed by a back arrow and the filename.

Examples:

```
-!←DEDPRG.99
-!←DATED.999
```

3.1.2 Message Lines

Message lines are commands that contain text enclosed in quotation marks. The text is typed at the terminal when the line is processed by FED. The text is ignored by FED and is not copied to the updated file. Message lines contain a minus sign followed by text in quotation marks. Message lines are not produced by CAM or COMP10; the user places them in the correction file when he is editing it.

Examples:

```
-"OPERATOR: PLEASE MOUNT NEW BASE DECTAPE"
-"FILE NEW.001 ADDED TO NEW BASE COMPLEX"
```

After the message is typed, control is returned to the monitor. To resume FED operations, the user must:

1. Type CONT) and when FED responds with an asterisk,
2. Type D or DONE)

3.1.3 Editing Commands

Editing commands cause lines to be inserted into, deleted from, or replaced in the updated file. Two types of editing commands can be present: insertion commands and deletion/insertion commands.

3.1.3.1 Insertion Command - The insertion command consists of a minus sign followed by a line number from the base file. The lines to be inserted after this line in the base file follow the insertion command.

Example:

```
-15
THIS IS THE FIRST LINE TO BE INSERTED
THIS IS THE SECOND LINE TO BE INSERTED
--THIS SOURCE LINE BEGINS WITH A MINUS SIGN
```

The three lines in the example are inserted into the updated file between lines 15 and 16 of the base file.

3.1.3.2 Deletion/Insertion Command - This editing command causes a line or lines of the base file to be deleted and replaced in the updated file with any lines that follow the command. The deletion/insertion command consists of a minus sign followed by two line numbers separated by a comma. Insertion lines, if any, follow the command. If a single line is to be deleted, its number is written twice in the command, separated by a comma.

Example:

```
-20,30
THIS LINE WILL BE INSERTED TO REPLACE LINE 20

THIS LINE WILL REPLACE LINES 21-30
-50,50
```

Lines 20 through 30 are deleted and replaced with the insertion lines that follow the command. Line 50 is also deleted.

3.1.4 SYNC Lines

After every fifth editing command, CAM inserts a SYNC command line into the correction file. This line is used for comparison with the base file to ensure that the correction file is being applied to the proper base file. The SYNC line contains a minus sign followed by the SOUP line number of the base file and a colon (:). Following the colon is the line as it appears in the base file. The FED program checks that the SYNC line matches the line in the base file. If the lines do not match, an error message is issued, and FED waits for action from the user.

Example:

```
-123:  LABEL:    ADD 10,TABLE1(2)
```

The characters in line 123 of the base file must be identical to the characters following the first colon of the SYNC line or FED issues an error message.

3.2 COMMENTS AND COMMENT LINES

Comments can be written into the correction file by the user when he is editing the file. CAM and COMP10 do not write comments in the correction file (unless they are comment lines in the new source file). The comments in a correction file can be placed on the same line as the commands as long as they are separated from the commands by spaces and/or tabs. A comment can also be placed on a separate line. In this case, it is preceded by a minus sign. Comments are bypassed by FED and are not copied into the updated file.

Examples:

```
-521                JUST A COMMENT IS ADDED
;  THIS ROUTINE     READS IN DTA DIRECTORY
-593,597            SUPPRESSES CALL TO SUBR A
-603,607            JUST DELETES BLANK LINES
- *** REST OF THIS ROUTINE IS UNCHANGED
```

3.3 CONFLICTS

When two correction files for the same base file are combined by means of the COMERGE portion of CAM, conflicts can arise. For example, one correction file can delete a line from the base file and replace it with another line while the other correction file has kept the line and added another. Every

time that CAM detects a line that is referenced by both correction files, it notes a conflict and places a message to that effect in the correction file. A conflict, as recognized by CAM, is a reference to the same line number by both correction files, even if the same operation is being performed on the line. CAM does not detect logical conflicts. That is, if one correction file changes a portion of coding such that the program does not operate in the desired manner, as long as the other correction file does not reference the same line numbers, CAM does not detect a conflict. If there is any possibility that logical conflicts could occur, the user should check the base file and his own version of the base file. Procedures for correcting conflicts and obtaining a listing of the base file are described below.

3.3.1 Indication of Conflicts

Conflicts are indicated in the correction file by CAM as illustrated in the following example. These conflicts must be resolved and the message removed from the correction file before FED can be run.

```
-/-/-/-/  BEGINNING OF CONFLICT 3 \-\-\-\-\
-20,21
B INSERT
C INSERT
D INSERT
-20,20
A INSERT
B INSERT
C INSERT
-/-/-/-/   END OF CONFLICT 3 \-\-\-\-\
```

In conflict 3, the first correction file (the user corrections) deleted lines 20 and 21 of the base file and replaced them with lines B, C, and D. The second correction file (the DEC corrections) only deleted line 20 and replaced it with lines A, B, and C. The user must decide how he wants this conflict resolved; however, this conflict and all conflicts must be resolved before FED can be run.

3.3.2 Editing the Correction File

Resolving conflicts or adding commands to the correction file is commonly done by means of TECO. Consult the TECO manual for a description of TECO operations. Care must be taken that no extraneous form-feed characters are added to the file by TECO, especially after file headers or in insertion lines. One method to avoid adding form-feed characters is as follows.

Step	Procedure
1	Place the entire correction file in core with the TECO command Y1000 <A> $$
2	Edit the file as desired.

Step	Procedure
3	Insert a CONTROL-Z (↑Z) as the last character in the file.
4	Output the entire file with the TECO command HP ↑G$$

When the file is processed by FED and a CONTROL-Z has been added, the /Z switch must be included in the command string to cause FED to recognize CONTROL-Z as the end-of-file marker.

3.4 SOUP LINE NUMBERS

The programs in SOUP use line numbers to refer to individual lines in the base file. Both FED and CAM keep an internal line count for the lines in the base file but do not add these line numbers to the source coding. These line numbers are not the same as those produced by CREF.

When CAM edits the base file and writes the correction file, it refers to a line from the base file by its number. FED reads these editing changes and corrects the specified lines in the base file. FED then renumbers the lines in the listing file to make them sequential. Thus, a line-numbered listing of the old base file and a line-numbered listing of the new base file will not necessarily have the same line numbers on the same lines of code.

Because the correction file refers to a line in the base file only by its SOUP line number, some confusion can arise when editing the correction file. If there is a question as to what line of source coding is being referenced, the user should obtain a line-numbered listing of the old base file. This can be accomplished by performing the following steps.

Step	Procedure
1	Run FED using the terminal as the correction file input device.
2	Type a file insertion header naming the base file as the old and new files.
3	Press the CONTROL and Z keys.

The listing file will contain the line-numbered listing of the base file.

Example:

```
.R FED
*DSK:,LPT:←DTA0:,TTY:)
-!BASE.MAC←BASE.MAC)
↑Z)
```

Chapter 4
CAM and COMP10

CAM is the SOUP program which compares base files and merges correction files. COMP10 is a simplified version of CAM with a larger buffer and is used only to compare very dissimilar files. CAM contains two portions, COMPARE and COMERGE. The COMPARE portion can be used alone to compare a base file and a user version of the base file to produce a correction file. The correction file sent from DEC is the output from COMPARE. The COMERGE portion can also be used alone to merge two correction files which have been derived from the same base file. The two portions of CAM can be used together to perform both comparison and merging. Three options are thus available when using CAM.

a. COMPARE-only - comparison of a base file or files and a user version of the base files to produce a correction file.

b. COMERGE-only - merging of two correction files derived from the same base file.

c. COMPARE and COMERGE (full CAM) - comparison of base and user files and simultaneous merging of the resultant corrections with a previously-generated correction file (e.g., the DEC correction file).

Output from all of the options of CAM and from COMP10 is a correction file which is used as input to the FED (final edit) program. The CAM program performs one of the above functions according to the way in which the command string is written.

Generally, CAM is used to update a user's version of a system file with the corrections issued by DEC. This is accomplished by comparing the base file with the user version of it and merging the resulting corrections with the correction file from DEC. The merged correction file is then applied to the old base file using the FED program to produce an updated user file. This procedure should be done separately from the updating of the base files.

In the COMPARE portion of CAM, lines are read from the base file and user file and are compared character by character. As long as the two files are identical, no output is produced. When a difference is detected, both the line from the base file and the line from the user file are stored, each in a separate buffer. Subsequent lines are read from each file alternately and compared each time to

all the lines present in the other file's buffer. When three consecutive lines from both files match, it is assumed that the end of the current difference has been found. An editing change is written into the correction file, the buffers are cleared, and line-by-line comparison is resumed. The editing change specifies the deletion from the base file of those lines in the base file buffer that precede the lines in the match, and the insertion at the same place of those lines in the user file buffer that precede the three lines in the match.

In the COMERGE portion of CAM, conflicts between the user corrections and the DEC corrections are detected and noted in the correction file. Two corrections are treated as conflicting by CAM if they both refer to the same line of the base file. The conflicts are noted in the log file and are put in the listing file and the correction file in the following manner.

```
-/-/-/-/  BEGINNING OF CONFLICT n  - - - -
      lines in conflict
-/-/-/-/  END OF CONFLICT n  - - - -
```

The merged correction file must be edited by the user to remove the conflicts before it can be input to the FED program. Note that CAM does not detect logical conflicts that result when the correction files are merged. The user must detect and correct logical conflicts himself.

When the correction files are merged, the subfile header written into the merged correction file for each subfile contains the name of the base file to which both files apply and the user file in the form:

```
-!user.ext←base.ext
```

If the last two characters of the extension of the user filename are numeric, CAM increments the number by 1. Thus, 19 is incremented to 20, and 99 to 00. For example, if the base filename is FILEA.B01 and the user filename is FILEA.U01, the subfile header in the correction file is written by CAM as:

```
-!FILEA.U02←FILEA.B01
```

Errors in the merged correction file are also noted by CAM. These errors appear when the user has either written or edited one of the correction files. If the correction files are produced by CAM (or COMP10), no errors appear.

4.1 COMPARE-ONLY

The COMPARE portion of CAM compares either single or multiple files. In single-file COMPARE, CAM compares a single-user file to a single-base file and produces a correction file with one subfile containing the changes to be made to the base file. In multifile COMPARE, CAM compares a group of user files to a group of base files to produce a correction file containing several subfiles. Files are processed automatically in the order of their appearance in the base directory. Figure 4-1 illustrates the COMPARE process.

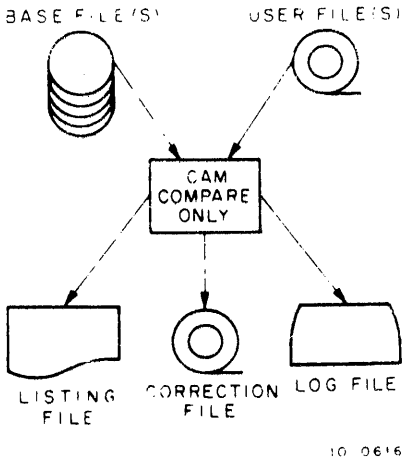

Figure 4-1 COMPARE Function of CAM

4.1.1 COMPARE Command Strings

The command string for single-file COMPARE differs from that for multifile COMPARE in that the input files are named. The general forms for both command strings are shown below.

The command string to CAM for single-file comparison has the general form:

dev:corr.ext,dev:list.ext,dev:log.ext ←dev:base.ext,dev:user.ext

The general form of the command string to CAM for multifile comparison is:

dev:corr.ext, dev:list.ext,dev:log.ext ←base dev:,user dev:

The default assumptions for the command strings are as follows.

a. If the name of the correction file is omitted, MERCOR is assumed.

b. If the listing device is omitted, no listing is produced.

c. If the name of the listing file is omitted, it is assumed to be LIST.L01.

d. If the log device is omitted, the Teletype is assumed.

e. If the name of the log file is omitted, it is assumed to be LOG.G01.

No other device or file name can be omitted from the command strings.

Examples:

```
.R CAM
*PTP:,LPT:←DTA4:FILEA.05,DTA5:FILEA.06
.R CAM
*DSK:,DSK:,DSK:←DTA4:FILEA.05,DSK:FILEA.06  ;NEW SWITCH

.R CAM
*PTP:,LPT:←DTA4:,DTA5:  ;REENTRANT CHANGES

.R CAM
*DSK:CUSP.C01,DSK:CUSP.L01←DTA4:,DSK: [11,107]
```

The switches that can be included in the command strings are described in Appendix A.

4.1.2 Input to COMPARE

Input to COMPARE consists of one or more base files and one or more user files. The base files must reside on a directory device. The user files can be read from any input device.

When single files are compared, the base file must be named in the command string. Naming of the user file in the command string is optional.

In multifile COMPARE, the base files must be the only files present on a directory device. Neither the base file complex nor the user file complex can be named in the command string. Each user file should have the same name as the base file to which it will be compared (the extension can be different). If the user files are on a nondirectory device, the name of each file is assumed to be the same as that of the base file. Also the files on a nondirectory device must be in the same order as the base files.

Each base file is matched with a similarly-named user file. CAM first attempts to find a user file with the same name and extension as the base file. If this fails, CAM then searches for a file of the same name disregarding the extension. The first such file is chosen.

4.1.3 Output From COMPARE

The output from the COMPARE portion of CAM consists of a correction file, a listing file, and a log file. All the output files can be written to any output device.

The correction file contains one or more subfiles, each of which has a header indicating that the file is to be corrected, transferred unchanged to the updated file complex, or deleted from the updated file complex. If a file is to be corrected, the corrections follow the subfile header. In single-file COMPARE, the correction file contains one subfile containing the editing changes to be made to the base file.

The listing file is a line-numbered listing of the correction file and is normally written on the line printer. The listing file is for informational purposes only and can be omitted if so desired. The log file contains the messages from the COMPARE portion of CAM that are normally typed at the terminal.

4.1.4 Messages From COMPARE

As each pair of files is compared, the following message is typed.

```
COMPAR:  BASE= file.ext        USER-NEW=file.ext
```

Successful completion of the CAM operation is indicated by typeout of an asterisk. At that time another command string can be entered or control can be returned to the monitor. However, if the files being compared are so dissimilar that CAM's buffers overflow, the following message is typed:

```
***INSUFFICIENT CORE FOR COMPARISON ON FILES
-! file.ext ←file.ext
```

and CAM places a message into the correction file.

```
COMPARISON NOT COMPLETED ON-!file.ext ←file.ext
```

Processing continues on any remaining files when CAM is operating in multifile mode.

When all of the base files on a DECtape have been exhausted, CAM types the following message and returns control to the monitor.

```
*** MOUNT NEXT INPUT DECTAPE ON UNIT DTAn
```

If there are no more input files, the user should:

Step	Procedure
1	Type CONT) and when CAM responds with an asterisk,
2	Press CARRIAGE RETURN to terminate processing.

CAM responds with the message:

```
***CAM TERMINATED -- INPUT FILES EXHAUSTED
```

and types an asterisk to indicate that it is ready for another command.

If there are more input base files, the user should:

Step	Procedure
1	Mount the next input tape on the same drive
2	Type ASSIGN DTAn)
3	Type CONT) and when CAM responds with an asterisk,
4	Type D) or DONE) to indicate that the tape mounting is done and processing can continue.

If CAM cannot find a matching file on the user device for a file on the base device, the following message is typed and control is returned to the monitor.

```
*** FILE filename NOT ON USER'S DEVICE device-name
```

If the file is not available, the user should:

Step	Procedure
1	Type CONT) and when CAM responds with an asterisk,
2	Press CARRIAGE RETURN.

CAM writes a file deletion header into the correction file so that the base file will not be copied into the updated file complex when FED is run.

If the named user file is present on another device, the user should:

Step	Procedure
1	Mount the file on the same device
2	Type ASSIGN dev) if the file is on a directory device.

Step	Procedure
3	Type CONT) and when CAM responds with an asterisk,
4	Type D) or DONE) to continue processing.

4.2 COMPARISON OF DISSIMILAR FILES - COMP10

COMP10 is a special enlarged version of the COMPARE portion of CAM that is used to compare files that are very dissimilar. This condition is generally the result of a user heavily modifying the system software to meet his own installation's needs.

COMP10 has comparison buffers that are twice as large as those of CAM. In addition, when COMP10 is assembled, the user can specify even larger buffers for COMP10.

COMP10 compares only single files and should be used instead of CAM either when the user is aware that his file is very different from the system file or when CAM cannot compare the files.

Care must be taken when running COMP10 that totally unlike files are not compared, because COMP10 will replace all the lines in the base file with the lines from the user file, thus exchanging the files.

4.2.1 COMP10 Command String

The general form of the command string for COMP10 is:

```
dev:corr.ext←dev:base.ext,dev:user.ext
```

Neither a listing file nor a log file can be specified in the command string for COMP10, and no device or file name can be omitted. Also, switches cannot be included in the command string.

Examples:

```
.R COMP10
*DTA6:DIFF.C01←DTA4:PRG.OLD,DTA5:PRG.NEW

.R COMP10
*DSK:CORR.BIG←DSK:FILEA.B01,DSK:FILEA.B02
```

4.2.2 Input to COMP10

The input to COMP10 consists of a base file and a user file. The base file must be named and must reside on a directory device. The user file can be read from any input device.

4.2.3 Output from COMP10

The output from COMP10 is a correction file that contains a subfile header followed by the corrections to be applied to the base file when FED is run. No listing file is produced by COMP10.

4.2.4 Messages from COMP10

Although no log file can be specified in the command string, any messages from COMP10 are output to the terminal.

When COMP10 has completed the comparison of the two files, it types the following message:

```
COMPAR COMPLETED ON: base.ext
                AND: user.ext
```

COMP10 then types an asterisk to indicate that another command can be entered or that control can be returned to the monitor.

4.3 COMERGE-ONLY

The COMERGE portion of CAM merges two correction files which apply to the same base file or base complex. CAM detects any conflicts that may exist in the two files. The merged correction file can be used as input to the FED program to produce the updated files.

4.3.1 COMERGE Command String

The general form of the command string for the COMERGE-only function of CAM is:

```
dev:merg-corr.ext,dev:list.ext,dev:log.ext ←,dev:user-corr.ext,dev:
manf-corr.ext
```

NOTE

The parameter for the base device cannot be specified in the command string. However, a comma must be placed in the command after the back arrow to indicate that the base device is not present.

The default assumptions for the command string are as follows.

a. If the name of the merged correction file is omitted, MERCOR is assumed.

b. If the listing device is omitted, no listing is produced.

c. If the name of the listing file is omitted, it is assumed to be LIST.L01.

d. If the log device is omitted, the Teletype is assumed.

e. If the name of the log file is omitted, it is assumed to be LOG.G01.

f. If the device for the manufacturer's correction file is omitted, the paper tape reader is assumed.

g. If the name of the manufacturer's (DEC) correction file is omitted, DECCOR is assumed.

No other device or filename can be omitted from the command string. The switches that can be included in the command string are described in Appendix A.

Examples:

```
.R CAM
*DSK:BOTH.COR,DSK:BOTH.LØ1←,DTA4:MY.COR,DSK:HIS.COR

.R CAM
*DTA6:MERCOR.CØ1,DSK:←,DSK:CORR1("ABC"),PTR:
```

Figure 4-2 illustrates the COMERGE-only operation of the CAM program.

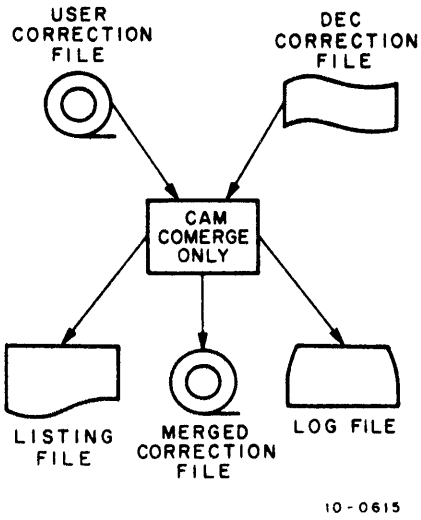

Figure 4-2 COMERGE Function of CAM

4.3.2 Input to COMERGE

The input to the COMERGE portion of CAM consists of a user correction file and the DEC correction file, both of which have been generated by CAM or COMP10. Both correction files contain editing changes to a common base file or base complex. The correction files can be read from any input devices, although DEC's correction file is normally read from paper tape or DECtape.

4.3.3 Output from COMERGE

The output from the COMERGE portion of CAM consists of a merged correction file, a listing file, and a log file. The merged file contains the editing changes to one or more base files and indications of any conflicts that have been detected by CAM. Each subfile header in the merged file contains the name taken from the subfile header of the user's correction file and the name of the base file. If the name of the user file is not desired as the filename, the merged correction file must be edited by the user to change it.

The listing file contains a line-numbered listing of the conflicts. If no conflicts have been detected, the listing file indicates this fact. The log file, normally output to the terminal, contains messages from CAM.

4.3.4 Messages from COMERGE

As CAM merges the correction files, it types the following message for each pair of subfiles.

```
COMERGE:  BASE = file.ext    USER-NEW = file.ext    MANF-NEW = file.ext
```

When the end of a correction file is reached, CAM types the following message and returns control to the monitor.

```
***END OF CORRECTION FILE  file.ext ON device-name
```

If the end-of-file has been reached on both correction files, the user should perform the following steps to terminate the run.

Step	Procedure
1	Type CONT↑ and when CAM responds with an asterisk,
2	Press CARRIAGE RETURN to indicate that the operation has been completed.

Both files must reach end-of-file for the run to terminate properly. If only one has reached end-of-file when the above message is typed, the user should:

Step	Procedure
1	Mount the next DECtape or paper tape
2	Type ASSIGN dev) (for a DECtape)
3	Type CONT) and when CAM responds with an asterisk,
4	Type D) or DONE) to continue processing.

The correction files that are continued in this manner can only be read from DECtape or paper tape. If the correction file is on more than one DECtape, the filename and extension for this file must be the same on all of the DECtapes.

4.4 COMPARE AND COMERGE - FULL CAM

Use of full CAM causes the comparison of a base file complex with a user file complex and the simultaneous merging of the resultant corrections with a previously generated correction file. Ordinarily, this operation is performed by a user who has modified his system files, yet wishes to update his versions of the files as well as the base files with the corrections from DEC. For example, the user who has modified the monitor complex receives a correction file from DEC to update the base monitor files. He must update the base files with the correction file (see Chapter 2). He can then compare his versions of the base files with the base files and merge his corrections with the correction file from DEC. The merged correction file can then be used as input to FED (if no conflicts are detected). Figure 4-3 shows the operation of full CAM (COMPARE and COMERGE).

4.4.1 CAM Command String

The general form of the command string for full CAM is:

```
dev:corr.ext,dev:list.ext,dev:log.ext ←base dev:,user dev:,dev:manf-corr.ext
```

The default assumptions for the command string are as follows:

a. If the name of the merged correction file is omitted, MERCOR is assumed.

b. If the listing device is omitted, no listing is produced.

c. If the name of the listing file is omitted, it is assumed to be LIST.L01.

d. If the log device is omitted, the Teletype is assumed.

e. If the name of the log file is omitted, it is assumed to be LOG.G01.

f. If the device for the manufacturer's correction file is omitted, the paper tape reader is assumed.

g. If the name of the manufacturer's correction file is omitted, DECCOR is assumed.

No other device or file name can be omitted from the command string. The switches that can be included in the command string are described in Appendix A.

Examples:

```
.R CAM
*DSK:,←DSK: [11,107],DSK:("CFB"),DTA4:

.R CAM
*DSK:MERCOR.C01,DSK:LIST.107,←DTA4:,DTA5:,PTR:
```

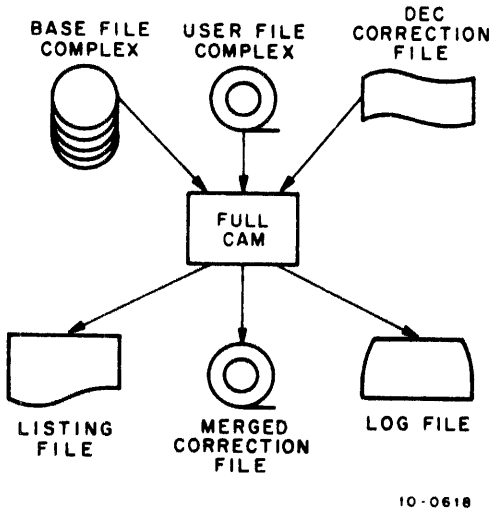

Figure 4-3 COMPARE and COMERGE (Full CAM)

4.4.2 Input to Full CAM

The input to CAM consists of a base file complex, a user file complex, and a correction file. The base files must reside on a directory device. If the base complex is on DECtape, more than one DECtape can be used to contain it. The base complex is not named in the command string.

The user file complex contains a group of new versions of the base files. The user files are compared with the base files to produce the user's corrections. These corrections are merged with the corrections from the DEC correction file. The new file complex can be read from any input device.

The correction file contains subfile headers followed by DEC's corrections to each base file. The correction file can be read from any input device, but it is normally read from paper tape or DECtape.

4.4.3 Output from Full CAM

The output from full CAM consists of a merged correction file, a listing file, and a log file. The output files can be written to any output device.

The merged correction file contains corrections to the base files from the user versions and from the manufacturer's new versions. If conflicts arise between the two versions, CAM notes them in the correction file. A conflict is defined as a reference by both versions to the same line of the base file. If conflicts exist, the correction file must be edited by the user.

The listing file contains a line-numbered listing of the correction file with conflicts noted if any exist.

The log file contains the messages ordinarily typed by CAM at the terminal.

4.4.4 Messages from CAM

CAM types the following message as the files are being compared and merged.

```
CAM:    BASE = file.ext USER-NEW  = file.ext      MANF-NEW  = file.ext
CCMERGE COMPLETED:     n ERROR(S) AND    n CONFLICT(S)  DETECTED
```

The following messages are typed by CAM depending on the conditions of the input files.

If the correction file is on a directory device, the following message is typed and control returns to the monitor when CAM detects the end of the correction file.

```
***END OF CORRECTION FILE name  ON DEVICE name
```

If there are no further correction files to process, the user should:

Step	Procedure
1	Type CONT) and when CAM responds with an asterisk,
2	Press CARRIAGE RETURN to terminate processing.

If the correction file occupies more than one DECtape, the user should:

Step	Procedure
1	Mount the next DECtape on the same drive.
2	Type ASSIGN DTAn)
3	Type CONT) and when CAM responds with an asterisk,
4	Type D) or DONE) to continue processing.

If the correction file is on paper tape, the following message is typed and control is returned to the monitor when CAM detects the end of the paper tape.

```
***END OF INPUT PAPER TAPE ON PTR
```

To terminate processing, the user should press CARRIAGE RETURN.

If more paper tape exists, the user should:

Step	Procedure
1	Put the next paper tape in the reader.
2	Type D) or DONE) to continue processing.

When CAM finds a base file named in the correction file that does not have a match in the base file complex, the following message is typed.

```
***FILE name NOT ON BASE DEVICE name
```

When CAM is unable to find a match on the user device for a file on the base device, it types the following message and returns control to the monitor.

```
***FILE name  NOT ON USER'S DEVICE name
```

If the user does not wish the base file to be copied when FED is run, he should:

Step	Procedure
1	Type CONT) and when CAM responds with an asterisk,
2	Press CARRIAGE RETURN and a file deletion header is written into the correction file (see Chapter 3).

If the file is on another device, the user should:

Step	Procedure
1	Mount the file on the same device.
2	Type ASSIGN dev) (for directory devices)
3	Type CONT) and when CAM responds with an asterisk,
4	Type D) or DONE) to continue processing.

When all the files on the base device have been exhausted, CAM types the message:

```
***MOUNT NEXT INPUT DECTAPE ON UNIT DTAn
```

and returns control to the monitor.

If there are no more input base files, the user should:

Step	Procedure
1	Type CONT) and when CAM responds with an asterisk,
2	Press CARRIAGE RETURN.

CAM responds with the message:

```
***CAM TERMINATED -- INPUT FILES EXHAUSTED
```

and types an asterisk to indicate that it is ready for another command.

However, if more input DECtapes exist, the user should:

Step	Procedure
1	Mount the next input DECtape on the same drive
2	Type ASSIGN DTAn)
3	Type CONT) and when CAM responds with an asterisk,
4	Type D) or DONE) to continue processing.

Chapter 5
The FED Progam

FED (Final Edit) is the SOUP program which enables the user to apply the corrections generated by CAM or COMP10 to the base file or files to produce updated files. This chapter describes the updating of user versions of system files using the FED Program. The updating of system files is described in Chapter 2.

The merged correction file containing the changes from DEC and the user's changes is read by FED to edit the base file, producing the updated user version of the base file. Any conflicts or errors that have been detected by CAM must be resolved by the user before the merged correction file can be input to FED. Figure 5-1 illustrates the FED updating process.

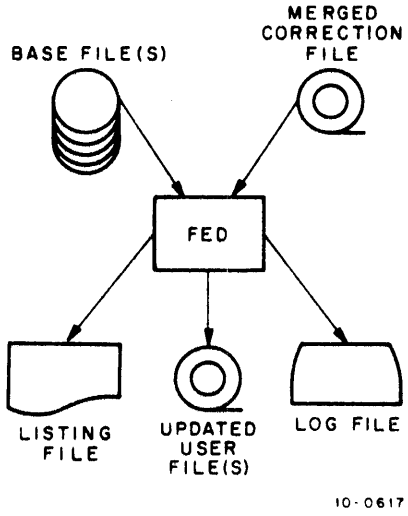

Figure 5-1 Updating User Files with FED

5.1 FED COMMAND STRING

The general form of the command string for FED is:

```
new-file dev:,list dev:,dev:log.ext ←base dev:,dev:corr.ext
```

The default assumptions for the command string are as follows:

a. If the listing device is omitted, no listing is produced.

b. If the log device is omitted, the Teletype is assumed.

c. If the name of the log file is omitted, it is assumed to be LOG.G01.

d. If the name of the correction file is omitted, DECCOR is assumed.

No other device or file name can be omitted from the command string. The switches that can be included in the command string are described in Appendix A.

Examples:

```
.R FED
*DTA0:,LPT:,TTY:←DSK:,DTA1:MERCOR.C01

.R FED
*DSK:,DSK:←DSK: [11,321],PTR:
```

5.2 INPUT TO FED

The input to FED consists of one or more base files and a correction file. The base files are the previous versions of the base files (i.e., the base files before updating with the changes from DEC). The base file complex must reside on a directory device and its name cannot be written in the command string. The correction file contains one or more subfiles, each of which is preceded by a header. These subfile headers can indicate to FED that a file is to be corrected, copied without changes, inserted into the updated complex, or deleted from the updated complex. The correction subfiles contain the changes from both DEC and the user. The correction file can be named and can be read from any input device.

5.3 OUTPUT FROM FED

The output from FED consists of one or more updated user files, a listing file for each updated file, and a log file. The output files can be written to any output device.

The updated user files are the new user versions of the updated base files. The name of the new user complex cannot be specified in the command string because the name of each file in the complex is taken from the user filename in the subfile header in the correction file.

The listing files are line-numbered listings of each of the new files. On each line that has been added by FED, *NEW* is placed in the right margin. Deleted lines are indicated by a minus number in the right margin (e.g., -5 indicates that five lines were deleted at that point in the old base file). If the user placed an identifier on those lines in the correction file that contained his corrections, the identifier is copied by FED into the listing of the new user file. The name of the listing file is the same as that of the new file except that FED changes the second character of the extension to L (e.g., FILEA.DLC is the listing of FILEA.DOC). The name of the listing file therefore cannot be included in the command.

The log file contains the messages from FED, which are normally output to the terminal.

5.4 FED MESSAGES

The messages described below are written by FED during execution.

As the corrections from each subfile are applied to a base file, FED types the following message.

```
FED COMPLETED ON FILES- luser.ext←base.ext
```

When the FED run is completed, the following is typed.

```
***FED COMPLETED SUCCESSFULLY
```

The user should not type CONTROL-C until FED types an asterisk after this message, because all output may not have been completed.

If FED detects an unrecognizable command in the correction file, it types the following message.

```
***IMPROPER COMMAND
```

The message:

```
***IMPROPER SEQUENCING
```

is typed either if a command in the correction file refers to a line that has already been deleted, or if a command refers to a line number that is less than the line number of the last insertion or deletion performed.

In both of the above cases, FED ignores the erroneous commands and continues processing. The above messages are also written into the listing file, if a listing file has been specified.

When a SYNC line (described in Chapter 3) in the correction file does not match the designated line in the base file, the following messages are typed by FED.

```
***UNMATCHING SYNC LINE
BASE FILE IS
base-line-number   text from base file
SYNC LINE IS
sync-line-number   text from correction file
```

The user can respond in one of two ways. He can:

> Press CARRIAGE RETURN, which causes FED to skip the rest of the subfile, close the new file, and resume processing with the next subfile (if one exists).

or:

> Type KEEP), which causes FED to ignore the error condition and to continue processing that file and subfile.

FED types the message:

```
***THE FOLLOWING NON-EXISTENT LINES NOT ALTERED
                    .
                    .
                    .
```

when a command in the correction file refers to a line number which is higher than the last line number in the base file. The message is followed by a typeout of the rest of the correction subfile up to the next subfile header or to the end of the correction file.

When a base complex is contained on more than one DECtape, the following message is typed and control is returned to the monitor when the input from each DECtape is exhausted.

```
***MOUNT NEXT INPUT DECTAPE ON UNIT DTAn
```

If more input exists, the user should:

Step	Procedure
1	Mount the next DECtape on the same drive
2	Type ASSIGN DTAn)
3	Type CONT) and when FED responds with an asterisk,
4	Type D) or DONE) to continue processing.

If there are no more input files, the user should:

Step	Procedure
1	Type CONT) when FED responds with an asterisk,
2	Press CARRIAGE RETURN to terminate processing.

Chapter 6
Examples of SOUP Operations

Two examples of SOUP operations are shown in this chapter. The first example illustrates a sample run in which user application files are updated. The second example shows the files as they are operated on by SOUP and the commands to the SOUP programs.

6.1 UPDATING USER APPLICATION FILES

A user has a billing system composed of three programs: CARD.MAC, SORT.MAC, and BILL.MAC. He wishes to update these files (the base files) with new versions written by another programmer (the user files). He also wishes to add a new file called BILL.OPR which explains the use of the programs.

There are two SOUP operations to be performed: comparing and updating. In addition, the BILL.OPR file must be added to the correction file before the updating procedure. Figure 6-1 illustrates the entire process of updating these files.

6.1.1 Comparing the Files

The COMPARE-only portion of CAM is used to compare the files. The base files are on DECtape, as are the user files. The correction file is written onto disk, the listing file is also written on the disk, and the log file is written on the terminal. The user should perform the following steps.

Step	Procedure
1	Mount the base DECtape on drive 1.
2	Mount the user DECtape on drive 2.
3	Call CAM and type the command string in response to the asterisk.

```
.R CAM
*DSK:BILL.COR,DSK:←DTA1:,DTA2:
```

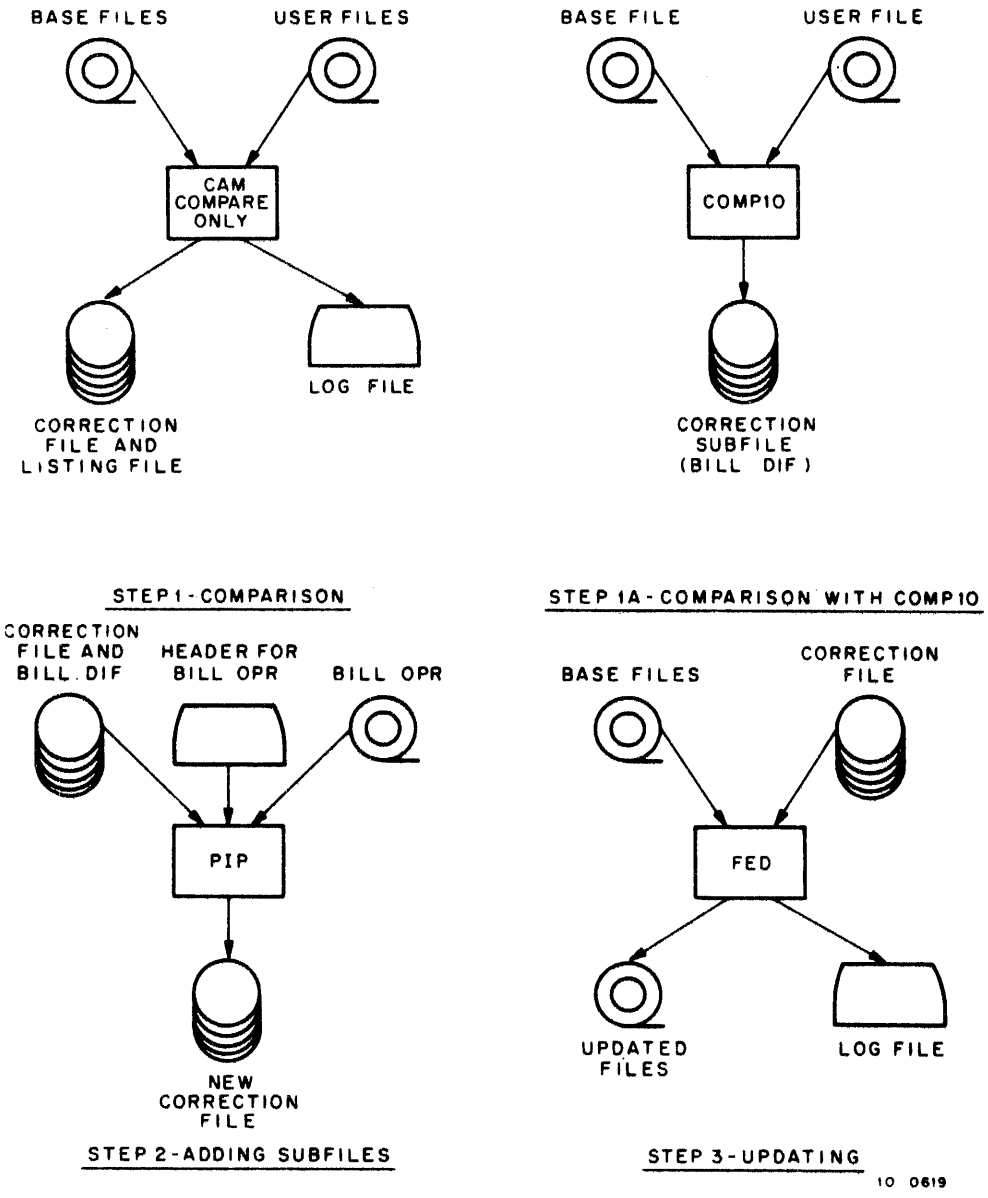

Figure 6-1 Updating User Application Files

CAM types the following messages as it compares the files.

```
COMPAR:  BASE=CARD.MAC USER-NEW=CARD.MAC
COMPAR:  BASE=SORT.MAC USER-NEW=SORT.MAC
?COMPAR  BUFFER OVERFLOW ERROR FROM BILL.MAC
*
```

The user version of BILL.MAC has been altered so drastically that CAM cannot compare the files. The user must compare them with COMP10.

```
.R COMP10
*DSK:BILL.DIF←DTA:BILL.MAC,DTA2:BILL.MAC
```

COMP10 types the following message when the comparison is complete.

```
COMPAR COMPLETED ON: BILL.MAC
                AND: BILL.MAC
```

The correction file contains a message stating that CAM could not compare the files. The user edits the file to remove the message.

6.1.2 Adding Subfiles to the Correction File

The user wants to add the correction subfile BILL.DIF and insert the file BILL.OPR into the correction file. This is accomplished through PIP. The BILL.OPR file is on a DECtape, and the correction file is placed on the disk area of user number 11,122. The user should perform the following steps.

Step	Procedure
1	Mount the DECtape containing BILL.OPR on drive 2.
2	Call PIP and enter a command string in response to the asterisk.

```
.R PIP
*DSK:BILL.COR [11,122]←DSK:BILL.COR,DSK:BILL.DIF,
*TTY:,DTA2:BILL.OPR
```

3	Add the subfile header from the terminal.

```
-1BILL.OPR ←
```

When PIP types an asterisk, the user returns control to the monitor.

6.1.3 Updating the Files

The user has the complete correction file BILL.COR on the disk area of number 11,122 and the base files on DECtape drive 1. The updated files are written onto DECtape, the log file is typed at the terminal, and the listing file is omitted. The user should perform the following steps.

Step	Procedure
1	Mount an empty DECtape on drive 0.
2	Call FED and enter the command string in response to the asterisk.

```
.R FED
*DTA0:←DTA1:,DSK: [11,122]
```

As each file is updated, FED types the following messages.

```
FED COMPLETED ON FILES-!CARD.MAC←CARD.MAC
FED COMPLETED ON FILES-!SORT.MAC←SORT.MAC
FED COMPLETED ON FILES-!BILL.MAC←BILL.MAC
FED COMPLETED ON FILES-!BILL.OPR←
```

When it has finished processing, FED types the message:

```
FED COMPLETED SUCCESSFULLY
*
```

The user types CONTROL-C to return control to the monitor.

6.2 UPDATING SAMPLE FILES

In the following example the files which are operated on and the files produced by CAM and FED are depicted.

The user has a copy of the base file called DECOCT.1.

```
TITLE   DECOCT  V.01
SUBTTL  DECIMAL TO OCTAL CONVERSION

F=0
A=1
N=2
C=3
P=4

START:  CALLI   0
        TTCALL  3,[ASCIZ/
/]
        SETZM   F               ;CLR FLAG
        SETZM   N               ;CLR NUMBR AC
        MOVE    P,[POINT 7,BUFR];SET BUFR PTR
READ:   TTCALL  0,A             ;GET CHAR
        CAIN    A,"."           ;TERMINATOR?
        JRST    CONVRT          ;YES
ZERORT: CAIL    A,"0"           ;DIGIT?
        CAILE   A,"9"
        JRST    ERROR           ;NO
        IDPB    A,P             ;STORE
        SUBI    A,"0"           ;CONVERT TO BINARY
        IMULI   N,^D10          ;PREV. TOTAL * 10
        ADD     N,A             ;+ NEW
        SETZM   A
        JRST    READ
ERROR:  TTCALL  1,["?"]
        JRST    START+1         ;TRY AGAIN
CONVRT: TTCALL  3,[ASCIZ/ = /]
        MOVSI   C,-^D12
WRITE:  SETZ    A,
        ROTC    A,3             ;HIGH 3 BITS OF N INTO A
        SKIPN   F               ;ANY DIGITS TYPED YET?
        JUMPE   A,.+4           ;NO, OMIT LEADING ZEROS
        ADDI    A,"0"           ;BINARY TO ASCII
        TTCALL  1,A             ;TYPE DIGIT
        AOS     F               ;HAVE TYPED A DIGIT
        AOBJN   C,WRITE         ;LOOP THRU 12 DIGITS
        SKIPN   F               ;ANY DIGITS TYPED?
        TTCALL  1,ZERORT        ;NO, TYPE ZERO
        JRST    START+1

BUFR:   BLOCK   10
        END     START
```

He receives a correction file from DEC to update the base file.

```
-!DECOCT.2←DECOCT.1
-1,1
TITLE   DECOCT  V.02
-13,14
        SETZB   F,N             ;CLR FLAG & NUMBR AC
-18
        CAIN    A,15            ;CR?
        TTCALL  0,A             ;GET LF TOO
-26,26
-30,30
CONVRT: TTCALL  3,[ASCIZ/ DECIMAL = /]
-31:    MOVSI   C,-↑D12
-38,38
        AOJ     F               ;HAVE TYPED A DIGIT
-41
        TTCALL  3,[ASCIZ/ OCTAL/]
```

The FED program is run to apply the corrections to DECOCT.1 producing the updated file DECOCT.2.

```
.R FED
*DSK:,DSK:←DSK:,DSK:DECCOR
FED COMPLETED ON FILES -!DECOCT.2←DECOCT.1
*** FED COMPLETED SUCCESSFULLY
*
```

```
TITLE   DECOCT  V.02
SUBTTL  DECIMAL TO OCTAL CONVERSION

F=0
A=1
N=2
C=3
P=4

START:  CALLI   0
        TTCALL  3,[ASCIZ/
/]
        SETZB   F,N             ;CLR FLAG & NUMBR AC
        MOVE    P,[POINT 7,BUFR];SET BUFR PTR
READ:   TTCALL  0,A             ;GET CHAR
        CAIN    A,"."           ;TERMINATOR?
        JRST    CONVRT          ;YES
        CAIN    A,15            ;CR?
        TTCALL  0,A             ;GET LF TOO
ZERORT: CAIL    A,"0"           ;DIGIT?
        CAILE   A,"9"
        JRST    ERROR           ;NO
        IDPB    A,P             ;STORE
        SUBI    A,"0"           ;CONVERT TO BINARY
        IMULI   N,^D10          ;PREV. TOTAL * 10
        ADD     N,A             ;+ NEW
        JRST    READ
ERROR:  TTCALL  1,["?"]
        JRST    START+1         ;TRY AGAIN
CONVRT: TTCALL  3,[ASCIZ/ DECIMAL = /]
        MOVSI   C,-^D12
WRITE:  SETZ    A,
        ROTC    A,3             ;HIGH 3 BITS OF N INTO A
        SKIPN   F               ;ANY DIGITS TYPED YET?
        JUMPE   A,.+4           ;NO, OMIT LEADING ZEROS
        ADDI    A,"0"           ;BINARY TO ASCII
        TTCALL  1,A             ;TYPE DIGIT
        AOJ     F               ;HAVE TYPED A DIGIT
        AOBJN   C,WRITE         ;LOOP THRU 12 DIGITS
        SKIPN   F               ;ANY DIGITS TYPED?
        TTCALL  1,ZERORT        ;NO, TYPE ZERO
        TTCALL  3,[ASCIZ/ OCTAL/]
        JRST    START+1

BUFR:   BLOCK   10
        END     START
```

The listing file produced from the FED run is printed to show where lines were deleted and inserted. This file is called DECOCT.2L.

```
1.  TITLE   DECOCT  V,02                                                    *-1*NEW*
2.  SUBTTL  DECIMAL TO OCTAL CONVERSION
3.
4.  F=0
5.  A=1
6.  N=2
7.  C=3
8.  P=4
9.
10. START:  CALLI   0
11.         TTCALL  3,[ASCIZ/
12. /]
13.         SETZB   F,N             ;CLR FLAG & NUMBR AC            *-2*NEW*
14.         MOVE    P,[POINT 7,BUFR];SET BUFR PTR
15. READ:   TTCALL  0,A             ;GET CHAR
16.         CAIN    A,","           ;TERMINATOR?
17.         JRST    CONVRT          ;YES
18.         CAIN    A,15            ;CR?                            *NEW*
19.         TTCALL  0,A             ;GET LF TOO                     *NEW*
20. ZERORT: CAIL    A,"0"           ;DIGIT?
21.         CAILE   A,"9"
22.         JRST    ERROR           ;NO
23.         IDPB    A,P             ;STORE
24.         SUBI    A,"0"           ;CONVERT TO BINARY
25.         IMULI   N,^D10          ;PREV, TOTAL * 10
26.         ADD     N,A             ;+ NEW
27.         JRST    READ                                            *-1
28. ERROR:  TTCALL  1,["?"]
29.         JRST    START+1         ;TRY AGAIN
30. CONVRT: TTCALL  3,[ASCIZ/ DECIMAL = /]                          *-1*NEW*
31.         MOVSI   C,-^D12
32. WRITE:  SETZ    A,
33.         ROTC    A,3             ;HIGH 3 BITS OF N INTO A
34.         SKIPN   F               ;ANY DIGITS TYPED YET?
35.         JUMPE   A,.+4           ;NO, OMIT LEADING ZEROS
36.         ADDI    A,"0"           ;BINARY TO ASCII
37.         TTCALL  1,A             ;TYPE DIGIT
38.         AOJ     F               ;HAVE TYPED A DIGIT             *-1*NEW*
39.         AOBJN   C,WRITE         ;LOOP THRU 12 DIGITS
40.         SKIPN   F               ;ANY DIGITS TYPED?
41.         TTCALL  1,ZERORT        ;NO, TYPE ZERO
42.         TTCALL  3,[ASCIZ/ OCTAL/]                               *NEW*
43.         JRST    START+1
44.
45. BUFR:   BLOCK   10
46.         END     START
```

The user has his own version of DECOCT, which he has called DECOCT.1A.

```
TITLE   DECOCT  V.01A
SUBTTL  DECIMAL TO OCTAL CONVERSION

F=0
A=1
N=2
C=3
P=4

START:  CALLI   0
        TTCALL  3,[ASCIZ/
INPUT: /]
        SETZM   F               ;CLR FLAG
        SETZM   N               ;CLR NUMBR AC
        MOVE    P,[POINT 7,BUFR];SET BUFR PTR
READ:   TTCALL  0,A             ;GET CHAR
        CAIN    A,"."           ;TERMINATOR?
        JRST    CONVRT          ;YES
        CAIN    A,15            ;CR?
        JRST    CRERR           ;YES
ZERORT: CAIL    A,"0"           ;DIGIT?
        CAILE   A,"9"
        JRST    ERROR           ;NO
        IDPB    A,P             ;STORE
        SUBI    A,"0"           ;CONVERT TO BINARY
        IMULI   N,↑D10          ;PREV. TOTAL * 10
        ADD     N,A             ;+ NEW
        SETZM   A
        JRST    READ
CRERR:  TTCALL  0,A             ;DISCARD LF
ERROR:  TTCALL  1,["?"]
        JRST    START+1         ;TRY AGAIN
CONVRT: TTCALL  3,[ASCIZ/ = /]
        MOVSI   C,-↑D12
WRITE:  SETZ    A,
        ROTC    A,3             ;HIGH 3 BITS OF N INTO A
        SKIPN   F               ;ANY DIGITS TYPED YET?
        JUMPE   A,.+4           ;NO, OMIT LEADING ZEROS
        ADDI    A,"0"           ;BINARY TO ASCII
        TTCALL  1,A             ;TYPE DIGIT
        AOS     F               ;HAVE TYPED A DIGIT
        AOBJN   C,WRITE         ;LOOP THRU 12 DIGITS
        SKIPN   F               ;ANY DIGITS TYPED?
        TTCALL  1,ZERORT        ;NO, TYPE ZERO
        JRST    START+1

BUFR:   BLOCK   10
        END     START
```

The user wants to update DECOCT.1A with the DEC corrections. The user first compares DECOCT.1A to DECOCT.1, and then merges the correction file with the correction file from DEC.

```
.R CAM
*DSK:USRCOR←DSK:DECOCT.1,DSK:DECOCT.1A
COMPAR:   BASE= DECOCT.1     USER-NEW = DECOCT.1A
*DSK:MERCOR←,DSK:USRCOR,DSK:DECCOR
COMERG:   BASE= DECOCT.1     USER-NEW = DECOCT.1A  MANF-NEW= DECOCT.2
   1. -!DECOCT.1A←DECOCT.1
   2. - /-/-/-/-/-/-/-/-/-/ BEGINNING OF CONFLICT   1 \-\-\-\-\-\-\-\-\-\
   3. -1,1
   4. TITLE     DECOCT  V.01A
   5. -1,1
   6. TITLE     DECOCT  V.02
   7. - /-/-/-/-/-/-/-/-/-/-/-/-/ END OF CONFLICT   1 \-\-\-\-\-\-\-\-\-\
  12. - /-/-/-/-/-/-/-/-/-/ BEGINNING OF CONFLICT   2 \-\-\-\-\-\-\-\-\-\
  13. -18
  14.   CAIN    A,15            ;CR?
  15.   JRST    CRERR           ;YES
  16. -18
  17.   CAIN    A,15            ;CR?
  18.   TTCALL  3,A             ;GET LF TOO
  19. - /-/-/-/-/-/-/-/-/-/-/-/-/-/ END OF CONFLICT   2 \-\-\-\-\-\-\-\-\-\
*** END OF CORRECTION FILE USRCOR.  ON DSK

.CONT

*
*** END OF CORRECTION FILE DECCOR.  ON DSK

.CONT

*
COMERG COMPLETED:     0 ERRORS AND     2 CONFLICT(S) DETECTED
*
```

The log file from the COMERGE operation shows that conflicts have been noted by CAM. The user lists the correction file produced by the COMPARE operation, and lists the correction file from the COMERGE operation.

```
-!DECOCT.1A←DECOCT.1
-1,1
TITLE   DECOCT  V.01A
-12,12
INPUT: /]
-18
        CAIN    A,15            ;CR?
        JRST    CRERR           ;YES
-27
CRERR:  TTCALL  0,A             ;DISCARD LF
```

```
-!DECOCT.1A←DECOCT.1
- /-/-/-/-/-/-/-/-/-/ BEGINNING OF CONFLICT   1 \-\-\-\-\-\-\-\-\-\
-1,1
TITLE   DECOCT  V.01A
-1,1
TITLE   DECOCT  V.02
- /-/-/-/-/-/-/-/-/-/-/-/ END OF CONFLICT   1 \-\-\-\-\-\-\-\-\-\
-12,12
INPUT: /]
-13,14
        SETZB   F,N             ;CLR FLAG & NUMBR AC
- /-/-/-/-/-/-/-/-/-/ BEGINNING OF CONFLICT   2 \-\-\-\-\-\-\-\-\-\
-18
        CAIN    A,15            ;CR?
        JRST    CRERR           ;YES
-18
        CAIN    A,15            ;CR?
        TTCALL  0,A             ;GET LF TOO
- /-/-/-/-/-/-/-/-/-/-/-/-/ END OF CONFLICT   2 \-\-\-\-\-\-\-\-\-\
-26,26
-27
CRERR:  TTCALL  0,A             ;DISCARD LF
-30,30
CONVRT: TTCALL  3,[ASCIZ/ DECIMAL = /]
-31:    MOVSI   C,-↑D12
-38,38
        AOJ     F               ;HAVE TYPED A DIGIT
-41
        TTCALL  3,[ASCIZ/ OCTAL/]
```

Because there are conflicts between the user changes and the DEC changes, the user gets a line-numbered listing of the old base file to use as a reference. Note that the line numbers in this listing are those used in the correction file.

```
 1.  TITLE   DECOCT  V.01
 2.  SUBTTL  DECIMAL TO OCTAL CONVERSION
 3.
 4.  F=0
 5.  A=1
 6.  N=2
 7.  C=3
 8.  P=4
 9.
10.  START:  CALLI   0
11.          TTCALL  3,[ASCIZ/
12.  /]
13.          SETZM   F                ;CLR FLAG
14.          SETZM   N                ;CLR NUMBR AC
15.          MOVE    P,[POINT 7,BUFR];SET BUFR PTR
16.  READ:   TTCALL  0,A              ;GET CHAR
17.          CAIN    A,"."            ;TERMINATOR?
18.          JRST    CONVRT           ;YES
19.  ZERORT: CAIL    A,"0"            ;DIGIT?
20.          CAILE   A,"9"
21.          JRST    ERROR            ;NO
22.          IDPB    A,P              ;STORE
23.          SUBI    A,"0"            ;CONVERT TO BINARY
24.          IMULI   N,↑D10           ;PREV. TOTAL * 10
25.          ADD     N,A              ;+ NEW
26.          SETZM   A
27.          JRST    READ
28.  ERROR:  TTCALL  1,["?"]
29.          JRST    START+1          ;TRY AGAIN
30.  CONVRT: TTCALL  3,[ASCIZ/ = /]
31.          MOVSI   C,-↑D12
32.  WRITE:  SETZ    A,
33.          ROTC    A,3              ;HIGH 3 BITS OF N INTO A
34.          SKIPN   F                ;ANY DIGITS TYPED YET?
35.          JUMPE   A,.+4            ;NO, OMIT LEADING ZEROS
36.          ADDI    A,"0"            ;BINARY TO ASCII
37.          TTCALL  1,A              ;TYPE DIGIT
38.          AOS     F                ;HAVE TYPED A DIGIT
39.          AOBJN   C,WRITE          ;LOOP THRU 12 DIGITS
40.          SKIPN   F                ;ANY DIGITS TYPED?
41.          TTCALL  1,ZERORT         ;NO, TYPE ZERO
42.          JRST    START+1
43.
44.  BUFR:   BLOCK   10
45.          END     START
```

The user decides how he wants to resolve the conflicts, and edits the file MERCOR with TECO.

```
.TECO MERCOR

*SDECOCT.1A$-2DI2A$@LT$$
-!DECOCT.2A←DECOCT.1
*LT$$
- /-/-/-/-/-/-/-/-/-/-/ BEGINNING OF CONFLICT   1 \-\-\-\-\-\-\-\-\-\
*K$$
*SV.@1A$-2DI2A$@LT$$
TITLE   DECOCT  V.@2A
*L3T$$
-1,1
TITLE   DECOCT  V.@2
- /-/-/-/-/-/-/-/-/-/-/-/-/-/-/ END OF CONFLICT   1 \-\-\-\-\-\-\-\-\-\
*3K$$
*4LT$$
- /-/-/-/-/-/-/-/-/-/-/ BEGINNING OF CONFLICT   2 \-\-\-\-\-\-\-\-\-\
*4K$$
*4T$$
-18
        CAIN    A,15            ;CR?
        TTCALL  @,A             ;GET LF TOO
- /-/-/-/-/-/-/-/-/-/-/-/-/-/-/ END OF CONFLICT   2 \-\-\-\-\-\-\-\-\-\
*3LK$$
*LT$$
-27
*2K$$
*EX$$

EXIT
↑C
```

Just to make certain that MERCOR (the merged correction file) contains the changes that he wants, the user lists it.

```
-!DECOCT.2A←DECOCT.1
-1,1
TITLE   DECOCT  V.@2A
-12,12
INPUT: /]
-13,14
        SETZB   F,N             ;CLR FLAG & NUMBR AC
-18
        CAIN    A,15            ;CR?
        TTCALL  @,A             ;GET LF TOO
-26,26
-3@,3@
CONVRT: TTCALL  3,[ASCIZ/ DECIMAL = /]
-31:    MOVSI   C,-↑D12
-38,38
        AOJ     F               ;HAVE TYPED A DIGIT
-41
        TTCALL  3,[ASCIZ/ OCTAL/]
```

The user can run the FED program to update DECOCT.1A with the corrections in MERCOR. He then lists the updated file which is called DECOCT.2A.

```
.R FED
*DSK:←DSK:,DSK:MERCOR
FED COMPLETED ON FILES -!DECOCT.2A←DECOCT.1
*** FED COMPLETED SUCCESSFULLY
*
```

```
TITLE   DECOCT  V.Ø2A
SUBTTL  DECIMAL TO OCTAL CONVERSION

F=Ø
A=1
N=2
C=3
P=4

START:  CALLI   Ø
        TTCALL  3,[ASCIZ/
INPUT: /]
        SETZB   F,N             ;CLR FLAG & NUMBR AC
        MOVE    P,[POINT 7,BUFR];SET BUFR PTR
READ:   TTCALL  Ø,A             ;GET CHAR
        CAIN    A,"."           ;TERMINATOR?
        JRST    CONVRT          ;YES
        CAIN    A,15            ;CR?
        TTCALL  Ø,A             ;GET LF TOO
ZERORT: CAIL    A,"Ø"           ;DIGIT?
        CAILE   A,"9"
        JRST    ERROR           ;NO
        IDPB    A,P             ;STORE
        SUBI    A,"Ø"           ;CONVERT TO BINARY
        IMULI   N,↑D1Ø          ;PREV. TOTAL * 1Ø
        ADD     N,A             ;+ NEW
        JRST    READ
ERROR:  TTCALL  1,["?"]
        JRST    START+1         ;TRY AGAIN
CONVRT: TTCALL  3,[ASCIZ/ DECIMAL = /]
        MOVSI   C,-↑D12
WRITE:  SETZ    A,
        ROTC    A,3             ;HIGH 3 BITS OF N INTO A
        SKIPN   F               ;ANY DIGITS TYPED YET?
        JUMPE   A,.+4           ;NO, OMIT LEADING ZEROS
        ADDI    A,"Ø"           ;BINARY TO ASCII
        TTCALL  1,A             ;TYPE DIGIT
        AOJ     F               ;HAVE TYPED A DIGIT
        AOBJN   C,WRITE         ;LOOP THRU 12 DIGITS
        SKIPN   F               ;ANY DIGITS TYPED?
        TTCALL  1,ZERORT        ;NO, TYPE ZERO
        TTCALL  3,[ASCIZ/ OCTAL/]
```

Appendix A
Summary of Switch Options

Switches can be included in the command strings of FED and CAM. Single-switch characters are written preceded by a slash (e.g., /H) and multiple-switch characters are enclosed in parentheses (e.g., ("DEC" HT)).

A.1 CAM SWITCHES

The switches that can be included in the command strings for CAM are described below. They are: /I, /P, /N = n, and ("xxx").

A.1.1 /I Switch

Normally, the COMERGE portion of CAM increments the extension of a user filename if the last two digits are numeric (e.g., C01 and 200) when it writes the filename into the subfile header in the correction file. For example, if the base file is named FILEA.B01 and the user file is named FILEA.U01, the header in the correction file is written as:

```
-!FILEA.U02←FILEA.B01
```

This new file name is then copied by FED into the updated user's file as FILEA.U02. The /I switch causes CAM to suppress the incrementing of the extension number. Thus, in the preceding example, the /I switch causes the filename to be written into the header in the correction file as:

```
-!FILEA.U01←FILEA.B01
```

and FED to copy the name FILEA.U01.

Examples:

```
.R CAM
*DTA4:MERCOR←DTA3:,DSK:,PTR:   /I

.R CAM
*DSK:,LPT:←,DTA4:MYCOR,PTR:   /I
```

A.1.2 /P Switch

The /P switch causes the command string to be typed at the terminal as CAM is reading it. The /P switch is useful when a group of command strings have been placed into a command file. As each command string in the file is read by CAM, it is typed. The user then can observe which CAM operations are being performed at that time.

Example:

```
.R CAM
DTA5:X.Y,TTY:←DTA4:A.B,DTA6:A.C   /P
```

A.1.3 /N=n Switch

Ordinarily, the COMPARE portion of CAM searches for three matching lines in the base and user files when a difference in the files occurs. The /N=n switch is used to change the number of lines that must match according to the value of n. The value of n can be from 1 through 9.

Examples:

```
.R CAM
*DTA3:CORR.01,LPT:←DSK:,DSK:   [13,132]   /N=5

.R CAM
DSK:←DTA4:,DTA3:   /N=6,PTR:
```

A.1.4 Identifier Switch ("xxx")

The identifier switch causes an identifier of up to three characters to be placed on each user correction line by the COMPARE portion of CAM. The correction line is placed into the correction file and the listing of the correction file as illustrated in the following example.

```
-521,                  CFB
; THIS LINE IS TO BE INSERTED
-593,597               CFB
```

CAM automatically adds the comma and tab following the command so that the identifier is set off from the command. This switch is useful when more than one programmer has a changed version of the base file. The changes from each programmer can then be noted in the correction file by means of the identifier. The identifier is also placed in the right margin of the listing of the updated file by FED.

Examples:

```
.R CAM
*DSK:,LPT:←DTA3:FILEA.B01,DTA4:FILEA.C01 ("CFB")

.R CAM
DTA0:,LPT:←DSK:  [11,123],DSK: ("ABC")
```

A.2 FED SWITCHES

The switches that can be placed in the command string for FED are described below. They are: /H, ("xxx"), /T, /P, and /Z.

A.2.1 /H Switch

The /H switch causes a heading to be placed at the beginning of the FED listing file and the file to be printed on numbered pages, 55 lines to a page. The heading has the form:

```
New-file name PREPARED FROM old-file name AND correction-file name (BY xxx)
mm-dd-yy hh:mm:ss  PAGE:n
```

The listing file produced is meant to be written on a line printer. If the terminal is specified as the listing device, FED attempts to fit each line on the terminal without folding. The phrase (BY xxx) is optional, but when it is desired by the user, an identifier switch must also be written in the command string in the form ("xxx"). The identifier switch is used in FED only with the /H switch.

Example:

```
.R FED
*DSK:,LPT:←DTA1:,DSK:LATEST.COR  ("DEC" H)
```

A.2.2 /T Switch

The /T switch causes a message to be inserted into the log file from FED. This message gives the beginning and ending time of the processing of a file by FED. The form of the message is:

```
hh:mm:ss  hh:mm:ss  FED COMPLETED ON FILES -! file.ext←file.ext
   ↑         ↑
START   FINISH
```

Example:

```
.R FED
*DTA3:,LPT:←DSK: [121,122], PTR:  /T
                  .
                  .
                  .
17:00:00  17:00:06  FED COMPLETED ON FILES -!FILEA.02←FILEA.01
   ↑         ↑
START     FINISH
```

A.2.3 /P Switch

The /P switch causes the command string to be typed at the terminal as it is read by FED. The /P switch is useful when a group of command strings have been placed in a command file. The user can then see which files are being processed by FED as each command string is typed.

Example:

```
.R CAM
*DSK:,DSK:←DTA4:,DTA2:  /P
```

A.2.4 /Z Switch

The /Z switch causes FED to recognize a CONTROL-Z character as the end of a correction file, regardless of what follows. This switch is useful when the terminal is used as an input device. It is also useful when a correction file is being edited by TECO. The CONTROL-Z character is used as the end-of-file rather than a form-feed because a form-feed could be copied into the source file.

Example:

```
.R FED
*DSK:,LPT:←DTA4:,TTY:  /Z
```

Appendix B
SOUP Messages

Generally, two kinds of messages are printed by the programs in SOUP: informative messages and error messages. Informative messages are usually preceded by asterisks or minus signs. Error messages begin with question marks.

B.1 INFORMATIVE MESSAGES

The informative messages printed at the terminal by the SOUP routines can be of two versions. One version notifies the user that an action is taking place or has taken place. The second type informs the operator that a condition exists in which he must take action. This action is usually mounting additional files for input. Table B-1 contains the informative messages, their meanings, and any action that must be taken by the user.

Table B-1
Informative Messages

Message	Meaning	Action by User
***MOUNT NEXT INPUT DECTAPE ON UNIT DTAn	The end of an input DECtape has been reached.	A. If more files exist: 1. Mount next input tape 2. Type ASSIGN DTAn⟩ 3. Type CONT⟩ when the program types an asterisk, 4. Type D⟩or DONE⟩
***END OF CORRECTION file.ext ON dev	An input correction file for COMERGE has reached end-of-file.	
***END OF CORRECTION FILE file.ext ON dev	The manufacturer's correction file has reached end-of-file.	B. If no more files exist: 1. Type CONT⟩ when the program types an asterisk, 2. Press CARRIAGE RETURN
COMPAR: BASE = file.ext USER-NEW = file.ext	CAM is about to compare the two named files.	None

Table B-1 (Cont)
Informative Messages

Message	Meaning	Action by User
***INSUFFICIENT CORE FOR COMPARE ON FILES -!base.ext ←user.ext	The files being compared are too dissimilar.	Compare the programs using COMP10.
***CAM TERMINATED - INPUT FILES EXHAUSTED	No more input base files exist for CAM.	Enter next command or return control to the monitor.
***FILE name NOT ON USER'S DEVICE dev	CAM cannot find a user file that matches a base file.	A. If the file is on another device: 1. Mount the file 2. Type ASSIGN dev) (directory device) 3. Type CONT) when CAM types an asterisk, 4. Type D) or DONE) B. If the file is to be deleted from the base complex: 1. Type CONT) when CAM types an asterisk, 2. Press CARRIAGE RETURN
COMERGE: BASE=file.ext USER-NEW=file.ext MANF-NEW=file.ext	CAM is merging correction subfiles	None
COMPAR COMPLETED ON: base.ext AND: new.ext	COMP10 has successfully compared the named files.	Enter next command or return control to the monitor.
CAM: BASE=file.ext USER-NEW=file.ext MANF-NEW=file.ext COMERGE COMPLETED: n ERROR(S) AND n CONFLICT(S) DETECTED	CAM has completed processing the named files.	Enter next command unless conflicts exist in correction file. In that case, edit correction file.
***END OF INPUT PAPER TAPE ON PTR	The end of the paper tape containing the manufacturer's correction file has been reached.	A. If there is more input: 1. Place the next tape in the reader 2. Type D) or DONE) B. If there is no more input: Press CARRIAGE RETURN
FILE name NOT ON BASE DEVICE dev	CAM cannot find the base file named in the manufacturer's correction file.	
FED COMPLETED ON FILES -!new.ext ←base.ext	FED has completed editing of the named file.	None

Table B-2 shows the conversion of ASCII code to SIXBIT code. The table does not show ASCII codes 000 through 037 because they all convert to SIXBIT 74 (\), except 11 (TAB) which converts to SIXBIT 00 (space).

Table B-2
ASCII to SIXBIT Conversion

Character	ASCII 7-bit	SIXBIT	Character	ASCII 7-bit	SIXBIT
Space	040	00	@	100	40
!	041	01	A	101	41
"	042	02	B	102	42
#	043	03	C	103	43
$	044	04	D	104	44
%	045	05	E	105	45
&	046	06	F	106	46
'	047	07	G	107	47
(	050	10	H	110	50
)	051	11	I	111	51
*	052	12	J	112	52
+	053	13	K	113	53
,	054	14	L	114	54
-	055	15	M	115	55
.	056	16	N	116	56
/	057	17	O	117	57
0	060	20	P	120	60
1	061	21	Q	121	61
2	062	22	R	122	62
3	063	23	S	123	63
4	064	24	T	124	64
5	065	25	U	125	65
6	066	26	V	126	66
7	067	27	W	127	67
8	070	30	X	130	70
9	071	31	Y	131	71
:	072	32	Z	132	72
;	073	33	[	133	73
<	074	34	\	134	74
=	075	35	]	135	75
>	076	36	↑	136	76
?	077	37	←	137	77

Table B-2 (Cont.)
ASCII to SIXBIT Conversion

ASCII code	ASCII character	SIXBIT code	SIXBIT character
140		74	\
141	a	41	A
142	b	42	B
143	c	43	C
144	d	44	D
145	e	45	E
146	f	46	F
147	g	47	G
150	h	50	H
151	i	51	I
152	j	52	J
153	k	53	K
154	l	54	L
155	m	55	M
156	n	56	N
157	o	57	O
160	p	60	P
161	q	61	Q
162	r	62	R
163	s	63	S
164	t	64	T
165	u	65	U
166	v	66	V
167	w	67	W
170	x	70	X
171	y	71	Y
172	z	72	Z
173	{	73	[
174	|	74	\
175	}	75	]
176	~	74	\
177	delete	74	\

Table B-2 (Cont)
Error Messages

Message	Meaning	Action by User
?LST TAPE TOO FULL ?MRG TAPE TOO FULL	The output tape for the listing or merged file has less than 20 free blocks. The run is aborted.	Mount a new tape and rerun the program.

INDEX

PDP-10 (SOUP)
SOFTWARE UPDATING PACKAGE
DEC-10-YUFA-D

READER'S COMMENTS

Digital Equipment Corporation maintains a continuous effort to improve the quality and usefulness of its publications. To do this effectively we need user feedback: your critical evaluation of this document. Please give specific page and line references when appropriate.

ERRORS NOTED IN THIS PUBLICATION: _____

SUGGESTIONS FOR IMPROVEMENT OF THIS PUBLICATION: _____

DEC also strives to keep its customers informed about current DEC software and publications. Thus, the following periodically distributed publications are available upon request. Please check the publication(s) desired.

☐ PDP-10 User's Bookshelf, a bibliography of current programming documents.

☐ Program Library Price List, a list of available software documents and programs.

Name _____ Date _____

Organization _____

Please describe your position _____

Street _____

City _____ State _____ Zip Code _____

--------Do Not Tear - Fold Here and Tape----------

digital

No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SOFTWARE PUBLICATIONS
200 FOREST STREET MR1-2/E37
MARLBOROUGH, MASSACHUSETTS 01752

------- Do Not Tear - Fold Here and Tape ----------

Cut Along Dotted Line