R. Bell

CONTROL DATA CORPORATION
Publications and Graphics Division
P. O. BOX 3492
SUNNYVALE, CALIFORNIA 94088-3492

DATE: September 18, 1984

TITLE: Update Version 1 Reference Manual

PUBLICATION NO.: 60449900

REVISION: F

REASON FOR CHANGE:

This revision documents Version 1.4 of the Update utility at PSR level 601. It includes the addition of the END directive and miscellaneous corrections and modifications.

INSTRUCTIONS:

To update this manual, for which the previous revision was E, make the following changes:

Remove	Insert
Front Cover/ Inside Front Cover	Front Cover/ Inside Front Cover
Title Page/ii	Title Page/ii
iii/iv	iii/iv
v/vi	v/vi
vii/viii	vii/viii
1-1/1-2	1-1/1-2
2-1 thru 2-4	2-1 thru 2-4
3-3/3-4	3-3/3-4
3-7/3-8	3-7/3-8
3-15	3-15/3-16
4-3 thru 4-8	4-3 thru 4-8
A-5/A-6	A-5/A-6
B-5/B-6	B-5/B-6
D-1/D-2	D-1/D-2
D-9/D-10	D-9/D-10
Index-1/Index-2	Index-1/Index-2
Comment Sheet/Mailer	Comment Sheet/Mailer
Inside Back Cover/ Back Cover	Inside Back Cover/ Back Cover

60449900

CONTROL DATA
CORPORATION

UPDATE
VERSION 1
REFERENCE MANUAL

CDC® OPERATING SYSTEMS:

NOS 1
NOS 2
NOS/BE 1
SCOPE 2

UPDATE DIRECTIVES INDEX

Directive	Parameters	Abbreviation	Page
*ABBREV		none	3-14
*ADDFILE	lfn,name	*AF	3-5
*BEFORE	line	*B	3-6
*CALL	deck	*CA	3-11
*CHANGE	oldid,newid,...,oldid,newid	*CH	3-6
*COMDECK	deck,NOPROP	*CD	3-5
*COMPILE	deck1.deck2	*C	3-11
*COMPILE	deck1,deck2,...,deckn	*C	3-11
*COPY	deck,line	*CY	3-6
*COPY	deck,line1,line2	*CY	3-6
*COPY	deck,line1,line2,lfn	*CY	3-6
*CWEOR	level	*CW	3-11
*DECK	deck	*DK	3-5
*DECLARE	deck	*DC	3-15
*DEFINE	name1,name2,...,namen	*DF	3-15
*DELETE	line1,line2	*D	3-7
*DELETE	line	*D	3-7
*DO	ident1,ident2,...,identn	none	3-12
*DONT	ident1,ident2,...,identn	*DT	3-12
*END		none	3-15
*ENDIF		*EI	3-12
*ENDTEXT		*ET	3-14
*IDENT	idname,B=num,K=ident,U=ident	*ID	3-7
*IF	type,name,num	none	3-12
*IF	-type,name,num	none	3-12
*INSERT	line	*I	3-8
*LIMIT	n	*LT	3-15
*LIST		*L	3-14
*MOVE	deck1,deck2	*M	3-8
*NOABBREV		*NA	3-14
*NOLIST		*NL	3-14
*PULLMOD	ident1,ident2,...,identn	*PM	3-15
*PURDECK	deck1,deck2,...,deckn	*PD	3-8
*PURDECK	deck1.deck2	*PD	3-8
*PURGE	ident1,ident2,...,identn	*P	3-9
*PURGE	ident1.ident2	*P	3-9
*PURGE	ident,*	*P	3-9
*READ	lfn	*RD	3-13
*RESTORE	line	*R	3-9
*RESTORE	line1,line2	*R	3-9
*REWIND	lfn	*RW	3-13
*SELPURGE	deck.ident1,deck2.ident2,...,deckn.identn	*SP	3-9
*SELYANK	deck1.ident1,deck2.ident2,...,deckn.identn	*SY	3-10
*SEQUENCE	deck1,deck2,...,deckn	*S	3-10
*SEQUENCE	deck1.deck2	*S	3-10
*SKIP	lfn,n	*SK	3-14
*TEXT		*T	3-14
*WEOR	level	*W	3-13
*WIDTH	linelen,idlen	*WI	3-13
*YANK	ident1,ident2,...,identn	*Y	3-10
*YANK	ident1.ident2	*Y	3-10
*YANKDECK	deck1,deck2,...,deckn	*YD	3-10
*/	comments	none	3-16

60449900

CONTROL DATA
CORPORATION

R. Bell

UPDATE
VERSION 1
REFERENCE MANUAL

CDC® OPERATING SYSTEMS:

NOS 1
NOS 2
NOS/BE 1
SCOPE 2

REVISION RECORD

Revision	Description
A (12/15/75)	Original printing. This manual is a successor to publication number 60342500 for users of NOS 1.0, NOS/BE 1.0, and SCOPE 2.1 operating systems.
B (03/31/78)	This revision reflects Version 1.3 of the Update utility at PSR level 472. Update has been modified to allow up to seven secondary old program libraries to be specified. This revision obsoletes all previous editions.
C (10/31/80)	This revision reflects Version 1.4 of the Update utility at PSR level 528, which adds the capability to maintain program libraries in ASCII (8-bit) code, and to use text lines with 256 characters or less.
D (11/23/81)	This revision reflects Version 1.4 of the Update utility at PSR level 552. This revision supersedes all previous editions.
E (03/25/82)	This revision reflects Version 1.4 of the Update utility at PSR level 564. It supports NOS Version 2.0 and includes miscellaneous technical corrections.
F (09/18/84)	This revision documents Version 1.4 of the Update utility at PSR level 601. It includes the addition of the END directive and miscellaneous corrections and modifications.

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

Address comments concerning this manual to:

CONTROL DATA CORPORATION
Publications and Graphics Division
P. O. Box 3492
SUNNYVALE, CALIFORNIA 94088-3492

or use Comment Sheet in the back of this manual

LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

Page	Revision
Front Cover	-
Inside Front Cover	F
Title Page	-
ii	F
iii/iv	F
v	F
vi	E
vii	F
viii	F
ix	D
1-1	F
1-2	F
1-3	E
2-1	D
2-2	F
2-3	D
2-4	F
3-1 thru 3-3	D
3-4	F
3-5 thru 3-7	E
3-8	F
3-9	D
3-10	D
3-11	E
3-12 thru 3-14	D
3-15	F
3-16	F
4-1	D
4-2	D
4-3 thru 4-5	F
4-6	D
4-7	D
4-8	F
5-1 thru 5-7	D
A-1 thru A-5	D
A-6	F
A-7 thru A-12	D
B-1 thru B-4	E
B-5	F
B-6	F
B-7	D
C-1	D
C-2	D
D-1	F
D-2 thru D-4	D
D-5	C
D-6 thru D-8	D
D-9	F
D-10	D
Index-1	F
Index-2	D
Comment Sheet/Mailer	F
Inside Back Cover	F
Back Cover	-

PREFACE

This manual describes the Update utility for maintaining and updating decks in compressed symbolic format on mass storage. As described in this publication, Update 1.4 operates under the control of the following operating systems:

- NOS 1 and NOS 2 for the CONTROL DATA® CYBER 180 Computer Systems; CYBER 170 Computer Systems; CYBER 70 Computer System models 71, 72, 73, and 74; and 6000 Computer Systems.
- NOS/BE 1 for the CDC® CYBER 180 Computer Systems; CYBER 170 Computer Systems; CYBER 70 Computer System models 71, 72, 73, and 74; and 6000 Computer Systems.
- SCOPE 2 for the CDC CYBER 170 Computer System model 176; CYBER 70 Computer System model 76; and 7600 Computer Systems.

The user is assumed to be familiar with the operating system and computer system in use.

The NOS 1, NOS 2, and NOS/BE Manual Abstracts are pocket-sized manuals containing brief descriptions of the contents and intended audience of NOS and NOS/BE and all the product set manuals of these two operating systems. The manual abstracts can be useful in determining which manuals are of greatest interest to a particular user.

The Software Publications Release History serves as a guide in determining which revision level of software documentation corresponds to the Programming System Report (PSR) level of installed site software.

The users of Update can find additional pertinent information in the Control Data Corporation manuals listed below. The manuals are listed alphabetically within groupings that indicate relative importance to readers of this manual. The applicable operating systems are also indicated.

The following manuals are of primary interest:

Publication	Publication Number	NOS 1	NOS 2	NOS/BE 1	SCOPE 2
NOS Version 1 Reference Manual, Volume 1 of 2	60435400	X			
NOS Version 2 Reference Set, Volume 3 System Commands	60459680		X		
NOS/BE Version 1 Reference Manual	60493800			X	
SCOPE Version 2 Reference Manual	60342600				X

The following manuals are of secondary interest:

Publication	Publication Number	NOS 1	NOS 2	NOS/BE 1	SCOPE 2
NOS Version 1 Diagnostic Index	60455720	X			
NOS Version 2 Diagnostic Index	60459390		X		
NOS/BE Version 1 Diagnostic Index	60456490			X	
NOS Version 1 Manual Abstracts	84000420	X			
NOS Version 2 Manual Abstracts	60485500		X		
NOS/BE Version 1 Manual Abstracts	84000470			X	
Software Publications Release History	60481000	X	X	X	X

CDC manuals can be ordered from Control Data Corporation, Literature and Distribution Services, 308 North Dale Street, St. Paul, Minnesota 55103.

This product is intended for use only as described in this document. Control Data cannot be responsible for the proper functioning of undescribed features or parameters.

CONTENTS

APPENDIXES

INDEX

FIGURES

TABLES

NOTATIONS

Throughout this manual, the following conventions are used to present Update directives:

UPPERCASE Uppercase letters indicate words, acronyms, or mnemonics either required by Update or produced as output by Update. All words printed entirely in uppercase letters have a preassigned meaning to Update. These words include command verbs and keywords.

lowercase Lowercase words identify variables for which values are supplied by the Update user or by Update as output. These words generally indicate the nature of the information they represent (numerical value, file or job name, and so forth).

... Ellipsis indicate that omitted entities repeat the form and function of the last entity given. An ellipsis immediately following a command element indicates it can be repeated at your option.

Δ The delta symbol represents a blank used as a separator.

INTRODUCTION 1

Update is a utility for maintaining and manipulating a mass storage file containing images of coded punched cards or text lines. Once these images have been made a part of an Update program library, physical punched cards or lines can be eliminated. Update can maintain 6-bit (display code) line images and 8-bit (ASCII) line images on the same program library. Line images can be as long as 256 characters. The length of line images written to the compile file can be controlled by an Update directive. The entire line appears in the output listing.

A file of line images to be manipulated by Update must be in a special format known as a program library. Three types of Update runs generate or manipulate a program library:

- A creation run generates a program library from the input stream text.
- A correction run manipulates the contents of an existing program library.
- A copy run changes the format of a program library from random to sequential or from sequential to random.

A separate new program library can be created with a correction run; the changes made during the correction run are permanently recorded in the new program library. Changes made during a correction run never permanently alter the existing program library. The changes become permanent only through the creation of a new program library.

As each line image is written to the program library, Update assigns it a unique identifier.

Groups of line images within the program library are known as decks. Each program library must have at least one deck; the maximum number of decks is 262143. Deck grouping is significant in terms of extracting line images from the program library in a format suitable for use by a compiler, assembler, or print routine. While an individual line can be referenced for purposes such as deletion of that line or insertion after that line, the smallest unit that can be extracted from the program library is the deck. Program libraries can be maintained either in the display code or ASCII character set. All ASCII input or output codes are 8-bit characters, right-justified in a 12-bit byte (ASCII 8/12).

Typically, use of Update involves maintenance of a group of compiler or assembly language routines. For convenience, the programmer often specifies each routine or group of related routines as an individual deck. One routine can then be changed or extracted without affecting other routines in the program library. Because each line image in a deck has its own identifier (a deck name) and an Update-supplied sequence number, the line image can be referenced individually in order to correct or change a routine. Then, the deck containing the modified routine can be extracted from the program library and used as if it had been entered into the system as a punched deck.

A deck can be composed of punched cards or images of punched cards. Update makes no assumptions about contents. While programs are the usual contents maintained by Update, this utility is equally applicable to a set of data cards or any other text.

The programmer controls Update operations through the following two mechanisms:

- The UPDATE control statement parameters specify the general operations to be performed. The file parameters control the files to be manipulated and influence the type of operations performed.
- The input stream directives specify the detailed operations to be performed and specify the line images to be made a part of the program library. The instructions for Update operation are called directives; the line images for the program library are called text. The input stream can be either part of the job deck containing the UPDATE control statement or a separate file.

FILE NAMES

Files used or generated by Update have generic names that are related to their default logical file names. The following names are used in the remainder of this manual in describing Update operations:

- Input file - the user-supplied file or part of the job deck that contains the input stream of Update directives and text.
- Output file - the listing file generated by Update that contains the status information produced during Update execution. It is in a format suitable for printing.
- Program library - the file generated by an Update creation run that contains the decks of line images. When the file is created, it is known as the new program library. When the file is corrected, it is known as the old program library. Line images in the program library are in a format that can be manipulated by Update, but the format is meaningless to most other formats and utilities.
- Compile file - the file generated by Update that contains line images restored to a format that is acceptable to a compiler or assembler. Decks written to the compile file during any given run are controlled by the Update mode selected, by control statement parameters, and by directives in the input stream.
- Source file - the file generated by Update that contains line images of an input stream that allows regeneration of the program library.

- Merge file - the file that contains a program library that Update merges with the old program library to create a new program library.
- Pullmod file - the file that contains directives and text of recreated correction sets.

Section 2 contains a detailed discussion of the files used or generated by Update.

DIRECTIVES

The directives for Update are interspersed with text in the input stream. They are distinguished by the presence of a control character contiguous with a directive keyword. More than 40 directives exist. The directives can be grouped according to the following operations:

- Identify decks.
- Control compile file contents.
- Manipulate primary or secondary input streams.
- Control overall handling of input files.
- Modify program library contents.

Section 3 contains a detailed discussion of Update directives.

CREATION RUN

A creation run constructs a program library. It is the original transfer of punched cards or line images into Update format. The input file of a creation run can consist of ASCII 8/12 or display code characters. ASCII characters must be right-justified in 12-bit bytes. The new program library is created in ASCII, if the input file uses ASCII and if the N or N8 parameter is specified on the UPDATE control statement.

A creation run exists when the first line read from the input stream is a DECK or COMDECK directive. A creation run also exists when one or more of the following ten directives precedes the first DECK or COMDECK directive:

ABBREV
DECLARE
LIMIT
LIST
NOABBREV
NOLIST
READ
REWIND
SKIP
/(comment)

The presence of any other directive before the first DECK or COMDECK directive causes Update to consider the run to be a correction run.

In addition to the preceding directives, the following are the only Update directives that can be used during a creation run:

CALL
CWEOR
ENDTEXT
ENDIF
TEXT
IF
WIDTH
WEOR

Each DECK or COMDECK directive defines a deck to be inserted into the program library that is being created. All text and directives following a DECK or COMDECK directive, until the next DECK or COMDECK directive, are considered to be part of the deck. Each line image receives the deck name and a unique sequence number so that the images can be referenced individually. The DECK or COMDECK directive defining the deck itself is assigned the sequence number one.

Update decks can be one of two types: a regular deck declared with a DECK directive, or a common deck declared with a COMDECK directive. DECK and COMDECK differ in that a common deck can be called by name so that it is inserted into the text of another deck when the compile file is being generated. One copy of the common deck exists on storage, but multiple copies can be part of a compile file.

When the library is created, Update generates a deck named YANK$$$ as the first deck on the library. The YANK$$$ deck contains all the YANK, SELYANK, YANKDECK and DEFINE directives that are encountered during Update runs. (The YANK$$$ deck is described further in appendix D, File Format and Structure.) Update also generates a deck list and directory during a creation run. The deck list contains the names of all decks in the library and the location of the first word for each deck (random library) or the relative order of the decks (sequential library). The directory contains one entry for each DECK, COMDECK, and IDENT directive that is used for the library.

CORRECTION RUN

A correction run, which is the most common use of Update, introduces changes into the existing program library. These changes exist only for the duration of the run unless a new program library is generated. Update recognizes a correction run when it encounters a directive other than one of the ten creation run directives prior to encountering DECK or COMDECK.

A correction run consists of a read-input-stream phase and a correction phase. During the first phase, Update reads directives and text, adds any new decks, and constructs a table of requested correction operations. During the second phase, Update performs the requested modifications on a deck-by-deck basis.

The order in which a correction run is processed is not always the same as specified. During a correction run using a RESTORE directive and then a DELETE directive on the same line image or deck, the DELETE directive is processed first. If the PURDECK directive is used, it is also processed first, assuming that one UPDATE directive is used.

The input file of a correction run can be in ASCII 8/12 or display code characters. Update uses ASCII for the program library, if the character set of the old program library uses ASCII and the N or N8 parameter is on the UPDATE control statement.

The corrections to the library (the newly inserted lines, replaced lines, and deleted lines) make up the correction sets. The IDENT directive assigns a unique identifier to each line image inserted by the correction directives. Each inserted line image is assigned a sequence number beginning with one for each IDENT name. All line images having the same correction set identifier comprise a correction set.

Update permits a user to remove (yank) the effects of a correction set or deck and later restore the correction set or deck. This feature is convenient for testing new code. Requests for yanking are maintained in the YANK$$$ deck. Before obeying a correction, Update checks the correction identifier against the YANK$$$ deck to see if the correction has already been yanked. If the correction has been yanked, an informative message is issued and processing continues. This effect on the YANK$$$ deck can be selectively controlled through DO and DONT directives within the decks.

The image of a line, even though deleted through DELETE or yanking, is maintained permanently on the program library with its current status (active or inactive) and a chronological history of modifications to its status. The images contain information known as correction history bytes. The history bytes that are generated by Update contain the history and status of the line and enable Update to reverse status. Deletion of a line, for example, is accomplished by the addition of a correction history byte to the line image rather than a physical deletion of the image. Consequently, the line can be reactivated at some later time.

Update also allows a complete and irreversible purging of correction sets and decks. When a correction set or deck is purged, it is physically removed from the library.

COPY RUN

A copy run changes the old program library format from sequential to random or from random to sequential. Update recognizes a copy run when either the A or B parameter is specified on the UPDATE control statement. Since Update does not read the input file on a copy run, no other operations are performed. The control statements COPY, COPYBF, COPYCF, COPYBR, or COPYCR should not be used on random access files since the operating system might not recognize that the copied file is a random access file.

DECK LIST AND DIRECTORY ORDER

Update maintains a deck list and directory for its internal use. The deck list and directory are only significant to the user when ranges of decks or correction sets are specified on Update directives. The output file (O parameter) lists the order of the deck names and correction set identifiers. The deck list and directory are always maintained in display code.

The deck list contains a list of all decks in the program library. The original entries of the deck list correspond to the order of the decks when written during the creation run. Subsequent entries are added to the end of the list as they are introduced in the program library. Therefore, deck list order might not reflect actual deck order in the program library, since the user determines deck location within the program library through directives.

The location of an entry in the deck list is significant in terms of parameters for PURDECK, SEQUENCE, and COMPILE directives in which a range of decks can be referenced. The order of names in a range reference must be the same as the order in the deck list. The decks named and all the decks between are then processed in accordance with the directive. An error exists if they are in reverse order.

Similarly, as each deck and correction set is introduced into the program library, Update creates an entry in an internal directory in chronological sequence. The location of an entry in the directory is significant in terms of parameters for PURGE and YANK directives in which a range of correction sets can be referenced. The order of reference must be the same as the order of the directory. The identified correction sets and all the sets between are processed in accordance with the directive. An error exists when a correction set range is not referenced in the order the sets were introduced into the library.

UPDATE MODE

The content of any compile file, source file, or new program library produced during a correction run is affected by the Update mode. (Table 2-2 in section 2 summarizes the effect of mode upon file content.) The mode of an Update run is determined by a combination of the omission or specification of the F and Q parameters on the Update control statement as summarized in table 1-1.

TABLE 1-1. UPDATE MODE

Parameter Specified	Mode
F	Full mode in which all decks on the old program library are processed.
Q	Quick mode in which only decks specified on COMPILE directives and decks added through ADDFILE directives are processed.
F and Q	Quick mode.
F and Q omitted	Normal selective mode in which the only decks processed are those modified or those specified on COMPILE directives.

The mode chosen depends on how extensively the user wishes to modify the program library and its size. If the library contains many decks and the user wishes to modify only a few decks, quick mode should be used. If there are many decks and the user wants all decks to be processed, full mode should be used. Normal selective mode should be used when only those decks modified or specified are wanted in the compile file.

UPDATE FILES

During its execution, Update manipulates as many as eight files that can be referenced by the user. The files involved with any given run depend on the following:

- The parameters selected by the UPDATE control statement.
- Whether the run is a creation run, correction run, or copy run.

The files that Update generates or uses are described in this section. Each of these files has a default name, but other names can be specified through the appropriate parameters on the UPDATE control statement.

File characteristics are summarized in table 2-1. The ASCII chracter set codes used in a file are 8-bit characters, right-justified, in 12-bit bytes.

Whether or not a file is optional, used, or not applicable on an Update run depends on the type of run, as follows:

- Creation run - the user must supply the input file. Update generates the new program library, compile file, and output file by default. The generation of a source file is optional. No other files are applicable on a creation run.
- Correction run - the user must supply the input file, the old program library, and the merge file (if a merge is to take place). Update generates, by default, the output and compile files. The creation of a new program library, source file, and pullmod file is optional on a correction run.

TABLE 2-1. FILE SUMMARY

File	Default Name	Contents	Mode	Default Position
Input	INPUT	The input stream.	Binary	Remains at the end of the record (end-of-section for SCOPE 2) terminating Update directives. If Update aborts, location of input file is unpredictable.
New program library	NEWPL	Updated library.	Binary	Rewound before and after run.
Old program library	OLDPL	Library to be updated.	Binary	Rewound before and after run.
Secondary old program library	None	Library from which common decks can be called.	Binary	Rewinding not necessary because file must be random.
Compile	COMPILE	Line images for assembly or compilation.	Binary	Rewound before and after run.
Output	OUTPUT	Information for the programmer.	Binary	Remains in current position. File is not rewound.
Source	SOURCE	Line images for regeneration of a new program library.	Binary	Rewound before and after run.
Merge	MERGE	Second library to be merged into new program library.	Binary	Rewound before and after run.
Pullmod	Source file	Re-created correction sets.	Binary	Rewound before and after run.

- Copy run - the user must supply the old program library. Update generates, by default, the new program library and the output listing file. No other files are applicable on a copy run and, if specified, are ignored.

The contents of any compile file, source file, or new program library produced during a run are affected by the Update mode and the file format of the old program library. The contents of these files are summarized in table 2-2.

INPUT FILE

The input file contains the input stream; it must contain coded lines or their equivalent. The input stream consists of directives that direct Update processing and text to be added to the program library. The directives allowed in the input stream are determined by the type of Update run. The input data can be equal to or less than 256 characters.

Update initially reads the input stream from the primary input file specified by the I parameter of the UPDATE control statement; default file name is INPUT. Update stops reading directives and text when it encounters a 7/8/9 card or its equivalent, or end-of-information (EOI).

If Update encounters a READ or ADDFILE directive in the input stream, it stops reading from the primary input file and starts reading from the file specified on the directive. Update reads one system-logical record (one section for SCOPE 2) from the secondary input file, then resumes reading from the primary input file.

The input file can only consist of ASCII 8/12 or display code characters during a creation or correction run. No attempt to input ASCII 6/12 data (on NOS) should be made. The ASCII 6/12 data must first be converted to ASCII 8/12 data using the NOS FCOPY control statement. Update uses ASCII 8/12 for the program library if the character set of the old program library uses ASCII 8/12 and if the N8 parameter is on the UPDATE control statement. The input file character set is determined from the first line of the input file. If other than ASCII 8/12 character set data is entered, the invalid code is translated into a blank. See appendix A for the character set tables.

PROGRAM LIBRARY FILES

A program library is created during an Update run and can be manipulated in later runs. The library consists of a file of line images and internal information in a special format that can be processed only by Update. The line images are grouped into decks. Each line image is represented in a compressed format with multiple space characters removed that adds a line identifier. The format also includes history and status information that is known as correction history bytes. Program libraries can be maintained in display code or ASCII code characters.

TABLE 2-2. FILE CONTENTS AND UPDATE MODE

File	Normal Selective Mode Contents	Full Mode Contents	Quick Mode Contents (Sequential OLDPL)	Quick Mode Contents (Random OLDPL)
New Program Library	Regular decks and common decks after corrections are made.	Regular decks and common decks after corrections are made.	All decks specified on COMPILE directives, any common decks they call, and any common decks encountered prior to all decks of COMPILE.	Decks specified on COMPILE directives and any common decks they call.
Compile File	Decks corrected or on COMPILE directives and decks calling a corrected common deck (unless the calling deck precedes the common deck or NOPROP is specified on COMDECK).	Active decks on old program library.	Decks on COMPILE directives and decks added via ADDFILE plus called common decks.	Decks on COMPILE directives and decks added via ADDFILE plus called common decks.
Source File	Active lines and decks required to re-create the the library.	Active lines and decks required to re-create the library.	Active lines from decks specified on COMPILE directives, any common decks they call, and any common decks encountered prior to all decks on COMPILE.	Active lines from decks specified on COMPILE directives and any common decks they call. A common deck called by a deleted *CALL directive.

The program library also contains a deck list and a directory. The deck list contains the names of all decks in the library. In addition to deck names, the directory also contains the names of all correction sets. Unless changed by the E parameter of the UPDATE control statement, the names in both the deck list and the directory exist in the order they were introduced.

Update can create and maintain program library files in two distinct formats: random and sequential. (These formats are described in detail in appendix D.) A random program library can be processed substantially faster than a sequential program library; however it can exist only on disk and not on tape.

NEW PROGRAM LIBRARY

A new program library is initially generated on a creation run. It contains directives and text in an updatable format. File content is determined by the file format of the old program library and Update mode as shown in table 2-2. The new program library name is specified by the N parameter of the UPDATE control statement; the default file name is NEWPL. The new program library character set is the same as the character set used in the input file during the creation run.

For subsequent correction runs, the previously generated new program library is identified as the old program library. A new program library that incorporates the changes made during the correction run is generated if requested. If the old program library is in display code, the correction run character set can be ASCII 8/12 or display code.

A new program library can be in random or sequential format. In the absence of the W parameter on the UPDATE control statement, the format is determined by file residence and record type as shown in table 2-3.

TABLE 2-3. NEW PROGRAM LIBRARY FORMAT

Format	NOS and NOS/BE	SCOPE 2
Random	File is on mass storage and W is not selected.	File is on mass storage, record type is W unblocked, and W is not selected.
Sequential	File is on magnetic tape or W is selected.	File is staged or online tape; or is on mass storage as record type S or record type W blocked; or W is selected, or R specifies no rewind.

A new program library can be written or appended to an existing permanent file according to the permission rules of NOS, NOS/BE, or SCOPE 2.

OLD PROGRAM LIBRARY

The old program library is the file that was generated as a new program library in a previous run. It contains a record of changes made since the program library was created. The old program library name is specified by the P parameter of the UPDATE control statement; the default file name is OLDPL.

An old program library is required for a correction run since it is the program library to be updated. On a copy run, the old program library is not modified, but is copied to a sequential or random new program library. If an old program library is specified on a creation run, it is ignored.

In addition to the old program library to be updated, up to seven additional (secondary) old program libraries can be specified by the P parameter of the UPDATE control statement. Decks on the old program library can call common decks from the old program library or from any of the other secondary program libraries. No Update directive other than CALL can be used to reference common decks on secondary old program libraries. Common decks on secondary old program libraries can call common decks that reside on any of the old program libraries. Program libraries are searched in the order specified to find the called common decks. The called common decks that reside on the secondary old program libraries are not added to a new program library.

The secondary old program libraries must be random, have a unique name, and have the same master control character as the old program library. If these conditions are not met, a diagnostic message is issued.

When creating a new program library on a creation run that contains calls to common decks that reside on secondary old program libraries, C=0 must be specified on the UPDATE control statement.

COMPILE FILE

The compile file contains copies of decks in the program library restored to a format that can be processed by a compiler or assembler. The decks written to the file are determined by Update mode and the file format of the old program library as shown in table 2-2. Through the WIDTH directive, the user can specify whether the text on the file is to have Update line identifiers on each line of text.

Compile file name is specified by the C or K parameter of the UPDATE control statement; default file name is COMPILE. If the K parameter is specified, then decks are written to the compile file in the order they appear on COMPILE directives. (Any decks not specified on COMPILE directives follow those specified.) If the C parameter is specified, then decks are written on the compile file in the order they appear in the deck list.

The user has control over the decks written to the compile file through the compile file directives. Common decks can be called conditionally or unconditionally according to compile file directives embedded in the program library decks. Additional control of compile file format is afforded the user through directives that cause a system-logical record (end-of-section for SCOPE 2) of the specified level to be written at the end of decks. The compile file directives can be in the original decks or can be inserted into the program library decks during correction runs. These directives are interpreted when the compile file is written; the directives themselves are not written on the compile file.

LISTABLE OUTPUT FILE

The listable output file is the print file containing information for use by the programmer. Content of the file is controlled by the L parameter of the UPDATE control statement with options that can select a listing of directives processed, errors, comments, and a list of line images in the program library. The locations of all CWEOR, WEOR, ENDIF, IF, and CALL directives are listed if a compile file is written. If L=0, all listable output is suppressed. Output file name is specified by the O parameter of the UPDATE control statement; default file name is OUTPUT. If the output file is connected to a terminal, the default is L=1.

In quick mode only, Update produces an ordered printout of the deck list of the program library under the heading DECK LIST AS READ FROM OLDPL PLUS ADDED NEW DECKS. A quick mode dummy Update run (no decks added) produces a deck listing of the old program library.

The output file always defaults to display code characters unless the 08 option is specified.

SOURCE FILE

The source file is an optional file generated during a correction or creation run. The source file consists of the line images of an input stream that allows generation of a new program library. Only currently active line images are in resequenced format during a subsequent creation run. Only active DECK, COMDECK, WEOR, CWEOR, WIDTH, CALL, TEXT, IF, ENDIF, and ENDTEXT directives, in addition to all active text, are part of the source file. The line images in the source file do not contain line identifiers.

The source file name is specified by the S parameter of the UPDATE control statement; default file name is SOURCE. The content of the file is determined by the T parameter of the UPDATE control statement and by Update mode and the file format of the old program library as shown in table 2-2. The user is responsible for routing the source file to a punch or other output device.

If either the S or S6 parameter is specified, the source file is written in display code. If the S8 parameter is specified, it is written in ASCII 8/12. The character set of the old program library has no effect on the S8 parameter.

MERGE FILE

The merge file contains a program library to be merged with the old program library into a new program library. Update adds the deck list and directory from the merge file to the deck list and directory on the old program library. Any names on the merge file that duplicate names on the old program library are modified to make them unique as follows:

- The last character of the name is changed by adding 01 (modulo 55_8) until all valid characters have been tried.

- A character is appended to the name and the first step is repeated. Characters are appended until the name reaches nine characters.

If no unique name can be generated by this method, the Update run is abnormally terminated. Directives that reference these changed names are modified to agree with the new name. All names that required modification are listed in the output file.

Merge file name is specified by the M parameter of the UPDATE control statement; default name is MERGE. All Update functions that are valid in a correction run are valid with the merge parameter. Care should be exercised when including modifications in a merge run. Update might change a name to which correction lines have been applied. In this case, corrections can refer to the wrong deck or correction set.

Decks from the merge file are added to the new program library after all decks from the old program library are added. This sequence of decks in the new program library can be altered by the MOVE directive if desired.

PULLMOD FILE

The pullmod file contains directives and text of recreated correction sets specified on PULLMOD directives. These re-created correction sets produce the same results as the original sets. This feature permits a user to take an earlier version of the library and apply selected correction sets. The file has the same format as an input file.

File name is specified by the G parameter of the UPDATE control statement. If no file is specified, pulled modifications are written to the source file specified by the S or T parameter; if no source file is specified, the re-created correction sets are written to a file named SOURCE.

UPDATE DIRECTIVES

Directives allow the user to create program libraries. Directives also extensively control and direct the correction and modification process. Directives perform the following operations:

- Identify decks.
- Control compile file contents.
- Manipulate primary or secondary input streams.
- Control overall handling of the input file.
- Modify program library contents

Each directive is summarized in table 3-1.

TABLE 3-1. SUMMARY OF UPDATE DIRECTIVES

Directive Keyword Abbreviation	Directive Format	Use
none	*ABBREV	Resume checking for abbreviated directives.
*AF	*ADDFILE lfn,name	Read creation directives and text from named file and insert after specified deck or line.
*B	*BEFORE line	Write subsequent text lines before line identified.
*CA	*CALL deck	Write common deck to compile file.
*CH	*CHANGE oldid,newid, . . . ,oldid,newid	Change correction set identifier.
*CD	*COMDECK deck,NOPROP	Define common deck and propagation parameter.
*C	*COMPILE deck1,deck2, . . . ,deckn	Write specified decks to compile file, source file, and new program library.
	*COMPILE deck1.deck2	Write inclusive range of decks to compile file, source file, and new program library.
*CY	*COPY deck,line	Copy and insert specified line from named deck.
	*COPY deck,line1,line2	Copy and insert specified range of lines from named deck.
	*COPY deck,line1,line2,lfn	Copy specified range of lines from named deck to specified file.
*CW	*CWEOR level	Conditionally write end-of-record (end-of-section for SCOPE 2) or end-of-file.
*DK	*DECK deck	Define deck to be included in program library.
*DC	*DECLARE deck	Restrict corrections to named deck.
*DF	*DEFINE name1,name2, . . . ,namen	Define names to be tested by IF directive while compile file is being written.
*D	*DELETE line	Deactivate specified line and optionally insert text in its place.
	*DELETE line1,line2	Deactivate inclusive range of lines and optionally insert text in their place.

TABLE 3-1. SUMMARY OF UPDATE DIRECTIVES (Contd)

Directive Keyword Abbreviation	Directive Format	Use
none	*DO ident1,ident2, . . . ,identn	Reactivate yanked lines in specified correction sets until a DONT is encountered.
*DT	*DONT ident1,ident2, . . . ,identn	Terminate the DO for specified correction sets.
*EI	*ENDIF	Indicate end of conditional text.
*ET	ENDTEXT	End delimiter for sequence of lines identifying text.
*ID	*IDENT idname,B=num,K=ident,U=ident	Define correction set, bias for seqnum, and whether specified correction sets must be known or unknown to process this set.
none	*IF type,name,num	Write specified number of following lines to the compile file if name of type DECK, IDENT, or DEF is known.
	*IF -type,name,num	Write specified number of following lines to the compile file if name of type DECK, IDENT, or DEF is unknown.
*I	*INSERT line	Write subsequent text lines after line identified.
*LT	*LIMIT n	Limit listable output to n lines.
*L	*LIST	Resume listing lines encountered in input stream.
*M	*MOVE deck1,deck2	Place deck1 after deck2.
*NA	*NOABBREV	Do not check for abbreviated directives.
*NL	*NOLIST	Disable list option 4.
*PM	*PULLMOD ident1,ident2, . . . ,identn	Re-create specified correction sets and write them to file specified by the G option.
*PD	*PURDECK deck1,deck2, . . . ,deckn	Permanently remove specified decks from program library.
	*PURDECK deck1.deck2	Permanently remove inclusive range of decks.
*P	*PURGE ident1,ident2, . . . ,identn	Permanently remove specified correction sets from program library.
	*PURGE ident1.ident2	Permanently remove inclusive range of correction sets.
	PURGE ident,	Permanently remove specified correction set and all sets introduced after it.
*RD	*READ lfn	Read directives and text from specified file.
*R	*RESTORE line	Reactivate specified line and optionally insert text after it.
	*RESTORE line1,line2	Reactivate inclusive range of lines and optionally insert text after them.
*RW	*REWIND lfn	Reposition named file to beginning-of-information.

TABLE 3-1. SUMMARY OF UPDATE DIRECTIVES (Contd)

Directive Keyword Abbreviation	Directive Format	Use
*SP	*SELPURGE deck1.ident1,deck2.ident2, . . . ,deckn.identn	Permanently remove all lines in specified deck that belong to specified correction set.
*SY	*SELYANK deck1.ident1,deck2.ident2, . . . ,deckn-identn	Deactivate all lines in specified deck that belong to specified correction set.
*S	*SEQUENCE deck1,deck2, . . . ,deckn	Resequence all active lines and purge all inactive lines in specified decks.
	*SEQUENCE deck1.deck2	Resequence all active lines and purge all inactive lines in inclusive range of decks.
*SK	*SKIP lfn,n	Reposition named file forward the specified number of logical records.
*T	*TEXT	Beginning delimiter for sequence of lines identifying text.
*W	*WEOR level	Write end-of-record or end-of-file according to specified level.
*WI	*WIDTH linelen,idlen	Reset size of line image written to compile file.
*Y	*YANK ident1,ident2, . . . ,identn	Temporarily remove specified correction sets from program library.
	*YANK ident1.ident2	Temporarily remove inclusive range of correction sets.
*YD	*YANKDECK deck1,deck2, . . . ,deckn	Temporarily deactivate decks specified.
none	*/comment	Copy text to listable output file.

DIRECTIVE FORMAT

The general format of Update directives is shown in figure 3-1. A directive must begin with the master control character in column one. Comments can be placed after the last parameter of the directive. The comment and final parameter must be separated by one or more blanks. Most directives have both a full keyword and an abbreviated keyword as shown in table 3-1; when the NOABBREV directive is in effect, Update does not recognize the abbreviated forms of directive names. Any line in the input stream that cannot be recognized as a directive is assumed to be text.

The master control chracter is recorded in the program library. For a correction run, the master control character should match the character used when the program library was created. If the characters do not match, Update uses the character specified in the program library.

Since Update scans all 256 columns when interpreting directives, comments or sequencing information from a previous run can be interpreted as the parameter list. Update interprets comments or sequencing information as the parameter list when a list is not specified on WEOR, CWEOR, DECLARE, or

```
*keyword p-list
```

*   Master control character that distinguishes a directive from a text line. Must appear in column 1. This character can be changed through the * parameter of the UPDATE control statement.

keyword   Name of one of the Update directives or an abbreviation for a directive. No blanks can occur between the master control character and the keyword; a comma or blank terminates the keyword.

p-list   Parameters identifying decks, cards, lines, or files. Some directives have no parameters. Multiple blanks can appear between the keyword and parameters. Parameters in the list are separated by commas; embedded blanks cannot appear in the list. A blank terminates the p-list.

Notice that several parameters contain a period as part of a single parameter.

Figure 3-1. General Update Directive Format

ADDFILE directives. To avoid this problem, a null parameter list should be specified on these directives in the following manner:

```
    *WEOR,,     *DECLARE,,

    *CWEOR,,    *ADDFILE,,,
```

Specifying a null parameter field ensures that Update will use the default values as parameters rather than using the comments or sequencing information. Errors will occur if Update tries to use the comment or sequencing information as the directive parameter list.

LINE IDENTIFIERS

Each line image in a program library is uniquely identified by an identifier and a sequence number. The identifier is the name of the deck or correction set from which the line image originated; Update supplies the sequence number. Line identifiers assigned by Update are usually permanent; they can be changed only through the use of the SEQUENCE and CHANGE directives.

Update recognizes one full form and two short forms of line identifiers. The full form line identifiers are shown in figure 3-2. The two short forms of line identifiers, which can be used on BEFORE, INSERT, DELETE, RESTORE, and COPY directives, are expanded.

```
ident.seqnum

ident.   1- through 9-character name of a correc-
         tion set or deck.  A period terminates
         the identifier.

seqnum   Decimal ordinal (1 through 131071)
         representing the sequence number of the
         line within the correction set or deck.
         Any character other than 0 through 9
         terminates the sequence number.
```

Figure 3-2. Full Form of Line Identification

In the short form (shown in figure 3-3), idname is assumed to be the last explicitly named identifier given on a BEFORE, INSERT, DELETE, RESTORE, or COPY directive, whether or not it is a deck name. The dname is assumed to be the last explicitly named identifier given on a BEFORE, INSERT, DELETE, RESTORE, or COPY directive that is known to be a deck name. Both of these default identifiers are originally set to YANK$$$; therefore, the first directive using a line identifier must use the full form to reset the default.

```
seqnum     Expands to idname.seqnum where idname
           is a correction set identifier, whether
           or not it is also a deck name.

.seqnum    Expands to dname.seqnum where dname is
           a deck name.
```

Figure 3-3. Expansion of Short Forms of Line Identification

All deck names are also identifiers (but all identifiers are not deck names). Thus, if EXAMPLE is the deck name last used, and there is no subsequent explicit reference to a correction set identifier, then both .281 and 281 expand to EXAMPLE.281 as the line identifier. If there is an explicit reference to a correction set identifier ABC after the explicit reference to the deck name, then 281 would expand to the line identifier ABC.281 while .281 would expand to EXAMPLE.281.

Figure 3-4 shows the differences in identifier expansion depending on the order of the directives. A is a deck name and B is a correction set identifier on an old program library.

```
*ID C
*INSERT A.2
     data line
*INSERT B.1
     data line
*D    2,  3    expands to *DELETE B.2,B.3
*D    4, .5    expands to *DELETE B.4,A.5
*D   .7,  5    expands to *DELETE A.7,B.5
*D   .9, .10   expands to *DELETE A.9,A.10

whereas:

*ID D
*INSERT B.1
     data line
*INSERT A.2
     data line
*D    2,  3    expands to *DELETE A.2,A.3
*D    4, .5    expands to *DELETE A.4,A.5
*D   .7,  5    expands to *DELETE A.7,A.5
*D   .9, .10   expands to *DELETE A.9,A.10
```

Figure 3-4. Examples of Line Identifier Expansion

DECK IDENTIFYING DIRECTIVES

Each deck to be placed on a program library must be introduced by a DECK or COMDECK directive during a creation or correction run. When Update encounters one of these directives in the input stream prior to any correction directive, the run is considered to be a creation run. When Update encounters one of these directives while inserting new text lines, it terminates the insert and adds the decks to the program library following the line specified.

When a deck is added through the use of a DECK or COMDECK directive during a creation run or an ADDFILE directive during a correction run, termination of that deck occurs when Update encounters another DECK or COMDECK directive, or the end of a system-logical record. Lines within that deck are identified by the name of the deck or common deck to which the lines belong and are numerically sequenced beginning with 1 for the DECK or COMDECK directive. When a deck is inserted during a correction run as if it were text (that is, through the use of an INSERT, DELETE, BEFORE, or RESTORE directive), the deck is terminated by any condition that normally terminates insertion. The contents of the deck, including the DECK or COMDECK line, are identified by the correction set name and are numerically sequenced as if they were normal insertion text.

Frequently, a DECK or COMDECK directive precedes each program or subprogram in a given program library. More than one subprogram, however, can be included in a deck, as is indicated in figure 3-5. Normally, two programs are grouped together if modification of one program requires reassembly of both programs.

```
        *DECK      FIRST
                   IDENT    FIRST
                   .....
                   END
                   IDENT    SECOND
                   .....
                   END

        *COMDECK FDATA
               BLOCK DATA
               COMMON/J3/A(10)
               DATA A/3*0., 7*1.0/
               END
```

Figure 3-5. Example of Deck Structure

Because DECK and COMDECK directives can be deactivated by DELETE, YANK, or SELYANK, line images belonging to one deck at the beginning of an Update run can belong to a different deck at the end of the run. When a DECK or COMDECK directive is deactivated, all line images in the deactivated deck become members of the preceding deck on the program library; they retain their original line identifiers. If there is no preceding deck, then they become part of the YANK$$$ deck.

DECK DIRECTIVE

The DECK directive establishes a deck in the program library. It is one of the two directives that establishes the existence of a creation run. The directive can also be used in any correction run to add a deck to the location indicated by a preceding INSERT, BEFORE, DELETE, or RESTORE directive. Each deck must have a unique name within the program library. The DECK directive itself is part of the program library and has a sequence number of one within the name established by the directive. DECK directive format is shown in figure 3-6.

```
*DECK deck

deck   Name of deck.  Must be 1 through 9 char-
       acters.  Any character in the CDC display
       code character set is allowed, except
       blank, period, comma, and colon.  Must
       not duplicate the name of any other deck
       in program library.
```

Figure 3-6. DECK Directive Format

COMDECK DIRECTIVE

The COMDECK directive establishes a common deck that can be called from other decks as they are being written to the compile file. It is one of the two directives that establishes the existence of a creation run. The directive can be used in any correction run to add a common deck to the location specified by a preceding INSERT, BEFORE, or RESTORE directive. Each common deck must have a unique name. The COMDECK directive itself is part of the program library and has a sequence number of one within the name established by the directive. The COMDECK directive format is shown in figure 3-7.

```
*COMDECK deck,NOPROP

deck      Name of deck.  Must be 1 through 9
          characters.  Any character in the CDC
          display code character set is allowed,
          except blank, period, comma, and colon.
          Must not duplicate the name of an
          existing deck.

NOPROP    Indicates that decks calling this
          common deck are not to be considered
          as modified when the common deck itself
          is modified; that is, the effects of
          common deck changes are not to be
          propagated during normal Update mode.
          Optional.
```

Figure 3-7. COMDECK Directive Format

The NOPROP parameter of the COMDECK directive determines whether a deck calling a corrected common deck is to be considered as having been corrected. If NOPROP is specified, only the common deck is considered to be corrected. On the other hand, if NOPROP is not specified, the common deck and the calling decks are considered to be corrected.

A common deck should be placed before any of the decks that call it. If the common deck is placed after a deck that calls it, Update might not be able to find it. In addition, decks calling a corrected common deck are not written to the compile file if the calling deck precedes the common deck and the mode is normal selective.

CORRECTION DIRECTIVES

Correction directives control updating of the old program library. New text is assigned a unique line identifier based on the correction set identifier. The corrected program library is written on the new program library; the old program library is not actually changed. Correction directives are illegal on a creation run.

ADDFILE DIRECTIVE

The ADDFILE directive causes Update to add a file of decks to the new program library. ADDFILE differs from the READ directive in that the contents of the specified file are limited to those allowed on a creation run. Unless the specified file is the primary input file, the READ directive cannot appear in the added file. The first line image of the specified file must be a DECK or COMDECK directive. If the INPUT file is specified, the READ directive can be the first image; a DECK or COMDECK directive must then be the first line image on the file specified by the READ directive. An ADDFILE directive cannot appear among directives read from the file specified by a READ directive. The ADDFILE directive format is shown in figure 3-8. If only one parameter is specified, it is assumed to be lfn.

```
*ADDFILE lfn,name

lfn     Name of local file from which decks are
        to be added.  If lfn is omitted, the
        default is the file specified by the I
        parameter of the Update control state-
        ment; the separators are still required.

name    Name of deck or identifier line after
        which decks are to be placed on the pro-
        gram library.  If omitted, the addition
        is made after the last deck on the pro-
        gram library.

        If the name parameter is *, it refers to
        the ident that is known to be a deck name
        most recently mentioned on a BEFORE,
        COPY, DELETE, INSERT, or RESTORE direc-
        tive.  If no such directive precedes the
        ADDFILE, the YANK$$$ deck is used.
```

Figure 3-8. ADDFILE Directive Format

When the specified file is not the primary input file, Update adds directives and text until the end of one system-logical record is encountered. Update then returns to the file specified by the I parameter of the UPDATE control statement and continues processing the primary input stream. The specified file must have the same character set as the primary input file. When the file specified on the ADDFILE directive is the primary input file, however, Update adds line images until a noncreation directive or the end of the system-logical record is encountered.

Update does not reposition the file specified on the ADDFILE directive. Any repositioning must be requested by the SKIP or REWIND directive.

BEFORE DIRECTIVE

The BEFORE directive inserts text line images and compile file directives in the program library before the specified line image. The line images inserted are placed immediately after the directive. Line images cannot be inserted into the YANK$$$ deck. The inserted line images receive line identifiers established by the correction set name of the preceding IDENT directive. The BEFORE directive format is shown in figure 3-9.

```
*BEFORE line

line    Line identifier of line before which the
        insertion is to be made.
```

Figure 3-9. BEFORE Directive Format

Unless a TEXT directive has been encountered, Update terminates an insertion when it encounters the next insertion directive or a PURGE, PURDECK, IDENT, SELPURGE, ADDFILE, or SEQUENCE directive. On the other hand, compile file directives are inserted as if they are text after Update checks for correct syntax. Update interprets all other directives without terminating insertion; however, the directives are not inserted into the deck.

CHANGE DIRECTIVE

The CHANGE directive renames correction set identifiers. It cannot be used to change deck names. As a secondary effect, changing the name of the correction set invalidates any YANK or SELYANK directives that refer to the set by its previous name. Since a CHANGE directive goes into effect immediately, any subsequent references to the correction set must use the new name. The CHANGE directive need not be part of a correction set. CHANGE directive format is shown in figure 3-10.

```
*CHANGE oldid,newid,...,oldid,newid

oldid    Name of correction set to be changed.

newid    New correction set name.  Must be 1
         through 9 characters.  Any character in
         the CDC display code character set is
         allowed, except blank, period, comma,
         and colon.  Must not duplicate the name
         of any other correction set in the pro-
         gram library.
```

Figure 3-10. CHANGE Directive Format

COPY DIRECTIVE

The COPY directive copies active line images from a deck on the old program library and inserts the images into another deck as if they are text in an input stream, or the COPY directive copies active line images to a specified file. Since Update copies the line images into a deck before applying corrections to them, line images can be copied and original images can be modified in the same run. An attempt to copy line images introduced during the same Update run produces an informative message. The COPY directive format for copying line images to a deck on the program library is shown in figure 3-11. The COPY directive format for copying line images to a file is shown in figure 3-12.

```
A.  Copy specified line.

    *COPY deck,line

    deck    Name of deck on old program library
            that contains the line to be copied.

    line    Line identifier of line to be copied.

B.  Copy range of lines.

    *COPY deck,line1,line2

    deck    Name of deck on old program library
            that contains lines to be copied.

    line1,  Line identifiers of first and last
    line2   lines in sequence of lines to be
            copied.
```

Figure 3-11. COPY Directive Format - Copy to Deck

```
*COPY deck,line1,line2,lfn

deck     Name of deck on old program library that
         contains lines to be copied.

line1,   Line identifiers of first and last lines
line2    in sequence of lines to be copied.

lfn      Name of file onto which lines are to be
         copied. The user is responsible for the
         disposition of this file since it is not
         positioned either before or after the
         copy. The file is written as a binary
         file that contains 256-column line images
         with one system-logical record (Section
         for SCOPE2) for each COPY directive;
         sequencing information is not included.
```

Figure 3-12. COPY Directive Format - Copy to File

An INSERT, DELETE, BEFORE, or RESTORE directive must be in effect to use COPY for copying line images to a deck. In figure 3-13, example A, the use of the COPY directive is valid because a preceding INSERT directive has initiated insertion. Line images BDECK.4 through BDECK.8 are copied and inserted after the text lines. The copied line images are sequenced as part of correction set X. The input stream in figure 3-13, example B, is not valid because insertion is not in effect to indicate where to write the line image copies.

```
A.  Valid use of COPY

    *IDENT X
    *INSERT BLAP.11
    (text lines)
    *COPY BDECK,BDECK.4,BDECK.8

B.  Invalid use of COPY.

    *IDENT X
    *COPY BDECK,BDECK.4,BDECK.8
```

Figure 3-13. COPY Directive Example

Placement in the input stream of a COPY directive that copies line images to a file is not restricted; COPY can appear anywhere in the primary input stream. Copying line images to a file is illegal, however, when a secondary input stream is being read as a result of a READ directive.

DELETE DIRECTIVE

The DELETE directive deactivates a line image or a group of line images and optionally inserts text and directives after the deleted line images. The line images to be inserted are placed immediately after the directive. The inserted line images receive line identifiers established by the correction set name of the preceding IDENT directive. The DELETE directive format depends on whether line images to be deactivated are specified by line identifier or by a range of lines, as shown in figure 3-14.

```
A.  Delete specified line

    *DELETE line

    line     Line identifier for single line
             to be deleted.

B.  Delete range of lines

    *DELETE line1,line2

    line1,   Line identifiers of first and
    line2    last lines, in sequence of lines
             to be deleted. Line line1 must
             appear before line2 in the exist-
             ing library. The range can in-
             clude lines already in a deact-
             ivated state.
```

Figure 3-14. DELETE Directive Format

Unless a TEXT directive has been encountered, Update terminates an insertion when it encounters the next insertion directive or a PURGE, PURDECK, IDENT, SELPURGE, ADDFILE, or SEQUENCE directive. On the other hand, compile file directives are inserted as if they are text after Update checks for correct syntax. Update interprets all other directives without terminating insertion; however, the directives are not inserted into the deck.

IDENT DIRECTIVE

The IDENT directive establishes the name for the set of corrections being made. Lines added in this correction set are sequenced within the name specified. All correction set names must be unique. If a new program library is not being generated, a correction set need not begin with an IDENT directive. In this case, Update uses the default name of .NO.ID. for new text lines. The established correction set identifier remains in effect until Update encounters another IDENT directive or a PURGE, SELPURGE, PURDECK, ADDFILE, or SEQUENCE directive. IDENT directive format is shown in figure 3-15.

```
*IDENT idname,B=num,K=ident,U=ident

idname    Name to be assigned to this correction
          set. Must be 1 through 9 characters.
          Any character in the CDC display code
          character set is allowed, except blank,
          period, comma, and colon. Must not
          duplicate the name of another cor-
          rection set or deck. This directive
          causes a new entry in the directory.

B=num     Bias to be added to sequence numbers
          within deck. Optional; 1 is default.

K=ident   Indicator that specified correction
          set name must exist in the directory
          of the library before corrections can
          be made. Optional.

U=ident   Indicator that specified correction
          set name must not exist in the direc-
          tory of the library. Optional.
```

Figure 3-15. IDENT Directive Format

Omitting idname causes a format error. If idname duplicates a name previously used, Update issues an error message. Both errors are nonfatal as long as no new program library is created in the same run.

The B, K, and U parameters on the IDENT directive can appear in any order. If more than one B parameter is specified, Update uses the last one encountered. More than one K or U parameter can be specified; in this instance, all correction set names must be known or unknown as specified before the correction set is processed. (An identifier is known whether it is active or inactive; an identifier that has been yanked is still known. To become unknown, an identifier must be purged.) If the criteria of these parameters is not met, Update skips the correction set and resumes processing with the next IDENT, PURGE, SELPURGE, PURDECK, or ADDFILE directive.

In the following example, the bias of 100 is added to all ZAP correction set line sequence numbers:

```
    *IDENT ZAP,B=100,K=ACE,U=NON,U=ARF
```

The first line image in correction set ZAP has a sequence number of 101, not 1. Update skips the correction set if ACE is unknown or either NON or ARF is known.

INSERT DIRECTIVE

The INSERT directive inserts text line images and compile file directives in the program library after the specified line image. The line images to be inserted are placed immediately after the directive. Line images cannot be inserted into the YANK$$$ deck. The inserted line images receive line identifiers established by the correction set name of the preceding IDENT directive. The range of line images cannot be used when inserting. This causes only the first line image to be processed. INSERT directive format is shown in figure 3-16.

```
*INSERT line

line    Line identifier of line after which in-
        sertion is to be made.
```

Figure 3-16. INSERT Directive Format

Unless a TEXT directive has been encountered, Update terminates an insertion when it encounters the next insertion directive or a PURGE, PURDECK, IDENT, SELPURGE, ADDFILE, or SEQUENCE directive. On the other hand, compile file directives are inserted as if they are text after Update checks for correct syntax. Update interprets all other directives without terminating insertion; however, the directives are not inserted into the text.

MOVE DIRECTIVE

The MOVE directive enables the user to reorder decks while producing a new program library. The deck to be repositioned is moved from its position on the old program library and placed after the specified deck on the new program library. The YANK$$$ deck cannot be moved. A MOVE referencing a deck introduced in the same Update run produces an informative message. This directive does not terminate insertion and need not be part of a correction set. MOVE directive format is shown in figure 3-17.

```
*MOVE deck1,deck2

deck1   Deck name on old program library to be
        moved.

deck2   Deck name after which deck1 is to be
        placed on new program library.
```

Figure 3-17. MOVE Directive Format

PURDECK DIRECTIVE

The PURDECK directive permanently removes a deck or group of decks from the program library. However, the YANK$$$ deck cannot be purged. Every line image in a deck is purged, regardless of the correction set that contains the line image. Purging, unlike yanking, cannot be rescinded. A PURDECK directive can appear anywhere in the input stream; its appearance terminates the current correction set. PURDECK directive format depends on whether decks to be purged are specified individually by deck name or by a range of deck names, as shown in figure 3-18.

```
A.  Purge decks listed

    *PURDECK deck1,deck2,...deckn

    deck          Name of deck to be purged.
                  Names can appear in any order.

B.  Purge range of decks

    *PURDECK deck1.deck2

    deck1.deck2   Names of first and last decks,
                  inclusive, to be purged. Names
                  must appear in the relative
                  order in which decks exist in
                  the deck list.
```

Figure 3-18. PURDECK Directive Format

The name of a purged deck is removed from the deck list; it can be reused as a deck name. An entry for the purged deck remains in the directory, however, until removed through the use of the E parameter on the UPDATE control statement. The deck name can also be removed from the directory by resequencing the library, that is, by creating a source file in one Update run and then using the source file as input on a subsequent creation run. Until a deck name is removed from the directory, it cannot be used as a correction set identifier. (See the PURGE directive.)

PURGE DIRECTIVE

The PURGE directive permanently removes a correction set or group of correction sets from the program library. Every line in the correction set is purged, regardless of its status as active or inactive. Purging, unlike yanking, cannot be rescinded. A new program library written during the same run treats the purged correction set as if it had never existed. A PURGE directive can appear anywhere in the input stream; it terminates the current correction set. PURGE directive format, as shown in figure 3-19, depends on whether correction sets to be purged are specified individually by correction set name, by a range of correction set names, or by relative time of introduction into the program library.

```
A.  Purge listed correction sets

    *PURGE ident1,ident2,...,identn

    ident            Identifier of a correction
                     set to be purged.  Identi-
                     fiers can appear in any
                     order.

B.  Purge range of correction sets

    *PURGE ident1.ident2

    ident1.ident2    Identifiers of first and last
                     correction sets, inclusive,
                     to be purged.  Identifiers
                     must appear in the relative
                     order in which the correction
                     sets were introduced into the
                     program library; that is,
                     they must appear in the order
                     they exist in the directory.

C.  Purge later correction sets

    *PURGE ident,*

    ident            Identifier of correction set
                     to be purged along with all
                     correction sets introduced
                     after the specified correc-
                     tion set.

    *                Indicator that the program
                     library is to return to an
                     earlier level.  Intervening
                     PURGE directives and SEQUENCE
                     directives prevent complete
                     return.
```

Figure 3-19. PURGE DIRECTIVE Format

If Update cannot locate a specified correction set, it issues an error message. Purged identifiers can be reused on subsequent correction sets provided they do not appear in the YANK$$$ DECK as a YANK directive parameter.

RESTORE DIRECTIVE

The RESTORE directive reactivates a line image or a group of line images previously deactivated through a DELETE directive. Any text line images and compile file directives immediately following the RESTORE directive are inserted after the last line image identified on the directive. Any inserted line images receive line identifiers established by the correction set name of the preceding IDENT directive. RESTORE directive format depends on whether line images to be reactivated are specified by a line identifier or by a range of lines, as shown in figure 3-20.

```
A.  Restore specified line.

    *RESTORE line

    line     Line identifier of line to be re-
             stored.

B.  Restore range of lines.

    *RESTORE line1,line2

    line1,   Line identifiers of first and last
    line2    lines, inclusive, in sequence of
             lines to be restored.  Line1 must
             appear before line2 in the existing
             library.  Any lines in the sequence
             that are already active are not
             affected.
```

Figure 3-20. RESTORE Directive Format

Unless a TEXT directive has been encountered, Update terminates an insertion when it encounters the next insertion directive or a PURGE, PURDECK, IDENT, SELPURGE, ADDFILE, or SEQUENCE directive. On the other hand, compile file directives are inserted as if they are text after Update checks for correct syntax. Update interprets all other directives without terminating insertion; however, the directives are not inserted into the deck.

SELPURGE DIRECTIVE

The SELPURGE directive permanently removes the effects of the specified correction set on the specified deck. Only the line images belonging to the specified correction set are purged from the specified deck. Line images belonging to the specified correction set that are in other decks are not purged. Line images in the YANK$$$ deck can be purged through SELPURGE. A SELPURGE directive can appear anywhere in the input stream; it terminates the current correction set. SELPURGE directive format is shown in figure 3-21.

```
*SELPURGE deck1.ident1,...,deckn.identn

deck     Name of deck from which correction set
         is to be removed.

ident    Name of correction set to which cards to
         be removed belong.  It must be separated
         from the deck by a period.
```

Figure 3-21. SELPURGE Directive Format

SELYANK DIRECTIVE

The SELYANK directive temporarily removes the effects of the specified correction set on the specified deck. Only the line images belonging to the specified correction set are yanked from the specified deck. Line images belonging to the specified correction set that are in other decks are not yanked. Line images in the YANK$$$ deck can be yanked through SELYANK. A SELYANK directive must be part of a correction set; it is placed in the YANK$$$ deck. The SELYANK directive format is shown in figure 3-22.

```
*SELYANK deck1.ident1,...,deckn.identn

deck     Name of deck from which correction set
         is to be removed.

ident    Name of correction set to which lines to
         be removed belong. It must be separated
         from deck by a period.
```

Figure 3-22. SELYANK Directive Format

SEQUENCE DIRECTIVE

The SEQUENCE directive resequences active lines and purges inactive lines from the specified deck(s). Only those decks explicitly mentioned on the SEQUENCE directive are resequenced. Thus, if a correction set (for example, SET1) affects more than one deck on a program library (for example, DECK1 and DECK2), and only DECK1 has been subsequently resequenced through SEQUENCE, the SEQUENCE directive does not affect SET1 lines within DECK2. The YANK$$$ deck cannot be resequenced. SEQUENCE directive format, as shown in figure 3-23, depends on whether decks to be resequenced are specified individually by name or are specified as a range of deck names.

```
A.  Resequence listed decks.

    *SEQUENCE deck1,deck2,...,deckn

    deck           Name of deck to be resequenced.

B.  Resequence range of decks.

    *SEQUENCE deck1.deck2

    deck1.deck2    Name of first and last decks,
                   inclusive, to be resequenced.
                   Deck1 must appear before deck2
                   in old program library.
```

Figure 3-23. SEQUENCE Directive Format

Update normally allows deck and correction sets having the same name to coexist on the old program library. If a deck having the same name as a correction set is resequenced and lines for the correction set are in other decks, Update purges any modifications made by that correction set outside the resequenced deck to prevent duplicate identifiers.

The SEQUENCE directive does not result in identifiers being deleted from the directory even if, as a result of resequencing, no references to an identifier are on the library. This situation arises when all the corrections of a correction set refer to a deck that is resequenced. Deletion of the identifier, in this case, requires an edit (E parameter) or PURGE in a subsequent Update run.

A deck cannot be renamed and resequenced in the same Update run. (To rename a deck, delete the first line of the deck and replace it with a new DECK directive containing the new name.)

YANK DIRECTIVE

The YANK directive temporarily removes a correction set or group of correction sets from the program library. Line images activated by the correction set are deactivated; line images deactivated by the correction set are reactivated. If a correction set has been yanked, it is ignored during compile file or source file generation. The effects of the YANK directive can be selectively nullified through the introduction of DO and DONT directives in the decks. Update places the YANK directive in the YANK$$$ deck. The YANK directive format, as shown in figure 3-24, depends on whether correction sets to be yanked are specified individually by correction set name or by a range of correction set names.

```
A.  Yank listed correction sets

    *YANK ident1,ident2,...,identn

    ident            Identifier of a correction
                     set to be yanked. Identi-
                     fiers can appear in any
                     order.

B.  Yank range of correction sets

    *YANK ident1.ident2

    ident1.ident2    Identifiers of first and last
                     correction sets, inclusive,
                     to be yanked. Identifiers
                     must appear in the relative
                     order in which the correction
                     sets were introduced into the
                     program library; that is,
                     they must appear in the order
                     they exist in the directory.
```

Figure 3-24. YANK Directive Format

The YANK directive differs from PURGE in several respects: YANK must be part of a correction set; YANK does not terminate the current correction set; and the effects of a YANK directive can be rescinded.

YANKDECK DIRECTIVE

The YANKDECK directive temporarily removes all lines within the decks specified. All lines are deactivated, even if they belong to a correction set. YANKDECK differs from PURDECK in several respects: YANKDECK must be part of a correction set; it does not terminate the current correction set; and its effects can be rescinded. The YANKDECK directive format is shown in figure 3-25.

```
*YANKDECK deck1,deck2,...,deckn

deck   Name of deck to be yanked.  Names can
       appear in any order.
```

Figure 3-25. YANKDECK Directive Format

The deck YANK$$$ cannot be deactivated as a whole. Individual YANK directives within this deck can be yanked by a YANK directive, however.

COMPILE FILE DIRECTIVES

Compile file directives provide control over the compile file. These directives are interpreted when the program library decks are being corrected and written onto the compile file. Calls for common decks result in the common deck being written on the compile file. Other directives allow control of file format. None of the compile file directives are written on the compile file.

The user can prepare the original deck with embedded compile file directives (except for DO or DONT) or the user can insert compile file directives into program library decks as a part of a correction set. Compile file directives are not processed when they are encountered in the input stream (except for COMPILE); they are simply considered as text lines to be inserted and sequenced accordingly after update checks for correct syntax. To be recognized while the compile file is being written, these directives must have the same master control character as defined when the library was created.

CALL DIRECTIVE

The CALL directive causes the active text of a common deck to be written onto the compile file. The directive itself is stored as part of a deck and can be referenced by its line identifier. CALL is effective only within a deck or common deck. Common decks can call other common decks, but a common deck must not either call itself or call a common deck that contains a call to the common deck. Neither the CALL directive nor the COMDECK directive which defined the deck is written to the compile file. The CALL directive format is shown in figure 3-26.

```
*CALL deck

deck   Name of an existing common deck to be
       written to the compile file.
```

Figure 3-26. CALL Directive Format

Common decks can also be called from secondary old program libraries. If COMDECK names are duplicated on any secondary old program libraries, Update uses the first COMDECK encountered according to the order of the secondary old program libraries as specified by the P parameter of the UPDATE control statement.

COMPILE DIRECTIVE

The COMPILE directive indicates which decks are written to the compile file. During normal mode, decks specified on COMPILE directives and corrected decks are written to the compile file. During quick mode, decks specified on COMPILE directives and any common decks called by the directives are written to the compile file. The directive is ignored during a full Update.

The directive also affects the contents of any new program library and source file as shown in table 2-2 in section 2. The COMPILE directive format, as shown in figure 3-27, depends on whether decks to be written are specified individually by name or are specified as a range of deck names.

```
A.  Compile listed decks

    *COMPILE deck1,deck2,...,deckn

    deck            Name of deck to be written to
                    the compile file, new program
                    library file, and source file.

B.  Compile range of decks

    *COMPILE deck1.deck2

    deck1.deck2     Names of first and last decks
                    in range, inclusive, to be
                    written to the compile file.
                    The name of deck1 must appear
                    before the name of deck2 in
                    the old program library deck
                    list.
```

Figure 3-27. COMPILE Directive Format

Decks are written to the compile file in the order that the decks exist on the old program library, unless the K option is selected on the UPDATE control statement. If the K option has been specified, the decks are written in the order they appear on the COMPILE directive.

When a deck is being introduced in the same run that contains a COMPILE directive for the deck, the DECK directive must appear before the COMPILE directive. Otherwise, COMPILE directives can be anywhere in the input stream. They do not affect the current correction set name.

CWEOR DIRECTIVE

The CWEOR directive writes an end of system-logical record (section for SCOPE 2) on the compile file if data has been written to the file since the start of UPDATE or since the last end of system-logical record was written. The CWEOR directive format is shown in figure 3-28.

```
*CWEOR level

level   Level of system-logical record.

        For SCOPE 2, the following:

           RT=W   0 thru 14   end-of-section
           RT=W   15          end-of-partition
           RT=S   0 thru 15   end-of-record
           RT=Z   0 thru 15   end-of-section
           BT=C   0 thru 15   end-of-section
```

Figure 3-28. CWEOR Directive Format

DO DIRECTIVE

The DO directive causes Update to rescind a yank of specified correction sets while writing text to the compile file. If a line was deactivated as a result of a YANK or SELYANK, the line is reactivated. Likewise, if a line was activated by a YANK or SELYANK, Update deactivates it. A DO remains in effect until a DONT directive is encountered. The DO directive can be placed anywhere in the library. If Update encounters a DO for an unyanked correction set, an informative message is issued and the DO is ignored. The DO directive format is shown in figure 3-29.

```
*DO ident1,ident2,...,identn

ident   Name of correction set for which yanking
        is to be rescinded or initiated.
```

Figure 3-29. DO Directive Format

DONT DIRECTIVE

The DONT directive terminates a DO directive. It can also be used to initiate a yank of an unyanked correction set. When Update encounters a DONT for a correction set that has not been yanked, it yanks the set until it encounters a DO directive for the set. If the correction set has already been yanked, Update issues an informative message and ignores the DONT. The DONT directive can be placed anywhere in the program library. The DONT directive format is shown in figure 3-30.

```
*DONT ident1,ident2,...,identn

ident   Name of correction set for which yanking
        is to be rescinded or initiated.
```

Figure 3-30. DONT Directive Format

ENDIF DIRECTIVE

The ENDIF directive indicates the end of conditional text. It is used with IF when the num parameter is omitted from the IF directive. ENDIF should not be used if num is specified on the IF directive. Since num takes precedence, the ENDIF directive is included in the count of active lines and is written on the compile file. The ENDIF directive format is shown in figure 3-31.

```
*ENDIF
```

Figure 3-31. ENDIF Directive Format

IF DIRECTIVE

The IF directive conditionally writes text on the compile file. When Update encounters an IF directive, the text following the directive is written or skipped depending on the condition. The IF directive format, as shown in figure 3-32, depends on whether the specified name is to be known or unknown for the text to be written on the compile file.

```
A.  Name must be known (on old program library).

    *IF type,name,num

B.  Name must be unknown (not on old program
    library).

    *IF -type,name,num

    type   Type of condition name.

           DECK     Name is deck name.  To be
                    known, it must be in the deck
                    list on the primary old pro-
                    gram library.

           IDENT    Name is correction set iden-
                    tifier.  To be known, it must
                    be in the directory on the
                    primary old program library.

           DEF      Name is defined through
                    DEFINE directive on the old
                    program library.

           When type is not preceded by a minus
           sign, the name must be known for text
           to be written.  When type is preceded
           by a minus sign, the name must not be
           known for text to be written.

    name   Deck name, correction set identifier,
           or defined name, according to type.

    num    Number of active line images to be
           skipped if condition is not met.
           Optional.
```

Figure 3-32. IF Directive Format

If the num parameter is omitted and the condition is not met, Update searches for an ENDIF directive and resumes processing of the deck at that point; if ENDIF is not found, then the remainder of the PL is skipped and the compile file stops at this point; no error message is written. When the condition is met, no lines are skipped.

When an IF directive is encountered on a secondary old program library, Update only searches the directory, deck list, and YANK$$$ deck on the primary old program library in trying to satisfy the conditional. The deck lists, directories, and YANK$$$ decks of the secondary old program libraries are not searched.

When both an IF directive is encountered as a result of a CALL and a matching ENDIF directive is found as the result of a second CALL, the range of the IF, ENDIF pair is unpredictable.

WEOR DIRECTIVE

The WEOR directive causes the termination of the current system-logical record on the compile file with the specified level. The WEOR directive format is shown in figure 3-33.

```
*WEOR level

level   Level of system-logical record.

        For SCOPE 2, the following:

           RT=W   0 thru 14   end-of-section
           RT=W   15          end-of-partition
           RT=S   0 thru 15   end-of-record
           RT=Z   0 thru 15   end-of-section
           BT=C   0 thru 17   end-of-section
```

Figure 3-33. WEOR Directive Format

WIDTH DIRECTIVE

The WIDTH directive overrides the default compile file line image width settings, as specified by D and/or 8 on the UPDATE control statement. WIDTH directives are ignored with compressed compile files. The format for the WIDTH directive is shown in figure 3-34.

```
*WIDTH linelen,idlen

linelen   Number of characters of line image
          text that is written.

idlen     Width of the identification field fol-
          lowing the line image.
```

Figure 3-34. WIDTH Directive Format

The sum of the length of linelen and idlen must be equal to or less than 256 characters. If idlen is set to 0 (zero), the identification field is suppressed. The format of the fields linelen and idlen are shown in figure 3-35. The sequence data (S) is positioned within the identifier name field (I) by the following procedure:

1. Blanking the field.

2. Putting in the identifier name, left-justified with truncation on the right as needed.

3. Placing the sequence number over the field, right-justified with truncation on the left as needed.

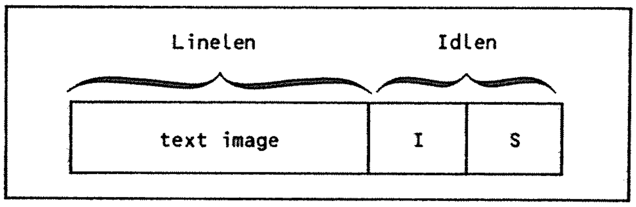


Figure 3-35. Fields of Line Image and Identification

If *WIDTH is specified with no parameters, the run default settings are restored. If only the length of the identification field is specified (*WIDTH ,idlen), then linelen is the previous setting used. If only linelen is specified (*WIDTH linelen), the previous setting of idlen is used.

FILE MANIPULATION DIRECTIVES

File manipulation directives control secondary input files during Update processing. These directives can only appear in the primary input stream. They are illegal on a secondary input file.

READ DIRECTIVE

The READ directive temporarily stops reading the primary input stream and begins reading an input stream from the specified file. READ differs from ADDFILE in that the content of the file specified by READ is not restricted except to prohibit the appearance of another READ directive or the ADDFILE, SKIP, and REWIND directives. Update reads from the specified file one system-logical record (section for SCOPE 2). Processing then continues with the main input stream. The READ directive format is shown in figure 3-36.

```
*READ lfn

lfn   Name of alternate file containing input
      stream.
```

Figure 3-36. READ Directive Format

The specified file cannot be one of the reserved files specified by a parameter on the UPDATE control statement. It can only be a local secondary input file. Also, the specified file must have the same character set as the primary input file.

REWIND DIRECTIVE

The REWIND directive repositions the specified file to beginning-of-information. The file to be rewound cannot be one of the reserved files. It can only be a secondary input file. The REWIND directive format is shown in figure 3-37.

```
*REWIND lfn

lfn   Name of file to be rewound.
```

Figure 3-37. REWIND Directive Format

SKIP DIRECTIVE

The SKIP directive repositions the named local file forward one or more system-logical records. A system-logical record (section for SCOPE 2) of level 17_8 or end-of-information terminates skipping. The SKIP directive format is shown in figure 3-38.

```
*SKIP lfn,n

lfn    Name of file to be positioned.

n      Number of logical records (sections for
       SCOPE 2) to be skipped in the forward
       direction. If n is omitted, Update skips
       one record (section).
```

Figure 3-38. SKIP Directive Format

INPUT STREAM CONTROL DIRECTIVES

The input stream control directives allow the user to specify whether or not Update is to recognize abbreviated directives, delimit text, or control which input stream lines are to be displayed on the listing file.

ABBREV DIRECTIVE

The ABBREV directive causes checking for abbreviated directives to be resumed. It is used in connection with the NOABBREV directive. The ABBREV directive format is shown in figure 3-39.

```
*ABBREV
```

Figure 3-39. ABBREV Directive Format

ENDTEXT DIRECTIVE

The ENDTEXT directive ends the condition established by a prior text directive. If ENDTEXT is encountered before TEXT, Update ignores it. The ENDTEXT directive format is shown in figure 3-40. Any information in columns 10 through 256 is taken as a comment.

```
*ENDTEXT
```

Figure 3-40. ENDTEXT Directive Format

LIST DIRECTIVE

The LIST directive causes listing of lines in the input stream to be resumed. It is used in connection with NOLIST. The LIST directive format is shown in figure 3-41.

```
*LIST
```

Figure 3-41. LIST Directive Format

NOABBREV DIRECTIVE

The NOABBREV directive causes Update to stop checking for the abbreviated forms of the directives. Update expands the name when it reads an abbreviated form so that it is a full name. The user has the option of not using abbreviations and of turning off the check through the NOABBREV feature. In this mode, an abbreviated directive is not recognized but is taken as text. The NOABBREV directive format is shown in figure 3-42.

```
*NOABBREV
```

Figure 3-42. NOABBREV Directive Format

NOLIST DIRECTIVE

The NOLIST directive disables list option 4. Update stops listing lines in the input stream when it encounters a NOLIST and resumes listing lines when it encounters a LIST. NOLIST directive format is shown in figure 3-43.

```
*NOLIST
```

Figure 3-43. NOLIST Directive Format

LIST and NOLIST can occur anywhere in the input stream. They do not terminate insertion or a correction set. The LIST/NOLIST directives are ignored if list option 0 is selected.

TEXT DIRECTIVE

The TEXT directive, used in connection with ENDTEXT, causes all following line images to be treated as text, whether or not they begin with the master control character and would otherwise be considered as directives. When Update encounters a TEXT directive, the TEXT directive line image and all line images following it, up to and including the ENDTEXT directive, are considered as text and are written on the program library. A TEXT directive in the input stream must be either in a deck or in text being inserted. The TEXT and ENDTEXT directives are maintained on the program library as text line images; however, they are not written on the compile file. The TEXT format is shown in figure 3-44. Any information in columns 10 through 256 is taken as a comment.

```
*TEXT
```

Figure 3-44. TEXT Directive Format

SPECIAL DIRECTIVES

The special directives provide extended features. With the exception of DEFINE and PULLMOD, they can appear any place in the input stream for creation or correction runs.

DECLARE DIRECTIVE

The DECLARE directive protects decks other than the declared deck from being inadvertently altered. Subsequent corrections are restricted to the named deck until Update encounters a DECLARE directive with no deck name or another DECLARE directive with a different deck name. This directive can only be used when the DECLKEY installation option has been assembled. The DECLARE directive format is shown in figure 3-45.

```
*DECLARE deck

deck   Name of deck to which following correc-
       tions are restricted. Omitting deck
       nullifies a previous DECLARE.
```

Figure 3-45. DECLARE Directive Format

When the DECLARE directive is encountered, the following restrictions go into effect:

- PURGE and YANK directives are illegal.
- INSERT, DELETE, RESTORE, and BEFORE directives can apply only to lines in the declared deck. if they do not, the operation is not performed and Update issues an informative message.
- Inserting or reactivating a DECK or COMDECK directive is illegal.

New decks inserted via the ADDFILE directive need not be named in a DECLARE directive.

DEFINE DIRECTIVE

The DEFINE directive establishes a condition to be tested by the IF directive. The names on a DEFINE directive are unrelated to correction set identifiers or deck names. Update places DEFINE directives in the YANK$$$ deck. A DEFINE directive can be placed anywhere in a correction set. The DEFINE directive format is shown in figure 3-46.

```
*DEFINE name1,name2,...,namen

name   Name for subsequent testing by IF
       directive.
```

Figure 3-46. DEFINE Directive Format

END DIRECTIVE

The END directive is ignored in the input stream. Update does not copy it onto the old program library.

The END directive provides compatibility with the SCOPE EDITSYM program. The END directive format is shown in figure 3-47.

```
*END
```

Figure 3-47. END Directive Format

LIMIT DIRECTIVE

The LIMIT directive changes the maximum size for the listable output file from the default value of 6000 lines to the specified number of lines. It should be one of the first lines encountered in the input stream. The LIMIT directive will not appear in the new program library. The LIMIT directive format is shown in figure 3-48.

```
*LIMIT n

n   New line limit for listable output.
```

Figure 3-48. LIMIT Directive Format

When the specified limit is reached, options 3 (line image, deck name, and modification key) and 4 (input stream) are turned off. Errors and directives are still listed, however, if options 1 and 2 were selected. Options 5 through 9 are not affected. Refer to L parameter in section 4.

PULLMOD DIRECTIVE

The PULLMOD directive causes the program library to be searched for all line images belonging to each specified correction set and reconstructs a set of directives and text. The reconstructed correction set produces the same results as the original set. The search of the library is performed at the end of the Update run. Therefore, any modifications made by the current run are reflected in the PULLMOD results. Each reconstructed correction set is written to the file specified by the G parameter on the UPDATE control statement. All of the sets are contained within one system-logical record (section for SCOPE 2) on the file. The PULLMOD directive format is shown in figure 3-49. The PULLMOD directive can be used only when the PMODKEY installation option has been assembled for Update.

```
*PULLMOD ident1,ident2,...,identn

ident   Name of correction set to be re-created.
```

Figure 3-49. PULLMOD Directive Format

The user is responsible for determining whether or not the reconstructed correction sets accurately reflect the original corrections. PULLMOD is unable to determine if line images have been purged subsequent to the addition of the correction sets requested.

A pullmod file has the same format as an input file. This feature permits a user to take an earlier version of the library and apply selected correction sets.

/ COMMENT DIRECTIVE

The / directive introduces a comment into the listable output file. Update ignores this line except to copy it to the listing file. A comment can appear at any place in the input stream. The slash can be redefined as another character through the / comment directive format as shown in figure 3-50. The slash must appear in column 2. Column 3 must be a comma or blank.

```
*/comment
```

Figure 3-50. Comment Directive Format

UPDATE CONTROL STATEMENT 4

The Update utility is called by the UPDATE control statement. Parameters specify options and files for the run. The format of the call is shown in figure 4-1. The word UPDATE must begin in column one. See the operating system reference manual for additional control statement syntax requirements.

```
UPDATE(p-list)

p-list   Parameters specifying options. Param-
         eters in the list are separated by
         commas. A left parenthesis or a comma
         must separate the list from the word
         UPDATE. A right parenthesis or a
         period terminates the statement.
```

Figure 4-1. UPDATE Control Statement Format

PARAMETERS

All update parameters are optional and can appear in any order. The parameters that specify files (C, G, I, K, M, N, O, P, S, T) optionally can be followed with either the digit 6 or 8, indicating 6-bit display code or 8-bit ASCII.

When using the C, G, K, O, P, S, or T parameters, the digit 6 forces the character set of the file to be 6-bit display code. The digit 8 forces the character set to be 8-bit ASCII. For example, C8=FILE specifies that the decks are to be written to the compile file named FILE using the ASCII character set. The 6 and the 8 cannot both be specified at once for the same file. These parameters each have a default of either display code or ASCII, which can be overridden by using either the 6 or 8 digit (6 overrides an ASCII default and 8 overrides a display code default).

When using the I parameter without the 6 or 8 digit, Update will determine the correct character set to use. For the N or N8 parameter, Update only uses ASCII if the old program library or input file has ASCII data. The N8 parameter does not force ASCII automatically. When using the M and N parameters, the character set is determined from the library's internal header.

By using the Update parameters, it is possible to convert a file of display code data to ASCII, or visa versa. This capability can be useful when your operating system does not have a standard utility to change the character set (such as SCOPE). The file to be converted must be in a legal input file format. An example of how a display code to ASCII conversion can be done is shown in the Update Control Statement Examples subsection, which appears at the end of this section.

The Update parameters are summarized in table 4-1 and are described in detail below.

A SEQUENTIAL-TO-RANDOM COPY

This parameter copies a sequential old program library to a random new program library. No other Update operations are performed; any I parameter is ignored. The only other control statement parameters that can be used with the A parameter are those specifying files, L=0, R, *, /, and H. An error results if the old program library is not sequential or the new program library is not random. For SCOPE 2, the new program library cannot be blocked.

- omitted

  No copy is made.

- A

  The sequential old program library is copied to a random new program library.

B RANDOM-TO-SEQUENTIAL COPY

This parameter copies a random old program library to a sequential new program library. No other Update operations are performed; any I parameter is ignored. The only other control statement parameters that can be used with the B parameter are those specifying files, L=0, R, *, and /. An error results if the old program library is not in random format.

- omitted

  No copy is made.

- B

  The random old program library is copied to a sequential new program library.

C COMPILE FILE NAME

This parameter specifies the name of the compile file. The content of the compile file is determined by the Update mode as shown in table 2-2 in section 2. The default character set is display code.

- omitted or C or C6 or C8

  Decks are written to the file named COMPILE.

- C=lfn or C6=lfn or C8=lfn

  Decks are written to file named lfn.

- C=PUNCH

  Decks are written to file named PUNCH. The D and 8 parameters are implied.

- C=0

  Compile file suppressed.

The C parameter is ignored if K is also specified.

D DATA WIDTH ON COMPILE FILE

This parameter specifies how many columns are to be used for data on the COMPILE file. Data width does not include sequencing information.

- omitted

  72 columns of data to be used.

- D

  80 columns of data to be used.

TABLE 4-1 SUMMARY OF UPDATE CONTROL STATEMENT PARAMETERS

Parameter[†]	Function
A	Copies a sequential old program library to a new random program library.
B	Copies a random old program library to a new sequential old program library.
C	Specifies the name of the compile file.
D	Defines the compile file line image width, excluding Update sequence information.
E	Removes from the directory previously purged identifiers and purge identifiers that exist simply as directory entries.
F	Selects full Update mode.
G	Specifies the name of the pullmod file.
H	Overrides the old program library character set.
I	Specifies the name of the primary input files.
K	Writes decks on compile file in order specified on COMPILE directives.
L	Selects listable output file contents.
M	Merges specified program library with an old program library.
N	Specifies the name of the new program library file.
O	Specifies the name of the listable output file; content is determined by L parameter.
P	Specifies the names of the old program library and secondary old program libraries.
Q	Selects quick update mode.
R	Specifies the particular files to rewind.
S	Specifies the name of the source file; content includes common decks and is determined by mode.
T	Same as S, but omits common decks.
U	Does not terminate execution if fatal error occurs.
W	Specifies the sequential new program library file.
X	Specifies the compressed format for the compile file.
8	Defines the compile file line image width including Update sequence information.
*	Redefines the master control character for directives.
/	Redefines the control character for comments.

[†] Parameters C, G, I, K, M, N, O, P, S, and T can be appended with either 6 (for display code) or 8 (for ASCII).

If specified, the WIDTH directive overrides the D parameter.

E EDIT OLD PROGRAM LIBRARY

This parameter specifies that the old program library is to be edited. During editing, the directory and deck list are rearranged to reflect the actual order of decks on the program library; all previously purged identifiers are removed. Identifiers that exist simply as entries in the directory and have no lines associated with them are purged. Any lines other than YANK, SELYANK, YANKDECK, or DEFINE that exist in the YANK$$$ deck are also purged.

Two edit runs are required to edit the library completely. The first edit run removes purged identifiers and flags unused identifiers as purged. The second edit run deletes the unused identifiers from the directory.

- omitted

  No editing is done.

- E

  The program library is edited.

The E parameter can only be used when the EDITKEY installation option has been assembled for Update.

F FULL UPDATE MODE

This parameter specifies full Update mode.

- omitted

  Normal selective Update mode, as long as Q is not specified.

- F

  Full Update mode.

G PULLMOD FILE NAME

This parameter specifies the name of the pullmod file. The default character set is display code. The G parameter can only be used when the PMODKEY installation option has been assembled for Update.

- omitted

  Output from PULLMOD directives is appended to the source file (S parameter).

- G=lfn or G6=lfn or G8=lfn

  Output from PULLMOD directives is written on file named lfn. The listable output file (O parameter) cannot be specified.

H CHARACTER SET CHANGE

This parameter allows the user to override the character set type specification in the old program library.

- omitted or H

  Update treats the old program library character set as the character set indicated in the old program library.

- H=3

  Update treats the old program library as a 63-character set program library regardless of the character set specified in the old program library.

- H=4

  Update treats the old program library as a 64-character set program library regardless of the character set specified in the old program library.

I INPUT STREAM FILE NAME

This parameter specifies the name of the primary input file. If the digit 6 or 8 is not specified (I or I=lfn), Update determines the input file character set by examining the first line image. Direct input from terminals, permitted only on NOS and NOS/BE, defaults to the display code character set unless I8 is specified on the UPDATE control statement. All auxiliary input files must be in the same character set as the primary input file. Input lines are read and stored up to 256 characters in length. No special parameter is necessary to use long lines. Lines exceeding 256 characters are truncated and an informative message is issued.

- omitted or I or I6 or I8

  Directives and text are on the file named INPUT.

- I=lfn or I6=lfn or I8=lfn

  Directives and text are on file named lfn.

K COMPILE FILE SEQUENCE

This parameter specifies that decks are to be written to the compile file in the order in which the deck names are encountered on COMPILE directives. If a deck name is mentioned more than once, its last specification determines the deck´s place within the compile file. The default character set is display code (K6). This parameter takes precedence over the C parameter. The K parameter is ignored if both the K parameter and the F parameter are specified.

- omitted

  Location determined by C parameter.

- K or K6 or K8

  Decks to be written to the file named COMPILE in COMPILE directive sequence.

- K=lfn or K6=lfn or K8=lfn

  Compile output decks to be written on file named lfn in COMPILE directive sequence.

L LISTABLE OUTPUT OPTIONS

This parameter specifies the content of the output file.

- omitted

  For a creation run, selects options A, 1, and 2.

  For a correction run, selects options A, 1, 2, 3, and 4.

  For a copy run, selects options A and 1.

  For an output file connected to a terminal, selects option 1.

- L=c...c

  Each character in string c...c selects one of the following options. Under NOS, up to seven options can be specified. The character 0 overrides any other options specified and suppresses the entire listing.

  A List known deck names and correction set identifiers (deck names and correction set identifiers must be on the primary old program library to be known), COMDECK directives that were processed, known definitions (DEFINE directive), and decks written to the compile file.

  F All options except 0.

  0 All listing is suppressed.

  1 List lines in error and the associated error messages. The flag *ERROR* appears to the left and right of an erroneous line image.

  2 List all active Update directives encountered either on the input file or on the old program library. Those directives encountered in input are flagged with five asterisks to the left unless the directive is abbreviated or the line identifier is in short form. In this case, the directive is flagged with five slashes. If the directive has been encountered on the old program library, the name of the deck to which this line belongs is printed in place of the five asterisks or slashes.

  3 Comment on each line that changed status during current run. Comments include the deck name, line image, line identifier, and an indicator of action taken for that line.

  I Line added.

  A Inactive line reactivated.

  D Active line deactivated.

  P Line purged. If the line was active, ACTIVE also appears.

  SEQ Line resequenced.

  4 List text lines encountered in the input stream. Lines read as a result of a READ directive are identified to the right with the file name. Lines inserted as a result of an ADDFILE directive are listed only when option 4 is explicitly selected. Lines inserted as a result of a COPY directive are identified to the right by the word copy.

  Option 4 can be turned on by a LIST directive and off by a NOLIST directive.

  5 List all active compile file directives.

  6 List number of active and inactive lines by deck name and correction set identifier.

  7 List all active lines; identify to the right with an A.

  8 List all inactive lines; identify to the right with an I.

  9 List correction history of all lines selected by list options 5, 7, and 8.

List options 5 through 9 are provided for auditing an old program library. These options are available only when the AUDITKEY installation option is assembled. Output is written to a temporary file and appended to the listable output file at the end of the Update run. When the F parameter is selected, options 5 through 9 apply to all decks on the old program library. If F is not selected, options 5 through 9 apply to decks listed on COMPILE directives only.

List options 3, 5, 6, 7, 8, and 9 do not apply to creation runs and are ignored if specified. However, list option 4 may be used to list creation run input.

If the A or B parameter is specified, the only list option honored is L=0.

If the old program library is sequential and F is not selected, called common decks that precede the decks that call them must be explicitly named on COMPILE directives to be audited. A common deck is audited automatically if it follows the deck that calls it. If the old program library is random, called common decks are audited automatically.

M MERGE PROGRAM LIBRARIES

This parameter merges two program libraries as one new program library. The M parameter is ignored on a creation run. The two program libraries must have the same master control character. The default character set is determined from the header.

- omitted

  No merge file.

- M or M6 or M8

  Program library to be merged with the old program library is on file MERGE.

- M=lfn

  Program library to be merged with old program library on file named lfn.

N NEW PROGRAM LIBRARY FILE NAME

This parameter specifies the name of the new program library. The default character set is ASCII if the P, M, or I file uses ASCII.

- omitted

  Suppress new program library generation if correction run, otherwise write new program library to file named NEWPL.

- N or N6 or N8

  Write new program library to file named NEWPL.

- N=lfn or N6=lfn or N8=lfn

  Write new program library to file named lfn.

O LISTABLE OUTPUT FILE NAME

This parameter specifies the name of the output file. Output file content is determined by the L parameter. The default character set is display code.

- omitted or O or O6 or O8

  Write output to file named OUTPUT.

- O=lfn or O6=lfn or O8=lfn

  Write output to file named lfn.

P OLD PROGRAM LIBRARY FILE NAME

This parameter specifies the name of the old program library; it is ignored on a creation run. The default character set is determined from the header.

- omitted or P or P6 or P8

  Old program library resides on file named OLDPL.

- P=lfn

  Old program library resides on file named lfn.

- P=lfn/sl/s2/.../s7

  Old program library resides on file named lfn. Secondary old program libraries reside on files sl, s2,...,s7.

- P=/sl/s2/.../s7

  Old program library resides on file OLDPL. Secondary old program libraries reside on files sl, s2,...,s7.

Q QUICK UPDATE MODE

This parameter specifies quick Update mode. It takes precedence when both F and Q are specified.

- omitted

  When F is also omitted, normal selective Update mode.

- Q

  Quick mode.

Corrections other than ADDFILE that reference lines in decks not specified on COMPILE directives are not processed in quick mode and Update abnormally terminates after printing the unprocessed corrections.

In Q mode, using a random old program library, a single correction set containing corrections to both a DECK and a COMDECK might cause trouble if the COMDECK logically precedes the DECK on the old program library. No errors will be detected, but if the same run is repeated with the N parameter specified on the UPDATE control statement and/or the old program library is sequential, the sequence numbers assigned to the text lines in the correction set will not be the same as they were in the Q mode run. This situation cannot be prevented without sacrificing the speed for which Q mode was designed. The correct sequence numbers are those assigned when N is specified or the old program library is sequential.

R REWIND FILES

This parameter specifies files to be rewound before and after an Update run.

- omitted

  Rewind the old program library, the new program library, the compile file, the source file, and the pullmod file.

- R

  Do not rewind any files. The new program library and the old program library are positioned before the end-of-file mark.

- R=c...c

  Each character in the string indicates a file to be rewound. The characters also apply to corresponding two-character control statement options.

  C Compile

  N New program library

  P Old program library and merge library

  S Source and pullmod

S SOURCE FILE NAME

This parameter specifies the name of the source file. The content of the source file is determined by the mode in which Update is operating, by the decks named on COMPILE directives, and by the format of the old program library in use (random or sequential).

- omitted

  Suppress source output file unless it is selected by the T parameter.

- S or S6 or S8

  Source output file to be written on file named SOURCE.

- S=lfn or S6=lfn or S8=lfn

  Source output file to be written on file named lfn.

T OMIT COMMON DECKS FROM SOURCE FILE

This parameter specifies that common decks are to be excluded from the source file. It takes precedence over the S parameter.

- omitted

  Suppress source file unless it is selected by the S parameter.

- T or T6 or T8

  Source output to be written on file named SOURCE, with common decks excluded.

- T=lfn or T6=lfn or T8=lfn

  Source output to be written on file named lfn, with common decks excluded.

U DEBUG HELP

The U parameter allows Update to proceed to pass 2 (correction phase) if errors are encountered in pass 1 (read-input-stream phase). The user should be aware that because of the method in which Update works, pass 1 errors could conceivably cause the flagging of pass 2 items which are not errors.

- omitted

  Update execution terminates when a fatal error is encountered.

- U

  Update execution is not terminated by a fatal error.

W SEQUENTIAL NEW PROGRAM LIBRARY FORMAT

This parameter specifies that the new program library is to have sequential format.

- omitted

  New program library format is determined by file residence as shown in table 2-3 in section 2.

- W

  New program library is a sequential file.

X COMPRESSED COMPILE FILE

This parameter specifies that the compile file is to be compressed.

- omitted

  Compile file is not written in compressed format.

- X

  Compile file is written in compressed format (appendix D).

8 LINE IMAGE WIDTH ON COMPILE FILE

This parameter specifies total line image width on the compile file including sequencing information (appendix D).

- omitted

  Compile file output is composed of 90-column line images.

- 8

  Compile file output is composed of 80-column line images.

If specified, the WIDTH directive overrides the 8 parameter.

* MASTER CONTROL CHARACTER

This parameter specifies the master control character. If the character specified for a correction run is not the same as the character used when the old program library was created, the old program library character is used.

- omitted

  The first character of each directive is *.

- *=c

  The first character of each directive for this Update run is c; c can be any character A through Z, 0 through 9, or + - * / $ or =. (The $ character should be specified as *=$$$$.)

/ COMMENT CONTROL CHARACTER

This parameter specifies the comment control character.

- omitted

  Comment control character is /.

- /=c

  The comment control character is c; c can be any character A through Z, 0 through 9, or + - * / $ or =. (The $ character should be specified as /=$$$$.) Note, however, that the character should not be changed to one of the abbreviated forms of a directive unless NOABBREV is in effect.

UPDATE CONTROL STATEMENT EXAMPLES

The Update control statement

```
UPDATE(C=0,I=IN,L=F,N=TEST2,P=TEST1,S,*=+)
```

selects the following options in addition to default values for the omitted parameters:

- C=0

  A compile file is not generated.

- I=IN

  The input stream is on the file named IN.

- L=F

  A full output listing is generated.

- N=TEST2

  A new program library named TEST2 is generated.

- P=TEST1

  The old program library is on the file named TEST1.

- S

  A source file is generated on file named SOURCE.

- *=+

  The master control character is +.

The Update control statement

```
UPDATE(P=OLDPL8,S8,I,O,N6=NUPL6)
```

selects the following values:

- P=OLDPL8

  Modify the program library named OLDPL8; the program library is assumed to be in ASCII.

- S8

  Generates an ASCII source file named SOURCE.

- I

  The input is in ASCII or display code on the file named INPUT. Update automatically determines the character set of the input file.

- O

  The output is in display code (the default) on the file named OUTPUT.

- N6=NUPL6

  Causes Update to generate a new program library in display code from the old program library in ASCII code on the file named NUPL6.

The Update control statement

```
UPDATE(C=0,I=0,N8=NUPL8,S)
```

selects the following options:

- N8=NUPL8

  An 8-bit (ASCII) NEWPL is generated if an ASCII old program library is input.

- C=0

  A compile file is not generated.

- I=0

  The 0 is an empty file; no corrections are applied.

- S

  A source file in display code named SOURCE is generated.

The Update control statement

```
UPDATE(A,N=RAN,P=SEQ)
```

causes Update to copy the sequential old program library, SEQ, to a random new program library named RAN. The L, O, R, *, and / parameters assume their default values. No other parameters are applicable when A is specified.

The Update control statement

```
UPDATE
```

selects the following default values:

- C=COMPILE

- G=SOURCE (correction run)

- I=INPUT
- L=A12 (creation run)
  L=A1234 (correction run)
  L=A1 (copy run)
- N=NEWPL (creation run)
- O=OUTPUT
- R=CNPS
- P=OLDPL (correction run)
- *=*
- /=/

In addition, the following defaults apply:

- The compile file has 90 columns with 72 columns for data.
- No editing is performed.
- Update mode is normal selective.
- The character set used is that specified in the library header. However, if there is ASCII data in INPUT, ASCII will be used in creating NEWPL.
- No merging is performed.
- Execution is terminated if a fatal error occurs.
- New program library file format is determined by file residence.
- The compile file is not in compressed format.

The Update control statement

```
UPDATE(C8=C812,F,I=F64,L=1,N=0,O=OUT,S8=S812,U)
```

causes Update to convert display code data in file F64 to ASCII data in file C812. The statement selects the following options:

- C8=C812

  A compile file is generated in ASCII.

- F

  Full Update mode is used.

- I=F64

  The input stream is on the file named F64. A *DECK directive must be the first line in the file.

- L=1

  The output file will contain lines in error and the associated error messages.

- N=0

  A new program library is not generated.

- O=OUT

  The output is written to the file named OUT.

- S8=S812

  An ASCII source file is generated on file named S812.

- U

  The Update execution will not be terminated by a fatal error.

EXAMPLES OF UPDATE RUNS 5

This section contains several examples of Update runs under NOS. The directives illustrated include PURGE, YANK, ADDFILE, and PULLMOD. Examples also show how to save or store a program library as a permanent file under the various operating systems. Also included in this section is an example of a FORTRAN program maintained as a program library.

LIBRARY FILE CREATION

Figure 5-1 shows an example of an Update creation run in which several COMPASS and FORTRAN routines become a program library. The UPDATE control statement indicates a new library is to be created with the name PL. Since no other parameters are specified, Update uses default values.

```
job statement
.
.
.
UPDATE(N=PL)
.
.
.
/EOR
*DECK COMGROUP
  COMPASS program
*DECK COMGROUP1
  COMPASS program
*WEOR
*DECK FORGROUP
  FORTRAN program
*DECK FORGROUP1
  FORTRAN program
/EOI
```

Figure 5-1. Update Creation Run

Since the first directive encountered is DECK, Update recognizes a creation run and begins construction of a new program library. All lines following the first DECK directive, up until the second DECK directive, are written as a deck with the name COMGROUP. The first line is assigned the identifier COMGROUP.2, the next COMGROUP.3, and so forth. (The DECK directive itself is also a part of the library and has the identifier COMGROUP.1.)

A new deck, with line identifiers in the form COMGROUP1.n, begins when Update encounters the second DECK directive. In this example (figure 5-1), two COMPASS programs form the first two decks; COMGROUP and COMGROUP1; and two FORTRAN programs make up the last two decks (FORGROUP and FORGROUP1). At the end of the Update run, a program library exists with four decks.

The compile file produced by the run in figure 5-1 contains two system-logical records as a result of the WEOR directive. All four decks are written to the compile file. It has the default name of COMPILE.

The example in figure 5-2 shows a creation run in which directives are read from the alternate input file REMTAPE. Update reads text and directives from REMTAPE until the end of the system-logical record (end-of-section for SCOPE 2) is encountered. Update then resumes reading from the main input file, INPUT. The resulting new program library contains decks A, B, C, and LOCAL.

```
A.  Update Job Deck.

    job statement
    .
    .
    .
    UPDATE(N)
    .
    .
    .
    /EOR
    *READ REMTAPE
    *DECK LOCAL
      text of LOCAL
    /EOI

B.  Contents of REMTAPE

    *DECK A
      text of A
    *DECK B
      text of B
    *DECK C
      text of C
```

Figure 5-2. Creation of Library
From Alternate Input File

The program library, NEWPL, created by the example in figure 5-3 contains four decks; two of them are common decks. The compile file that is produced by default contains decks XA and XB in that order. Deck XB is expanded by Update to contain common deck D2 on the compile file.

ALTERNATIVE INPUT FILES

Text and directives do not have to be part of the job deck. They can be in a file specified by the I parameter of the UPDATE control statement. In figure 5-4, Update creates a program library from information contained in file A1. The library that is produced contains three decks having lines identified by their deck name and sequence number as shown in figure 5-5.

```
job statement
.
.
.
UPDATE(N)
.
.
.
/EOR
*COMDECK D1
  text of D1
*COMDECK D2
  text of D2
*DECK XA
  text of XA
*DECK XB
  text of XB
*CALL D2
/EOI
```

Figure 5-3. Creation of Library With Common Decks

```
A.  Update Job Deck

    job statement
    .
    .
    .
    UPDATE(I=A1,N)
    .
    .
    .
    /EOI

B.  Contents of A1

    *COMDECK CSET
          COMMON A,B,C
    *DECK SET1
          PROGRAM ZIP
    C     A DO-NOTHING JOB
          STOP
          END
    *DECK SET2
          SUBROUTINE JIM
          A = B - SIN(C)
          RETURN
          END
```

Figure 5-4. Input File Not INPUT

```
*COMDECK CSET                               CSET.1
      COMMON A,B,C                          CSET.2
*DECK SET1                                  SET1.1
      PROGRAM ZIP                           SET1.2
C     A DO-NOTHING JOB                      SET1.3
      STOP                                  SET1.4
      END                                   SET1.5
*DECK SET2                                  SET2.1
      SUBROUTINE JIM                        SET2.2
      A = B - SIN(C)                        SET2.3
      RETURN                                SET2.4
      END                                   SET2.5
```

Figure 5-5. Program Library Contents

INSERTING, DELETING, AND COPYING

The Update run illustrated in figure 5-6 modifies the decks SET1 and SET2 of the program library created by the run in figure 5-4. As a result of the correction run, SET1 appears in the compile file as shown in figure 5-7.

```
job statement
.
.
.
UPDATE(N,F)
.
.
.
/EOR
*IDENT ADD1
*DELETE SET1.3,SET2.5
*CALL CSET
      B=1.0
      C=3.14159
      CALL JIM
*COPY SET1,SET1.4,SET1.5
*COPY SET2,SET2.2
*CALL CSET
*COPY SET2,SET2.3,SET2.5
/EOI
```

Figure 5-6. Modify Old Program Library

```
PROGRAM ZIP                                   SET1  2
COMMON A,B,C                                  CSET  2
B=1.0                                         ADD1  2
C=3.14159                                     ADD1  3
CALL JIM                                      ADD1  4
STOP                                          ADD1  5
END                                           ADD1  6
SUBROUTINE JIM                                ADD1  7
COMMON A,B,C                                  CSET  2
A = B - SIN(C)                                ADD1  9
RETURN                                        ADD1 10
END                                           ADD1 11
```

Figure 5-7. Compile File Contents

Figure 5-8 shows the modification of an old program library named FN and the production of an assembly listing. The compile file that is read by COMPASS contains deck XA after that deck was modified by Update.

PURGING AND YANKING

The purge directives differ from the yank directives in that yank operations are temporary; lines yanked from the program library are temporarily deactivated. The lines can be reactivated by a subsequent yank of the yank directive that deactivated the line images.

In contrast, any change made to a program library through a purge directive is permanent. A reversal of a purge operation is possible only through the reintroduction of the lines into the library as if they had not previously existed.

```
job statement
.
.
.
UPDATE(P=FN)
.
.
.
COMPASS(I=COMPILE)
.
.
.
/EOR
*IDENT CS1
*INSERT XA.1
  Insertions
*DELETE XA.20,XA.23
/EOI
```

Figure 5-8. Correction Run

The YANK directive in figure 5-9 becomes the first line on the new program library. The identifier for this line is NEGATE.1. The effects of the YANK can be nullified in future runs (consequently the effects of the correction set GOTTOGO are restored) by specifying the following:

```
    *IDENT RESTORE
    *DELETE NEGATE.1
or
    *IDENT RESTORE
    *YANK NEGATE
or
    *PURGE NEGATE
```

```
job statement
.
.
.
UPDATE(P=LIB,N=NEWLIB)
.
.
.
/EOR
*IDENT NEGATE
*YANK GOTTOGO
/EOI
```

Figure 5-9. Use of YANK

If the correction set NEGATE contained other corrections as well as the YANK, the YANK could be permanently removed by specifying the following:

```
    *SELPURGE YANK$$$.NEGATE
```

or it could be temporarily removed by specifying:

```
    *SELYANK YANK$$$.NEGATE
```

```
job statement
.
.
.
UPDATE(N=LIBMAY,P=LIBAUG,C=0)
.
.
.
/EOR
*PURGE JUNMOD1,*
/EOI
```

Figure 5-10. Return to Previous Level

The Update run in figure 5-10 returns a program library to a previous level. The program library LIBAUG was modified periodically over a number of months. LIBAUG is the most recent (August) version of the program library. This run re-creates a library modified only through May. The run purges all modifications made after May (beginning with JUNMOD1 in the directory).

The run in figure 5-11 permanently removes deck BAD from the library. LIB is the most recent program library. NEWBAD is the new program library with BAD purged. *PURDECK BAD operates so that any lines having the identifier BAD but physically located outside of the deck BAD are not purged.

```
job statement
.
.
.
UPDATE(P=LIB,N=NEWBAD,C=0)
.
.
.
/EOR
*PURDECK BAD
/EOI
```

Figure 5-11. Use of PURDECK

As a means of comparing the effects of YANK, SELYANK, and YANKDECK, consider the following:

- *YANK OLDMOD

  This directive causes all effects of the correction set OLDMOD on the entire library to be nullified. Line images introduced by OLDMOD are deactivated; line images deactivated by OLDMOD are reactivated.

- *SELYANK OLDDECK.OLDMOD

  This directive accomplishes the same effect as the *YANK OLDMOD directive except its effect is limited to line images within the deck OLDDECK.

- *YANKDECK OLDDECK

  This directive affects all line images in OLDDECK, without regard to which correction set they belong.

The effects of the purge directives PURGE, SELPURGE, and PURDECK work the same as the yank directives except the results are permanent.

SELECTIVE YANKING

The text stream in figure 5-12 illustrates the use of the DO and DONT directives. The deck ZOTS had contained lines introduced by the correction set DART; a later correction set contained a YANK directive that yanked correction set DART. The user wishes to nullify a portion of the YANK directive that affects the lines following ZOTS.19 through ZOTS.244; all other lines belonging to the correction set DART are to remain yanked. Inserting a DO at ZOTS.19 and a DONT at ZOTS.244 causes Update to rescind the YANK directive while writing the deck ZOTS to the compile file.

```
*IDENT          REST
*INSERT         ZOTS.19
*DO             DART
*INSERT         ZOTS.244
*DONT           DART
```

Figure 5-12. Use of DO and DONT

SELECTIVE WRITING TO COMPILE FILE

During the correction phase Update processes the following directive:

```
    *DEFINE ABC
```

It is automatically placed in the YANK$$$ deck (*INSERT is not needed). PROG2, a deck to be written on the compile file, contains the sequence shown in figure 5-13.

```
*DECK PROG2
  .
  .
  .
*IF DEF,ABC
  .
  .
  .
*ENDIF
```

Figure 5-13. Sequence of Deck

Since ABC is defined, all active lines between the IF and ENDIF pair are written as part of PROG2. Removing the DEFINE from the YANK$$$ deck would cause these text lines to be skipped.

The input stream in figure 5-14 has mutually exclusive requirements depending on the availability of correction set IDC. If IDC is known, the first 15 active lines after the first IF are written to the compile file. If IDC is not known, the lines following the second IF through the ENDIF are written to the compile file.

```
*DECK DECKA
.
.
.
*IF IDENT,IDC,15
*IF - IDENT,IDC
  active text lines
*ENDIF
```

Figure 5-14. Use of IF and ENDIF

Nesting of IF directives is illustrated in figure 5-15. The deck ROCK has an IF-controlled sequence containing a second IF-controlled sequence. The text following the first IF is written if PEBBLE is known (on the old program library); the text following the second IF is written if both PEBBLE and STONE are known. The ENDIF terminates both IF-controlled sequences.

```
*DECK ROCK
.
.
.
*IF IDENT,PEBBLE
.
.
.
*IF IDENT,STONE
.
.
.
*ENDIF
```

Figure 5-15. Nested IF Directives

ADDITION OF DECKS

A new program library, NEWPL, is constructed from the old program library, OLDPL, with the addition of one new common deck and two new decks. The new common deck, D1A, is the first deck after the YANK$$$ deck; the new deck XC follows deck SX; and the new deck SYSTEXT is the last deck on the new program library. No compile file is produced. All three of the ADDFILEs in figure 5-16 are to be read from the main input file INPUT. The ADDFILEs in figure 5-17 are to be read from the Update input file FNAME. In both these cases, the input file need not be specified but the two separators must be included (either space and comma or two commas). Each of the ADDFILE directives in figure 5-18 causes Update to read from a separate file that is not the main input file. Common deck D1A and its text are on FILEA; deck SYSTEXT and its text are on FILEB; deck XC and its text are on FILEC.

PULLMOD OPTION

The program library created by the example in figure 5-4 (Input File Not INPUT) has been altered by the correction run in figure 5-19. As a consequence of the run, the deck SET1 contains the lines shown in figure 5-20.

```
job statement
.
.
.
UPDATE(N,C=0)
.
.
.
/EOR
*ADDFILE INPUT,YANK$$$ or *ADDFILE,,YANK$$$
*COMDECK D1A
.
.
.
*ADDFILE INPUT or *ADDFILE
*DECK SYSTEXT
.
.
.
*ADDFILE INPUT,XB or *ADDFILE,,XB
*DECK XC
.
.
.
/EOI
```

Figure 5-16. ADDFILE Input on File INPUT

```
A.  Update Run
    job statement
    .
    .
    .
    UPDATE(N,C=0,I=FNAME)
    .
    .
    .
    /EOI

B.  Contents of file FNAME
    *ADDFILE FNAME,YANK$$$ or *ADDFILE,,YANK$$$
    *COMDECK D1A
    .
    .
    .
    *ADDFILE FNAME or *ADDFILE
    *DECK SYSTEXT
    .
    .
    .
    *ADDFILE FNAME,XB or *ADDFILE,,XB
    *DECK XC
    .
    .
    .
```

Figure 5-17. ADDFILE Input on File FNAME

The Update run in figure 5-21 recreates the correction set that changed SET1; the file PMFILE contains the recreated correction set shown in figure 5-22.

```
job statement
.
.
.
UPDATE(N,C=0)
.
.
.
/EOR
*ADDFILE FILEA,YANK$$$
*ADDFILE FILEB
*ADDFILE FILEC,XB
/EOI
```

Figure 5-18. ADDFILE Input on Secondary Input Files

```
job statement
.
.
.
UPDATE(N=PL2)
.
.
.
/EOR
*IDENT PMEX
*DELETE SET1.3
C         THIS IS FOR PULLMOD EXAMPLE
*COMPILE SET1
/EOI
```

Figure 5-19. Correction Run for PULLMOD Example

```
*DECK SET1
      PROGRAM ZIP
C     THIS IS FOR PULLMOD EXAMPLE
      STOP
      END
```

Figure 5-20. File Contents After Correction Run

```
job statement
.
.
.
UPDATE(G=PMFILE,P=PL2)
/EOR
*PULLMOD PMEX
/EOI
```

Figure 5-21. Pull Modifications

```
*IDENT PMEX
*DELETE SET1.3,SET1.3
C       THIS IS FOR PULLMOD EXAMPLE
```

Figure 5-22. Recreated Correction Run

PROGRAM LIBRARY AS A PERMANENT FILE

The job deck in figure 5-23 illustrates the creation and saving of a program library as a permanent file under NOS/BE and SCOPE 2; the deck in figure 5-24 saves a program library as an indirect access file under NOS. See the appropriate operating system reference manual for additional details.

```
job statement
accounting statements
REQUEST(PL,*PF)
UPDATE(N=PL,W,L=1234)
CATALOG(PL,PLIB,ID=JONES)
/EOR
*DECK ONE
.
.
.
/EOI
```

Figure 5-23. Permanent File Under NOS/BE or SCOPE 2

```
job statement
accounting statements
UPDATE(N=PL,W,L=1234)
SAVE(PL=UPLIB)
/EOR
*DECK ONE
.
.
.
/EOI
```

Figure 5-24. Permanent File Under NOS

SAMPLE FORTRAN PROGRAM

This set of Update examples illustrates how Update can be used for maintaining a FORTRAN program in program library format. The FORTRAN program calculates the area of a triangle obtaining the base and height from the data record.

The job in figure 5-25 places the FORTRAN program and subroutine as a single deck (ONE) on the new program library (NEWPL) and on the compile file (COMPILE). Following Update execution, FTN5 is called to compile the program; the source is on the COMPILE file. LGO calls for execution of the compiled program. This program does not execute because of an error in the SUBROUTINE statement. The name of the subroutine should be MSG, not MSA.

Examination of update output from the creation job reveals that the erroneous SUBROUTINE statement has line identifier ONE.20. The job in figure 5-26 corrects the error and generates a new program library.

```
job statement
.
.
.
UPDATE(N,F)
FTN5(I=COMPILE)
LGO.
.
.
.
/EOR
*DECK ONE
      PROGRAM ONE
      PRINT 5
5     FORMAT(1H1)
10    READ (*,100,END=120)BASE,HEIGHT
100   FORMAT (BZ,2F10.2,I1)
      IF (BASE.LE.0) GO TO 105
      IF (HEIGHT.LE.0) GO TO 105
      GO TO 106
105   CALL MSG
106   AREA = .5 * BASE * HEIGHT
      PRINT 110, BASE,HEIGHT,AREA
110   FORMAT (///,'BASE=',F20.5,'HEIGHT='
     C   ,F18.5,'AREA=',F20.5)
      WRITE (1) AREA
      GO TO 10
120   STOP
      END
      SUBROUTINE MSA
      PRINT 400
400   FORMAT (///,'FOLLOWING INPUT DATA
     C   NEGATIVE OR ZERO')
      RETURN
      END
/EOR
data
/EOI
```

Figure 5-25. FORTRAN Program Library - 1

```
job statement
.
.
.
ATTACH(OLDPL=MYLIB)
UPDATE(N,F)
FTN5(I=COMPILE)
LGO.
.
.
.
/EOR
*IDENT MOD1
*DELETE ONE.20
      SUBROUTINE MSG
/EOR
data
/EOI
```

Figure 5-26. Correction of SUBROUTINE Statement

The job in figure 5-27 uses the same input as the job in figure 5-25. However, the program in figure 5-27 is divided into two decks, MSG and ONE. Deck MSG is a common deck. A CALL directive is inserted into deck ONE to assure that whenever deck ONE is written on the compile file, MSG is also written on the compile file.

```
job statement
.
.
.
UPDATE(N,F)
FTN5(I=COMPILE)
LGO.
.
.
.
/EOR
*COMDECK MSG
       SUBROUTINE MSG
       PRINT 400
400    FORMAT (///,'FOLLOWING INPUT DATA
      C   NEGATIVE OR ZERO')
       RETURN
       END
*DECK ONE
       PROGRAM ONE
       PRINT 5
5      FORMAT(1H1)
10     READ (*,100,END=120)BASE,HEIGHT
100    FORMAT (BZ,2F10.2,I1)
       IF (BASE.LE.0) GO TO 105
       IF (HEIGHT.LE.0) GO TO 105
       GO TO 106
105    CALL MSG
106    AREA = .5 * BASE * HEIGHT
       PRINT 110,BASE,HEIGHT,AREA
110    FORMAT (///,'BASE=',F20.5,'HEIGHT='
      C   ,F18.5,'AREA=',F20.5)
       WRITE (1) AREA
       GO TO 10
120    STOP
       END
/EOR
data
/EOI
```

Figure 5-27. FORTRAN Program Library - 2

The example in figure 5-28 adds a deck to the library created in the previous example (figure 5-27). Since no new program library is generated (N is omitted from Update call), the addition is temporary.

```
job statement
.
.
.
ATTACH(OLDPL=MYLIB)
UPDATE.
FTN5(I=COMPILE)
LGO.
.
.
.
/EOR
*IDENT MOD2
*INSERT ONE.20
*DECK TWO
       PROGRAM TWO
       .
       .
       .
       END
*CALL MSG
*DELETE MSG.3
400    FORMAT (///,'FOLLOWING INPUT DATA
      C   POSITIVE')
/EOR
data
/EOI
```

Figure 5-28. Add Deck to FORTRAN Program Library

This appendix describes the code and character sets used by host computer operating system local batch device drivers, magnetic tape drivers, and terminal communication products. Some software products assume that certain graphic or control characters are associated with specific binary code values for collating or syntax processing purposes. This appendix does not describe those associations for all products.

All references within this manual to the ASCII character set or the ASCII code set refer to the character set and code set defined in the American National Standard Code for Information Interchange (ASCII, ANSI Standard X3.4-1977). References in this manual to the ASCII character set do not necessarily apply to the ASCII code set.

CHARACTER SETS AND CODE SETS

A character set differs from a code set. A character set is a set of graphic and/or control characters. A code set is a set of codes used to represent each character within a character set. Characters exist outside the computer system and communication network; codes are received, stored, retrieved, and transmitted within the computer system and network.

GRAPHIC AND CONTROL CHARACTERS

A graphic character can be displayed at a terminal or printed by a line printer. Examples of graphic characters are the characters A through Z, a blank, and the digits 0 through 9. A control character initiates, modifies, or stops a control operation. An example of a control character is the backspace character, which moves the terminal carriage or cursor back one space. Although a control character is not a graphic character, some terminals can produce a graphic representation when they receive a control character.

CODED AND BINARY CHARACTER DATA

Character codes can be interpreted as coded character data or as binary character data. Coded character data is converted from one code set representation to another as it enters or leaves the computer system; for example, data received from a terminal or sent to a magnetic tape unit is converted. Binary character data is not converted as it enters or leaves the system. Character codes are not converted when moved within the system; for example, data transferred to or from mass storage is not converted.

The distinction between coded character data and binary character data is important when reading or punching cards and when reading or writing magnetic tape. Only coded character data can be properly reproduced as characters on a line printer. Only binary character data can properly represent characters on a punched card when the data cannot be stored as display code.

The distinction between binary character data and characters represented by binary data (such as peripheral equipment instruction codes) is also important. Only such binary noncharacter data can properly reproduce characters on a plotter.

FORMATTED AND UNFORMATTED CHARACTER DATA

Character codes can be interpreted by a product as formatted character data or as unformatted character data. Formatted data can be stored or retrieved by a product in the form of the codes described for coded character data in the remainder of this appendix, or formatted data can be altered to another form during storage or retrieval; for example, 1 can be stored as a character code or as an integer value. Treatment of unformatted data by a product includes both coded character data and binary character data as described in this appendix.

NETWORK OPERATING SYSTEMS

The Network Operating System (NOS) and the Network Operating System/Batch Environment (NOS/BE) support the following character sets:

- CDC graphic 64-character set
- CDC graphic 63-character set
- ASCII graphic 64-character set
- ASCII graphic 63-character set
- ASCII graphic 95-character set

In addition, NOS supports the ASCII 128-character graphic and control set.

Each installation must select either a 64-character set or a 63-character set. The differences between the codes of a 63-character set and the codes of a 64-character set are described under Character Set Anomalies. Any reference in this appendix to a 64-character set implies either a 63- or 64-character set unless otherwise stated.

To represent its six listed character sets in central memory, NOS supports the following code sets:

- 6-bit display code
- 12-bit ASCII code (ASCII 8/12)
- 6/12-bit display code (ASCII 6/12)

Update only utilizes ASCII 8/12 or display code characters. No attempt to input ASCII 6/12 data (on NOS) should be made. The ASCII 6/12 data must first be converted to ASCII 8/12 data using the NOS FCOPY control statement.

To represent its five listed character sets in central memory, NOS/BE supports the following code sets:

- 6-bit display code
- 12-bit ASCII code (ASCII 8/12)

Under both NOS and NOS/BE, the 6-bit display code is a set of 6-bit codes from 00_8 to 77_8.

Under both NOS and NOS/BE, the 12-bit ASCII code is the ASCII 7-bit code (as defined by ANSI Standard X3.4-1977) right-justified in a 12-bit byte. Assuming that the bits are numbered from the right starting with 0, bits 0 through 6 contain the ASCII code, bits 7 through 10 contain zeros, and bit 11 distinguishes the 12-bit ASCII 0000_8 code from the end-of-line byte. The 12-bit codes are 0001_8 through 0177_8 and 4000_8.

Under NOS, the 6/12-bit display code is a combination of 6-bit codes and 12-bit codes. The 6-bit codes are 00_8 through 77_8, excluding 74_8 and 76_8. (The interpretation of the 00_8 and 63_8 codes is described under Character Set Anomalies later in this appendix.) The 12-bit codes begin with either 74_8 or 76_8 and are followed by a 6-bit code. Thus, 74_8 and 76_8 are considered escape codes and are never used as 6-bit codes within the 6/12-bit display code set. The 12-bit codes are 7401_8, 7402_8, 7404_8, 7407_8, and 7601_8 through 7677_8. All other 12-bit codes ($74xx_8$ and 7600_8) are undefined.

CHARACTER SET ANOMALIES

The operating system input/output software and some products interpret two codes differently when the installation selects a 63-character set rather than a 64-character set. If an installation uses a 63-character set, the colon graphic character is always represented by a 63_8 code, display code 00_8 is undefined (it has no associated graphic or punched card code), and the % graphic does not exist.

However, under NOS, if the installation uses a 64-character set, output of a 7404_8 6/12-bit display code or a 00_8 display code produces a colon. A colon can be input only as a 7404_8 6/12-bit display code. The use of undefined 6/12-bit display codes in output files produces unpredictable results and should be avoided.

Under NOS/BE, if the installation uses a 64-character set, output of a 00_8 display code produces a colon. Display code 63_8 is the colon when a 63-character set is used. The % graphic and related card codes do not exist on the 63-character set system and translations yield a blank (55_8).

Under both NOS and NOS/BE, two consecutive 00_8 codes can be confused with an end-of-line byte and should be avoided.

CHARACTER SET TABLES

The character set tables A-1 and A-2 are designed so that the user can find the character represented by a code (such as in a dump) or find the code that represents a character. To find the character represented by a code, the user looks up the code in the column listing the appropriate code set and then finds the character on that line in the column listing the appropriate character set. To find the code that represents a character, the user looks up the character and then finds the code on the same line in the appropriate column.

Conversational Terminal Users

Table A-1 shows the character sets and code sets available to an Interactive Facility (IAF) user at an ASCII code terminal using an ASCII character set. (Under NOS using network product software, certain Terminal Interface Program commands require specification of an ASCII code.)

IAF Usage

When in normal time-sharing mode (specified by the IAF NORMAL command), IAF assumes the ASCII graphic 64-character set is used and translates all input and output to or from display code. When in ASCII time-sharing mode (specified by the IAF ASCII command), IAF assumes the ASCII 128-character set is used and translates all input and output to or from 6/12-bit display code.

Update does not support 6/12-bit display code. The IAF user can convert a 6/12-bit code file to a 12-bit ASCII code file using the NOS FCOPY control statement. The resulting 12-bit ASCII file can be routed to a line printer but cannot be output through IAF.

TABLE A-1. CONVERSATIONAL TERMINAL CHARACTER SETS

ASCII Graphic (64-Character Set)	ASCII Character (128-Character Set)	Octal 6-Bit Display Code	Octal 6/12-Bit Display Code†	Octal 12-Bit ASCII Code	ASCII Graphic (64-Character Set)	ASCII Character (128-Character Set)	Octal 6-Bit Display Code	Octal 6/12-Bit Display Code†	Octal 12-Bit ASCII Code
: colon††		00††				^ circumflex		7402	0136
A	A	01	01	0101		: colon††		7404††	0072
B	B	02	02	0102		` grave accent		7407	0140
C	C	03	03	0103		a		7601	0141
D	D	04	04	0104		b		7602	0142
E	E	05	05	0105		c		7603	0143
F	F	06	06	0106		d		7604	0144
G	G	07	07	0107		e		7605	0145
H	H	10	10	0110		f		7606	0146
I	I	11	11	0111		g		7607	1047
J	J	12	12	0112		h		7610	0150
K	K	13	13	0113		i		7611	0151
L	L	14	14	0114		j		7612	0152
M	M	15	15	0115		k		7613	0153
N	N	16	16	0116		l		7614	0154
O	O	17	17	0117		m		7615	0155
P	P	20	20	0120		n		7616	0156
Q	Q	21	21	0121		o		7617	0157
R	R	22	22	0122		p		7620	0160
S	S	23	23	0123		q		7621	0161
T	T	24	24	0124		r		7622	0162
U	U	25	25	0125		s		7623	0163
V	V	26	26	0126		t		7624	0164
W	W	27	27	0127		u		7625	0165
X	X	30	30	0130		v		7626	0166
Y	Y	31	31	0131		w		7627	0167
Z	Z	32	32	0132		x		7630	0170
0	0	33	33	0060		y		7631	0171
1	1	34	34	0061		z		7632	0172
2	2	35	35	0062		{ left brace		7633	0173
3	3	36	36	0063		| vert. line		7634	0174
4	4	37	37	0064		} right brace		7635	0175
5	5	40	40	0065		~ tilde		7636	0176
6	6	41	41	0066		NUL		7640	4000
7	7	42	42	0067		SOH		7641	0001
8	8	43	43	0070		STX		7642	0002
9	9	44	44	0071		ETX		7643	0003
+ plus	+ plus	45	45	0053		EOT		7644	0004
- minus	- minus	46	46	0055		ENQ		7645	0005
* asterisk	* asterisk	47	47	0052		ACK		7646	0006
/ slash	/ slash	50	50	0057		BEL		7647	0007
(l. paren.	(l. paren.	51	51	0050		BS		7650	0010
) r. paren.	) r. paren.	52	52	0051		HT		7651	0011
$ dollar	$ dollar	53	53	0044		LF		7652	0012
= equal to	= equal to	54	54	0075		VT		7653	0013
space	space	55	55	0040		FF		7654	0014
, comma	, comma	56	56	0054		CR		7655	0015
. period	. period	57	57	0056		SO		7656	0016
# number	# number	60	60	0043		SI		7657	0017
[l. bracket	[l. bracket	61	61	0133		DEL		7637	0177
] r. bracket	] r. bracket	62	62	0135		DLE		7660	0020
% percent††	% percent††	63††	63††	0045		DC1		7661	0021
" quote	" quote	64	64	0042		DC2		7662	0022
_ underline	_ underline	65	65	0137		DC3		7663	0023
! exclam.	! exclam.	66	66	0041		DC4		7664	0024
& ampersand	& ampersand	67	67	0046		NAK		7665	0025
' apostrophe	' apostrophe	70	70	0047		SYN		7666	0026
? question	? question	71	71	0077		ETB		7667	0027
< less than	< less than	72	72	0074		CAN		7670	0030
> grtr. than	> grtr. than	73	73	0076		EM		7671	0031
@ coml. at		74				SUB		7672	0032
\ rev. slant	\ rev. slant	75	75	0134		ESC		7673	0033
^ circumflex		76				FS		7674	0034
; semicolon	; semicolon	77	77	0073		GS		7675	0035
	@ coml. at		7401	0100		RS		7676	0036
						US		7677	0037

†Generally available only on NOS, or through BASIC on NOS/BE.

††The interpretation of this character or code depends on its context. Refer to Character Set Anomalies in the text.

TABLE A-2. LOCAL BATCH DEVICE CHARACTER SETS

CDC Graphic (64-Character Set)	ASCII Graphic (64-Character Set)	ASCII Graphic (95-Character Set)	Octal 6-Bit Display Code	Octal 6/12-Bit Display Code[†]	Octal 12-Bit ASCII Code	Card Keypunch Code	
						026	029
: colon[††]	: colon[††]		00[††]			8-2	8-2
A	A	A	01	01	0101	12-1	12-1
B	B	B	02	02	0102	12-2	12-2
C	C	C	03	03	0103	12-3	12-3
D	D	D	04	04	0104	12-4	12-4
E	E	E	05	05	0105	12-5	12-5
F	F	F	06	06	0106	12-6	12-6
G	G	G	07	07	0107	12-7	12-7
H	H	H	10	10	0110	12-8	12-8
I	I	I	11	11	0111	12-9	12-9
J	J	J	12	12	0112	11-1	11-1
K	K	K	13	13	0113	11-2	11-2
L	L	L	14	14	0114	11-3	11-3
M	M	M	15	15	0115	11-4	11-4
N	N	N	16	16	0116	11-5	11-5
O	O	O	17	17	0117	11-6	11-6
P	P	P	20	20	0120	11-7	11-7
Q	Q	Q	21	21	0121	11-8	11-8
R	R	R	22	22	0122	11-9	11-9
S	S	S	23	23	0123	0-2	0-2
T	T	T	24	24	0124	0-3	0-3
U	U	U	25	25	0125	0-4	0-4
V	V	V	26	26	0126	0-5	0-5
W	W	W	27	27	0127	0-6	0-6
X	X	X	30	30	0130	0-7	0-7
Y	Y	Y	31	31	0131	0-8	0-8
Z	Z	Z	32	32	0132	0-9	0-9
0	0	0	33	33	0060	0	0
1	1	1	34	34	0061	1	1
2	2	2	35	35	0062	2	2
3	3	3	36	36	0063	3	3
4	4	4	37	37	0064	4	4
5	5	5	40	40	0065	5	5
6	6	6	41	41	0066	6	6
7	7	7	42	42	0067	7	7
8	8	8	43	43	0070	8	8
9	9	9	44	44	0071	9	9
+ plus	+ plus	+ plus	45	45	0053	12	12-8-6
- minus	- minus	- minus	46	46	0055	11	11
* asterisk	* asterisk	* asterisk	47	47	0052	11-8-4	11-8-4
/ slash	/ slash	/ slash	50	50	0057	0-1	0-1
(left paren.	(left paren.	(left paren.	51	51	0050	0-8-4	12-8-5
) right paren.	) right paren.	) right paren.	52	52	0051	12-8-4	11-8-5
$ dollar	$ dollar	$ dollar	53	53	0044	11-8-3	11-8-3
= equal to	= equal to	= equal to	54	54	0075	8-3	8-6
space	space	space	55	55	0040	no punch	no punch
, comma	, comma	, comma	56	56	0054	0-8-3	0-8-3
. period	. period	. period	57	57	0056	12-8-3	12-8-3
≡ equivalence	# number	# number	60	60	0043	0-8-6	8-3
[left bracket	[left bracket	[l. bracket	61	61	0133	8-7	12-8-2 or 12-0[†††]
] right bracket	] right bracket	] r. bracket	62	62	0135	0-8-2	11-8-2 or 11-0[†††]
% percent[††]	% percent[††]	% percent[††]	63	63	0045	8-6	0-8-4

TABLE A-2. LOCAL BATCH DEVICE CHARACTER SETS (Contd)

CDC Graphic (64-Character Set)	ASCII Graphic (64-Character Set)	ASCII Graphic (95-Character Set)	Octal 6-Bit Display Code	Octal 6/12-Bit Display Code[†]	Octal 12-Bit ASCII Code	Card Keypunch Code	
						026	029
≠ not equal	" quote	" quote	64	64	0042	8-4	8-7
↱ concat.	_ underline	_ underline	65	65	0137	0-8-5	0-8-5
∨ logical OR	! exclamation	! exclamation	66	66	0041	11-0	12-8-7
∧ logical AND	& ampersand	& ampersand	67	67	0046	0-8-7	12
↑ superscript	' apostrophe	' apostrophe	70	70	0047	11-8-5	8-5
↓ subscript	? question	? question	71	71	0077	11-8-6	0-8-7
< less than	< less than	< less than	72	72	0074	12-0	12-8-4
> greater than	> greater than	> greater than	73	73	0076	11-8-7	0-8-6
≤ less/equal	@ commercial at		74			8-5	8-4
≥ greater/equal	\ reverse slant	\ rev. slant	75	75	0134	12-8-5	0-8-2
¬ logical NOT	^ circumflex		76			12-8-6	11-8-7
; semicolon	; semicolon	; semicolon	77	77	0073	12-8-7	11-8-6
		@ coml. at		7401	0100		
		^ circumflex		7402	0136		
		: colon[††]		7404[††]	0072		
		` grave accent		7407	0140		
		a		7601	0141		
		b		7602	0142		
		c		7603	0143		
		d		7604	0144		
		e		7605	0145		
		f		7606	0146		
		g		7607	0147		
		h		7610	0150		
		i		7611	0151		
		j		7612	0152		
		k		7613	0153		
		l		7614	0154		
		m		7615	0155		
		n		7616	0156		
		o		7617	0157		
		p		7620	0160		
		q		7621	0161		
		r		7622	0162		
		s		7623	0163		
		t		7624	0164		
		u		7625	0165		
		v		7626	0166		
		w		7627	0167		
		x		7630	0170		
		y		7631	0171		
		z		7632	0172		
		{ left brace		7633	0173		
		| vert. line		7634	0174		
		} right brace		7635	0175		
		~ tilde		7636	0176		

[†] Generally available only on NOS, or through BASIC on NOS/BE.

[††] The interpretation of this character or code depends on its context. Refer to Character Set Anomalies in the text.

[†††] Available for input only, on NOS.

IAF supports both character mode and transparent mode transmissions through the network. These transmission modes are described under Network Access Method Terminal Transmission Code Sets in the Remote Batch Facility (RBF) reference manual. IAF treats character mode transmissions as coded character data; IAF converts these transmissions to or from either 6-bit or 6/12-bit display code. IAF treats transparent mode transmissions as binary character data; transparent mode communication between IAF and ASCII terminals using any parity setting occurs in the 12-bit ASCII code shown in table A-1.

Local Batch Users

Table A-2 lists the CDC graphic 64-character set, the ASCII graphic 64-character set, and the ASCII graphic 95-character set. This table also lists the code sets and card keypunch codes (026 and 029) that represent the characters.

The 64-character sets use display code as their code set; the 95-character set uses 12-bit ASCII code. The 95-character set is composed of all the characters in the ASCII 128-character set that can be printed at a line printer (refer to Line Printer Output). Only 12-bit ASCII code files can be printed using the ASCII graphic 95-character set. To print a 6/12-bit display code file (usually created in IAF ASCII mode), the user must convert the file to 12-bit ASCII code. To do this, the NOS FCOPY control statement must be issued. The 95-character set is represented by the 12-bit ASCII codes 0040_8 through 0176_8.

Line Printer Output

The batch character set printed depends on the print train used on the line printer to which the file is sent. The following are the print trains corresponding to each of the batch character sets:

Character Set	Print Train
CDC graphic 64-character set	596-1
ASCII graphic 64-character set	596-5
ASCII graphic 95-character set	596-6

The characters of the default 596-1 print train are listed in the table A-2 column labeled CDC Graphic (64-Character); the 596-5 print train characters are listed in the table A-2 column labeled ASCII Graphic (64-Character); and the 596-6 print train characters are listed in the table A-2 column labeled ASCII Graphic (95-Character).

If a transmission error occurs during the printing of a line, NOS prints the line again. The CDC graphic print train prints a concatenation symbol (↱) in the first printable column of a line containing errors. The ASCII print trains print an underline instead of the concatenation symbol.

If an unprintable character exists in a line (that is, a 12-bit ASCII code outside of the range 0040_8 through 0176_8), the number sign (#) appears in the first printable column of a print line and a space replaces the unprintable character.

Punched Card Input and Output

Under NOS, coded character data is exchanged with local batch card readers or card punches according to the translations shown in table A-2. As indicated in the table, additional card keypunch codes are available for input of the ASCII and CDC characters] and [. The 95-character set cannot be read or punched as coded character data.

Depending on an installation or deadstart option, NOS assumes an input deck has been punched either in 026 or 029 keypunch code (regardless of the character set in use). The alternate keypunch codes can be specified by a 26 or 29 punched in columns 79 and 80 of any 6/7/9 card or 7/8/9 card. The specified code translation remains in effect throughout the job unless it is reset by specification of the alternate code translation on a subsequent 6/7/9 card or 7/8/9 card.

NOS keypunch code translation can also be changed by a card containing a 5/7/9 punch in column 1. A blank (no punch) in column 2 indicates 026 conversion mode; a 9 punch in column 2 indicates 029 conversion mode. The conversion change remains in effect until another change card is encountered or the job ends.

The 5/7/9 card also allows literal input when 4/5/6/7/8/9 is punched in column 2. Literal input can be used to read 80-column binary character data within a punched card deck of coded character data.

Literal cards are stored with each column in a 12-bit byte (a row 12 punch is represented by a 1 in bit 11, row 11 by bit 10, row 0 by bit 9, and rows 1 through 9 by bits 8 through 0 of the byte), 16 central memory words per card. Literal input cards are read until a card identical to the previous 5/7/9 card (4/5/6/7/8/9 in column 2) is read. The next card can specify a new conversion mode.

Remote Batch Users

When card decks are read from remote batch devices, the ability to select alternate keypunch code translations depends upon the remote terminal equipment.

NOS Usage

Remote batch terminal line printer, punched card, and plotter character set support is described under Input Deck Structure in the Remote Batch Facility reference manual. RBF supports only character mode transmission to and from consoles through the network. Character mode is described under Network Access Method Terminal Transmission Code Sets in the Remote Batch Facility reference manual.

NOS/BE Usage

Remote batch terminal line printer and punched card character set support is described in the INTERCOM reference manual.

Magnetic Tape Users

Coded character data to be copied from mass storage to magnetic tape is assumed to be represented in display code. NOS converts the data to external BCD code when writing a coded 7-track tape and to ASCII or EBCDIC code (as specified on the tape assignment statement) when writing a coded 9-track tape.

Because only 63 characters can be represented in 7-track even parity, one of the 64 display codes is lost in conversion to and from external BCD code. Figure A-1 shows the differences in conversion that depend on which character set (63 or 64) the system uses. The ASCII character for the specified character code is shown in parentheses. The output arrow shows how the display code changes when it is written on tape in external BCD. The input arrow shows how the external BCD code changes when the tape is read and converted to display code.

63-Character Set

Display Code		External BCD		Display Code
00		16 (%)		00
33 (0)	Output →	12 (0)	Input →	33 (0)
63 (:)		12 (0)		33 (0)

64-Character Set

Display Code		External BCD		Display Code
00 (:)		12 (0)		33 (0)
33 (0)	Output →	12 (0)	Input →	33 (0)
63 (%)		16 (%)		63 (%)

Figure A-1. Magnetic Tape Code Conversions

Tables A-3 and A-4 show the character set conversions for nine-track tapes. Table A-3 lists the conversions to and from 7-bit ASCII character code and 6-bit display code. Table A-4 lists the conversions between 8-bit EBCDIC character code and 6-bit display code. Table A-5 shows the character set conversions between 6-bit external BCD and 6-bit display code for seven-track tapes.

If a lowercase ASCII or EBCDIC code is read from a 9-track coded tape, it is converted to its uppercase 6-bit display code equivalent. To read and write lowercase ASCII or EBCDIC characters, the user must assign the tape in binary mode and then convert the binary character data.

During binary character data transfers to or from 9-track magnetic tape, the 7-bit ASCII codes shown in table A-3 are read or written unchanged; the 8-bit hexadecimal EBCDIC codes shown in table A-4 also can be read or written unchanged. ASCII and EBCDIC codes cannot be read or written to 7-track magnetic tape as binary character data.

Tables A-6 and A-7 list the magnetic tape codes and their punch code equivalents on IBM host computers.

Two CDC utility products, FORM (not supported on SCOPE 2) and the 8-Bit Subroutines, can be used to convert to and from EBCDIC data. Table A-7 contains the octal values of each EBCDIC code right-justified in a 12-bit byte with zero fill. This 12-bit EBCDIC code can also be produced using FORM and the 8-Bit Subroutines.

TABLE A-3. ASCII 9-TRACK CODED TAPE CONVERSION

ASCII				Display Code†††	
Code Conversion†		Character and Code Conversion††			
Code (Hex)	Char	Code (Hex)	Char	ASCII Char	Code (Octal)
20	space	00	NUL	space	55
21	!	7D	}	!	66
22	"	02	STX	"	64
23	#	03	ETX	#	60
24	$	04	EOT	$	53
25	%	05	ENQ	%	63
25	%	05	ENQ	space	55
26	&	06	ACK	&	67
27	'	07	BEL	'	70
28	(	08	BS	(	51
29	)	09	HT	)	52
2A	*	0A	LF	*	47
2B	+	0B	VT	+	45
2C	,	0C	FF	,	56
2D	–	0D	CR	–	46
2E	.	0E	SO	.	57
2F	/	0F	SI	/	50
30	0	10	DLE	0	33
31	1	11	DC1	1	34
32	2	12	DC2	2	35
33	3	13	DC3	3	36
34	4	14	DC4	4	37
35	5	15	NAK	5	40
36	6	16	SYN	6	41
37	7	17	ETB	7	42
38	8	18	CAN	8	43
39	9	19	EM	9	44
3A	:	1A	SUB	:	00
Display code 00 is undefined at sites using the 63-character set.					
3A	:	1A	SUB	:	63
3B	;	1B	ESC	;	77
3C	<	7B	{	<	72
3D	=	1D	GS	=	54
3E	>	1E	RS	>	73
3F	?	1F	US	?	71

ASCII				Display Code†††	
Code Conversion†		Character and Code Conversion††			
Code (Hex)	Char	Code (Hex)	Char	ASCII Char	Code (Octal)
40	@	60	`	@	74
41	A	61	a	A	01
42	B	62	b	B	02
43	C	63	c	C	03
44	D	64	d	D	04
45	E	65	e	E	05
46	F	66	f	F	06
47	G	67	g	G	07
48	H	68	h	H	10
49	I	69	i	I	11
4A	J	6A	j	J	12
4B	K	6B	k	K	13
4C	L	6C	l	L	14
4D	M	6D	m	M	15
4E	N	6E	n	N	16
4F	O	6F	o	O	17
50	P	70	p	P	20
51	Q	71	q	Q	21
52	R	72	r	R	22
53	S	73	s	S	23
54	T	74	t	T	24
55	U	75	u	U	25
56	V	76	v	V	26
57	W	77	w	W	27
58	X	78	x	X	30
59	Y	79	y	Y	31
5A	Z	7A	z	Z	32
5B	[	1C	FS	[	61
5C	\	7C	|	\	75
5D	[	01	SOH	]	62
5E	^	7E	~	^	76
5F	_	7F	DEL	_	65

† When these characters are copied from or to a tape, the characters remain the same and the code changes from/to ASCII to/from display code.

†† These characters do not exist in display code. When the characters are copied from a tape, each ASCII character is changed to an alternate display code character. The corresponding codes are also changed. Example: When the system copies a lowercase a, 61_{16}, from tape, it writes an uppercase A, 01_8.

††† A display code space always translates to an ASCII space.

TABLE A-4. EBCDIC 9-TRACK CODED TAPE CONVERSION

EBCDIC				Display Code†††	
Code Conversion†		Character and Code Conversion††			
Code (Hex)	Char	Code (Hex)	Char	ASCII Char	Code (Octal)
40	space	00	NUL	space	55
4A	¢	1C	IFS	[	61
4B	.	0E	SO	.	57
4C	<	C0	{	<	72
4D	(	16	BS	(	51
4E	+	0B	VT	+	45
4F	|	D0	}	!	66
50	&	2E	ACK	&	67
5A	!	01	SOH	]	62
5B	$	37	EOT	$	53
5C	*	25	LF	*	47
5D	)	05	HT	)	52
5E	;	27	ESC	;	77
5F	¬	A1	~	/	76
60	-	0D	CR	-	46
61	'	0F	SI	'	50
6B	,	0C	FF	,	56
6C	%	2D	ENQ	%	63
6C	%	2D	ENQ	space	55
6D		07	DEL		65
6E	≥	1E	IRS	≥	73
6F	?	1F	IUS	?	71
7A	:	3F	SUB	:	00
Display code 00 is undefined at sites using the 63-character set.					
7A	:	3F	SUB	:	63
7B	#	03	ETX	#	60
7C	@	79	\	@	74
7D	'	2F	BEL	'	70
7E	=	1D	IGS	=	54
7F	"	02	STX	"	64
C1	A	81	a	A	01
C2	B	82	b	B	02
C3	C	83	c	C	03
C4	D	84	d	D	04
C5	E	85	e	E	05

EBCDIC				Display Code†††	
Code Conversion†		Character and Code Conversion††			
Code (Hex)	Char	Code (Hex)	Char	ASCII Char	Code (Octal)
C6	F	86	f	F	06
C7	G	87	g	G	07
C8	H	88	h	H	10
C9	I	89	i	I	11
D1	J	91	j	J	12
D2	K	92	k	K	13
D3	L	93	l	L	14
D4	M	94	m	M	15
D5	N	95	n	N	16
D6	O	96	o	O	17
D7	P	97	p	P	20
D8	Q	98	q	Q	21
D9	R	99	r	R	22
E0	\	6A	|	\	75
E2	S	A2	s	S	23
E3	T	A3	t	T	24
E4	U	A4	u	U	25
E5	V	A5	v	V	26
E6	W	A6	w	W	27
E7	X	A7	x	X	30
E8	Y	A8	y	Y	31
E9	Z	A9	z	Z	32
F0	0	10	DLE	0	33
F1	1	11	DC1	1	34
F2	2	12	DC2	2	35
F3	3	13	TM	3	36
F4	4	3C	DC4	4	37
F5	5	3D	NAK	5	40
F6	6	32	SYN	6	41
F7	7	26	ETB	7	42
F8	8	18	CAN	8	43
F9	9	19	EM	9	44

†All EBCDIC codes not listed translate to display code 55_8 (space). A display code space always translates to an EBCDIC space.

††These characters do not exist in display code. When the characters are copied from a tape, each EBCDIC character is changed to an alternate display code character. The corresponding codes are also changed. Example: When the system copies a lowercase a, 81_{16}, from tape, it writes an uppercase A, 01_8.

†††When these characters are copied from or to a tape, the characters remain the same (except EBCDIC codes $4A_{16}$, $4F_{16}$, $5A_{16}$, and $5F_{16}$) and the code changes from/to EBCDIC to/from display code.

TABLE A-5. 7-TRACK CODED TAPE CONVERSIONS

External BCD	ASCII Character	Octal Display Code	External BCD	ASCII Character	Octal Display Code
01	1	34	40	-	46
02	2	35	41	J	12
03	3	36	42	K	13
04	4	37	43	L	14
05	5	40	44	M	15
06	6	41	45	N	16
07	7	42	46	O	17
10	8	43	47	P	20
11	9	44	50	Q	21
12†	0	33	51	R	22
13	=	54	52	!	66
14	"	64	53	$	53
15	@	74	54	*	47
16†	%	63	55	'	70
17	[	61	56	?	71
20	space	55	57	>	73
21	/	50	60	+	45
22	S	23	61	A	01
23	T	24	62	B	02
24	U	25	63	C	03
25	V	26	64	D	04
26	W	27	65	E	05
27	X	30	66	F	06
30	Y	31	67	G	07
31	Z	32	70	H	10
32	]	62	71	I	11
33	,	56	72	<	72
34	(	51	73	.	57
35		65	74	)	52
36	$\overline{\#}$	60	75	\	75
37	&	67	76	^	76
			77	;	77

† As explained in the text of this appendix, conversion of these codes depends on whether the tape is being read or written.

TABLE A-6. AMERICAN NATIONAL STANDARD CODE FOR INFORMATION INTERCHANGE (ASCII) PUNCHED CARD CODES AND EBCDIC TRANSLATION

	b8 b7 b6 b5	0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1	1 0 0 0	1 0 0 1	1 0 1 0	1 0 1 1	1 1 0 0	1 1 0 1	1 1 1 0	1 1 1 1
b4 b3 b2 b1	COL ROW	0	1	2	3	4	5	6	7	8	9	10 (A)	11 (B)	12 (C)	13 (D)	14 (E)	15 (F)
0 0 0 0	0	NUL 12-0-9-8-1 NUL 00	DLE 12-11-9-8-1 DLE 10	SP no-punch SP 40	0 0 0 F0	@ 8-4 @ 7C	P 11-7 P D7	` 8-1 ` 79	p 12-11-7 p 97	11-0-9-8-1 DS 20	12-11-0-9-8-1 30	12-0-9-1 41	12-11-9-8 58	12-11-0-9-6 76	12-11-8-7 9F	12-11-0-8 88	12-11-9-8-4 DC
0 0 0 1	1	SOH 12-9-1 SOH 01	DC1 11-9-1 DC1 11	! 12-8-7 | 4F	1 1 1 F1	A 12-1 A C1	Q 11-8 Q D8	a 12-0-1 a 81	q 12-11-8 q 98	0-9-1 SOS 21	9-1 31	12-0-9-2 42	11-8-1 59	12-11-0-9-7 77	11-0-8-1 A0	12-11-0-9 89	12-11-9-8-5 DD
0 0 1 0	2	STX 12-9-2 STX 02	DC2 11-9-2 DC2 12	" 8-7 " 7F	2 2 2 F2	B 12-2 B C2	R 11-9 R D9	b 12-0-2 b 82	r 12-11-9 r 99	0-9-2 FS 22	11-9-8-2 CC 1A	12-0-9-3 43	11-0-9-2 62	12-11-0-9-8 78	11-0-8-2 AA	12-11-0-8-2 BA	12-11-9-8-6 DE
0 0 1 1	3	ETX 12-9-3 ETX 03	DC3 11-9-3 TM 13	# 8-3 # 7B	3 3 3 F3	C 12-3 C C3	S 0-2 S E2	c 12-0-3 c 83	s 11-0-2 s A2	0-9-3 23	9-3 33	12-0-9-4 44	11-0-9-3 63	12-0-8-1 80	11-0-8-3 AB	12-11-0-8-3 BB	12-11-9-8-7 DF
0 1 0 0	4	EOT 9-7 EOT 37	DC4 9-8-4 DC4 3C	$ 11-8-3 $ 5B	4 4 4 F4	D 12-4 D C4	T 0-3 T E3	d 12-0-4 d 84	t 11-0-3 t A3	0-9-4 BYP 24	9-4 PN 34	12-0-9-5 45	11-0-9-4 64	12-0-8-2 8A	11-0-8-4 AC	12-11-0-8-4 BC	11-0-9-8-2 EA
0 1 0 1	5	ENQ 0-9-8-5 ENQ 2D	NAK 9-8-5 NAK 3D	% 0-8-4 % 6C	5 5 5 F5	E 12-5 E C5	U 0-4 U E4	e 12-0-5 e 85	u 11-0-4 u A4	11-9-5 NL 15	9-5 RS 35	12-0-9-6 46	11-0-9-5 65	12-0-8-3 8B	11-0-8-5 AD	12-11-0-8-5 BD	11-0-9-8-3 EB
0 1 1 0	6	ACK 0-9-8-6 ACK 2E	SYN 9-2 SYN 32	& 12 & 50	6 6 6 F6	F 12-6 F C6	V 0-5 V E5	f 12-0-6 f 86	v 11-0-5 v A5	12-9-6 LC 06	9-6 UC 36	12-0-9-7 47	11-0-9-6 66	12-0-8-4 8C	11-0-8-6 AE	12-11-0-8-6 BE	11-0-9-8-4 ⑂ EC
0 1 1 1	7	BEL 0-9-8-7 BEL 2F	ETB 0-9-6 ETB 26	' 8-5 ' 7D	7 7 7 F7	G 12-7 G C7	W 0-6 W E6	g 12-0-7 g 87	w 11-0-6 w A6	11-9-7 IL 17	12-9-8 GE 08	12-0-9-8 48	11-0-9-7 67	12-0-8-5 8D	11-0-8-7 AF	12-11-0-8-7 BF	11-0-9-8-5 ED
1 0 0 0	8	BS 11-9-6 BS 16	CAN 11-9-8 CAN 18	(12-8-5 (4D	8 8 8 F8	H 12-8 H C8	X 0-7 X E7	h 12-0-8 h 88	x 11-0-7 x A7	0-9-8 28	9-8 38	12-8-1 49	11-0-9-8 68	12-0-8-6 8E	12-11-0-8-1 B0	12-0-9-8-2 CA	11-0-9-8-6 EE
1 0 0 1	9	HT 12-9-5 HT 05	EM 11-9-8-1 EM 19	) 11-8-5) 5D	9 9 9 F9	I 12-9 I C9	Y 0-8 Y E8	i 12-0-9 i 89	y 11-0-8 y A8	0-9-8-1 29	9-8-1 39	12-11-9-1 51	0-8-1 69	12-0-8-7 8F	12-11-0-1 B1	12-0-9-8-3 CB	11-0-9-8-7 EF
1 0 1 0	10 (A)	LF 0-9-5 LF 25	SUB 9-8-7 SUB 3F	* 11-8-4 * 5C	8-2 7A	J 11-1 J D1	Z 0-9 Z E9	j 12-11-1 j 91	z 11-0-9 z A9	0-9-8-2 SM 2A	9-8-2 3A	12-11-9-2 52	12-11-0 70	12-11-8-1 90	12-11-0-2 B2	12-0-9-8-4 ⑁ CC	12-11-0-9-8-2 |(LVM) FA
1 0 1 1	11 (B)	VT 12-9-8-3 VT 0B	ESC 0-9-7 ESC 27	+ 12-8-6 + 4E	; 11-8-6 ; 5E	K 11-2 K D2	[12-8-2 ¢ 4A	k 12-11-2 k 92	{ 12-0 { C0	0-9-8-3 CU2 2B	9-8-3 CU3 3B	12-11-9-3 53	12-11-0-9-1 71	12-11-8-2 9A	12-11-0-3 B3	12-0-9-8-5 CD	12-11-0-9-8-3 FB
1 1 0 0	12 (C)	FF 12-9-8-4 FF 0C	FS 11-9-8-4 IFS 1C	, 0-8-3 , 6B	< 12-8-4 < 4C	L 11-3 L D3	\ 0-8-2 \ E0	l 12-11-3 l 93	¦ 12-11 ¦ 6A	0-9-8-4 2C	12-9-4 PF 04	12-11-9-4 54	12-11-0-9-2 72	12-11-8-3 9B	12-11-0-4 B4	12-0-9-8-6 ⑃ CE	12-11-0-9-8-4 FC
1 1 0 1	13 (D)	CR 12-9-8-5 CR 0D	GS 11-9-8-5 IGS 1D	- 11 - 60	= 8-6 = 7E	M 11-4 M D4	] 11-8-2 ! 5A	m 12-11-4 m 94	} 11-0 } D0	12-9-8-1 RLF 09	11-9-4 RES 14	12-11-9-5 55	12-11-0-9-3 73	12-11-8-4 9C	12-11-0-5 B5	12-0-9-8-7 CF	12-11-0-9-8-5 FD
1 1 1 0	14 (E)	SO 12-9-8-6 SO 0E	RS 11-9-8-6 IRS 1E	. 12-8-3 . 4B	> 0-8-6 > 6E	N 11-5 N D5	^ 11-8-7 ¬ 5F	n 12-11-5 n 95	~ 11-0-1 ~ A1	12-9-8-2 SMM 0A	9-8-6 3E	12-11-9-6 56	12-11-0-9-4 74	12-11-8-5 9D	12-11-0-6 B6	12-11-9-8-2 DA	12-11-0-9-8-6 FE
1 1 1 1	15 (F)	SI 12-9-8-7 SI 0F	US 11-9-8-7 IUS 1F	/ 0-1 / 61	? 0-8-7 ? 6F	O 11-6 O D6	_ 0-8-5 _ 6D	o 12-11-6 o 96	DEL 12-9-7 DEL 07	11-9-8-3 CU1 1B	11-0-9-1 E1	12-11-9-7 57	12-11-0-9-5 75	12-11-8-6 9E	12-11-0-7 B7	12-11-9-8-3 DB	EO 12-11-0-9-8-7 FF

LEGEND

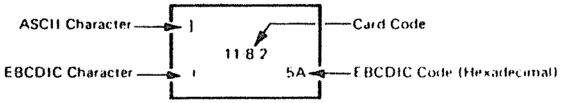
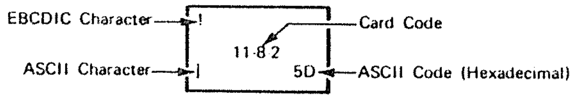


TABLE A-7. EXTENDED BINARY CODED DECIMAL INTERCHANGE CODE (EBCDIC) PUNCHED CARD CODES AND ASCII TRANSLATION

BITS 4 5 6 7	BITS 0 1 2 3 / 1ST HEX / 2ND	0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1	1 0 0 0	1 0 0 1	1 0 1 0	1 0 1 1	1 1 0 0	1 1 0 1	1 1 1 0	1 1 1 1
		0	1	2	3	4	5	6	7	8	9	A (10)	B (11)	C (12)	D (13)	E (14)	F (15)
0 0 0 0	0	NUL 12-0-9-8-1 NUL 00	DLE 12-11-9-8-1 DLE 10	DS 11-0-9-8-1 80	12-11-0-9-8-1 90	SP no punch SP 20	& 12 & 26	- 11 - 2D	12-11-0 BA	12-0-8-1 C3	12-11-8-1 CA	11-0-8-1 D1	12-11-0-8-1 D8	{ 12-0 { 7B	} 11-0 } 7D	\ 0-8-2 \ 5C	0 0 0 30
0 0 0 1	1	SOH 12-9-1 SOH 01	DC1 11-9-1 DC1 11	SOS 0-9-1 81	9-1 91	12-0-9-1 A0	12-11-9-1 A9	/ 0-1 / 2F	12-11-0-9-1 BB	a 12-0-1 a 61	j 12-11-1 j 6A	~ 11-0-1 ~ 7E	12-11-0-1 D9	A 12-1 A 41	J 11-1 J 4A	11-0-9-1 9F	1 1 1 31
0 0 1 0	2	STX 12-9-2 STX 02	DC2 11-9-2 DC2 12	FS 0-9-2 82	SYN 9-2 SYN 16	12-0-9-2 A1	12-11-9-2 AA	11-0-9-2 B2	12-11-0-9-2 BC	b 12-0-2 b 62	k 12-11-2 k 6B	s 11-0-2 s 73	12-11-0-2 DA	B 12-2 B 42	K 11-2 K 4B	S 0-2 S 53	2 2 2 32
0 0 1 1	3	ETX 12-9-3 ETX 03	TM 11-9-3 DC3 13	0-9-3 83	9-3 93	12-0-9-3 A2	12-11-9-3 AB	11-0-9-3 B3	12-11-0-9-3 BD	c 12-0-3 c 63	l 12-11-3 l 6C	t 11-0-3 t 74	12-11-0-3 DB	C 12-3 C 43	L 11-3 L 4C	T 0-3 T 54	3 3 3 33
0 1 0 0	4	PF 12-9-4 9C	RES 11-9-4 9D	BYP 0-9-4 84	PN 9-4 94	12-0-9-4 A3	12-11-9-4 AC	11-0-9-4 B4	12-11-0-9-4 BE	d 12-0-4 d 64	m 12-11-4 m 6D	u 11-0-4 u 75	12-11-0-4 DC	D 12-4 D 44	M 11-4 M 4D	U 0-4 U 55	4 4 4 34
0 1 0 1	5	HT 12-9-5 HT 09	NL 11-9-5 85	LF 0-9-5 LF 0A	RS 9-5 95	12-0-9-5 A4	12-11-9-5 AD	11-0-9-5 B5	12-11-0-9-5 BF	e 12-0-5 e 65	n 12-11-5 n 6E	v 11-0-5 v 76	12-11-0-5 DD	E 12-5 E 45	N 11-5 N 4E	V 0-5 V 56	5 5 5 35
0 1 1 0	6	LC 12-9-6 86	BS 11-9-6 BS 08	ETB 0-9-6 ETB 17	UC 9-6 96	12-0-9-6 A5	12-11-9-6 AE	11-0-9-6 B6	12-11-0-9-6 C0	f 12-0-6 f 66	o 12-11-6 o 6F	w 11-0-6 w 77	12-11-0-6 DE	F 12-6 F 46	O 11-6 O 4F	W 0-6 W 57	6 6 6 36
0 1 1 1	7	DEL 12-9-7 DEL 7F	IL 11-9-7 87	ESC 0-9-7 ESC 1B	EOT 9-7 EOT 04	12-0-9-7 A6	12-11-9-7 AF	11-0-9-7 B7	12-11-0-9-7 C1	g 12-0-7 g 67	p 12-11-7 p 70	x 11-0-7 x 78	12-11-0-7 DF	G 12-7 G 47	P 11-7 P 50	X 0-7 X 58	7 7 7 37
1 0 0 0	8	GE 12-9-8 97	CAN 11-9-8 CAN 18	0-9-8 88	9-8 98	12-0-9-8 A7	12-11-9-8 B0	11-0-9-8 B8	12-11-0-9-8 C2	h 12-0-8 h 68	q 12-11-8 q 71	y 11-0-8 y 79	12-11-0-8 E0	H 12-8 H 48	Q 11-8 Q 51	Y 0-8 Y 59	8 8 8 38
1 0 0 1	9	RLF 12-9-8-1 8D	EM 11-9-8-1 EM 19	0-9-8-1 89	9-8-1 99	12-8-1 A8	11-8-1 B1	0-8-1 B9	` 8-1 ` 60	i 12-0-9 i 69	r 12-11-9 r 72	z 11-0-9 z 7A	12-11-0-9 E1	I 12-9 I 49	R 11-9 R 52	Z 0-9 Z 5A	9 9 9 39
1 0 1 0	A (10)	SMM 12-9-8-2 8E	CC 11-9-8-2 92	SM 0-9-8-2 8A	9-8-2 9A	¢ 12-8-2 [5B	! 11-8-2] 5D	¦ 12-11 ¦ 7C	: 8-2 3A	12-0-8-2 C4	12-11-8-2 CB	11-0-8-2 D2	12-11-0-8-2 E2	12-0-9-8-2 E8	12-11-9-8-2 EE	11-0-9-8-2 F4	|(LVM) 12-11-0-9-8-2 FA
1 0 1 1	B (11)	VT 12-9-8-3 VT 0B	CU1 11-9-8-3 8F	CU2 0-9-8-3 8B	CU3 9-8-3 9B	. 12-8-3 . 2E	$ 11-8-3 $ 24	, 0-8-3 , 2C	# 8-3 # 23	12-0-8-3 C5	12-11-8-3 CC	11-0-8-3 D3	12-11-0-8-3 E3	12-0-9-8-3 E9	12-11-9-8-3 EF	11-0-9-8-3 F5	12-11-0-9-8-3 FB
1 1 0 0	C (12)	FF 12-9-8-4 FF 0C	IFS 11-9-8-4 FS 1C	0-9-8-4 8C	DC4 9-8-4 DC4 14	< 12-8-4 < 3C	* 11-8-4 * 2A	% 0-8-4 % 25	@ 8-4 @ 40	12-0-8-4 C6	12-11-8-4 CD	11-0-8-4 D4	12-11-0-8-4 E4	⑀ 12-0-9-8-4 EA	12-11-9-8-4 F0	⑁ 11-0-9-8-4 F6	12-11-0-9-8-4 FC
1 1 0 1	D (13)	CR 12-9-8-5 CR 0D	IGS 11-9-8-5 GS 1D	ENQ 0-9-8-5 ENQ 05	NAK 9-8-5 NAK 15	(12-8-5 (28	) 11-8-5) 29	0-8-5 5F	' 8-5 ' 27	12-0-8-5 C7	12-11-8-5 CE	11-0-8-5 D5	12-11-0-8-5 E5	12-0-9-8-5 EB	12-11-9-8-5 F1	11-0-9-8-5 F7	12-11-0-9-8-5 FD
1 1 1 0	E (14)	SO 12-9-8-6 SO 0E	IRS 11-9-8-6 RS 1E	ACK 0-9-8-6 ACK 06	9-8-6 9E	+ 12-8-6 + 2B	; 11-8-6 ; 3B	> 0-8-6 > 3E	= 8-6 = 3D	12-0-8-6 C8	12-11-8-6 CF	11-0-8-6 D6	12-11-0-8-6 E6	⑂ 12-0-9-8-6 EC	12-11-9-8-6 F2	11-0-9-8-6 F8	12-11-0-9-8-6 FE
1 1 1 1	F (15)	SI 12-9-8-7 SI 0F	IUS 11-9-8-7 US 1F	BEL 0-9-8-7 BEL 07	SUB 9-8-7 SUB 1A	| 12-8-7 ! 21	¬ 11-8-7 ^ 5E	? 0-8-7 ? 3F	" 8-7 " 22	12-0-8-7 C9	12-11-8-7 D0	11-0-8-7 D7	12-11-0-8-7 E7	12-0-9-8-7 ED	12-11-9-8-7 F3	11-0-9-8-7 F9	12-11-0-9-8-7 EO FF

LEGEND



DIAGNOSTICS B

Diagnostic messages can either appear in the dayfile or are intermixed with Update output in the output file. In addition to detecting errors, Update detects overlapping corrections when the EXTOVLP installation option has been assembled.

DIAGNOSTIC MESSAGES

All diagnostic messages that can be issued during an Update run are listed in alphabetic order in table B-1. One of the following codes is included for each diagnostic:

Type	Meaning
I	An informative message; processing continues.
N	A nonfatal error; processing continues.
F	A fatal error; processing is terminated.

OVERLAPPING CORRECTIONS

Update can detect four overlapping correction situations. When any of these types are detected, Update prints the line in error with the words TP.n OVLP appended on the far right of this line. Type n is one of the following:

Type	Meaning
1	Two or more modifications are made to one line by a single correction set.
2	A modification attempts to activate an already active line.
3	A modification attempts to deactivate an already inactive line.
4	A line is inserted after a line which is inactive on the old program library and is inactive on the new program library.

The listing of overlap lines is controlled by list option 3.

Detection of an overlap does not necessarily indicate a user error. Overlap messages are advisory, and they point to conditions in which the probability of error is greater than normal. If any overlap condition is encountered, a dayfile message is printed.

Type TP.2 and TP.3 are detected by comparing existing correction history bytes with those to be added. Complex operations involving YANK and PURGE might generate these overlap messages even though no overlap occurs.

TABLE B-1. DIAGNOSTICS

Message	Type	Significance	Action
A OPTION INVALID WITH RANDOM OLDPL OR SEQUENTIAL NEWPL	F	The old program library is not sequential or the new program library is not random or is not on a random device for a sequential-to-random copy.	Correct the error.
ADDFILE DIRECTIVE INVALID ON REMOTE FILE	F	The ADDFILE directive cannot be used in the file specified by a READ directive.	Remove the ADDFILE directive from the file specified by the READ directive.
ADDFILE FIRST LINE MUST BE DECK OR COMDECK	F	The first line on the file specified by the ADDFILE directive is not a DECK or COMDECK directive.	Correct the error.
ALL YANK, SELYANK, YANKDECK, AND CALL DIRECTIVES AFFECTED HAVE BEEN CHANGED	I	If Update changes any identifiers during a merge, it also changes the corresponding YANK, SELYANK, YANKDECK, and CALL directives.	None.
B OPTION INVALID WITH SEQUENTIAL OLDPL	E	The old program library is not random for a random-to-sequential copy.	Do not specify B on the control statement.

TABLE B-1. DIAGNOSTICS (Contd)

Message	Type	Significance	Action
BAD ORDER ON YANK DIRECTIVE	N	Identifiers separated by a period on the YANK directive are in the wrong order.	Correct the order of the identifiers.
LINE NUMBER ZERO OR INVALID CHARACTER IN NUMERIC FIELD	F	Sequence number field on a correction directive is erroneous.	Correct the sequence number.
DIRECTIVE INVALID OR MISSING	F	Update detected a format error on a directive, deleted a directive that was unrecognizable, or detected an illegal file name. Illegal operations such as INSERT prior to an IDENT could also have been attempted.	Correct the error.
COPY TO EXTERNAL FILE NOT ALLOWED WHEN READING ALTERNATE INPUT UNIT	N	No copy is made.	Correct the error.
COPYING INPUT TO TEMPORARY NEWPL	I	A sequential new program library was requested on a creation run.	None.
COPYING OLDPL TO A RANDOM FILE	I	The old program library is being copied to a random file.	None.
CREATING NEW PROGRAM LIBRARY	I	Indicates that a new program library is being created.	None.
DECK NAME ON ABOVE LINE NOT LAST DECLARED DECK	I	When a DECLARE directive is in effect, only line images belonging to decks specified can be modified or referenced.	Add appropriate DECLARE directives or remove directives that reference non-declared decks.
DECK SPECIFIED ON MOVE OR COPY DIRECTIVE NOT ON OLDPL, DIRECTIVE WILL BE IGNORED	I	The specified deck will not be moved or copied.	Correct the error.
DECK STRUCTURE CHANGED	I	A deck has been moved or deleted.	None.
DO/DONT IDENT idname IS NOT YANKED/YANKED NULL DO/DONT	I	A DO directive, to negate the effect of a YANK, references an identifier that has been yanked; or a DONT directive, to restore a YANK, references an identifier that was already yanked.	None.
DUPLICATE DECK dname NEWPL ILLEGAL	F/N	Update encountered an active DECK or COMDECK directive that duplicates a previous directive. This condition is fatal if a new program library is being created; nonfatal if a new program library is not being created.	Change one of the deck names.
DUPLICATE FILE NAME OF file, JOB ABORTED	F	The same file name has been assigned to two Update files.	Change one of the file names.
DUPLICATE IDENT CHANGED TO ident	N	Update changed a duplicate identifier name to a unique one.	None.

TABLE B-1. DIAGNOSTICS (Contd)

Message	Type	Significance	Action
DUPLICATE IDENT NAME	F	During a merge run, Update encountered a duplicate identifier name that it could not make unique.	Change one of the identifiers.
DUPLICATE IDENT NAME IN ADDFILE	F	The name of a correction set to be added as a result of an ADDFILE directive duplicates a correction set name on the old program library.	Change the name of the correction set.
DUPLICATE SECONDARY OLDPL IGNORED	I	Two secondary old program libraries have the same name.	Correct the error or ignore.
ERROR. NO PERMISSION TO WRITE NEWPL	F	Both MODIFY and EXTEND permission must be present to overwrite a permanent (direct access) file, or a write ring must be in place to store the information.	Attach file with correct access permissions. The tape requested or labeled for output must have a write ring. (Refer to the NOS, NOS/BE, or SCOPE reference manuals.)
ERRORNOT ALL MODS WERE PROCESSED***	F	All changes indicated in the input stream were not processed.	Make sure that names specified on correction directives correspond to identifiers on the old program library (or on the COMPILE directive if in quick mode).
ERROR. WIDTH EXCEEDS 256 CHARACTERS	N	Total of statement width plus ident field width is greater than 256 characters on *WIDTH statement.	Correct *WIDTH statement.
FILENAME OF file IS TOO LONG, UPDATE ABORTED	F	A file name exceeds seven characters.	Correct the file name.
FILENAME ON ABOVE DIRECTIVE GREATER THAN SEVEN CHARACTERS	F	A file name exceeds seven characters.	Correct the file name.
FILE NAME ON UPDATE CONTROL STATEMENT GR 7 CHARACTERS	F	A file name on the UPDATE control statement is greater than seven characters.	Correct the file name.
G AND O FILES CANNOT HAVE SAME FILENAME	F	The G and O control statement options specify the same file name.	Change one of the names.
GARBAGE IN OLDPL HEADER, UPDATE ABORTED	F	Invalid data was found in the random index.	Rerun job/re-create program library. If the problem still exists, follow site-defined procedures for reporting software errors or operational problems.
IDENT DIRECTIVE MISSING, NO NEWPL REQUESTED, DEFAULT IDENTIFIER OF .NO.ID. USED	I/F	If no new program library is generated, then a correction set need not be introduced by an IDENT directive. The identifier .NO.ID. is used.	Add IDENT directive if new program library is to be generated.
IDENT xxxxx WILL NOT BE PROCESSED	I	Named correction set not processed because dependency condition (K or U parameter on the IDENT directive) has not been met.	None.

TABLE B-1. DIAGNOSTICS (Contd)

Message	Type	Significance	Action
IDENT LONGER THAN NINE CHARACTERS	F	An identifier can only have up to nine characters.	Correct the identifier.
IDENTIFIERS SEPARATED BY PERIOD IN WRONG ORDER	F	The specified identifiers are not in the correct order.	Switch the identifiers.
ILLEGAL CONTROL STATEMENT IN ADDFILE	F	ADDFILE insertions cannot contain correction directives.	Remove the correction directives.
IMPROPER MASTER CHARACTER CHANGED TO char	N	The character specified on the * control statement parameter is not the same as the master control character on the old program library.	Use the same master control character as on the old program library.
INSUFFICIENT FIELD LENGTH, UPDATE ABORT	F	The table manager ran out of room for internal tables.	Allocate more field length.
IT MAY EXIST IN A DECK NOT MENTIONED ON A COMPILE DIRECTIVE	F	An identifier references a line in a deck not specified on a COMPILE directive (only if in quick mode).	Correct the error.
INVALID NUMERIC FIELD	F	The directive does not contain required numeric field.	Correct the directive.
LENGTH ERROR ON OLDPL. UNUSABLE OLDPL OR HARDWARE ERROR	F	Line length on old program library is greater than the maximum allowed or is less than one.	Rerun job. If problem still exists, then recreate the program library.
LISTED BELOW ARE ALL IDENT NAMES WHICH WERE CHANGED DURING THE MERGE	I	Update changes any duplicate identifiers to make them unique when merging two program libraries.	None.
NEW IDENT ON CHANGE DIRECTIVE IS ALREADY KNOWN	F	An attempt was made to change a correction set identifier to one already in existence.	Correct the error.
NO ACTIVE LINES WERE FOUND WITHIN THE COPY RANGE. NULL COPY	N	All line images within the specified range are inactive.	None.
NO DECK NAME ON DECK DIRECTIVE	F	No name was specified on the DECK directive.	Specify a name.
NO INPUT FILE, Q MODE, UPDATE ABORT	F	In quick mode, Update relies on the input file to determine what is written to the compile file.	Put appropriate COMPILE directives in the input file.
NO OLDPL, NOT CREATION RUN, UPDATE ABORT	F	No old program library was supplied on a non-creation run.	Correct the error.
NULL ADDFILE	I	The first read on the file specified by ADDFILE encountered an end-of-record. If the input file was specified, the first read encountered an illegal directive.	Correct the error.
NULL IDENT	F	An identifier was not found on a directive where one was expected.	Correct the directive.

TABLE B-1. DIAGNOSTICS (Contd)

Message	Type	Significance	Action
NULL DECK NAME	F	During ADDFILE or a CREATION run, Update encountered a DECK or COMDECK directive that did not have a name.	Correct the directive.
OLDPL READ ERROR - ATTEMPTING RECOVERY ***READ RECOVERED - DATA LOST BEFORE THE FOLLOWING LINE*** -line image-	F	A parity error or other error has occurred while processing an old program library. As a result, Update is uncertain of the position of the old program library. When Update finds the next valid line following the error, the second message and the image of that line are printed.	Rerun the job. If the U option is used, line images might be lost on the NEWPL.
OLDPLS HAVE DIFFERENT CHARACTERS SETS	I	The merging of two old program libraries with different character sets is not allowed.	Use program libraries with the same character set.
OUTPUT LINE LIMIT EXCEEDED. LIST OPTIONS 3 AND 4 DEFEATED	N	Update output exceeds the line limit specified by default or by the LIMIT directive.	Use the LIMIT directive to increase one unit.
PLS HAVE DIFFERENT CONTROL CHARACTERS, ABORT	F	The merging of two program libraries with different control characters is not allowed.	Use program libraries with the same control characters.
PREMATURE END OF RECORD ON OLD PROGRAM LIBRARY	F	A PRU of level 0 was encountered in the line image.	Rerun the job. If error still exists, recreate the program library.
READING INPUT	I	The input file is being read by Update.	None.
RECURSIVE CALL ON COMDECK dname IGNORED. FATAL ERROR	F	A common deck has called itself or common decks that contain calls to the specified common deck.	Correct the error.
SECONDARY OLDPL NOT RANDOM	F	Secondary old program libraries must be random.	Use random secondary old program libraries.
SEQUENCE NUMBER EXCEEDS 131071	F	The proper range of sequence numbers is 1 through 131071.	Correct the error.
STACK DEPTH EXCEEDED	F	Stack in which line images are placed became full while processing a BEFORE or ADDFILE directive.	Follow site-defined procedures for reporting software errors or operational problems. (increase RECURDEP).
TABLE MANAGER LOGIC ERROR	F	There is not enough table space to accommodate the old program library tables.	Increase field length.
THE ABOVE CALLED COMMON DECK WAS NOT FOUND	F	The called common deck could not be found.	Check the spelling of the deck name. If creating a program library with calls to secondary old program libraries, set C=0 on the UPDATE control statements.
THE ABOVE CARD AFFECTS A DECK OTHER THAN THE DECLARED DECK	I	Corrections are restricted to the named deck.	Change the declared deck or correct the identifier name.

TABLE B-1. DIAGNOSTICS (Contd)

Message	Type	Significance	Action
THE ABOVE DIRECTIVE IS ILLEGAL DURING A CREATION RUN	F	A directive that is not allowed on a creation run was encountered.	Remove the illegal directive.
THE ABOVE DIRECTIVE IS ILLEGAL IN AN ALTERNATE FILE. IGNORED	N	Directives *READ, *SKIP, or *REWIND are illegal in an alternate file.	Remove the illegal directives.
THE ABOVE DIRECTIVE IS ILLEGAL AFTER A DECK HAS BEEN DECLARED	N	CHANGE, PURGE, and YANK directives are illegal after a deck has been specified on a DECLARE directive. They are ignored.	Remove the illegal directives.
THE ABOVE LISTED DIRECTIVES CANNOT EXIST IN THE YANK DECK AND HAVE BEEN PURGED DURING EDITING	I	Only YANK, YANKDECK, SEL-YANK, and DEFINE directives are kept in the YANK$$$ deck.	None.
THE ABOVE OPERATION IS NOT LEGAL WHEN REFERENCING THE YANK DECK	F	The specified operation is illegal when referencing the YANK$$$ deck.	Correct the error.
THE ABOVE SPECIFIED LINE WAS NOT ENCOUNTERED	F	Update could not locate the specified line on the old program library.	Make sure that the correct identifier is specified.
THE INITIAL LINE OF THE COPY RANGE WAS NOT FOUND. NULL COPY	N	No copy was made.	Make sure that the correct identifier is specified.
THE TERMINAL LINE OF THE COPY RANGE WAS NOT FOUND. COPY ENDS AT END OF SPECIFIED DECK	I	The last line specified was not found; the rest of the deck was copied.	Make sure that the correct identifier is specified.
THE TERMINAL LINE SPECIFIED WAS NOT ENCOUNTERED	F	While processing a line range, Update could not locate the last line of the range.	Make sure that the correct identifier is specified.
THIS UPDATE REQUIRED n WORDS OF CORE	I	It took n words of memory for the update.	None.
TOO MANY CHBS -- INCREASE L.CHB	F	Correction history bytes exceed the specified limit of 100_8 for a line.	Increase of value of L.CHB in Update and reinstall it.
TOO MANY SECONDARY OLDPLS SPECIFIED	F	Up to seven secondary old program libraries can be specified.	Specify seven or fewer secondary old program libraries.
UNBALANCED TEXT/ENDTEST DIRECTIVES	N	TEXT/ENDTEXT directives encountered in the run were not matching pairs.	Make TEXT/ENDTEXT directives matching pairs.
UNKNOWN IDENTIFIER idname	F	A correction directive references an identifier not found in the directory.	Make sure that the correct identifier is specified.
UPDATE COMPLETE	I	The update is completed.	None.
UPDATE CONTROL STATEMENT ERROR(S)	F	The UPDATE control statement contains unacceptable parameters. The erroneous parameters are listed on the next line.	Correct the erroneous parameters.

TABLE B-1. DIAGNOSTICS (Contd)

Message	Type	Significance	Action
UPDATE CREATION RUN	I	This Update run was a creation run.	None.
WAITING FOR 45000B WORDS	I	Update is waiting for the operating system to allocate enough memory.	None.
OLDPL CHECKSUM ERROR	F	At least one updated deck from the old program library is bad.	Rerun Job. If problem still exists, follow site-defined procedures for reporting software errors or operational problems.
YANK, SELYANK, OR YANKDECK ident NOT KNOWN	N	The identifier referenced on a YANK, SELYANK, or YANKDECK has probably been purged; this applies to lines already on the library.	Remove the yank directive from the YANK$$$ deck.
deckname IS NOT A VALID DECK NAME	F	A deck name has 1 through 9 characters; legal characters are: A through Z, 0 through 9, and + - * /() $=.	Correct the deck name.
n ERRORS IN INPUT	I	Update encountered n fatal errors in the input stream. Processing continues in order to detect additional errors. This message is issued only if the U parameter is specified on the control statement.	None.
n ERRORS IN INPUT, NEWPL, COMPILE, SOURCE SUPPRESSED	I	Update encountered n fatal errors in the input stream. Processing continues in order to detect errors. A new program library, a compile file, and a source file are not generated.	None.
n ERRORS IN UPDATE INPUT	I	First pass of Update processing encountered n fatal errors while reading a correction set.	None.
n DECLARE ERRORS	I	Indicates the number of directives that reference line images in decks not specified on DECLARE directives.	None.
n FATAL ERRORS	I	Indicates the number of errors that caused Update to abort.	None.
n NONFATAL ERRORS	I	Indicates the number of errors that did not cause Update to abort.	None.
n OVERLAPPING CORRECTIONS	I	A correction set changed the status of some lines more than once or referenced an inactive line image.	None.
n UPDATE ERRORS, JOB ABORTED	I	Errors were encountered in reading the input file.	None.

GLOSSARY C

ASCII -
American Standard Code for Information Interchange. ASCII input and output codes for Update are 8-bit characters right-justified in a 12-bit byte.

Common Deck -
A deck that is written on a compile file as a result of a CALL directive. The COMDECK directive introduces a common deck.

Compile File -
The file generated by Update that contains line images restored to a format that is acceptable to a compiler or assembler.

Copy Run -
An Update run that performs a sequential-to-random or random-to-sequential copy of a program library. Contrast with creation run and correction run.

Correction History Byte -
A byte added to a line image by Update each time the status of the line image changes. The correction history byte tells Update whether or not a line image is active or inactive and which correction set modified the line image. A maximum of 100B correction history bytes may exist for each line.

Correction Run -
An Update run in which changes can be made to a program library. Contrast with copy run and creation run.

Correction Set -
A set of directives and text that direct Update to modify a program library. The IDENT directive introduces a correction set.

Creation Run -
An Update run that constructs a program library. It is the original transfer of lines into Update format. Contrast with copy run and correction run.

Deck -
A deck consists of a DECK or COMDECK directive and all text and directives until the next DECK or COMDECK directive. It is the smallest unit that can be extracted from a program library.

Deck List -
A list internal to Update that contains the names of all decks in the program library and the location of the first word for each deck.

Directory -
A list that contains one entry for each DECK, COMDECK, and IDENT directive that is used for the program library.

Full Update Mode -
An Update run in which the F parameter is selected on the control statement causing Update to process all decks on the library. Contrast with normal selective mode and quick Update mode.

Identifier -
The name of a deck, common deck, or correction set.

Input File -
The user-supplied file or part of the job deck that contains the input stream of Update directives and text.

Known -
The status of a deck name or identifier that is on the primary old program library. The deck name must be in the deck list on the primary old program library and an identifier must be in the directory on the primary old program library.

Line Identifier -
The combination of identifier and sequence number that uniquely identifies each line image in a program library.

Master Control Character -
A character in column 1 that informs Update that the line contains a directive.

Merge File -
The file that contains a program library to be merged with the old program library into a new program library.

New Program Library -
The program library either automatically generated by a creation run or optionally generated by a correction run.

Normal Selective Mode -
An Update run in which the F and Q options are not selected on the control statement. All decks specified on COMPILE directives as well as all corrected decks are processed. Contrast with full Update mode and quick Update mode.

Old Program Library -
The program library to be modified.

Output File -
The print file generated by Update that contains the status information produced during Update execution. It is in a form suitable for printing.

Program Library -
The file generated by an Update run that contains decks of line images that can be manipulated by Update.

Pullmod File -
A file that contains directives and text or re-created correction sets specified on PULLMOD directives.

Quick Update Mode -
An Update run in which the Q option is selected on the control statement. Only decks specified on COMPILE directives and called common decks are processed. Contrast with full Update mode and normal selective mode.

Secondary Old Program Library -
A program library from which decks on the old program library can call common decks.

Sequence Number -
A number supplied by Update that uniquely identifies a line image.

Source File -
An optional file generated by Update that uses line images of an input stream to generate a new program library.

System-Logical Record -
Under NOS/BE, a data grouping that consists of one or more PRUs terminated by a short PRU or zero-length PRU. These records can be transferred between devices without loss of structure. Equivalent to a logical record under NOS. Table C-1 shows equivalency under SCOPE 2.

TABLE C-1. RECORD TYPE UNDER SCOPE 2

Type	Level	Equivalency
RT=W	0 thru 16_8	end-of-section
RT=W	17_8	end-of-partition
RT=S	0 thru 17_8	end-of-record
RT=Z	0 thru 17_8	end-of-section
BT=C	0 thru 17_8	end-of-section

Unknown -
The status of a deck name or identifier that is not on the old program library. A deck name or identifier that is purged has the status of unknown.

FILE FORMAT AND STRUCTURE D

The files generated and used by Update have formats determined by both the operating system in use and the user. This appendix describes default file formats, allowed file formats, and the interchangeability of files among operating systems. Table D-1 summarizes file structure according to the operating system used.

LIBRARY FILE FORMATS

Update can create and maintain library files in two distinctly different formats: random and sequential. These formats are described in detail below. Random format should be used whenever possible because it can be processed substantially faster than sequential format.

RANDOM FORMAT

On a random format library, each deck is a system-logical record as shown in figure D-1. The deck records are followed by separate records containing the deck list, the directory, and the random index.

Random Index

The random index tells Update the beginning point and length of the directory and the deck list. The index also contains such information as the master control character and the character set used when the library was generated. Random index format is shown in figure D-2.

Two copies of the random index are generated under SCOPE 2 because Update generates another copy when it closes the file. The closing of the file is a process internal to Update.

Under SCOPE 2, Update adds a 2-word header to the random index that indicates the number of words in the index. SCOPE 2 header format is shown in figure D-3.

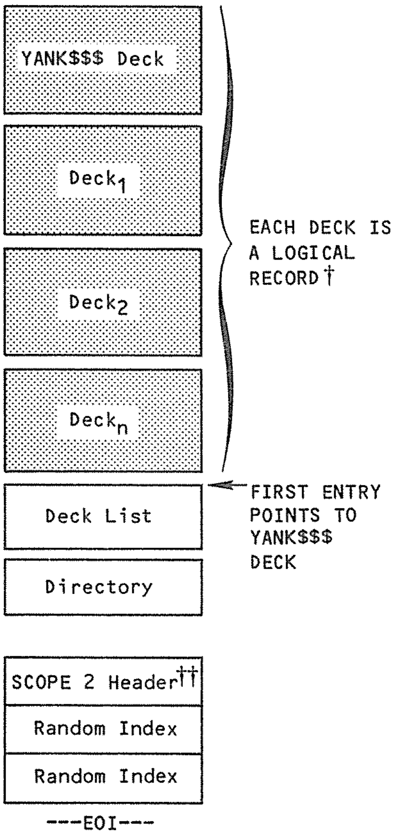
Copying to Tape

Random program libraries should be copied to tape through Update parameters. To copy a random program library to tape under NOS or NOS/BE, use the UPDATE control statement:

```
    UPDATE(B,P=plname,N=lfn)
```

where plname is the library name and lfn is the tape file. To copy the library back to mass storage, use:

```
    UPDATE(A,P=lfn,N=newpl)
```



†For SCOPE 2, each deck is a section.
††Header applies to SCOPE 2 only.

Figure D-1. Random Program Library Format

where lfn is the tape file and newpl is the new program library name.

Under SCOPE 2, use the UPDATE control statement:

```
    UPDATE(F,P=plname,N=lfn)
```

to copy a random program library to tape. The program library name is plname and lfn is the tape file. To copy the library back to mass storage, use:

```
    UPDATE(F,P=lfn,N=newpl)
```

where lfn is the tape file and newpl is the new program library name.

TABLE D-1. FILE STRUCTURE VERSUS OPERATING SYSTEM

Update File	NOS/BE		NOS		SCOPE 2	
	Tape	Mass Storage	Tape	Mass Storage	Tape	Mass Storage
P=OLDPL	Binary SI tape	Random or sequential	Binary SI tape or I tape	Random or sequential	Binary, sequential[†] RT=W or S	Random: RT=W, unblocked Sequential: RT=W, unblocked RT=W
N=NEWPL	Binary SI tape	Random or W - sequential	Binary SI tape or I tape	Random W - sequential	Binary, sequential RT=W or S	Random if unblocked Sequential if blocked or if W specified on Update control statement. RT=W,unblocked by default. RT=blocked W or S; specified through FILE control statement. Cannot be blocked if random.
C=COMPILE	Binary	Sequential RT=Z	Binary	Sequential RT=Z	RT=W,I blocked. Other types (F or Z only) determined by FILE control statement.	RT=W,unblocked. Other types (F or Z only) determined by FILE control statement.
I=INPUT	Binary	Sequential RT=Z	Binary	Sequential RT=Z	RT=W,I blocked. Other blocking or RT=Z, FL ≤ 256 through FILE control statement.	RT=W,unblocked RT=W blocked or RT=Z, FL ≤ 256 through FILE control statement.
O=OUTPUT	Binary	Sequential RT=Z	Binary	Sequential RT=Z	RT=W,I blocked. Other types possible through FILE control statement.	RT=W,unblocked
S=SOURCE	Binary	Sequential RT=Z	Binary	Sequential RT=Z	RT=W,I blocked. Other blocking or RT=Z, FL ≤ 256 through FILE control statement.	RT=W,unblocked RT=W blocked or RT=Z if specified through FILE control statement.
*READ lfn or *ADDFILE lfn	Binary	Sequential RT=Z	Determined by REQUEST or LABEL control statement	Sequential RT=Z	RT=W,I blocked. Other blocking or RT=Z, FL ≤ 256 through FILE control statement.	RT=W,unblocked RT=W blocked or RT=Z if specified through FILE control statement.

[†] Random files can be put on tape by copying the file to tape. To access this file, it must first be copied to a W unblocked file. W records are 5120 characters in length. SCOPE 2 Update checks for presence of directory header containing DIRECT$ to identify random file and for presence of CHECK in word 1 of sequential file. If both tests fail, library format is unacceptable. Random format library must be unblocked W records.

NOTE

Update uses 7000 record manager for I/O, but Update does not use 6000 record manager (BAM) Basic Access Method. A FILE control statement can be used with SCOPE, but this control statement is ignored under NOS and NOS/BE.

<table>
<tr><th>59</th><th>47</th><th>29</th><th>24</th><th>23</th><th>17</th><th>11</th><th>5 0</th></tr>
<tr><td>7000</td><td>dll</td><td colspan="6">dllra</td></tr>
<tr><td>unused</td><td>dirl</td><td colspan="6">dirra</td></tr>
<tr><td colspan="3">unused</td><td>m</td><td>x</td><td>lab</td><td>y</td><td>c</td></tr>
<tr><td colspan="8">label</td></tr>
<tr><td colspan="8">label (contd)</td></tr>
</table>

7000 Identifies random directory record.

dll Length of the deck list in words.

dllra Random address of first word of deck list.

dirl Length of directory in words.

dirra Random address of first word of directory.

m Indicates presence of deck bits in deck list

1	Deck bits present.
other	Deck bits not present.

x Character set identifier determined by IP.CSET parameter.

3 (36_8)	IP.CSET is set for a 63-character set.
4 (37_8)	IP.CSET is set for a 64-character set.
7 (42_8)	IP.CSET is set for 63-character set plus ASCII.
8 (43_8)	IP.CSET is set for 64-character set plus ASCII.

lab Label flag:

nonzero	Words 3 and 4 contain tape label.
0	Words 3 and 4 not present.

SCOPE 2 does not recognize tape labels.

y Indicates which character set was used when the library was generated.

Y or null	64-character set used.
other	63-character set used.

c Indicates master control character in use when the library was created.

Figure D-2. Random Index Format

<table>
<tr><th>59</th><th>17 0</th></tr>
<tr><td>DIRECT$</td><td>unused</td></tr>
<tr><td colspan="2">n</td></tr>
</table>

n Number of words in the random index.

Figure D-3. SCOPE 2 Random Index Header Format

SEQUENTIAL FORMAT

Update optionally creates new program libraries in sequential format. On magnetic tape, a sequential library (I tape format on NOS, SI tape format on NOS/BE, or RT=S on SCOPE) is written as one record in binary (figure D-4). The first word in the file is a display code key word (figure D-5); the second is a counter word containing the number of deck names in the deck list and the count of correction set identifiers in the directory (figure D-6). The last word in the file is a checksum (figure D-7).

YANK$$$ DECK

The YANK$$$ deck is automatically created on a creation run as the first deck on the program library. It does not have a DECK line as its first line image. On correction runs, Update inserts into the YANK$$$ deck any YANK, SELYANK, YANKDECK, and DEFINE directives that it encounters during the read-input-stream phase. These directives acquire identification and sequence information from the correction set from which they originate. On a merge, the two YANK$$$ decks are merged into a single deck.

Although the YANK$$$ deck as a whole cannot be yanked or purged, lines in the deck can be deleted, yanked, or purged. If information other than the four directive types mentioned inadvertently gets into the YANK$$$ deck, this information can be purged through the E option on the Update control statement or through the SELYANK directive. The YANK$$$ deck is maintained in display code.

DECK LIST

The deck list is a table that contains an entry for each deck on the program library. Each entry on a sequential program library consists of one word containing the deck name; bit three is reserved for the deck bit that indicates whether or not the deck is a common deck. Each deck list entry on a random program library consists of two words as shown in figure D-8. The deck list is maintained in display code.

DIRECTORY

The directory is a table that contains one entry for each DECK, COMDECK, and IDENT that has ever been used for this library. Directory entries each consist of one word containing the 1 through 9 character identifier in display code, left-justified with zero fill. Correction set identifiers and deck names are listed chronologically as they are introduced into the library. The directory is maintained in display code.

A deck name that has been purged remains in the table although it is not printed on the listable output file. The purged deck names are not removed from the table unless the E (edit) parameter is specified on the Update control statement.

The number of identifiers in the directory is limited by the amount of central memory (or small core memory) available.

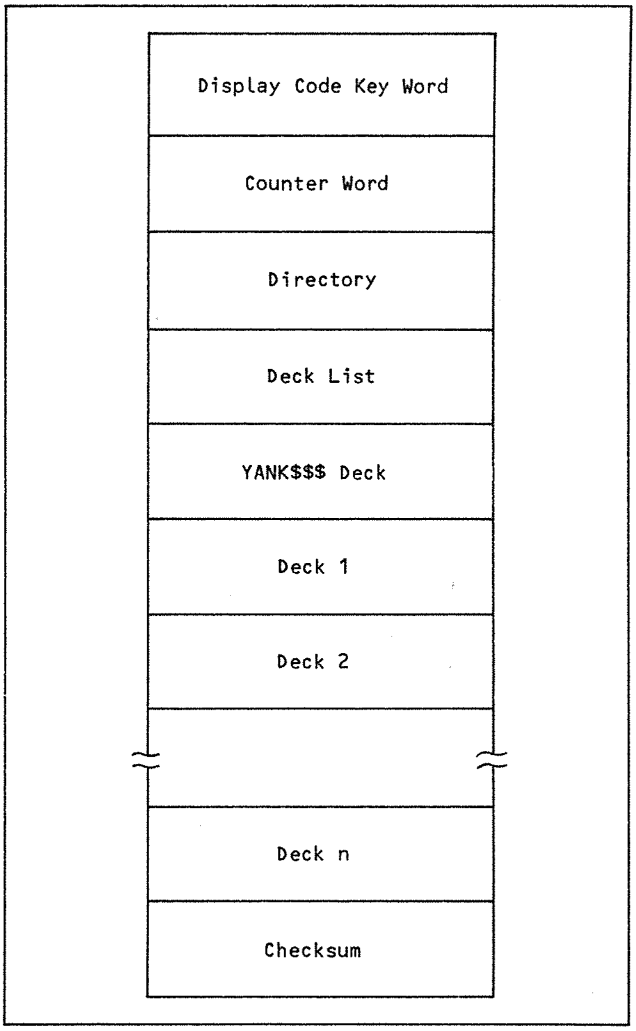


Figure D-4. Sequential Program Library Format

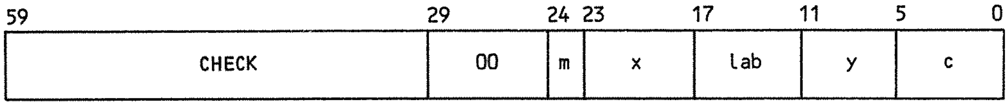
Each directory entry has the format shown in figure D-9. For a purged identifier, bits 59 through 6 are zeros, and bits 5 through 0 contain a 20_8.

COMPRESSED TEXT FORMAT

Text is an indefinite number of words that contain a correction history and the compressed image of each line in the deck. Information for each line is in the format shown in figure D-10.

OLD SEQUENTIAL FORMAT

Update accepts library files in the old (pre-SCOPE 3.4) Update sequential format as shown in figure D-11. These libraries resemble the new sequential format but do not contain the CHECK word or checksum, and the text format and correction history bytes are different. Word 2 on the new format is the same as word 1 on the old format. Update no longer generates this obsolete sequential format.



CHECK Identifies the file as being a sequential file.

m Indicates presence of deck bits in deck list:

1 Deck bits present.
other Deck bits not present.

x Character set identifier determined by IP.CSET parameter:

3 (36_8) IP.CSET is set for a 63-character set.
4 (37_8) IP.CSET is set for a 64-character set.
7 (42_8) IP.CSET is set for 63-character set plus ASCII.
8 (43_8) IP.CSET is set for 64-character set plus ASCII.

lab Label flag:

L Indicates labeled tape.
null Indicates unlabeled tape.

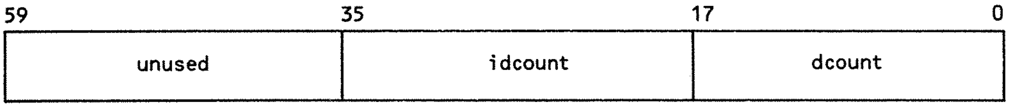
SCOPE 2 does not recognize tape labels.

y Indicates which character set used when the library was generated:

Y or null 64-character set used.
other 63-character set used.

c Indicates master control character in use when the library was created.

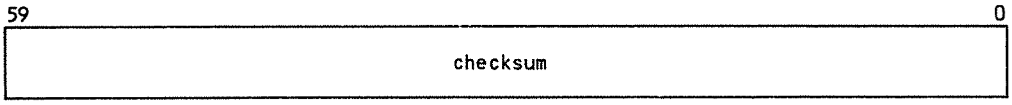
Figure D-5. Display Code Key Word Format



idcount Number of identifiers in the directory.

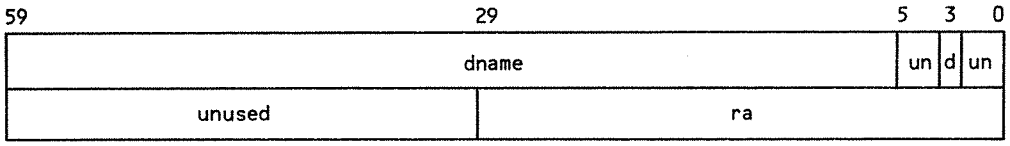
dcount Number of deck names in the deck list.

Figure D-6. Counter Word Format



checksum Count of bits in the program library.

Figure D-7. Checksum Format



dname — 1 through 9 alphanumeric character deck name obtained from DECK or COMDECK directive when deck was placed on library. The first dname is YANK$$$.

un — Unused.

d — Deck bit. Indicates kind of deck.

    0 Common deck.
    1 Regular deck.

ra — Random address of first word of compressed text for the deck.

Figure D-8. Random Program Library Deck List Format

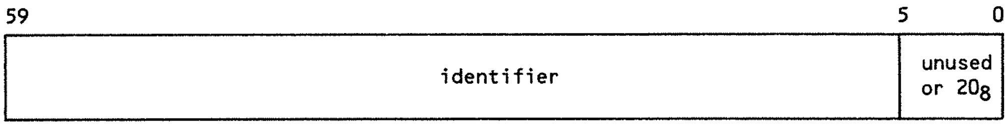
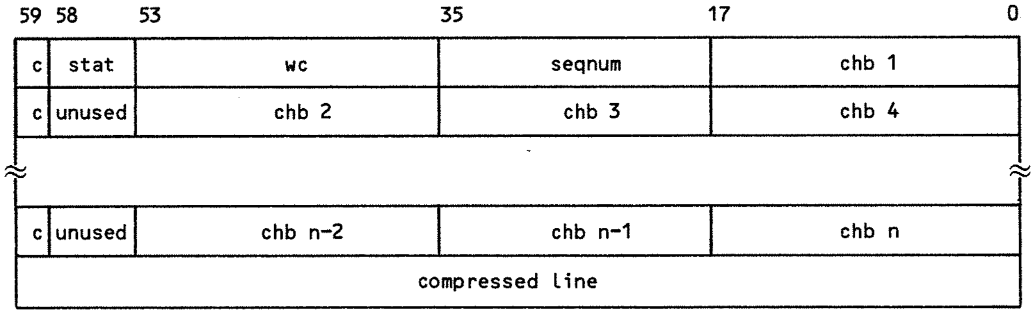


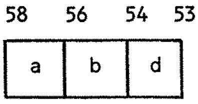
Figure D-9. Directory Format



c — Correction history byte flag. Indicates the last word containing correction history bytes.

    0 Not last word.
    1 Last word.

Figure D-10. Compressed Text Format on Program Library (Sheet 1 of 3) stat — Line status:



a — Activity bit:

0 Line is inactive.
1 Line is active.

b — Character set mode:

0 Character set is display code.
1 Character set is ASCII.

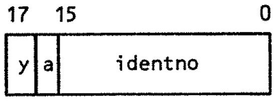
d — Yank deck indicator (*DECK directive only):

0 Deck not yanked.
1 Deck yanked.

wc — Number of words of compressed text for this line, excluding words containing correction history bytes.

seqnum — Sequence number of line (octal) according to position in deck or correction set identified by chb 1.

chb_i — Correction history byte. Update creates a byte for each correction set that changes the status of the line. The format of chb is:



y — Yank bit:

0 Line not yanked.
1 Line has been yanked.

a — Activity bit:

0 Correction set deactivated the line.
1 Correction set activated the line.

identno — Index to the entry in the directory that contains the name of the correction set or deck that introduced the line or changed the line status.

Compressed line in display code — The compressed image of the line in display code. Single and double spaces are unaltered. Three or more embedded spaces are replaced in the image as follows:

3	spaces replaced by 0002_8
4	spaces replaced by 0003_8
5	spaces replaced by 0004_8
.	. .
.	. .
.	. .
64	spaces replaced by 0077_8
65	spaces replaced by 007755_8
66	spaces replaced by 00775555_8
67	spaces replaced by 00770002_8, etc.

Trailing spaces are not considered as embedded and are not included in the line image. A 4-digit octal code 0000 or word count (wc) reached marks the end of the line. This is conditional on the CHAR64 option.

Figure D-10. Compressed Text Format on Program Library (Sheet 2 of 3)

When the full-character set installation option is assembled, a byte of 0001 represents a colon.

Compressed line in ASCII code

The compressed image of the line in ASCII. One or more spaces are replaced in the image as follows:

1	space replaced by 040_8
2	spaces replaced by 001_8
3	spaces replaced by 002_8
4	spaces replaced by 003_8
⋮	⋮
31	spaces replaced by 036_8
32	spaces replaced by 037_8 040_8
33	spaces replaced by 037_8 041_8
000_8 (NUL)	replaced by 037_8 000_8
001_8 (SH)	replaced by 037_8 001_8
⋮	⋮
037_8 (US)	replaced by 037_8 037_8

Compressed ASCII characters are stored as 7-1/2 eight-bit characters per 60-bit word, with multiple blanks compressed and trailing blanks removed. A four-digit octal code 0000_8 marks the end of the line, if the code occurs before the end of the last word is reached. Only the lower 8 bits of each 12-bit byte are saved; the upper 4 bits are ignored, unless expanding a compressed line image. When expanding an ASCII compressed line image, the upper 4 bits of each character are set to zero, unless the character is NUL (000_8). If the character is NUL, the 12-bit value 4000_8 is returned. Characters in the range 041_8 to 377_8 are stored unchanged.

Figure D-10. Compressed Text Format on Program Library (Sheet 3 of 3)

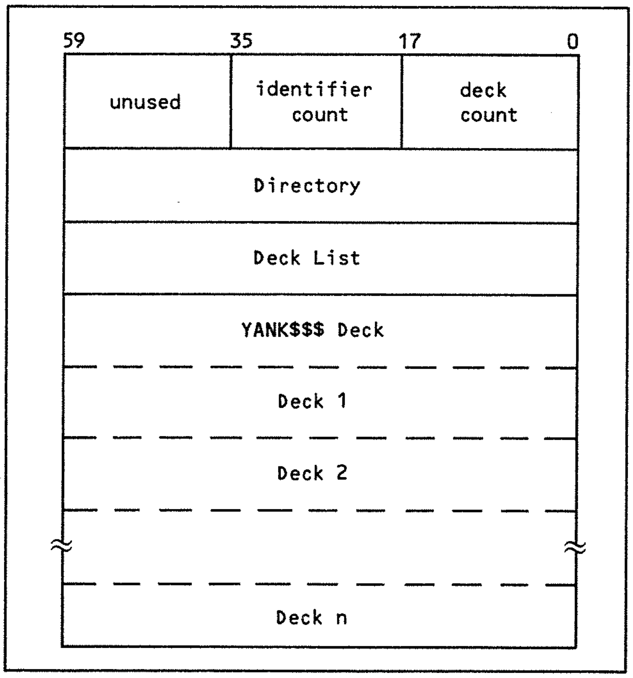


Figure D-11. Old Sequential Program Library Format

INTERCHANGEABILITY OF LIBRARIES

When the random format libraries have been copied to tapes, the libraries have limited interchangeability among the operating systems. This interchangeability is shown in table D-2.

The control statements COPY, COPYBF, COPYBR, COPYCF, or COPYCR should not be used on random access files on NOS/BE or on SCOPE since these operating systems might not recognize that the copied file is a random access file.

Sequential program libraries are interchangeable among operating systems when they are system-logical records (Record Manager type S records).

COMPILE FILE FORMAT

Through control statement parameters, the user can specify whether the text on the compile file is to be compressed or expanded, and sequenced or unsequenced. The expanded compile file format for each line consists of 72 or 80 columns of data followed by 0 to 18 columns of sequence information. The maximum size of a line image is 90 columns.

TABLE D-2. FILE INTERCHANGEABILITY

System That Generated Random Library on Tape	System to Read Random Library From Tape[†]		
	NOS	NOS/BE	SCOPE 2
NOS	Yes	No	No
NOS/BE	Yes	Yes	No[††]
SCOPE	No	No	Yes

[†] A yes indicates the tape can be read; a no indicates it cannot.

[††] Must be copied to unblocked mass storage file when read in.

Update attempts to place sequence information in the columns remaining in the line image after the data columns have been allocated. When the data field is 72 and the line image is 90 columns, column 73 is blank and 17 columns are available for sequencing information. In this case, the 1 to 9 character identifier is left-justified in column 74, and the sequence number is right-justified in column 86.

When the data field is 72 and the line image is 80 columns, 8 columns are available for sequencing information. If the data field is 80 and the line image is 90, 10 columns are available for sequencing information. In either of these cases, if the identifier and sequence number exceed the field, Update truncates the least significant (right-most) characters of the identifier leaving the sequence number intact.

If the data field and line image are both 80, the compile file output cannot have sequence information appended.

The width statement overrides the values specified by D and 8. The table D-3 shows the equivalence of the D and 8 parameter options to the *WIDTH directive.

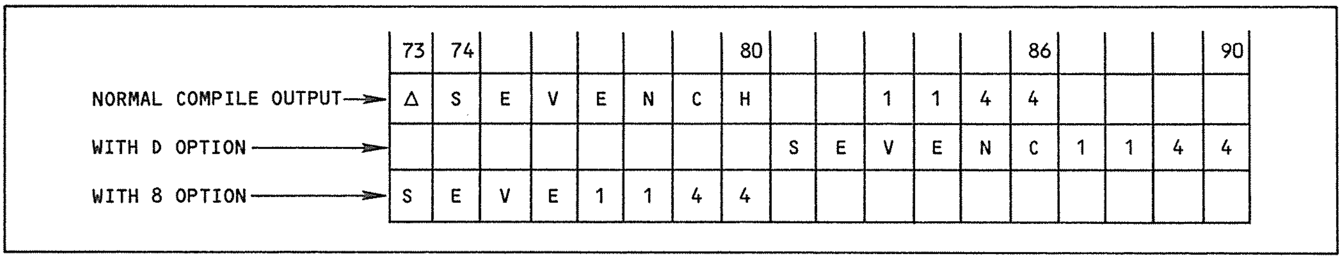
TABLE D-3. WIDTH DIRECTIVE EQUIVALENCE TO D AND 8 OPTIONS

D and 8 Options	*WIDTH Equivalent
neither D nor 8 option	*WIDTH 72,14
D option	*WIDTH 80,10
8 option	*WIDTH 72,8
D,8 option	*WIDTH 80,0

The examples in figure D-12 show how Update positions sequencing information for the various control statement options.

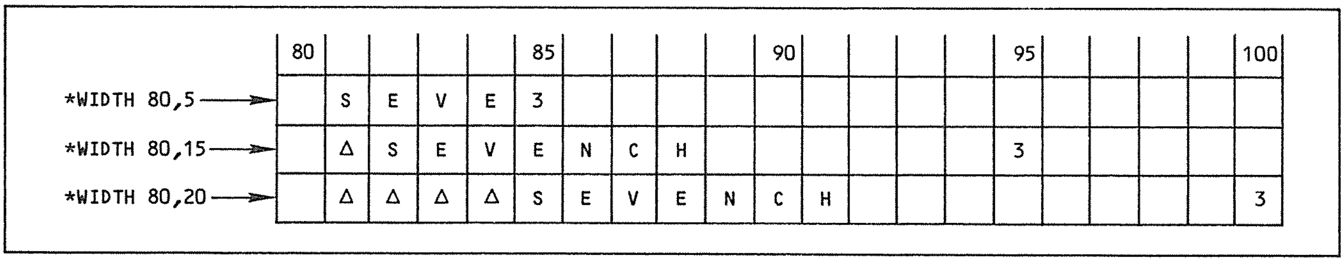
In addition, figure D-13 shows possible widths of the identification field and the positioning of the identifier name and sequence number. The total length possible for the identification field is 17. If the identification field length is larger than 17, extra blanks will be inserted between the sequence fields.

If the 80- or 90-character line image on the compile file has two blanks as the last two characters, these are converted to a 0000 line terminator and the line image is 8 (or 9) words long. If the last two columns do not con-



	73	74						80						86				90
NORMAL COMPILE OUTPUT →	Δ	S	E	V	E	N	C	H			1	1	4	4				
WITH D OPTION →									S	E	V	E	N	C	1	1	4	4
WITH 8 OPTION →	S	E	V	E	1	1	4	4										

Figure D-12. Sequencing Format for Compile File

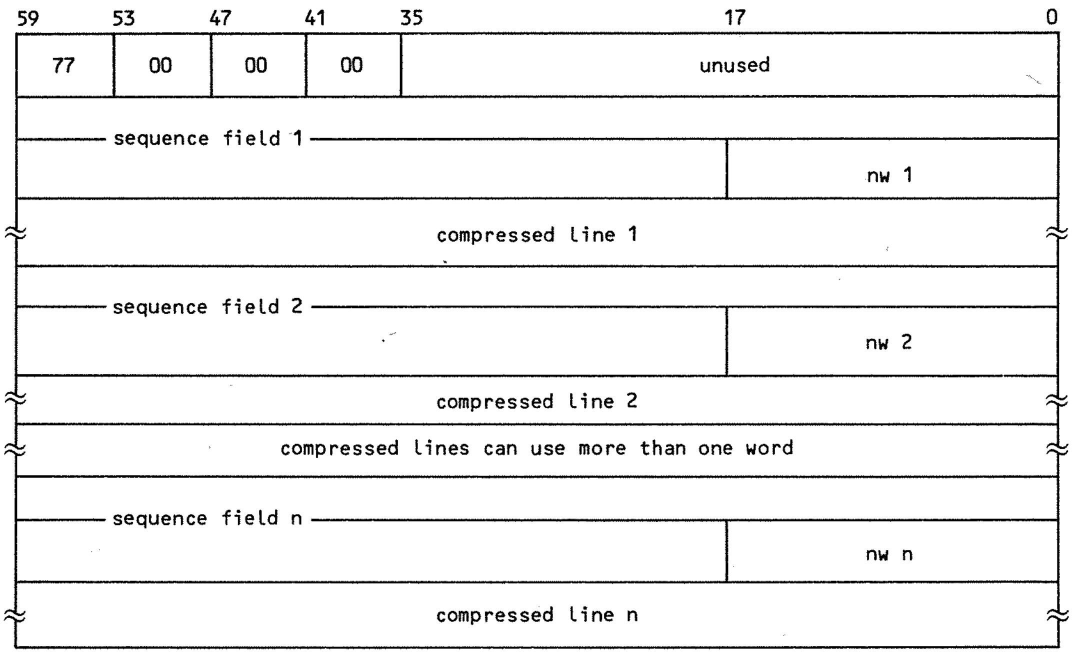


	80					85					90					95					100
*WIDTH 80,5 →		S	E	V	E	3															
*WIDTH 80,15 →		Δ	S	E	V	E	N	C	H							3					
*WIDTH 80,20 →		Δ	Δ	Δ	Δ	S	E	V	E	N	C	H									3

Figure D-13. Sequence Number Overlay

tain blanks, a word containing 8 blanks and a zero-byte line terminator are added, thus making the line image 9 (or 10) words long. This same procedure is used for creation of the source file.

The format of the compressed compile file is shown in figure D-14. The first word is a loader prefix table (77_8). Compressed format is generated through the X option on the UPDATE control statement.



sequence $field_i$ — 17 characters comprising card columns 74 through 90. Column 73 is always blank.

nw_i — Binary number of words in compressed line.

compressed $line_i$ — Each 00 character is replaced by the 12-bit value 0001, and three or more consecutive blanks (to a maximum of 64) are replaced by a 12-bit value 0002 through 0077_8. A single blank is represented in display code (55_8); two consecutive blanks are represented by the 12-bit value 5555_8. If the last word is not full, it is padded on the right with binary zeros. Because word count nw is present, an extra all-zero word is not required to guarantee 12 zero bits. *WIDTH directives are ignored with compressed compile files. The full line image is always present, and the sequence field information is always a full 17 characters.

Figure D-14. Compile File Compressed Format

INDEX

COMMENT SHEET

MANUAL TITLE: Update Version 1 Reference Manual

PUBLICATION NO.: 60449900

REVISION: F

This form is not intended to be used as an order blank. Control Data Corporation welcomes your evaluation of this manual. Please indicate any errors, suggested additions or deletions, or general comments on the back (please include page number references).

_____ Please reply _____ No reply necessary

CUT ALONG LINE

FOLD FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 8241 MINNEAPOLIS, MINN.

POSTAGE WILL BE PAID BY
CONTROL DATA CORPORATION
Publications and Graphics Division
P.O. BOX 3492
Sunnyvale, California 94088-3492

FOLD FOLD

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.
FOLD ON DOTTED LINES AND TAPE

NAME:

COMPANY:

STREET ADDRESS:

CITY/STATE/ZIP:

TAPE TAPE

UPDATE CONTROL STATEMENT PARAMETERS

UPDATE(p1,p2,...,pn)

A Sequential-to-Random Copy

omitted	no copy
A	copy

B Random-to-Sequential Copy

omitted	no copy
B	copy

C Compile File Name

omitted or C or C6 or C8	COMPILE
C=lfn or C6=lfn or C8=lfn	lfn
C=PUNCH	PUNCH
C=0	none

D Data Width On Compile File

omitted	72 columns
D	80 columns

E Edit Old Program Library

omitted	no editing
E	editing

F Full Update Mode

omitted	normal selective mode
F	full mode

G Pullmod File Name

omitted	source file
G=lfn or G6=lfn or G8=lfn	lfn

H Character Set Change

omitted or H	default set
H=3	63
H=4	64

I Input Stream File Name

omitted or I or I6 or I8	INPUT
I=lfn or I6=lfn or I8=lfn	lfn

K Compile File Sequence

omitted	C parameter determines deck location
K or K6 or K8	COMPILE directive sequence on file COMPILE
K=lfn or K6=lfn or K8=lfn	COMPILE directive sequence on file lfn

L Listable Output Options

omitted	creation run: A, 1, 2 correction run: A, 1, 2, 3, 4 copy run: A, 1 terminal output: 1
L=0	suppress listing
L=c...c	options 1 thru 9, A or F

M Merge Program Libraries

omitted	no merge
M or M6 or M8	MERGE
M=lfn	lfn

N New Program Library File Name

omitted	NEWPL; suppress if correction run
N or N6 or N8	NEWPL
N=lfn or N6=lfn or N8=lfn	lfn

O Listable Output File Name

omitted or O or O6 or O8	OUTPUT
O=lfn or O6=lfn or O8=lfn	lfn

P Old Program Library File Name

omitted or P or P6 or P8	OLDPL
P=lfn	lfn
P=lfn/s1/s2/...	lfn; secondaries on si
P=/s1/s2...	OLDPL; secondaries on si

Q Quick Update Mode

omitted	normal selective mode
Q	quick mode

R Rewind Files

omitted	rewind files
R	no rewinding
R=c...c	rewind specified files (C, N, P, S)

S Source File Name

omitted	none
S or S6 or S8	SOURCE
S=lfn or S6=lfn or S8=lfn	lfn

T Omit Common Decks From Source File

omitted	none
T or T6 or T8	SOURCE
T=lfn or T6=lfn or T8=lfn	lfn

U Debug Help

omitted	fatal error ends execution
U	fatal errors do not end execution

W Sequential New Program Library Format

omitted	random if possible
W	sequential

X Compressed Compile File

omitted	not in compressed format
X	in compressed format

8 Line Image Width On Compile File

omitted	90 columns
8	80 columns

* Master Control Character

omitted	*
*=c	c

/ Comment Control Character

omitted	/
/=c	c

CORPORATE HEADQUARTERS, P.O. BOX 0, MINNEAPOLIS, MINN. 55440
SALES OFFICES AND SERVICE CENTERS IN MAJOR CITIES THROUGHOUT THE WORLD

LITHO IN U.S.A.

CONTROL DATA CORPORATION

To list COMMON decks
from an UPDATE library

```
ATTACH(OLDPL, — — —
UPDATE(Q)
~~REWIND(COMPILE)~~
~~CEB~~ WHILE,.NOT.(FILE(COMPILE,EOI)),ENDLST.
     COPYSP(COMPILE)
   ENDW,ENDLST.

*ADDFILE
*DECK NULL
*CALL DOKUMENT1
*WEOR
 ALL GRIDDOC
*WEOR
*CALL ARRAYDOC
```

360 ↑

3803

UPDATE notes.

1. p34. For short form directives, the assumed id or deck name can be that given earlier on the same directive.
e.g ~~*ID.~~ *D $id_1.n_1$
*D $id_2.n_2, n_3$ → *D $id_2.n_2, id_2.n_3$
*copy dname, n_1 ⇒ *copy dname, dname.n_1

2. A source file contains only active cards
Pullmod files contain all cards from correction set (except, of course, comments).

3. Always put *YANK id and *YANKDECK dn directives with only one id or dname per directive, for easier restoration later.

4. Update would be improved if there was a dummy card ident meaning 1st and last cards in deck, e.g. *D id_1, *

1981-04-15 Rev C.

1982-07-20 Rev D

1982-08-11 , Source file has 80 or 81 character records by default.
– How can original record length be preserved.

1983 Mar 29 Rev E
1984 Dec 12 Rev F

17.1.1979

Update

1) Is the following legal?

*ADDFILE INPUT, dname

*DECK D1

test

*COPY $_4$ $dname_2$, cid_1, cid_2

! Yes — hooray!

2) Shortened card ids

~~*IDENT TEST~~

~~*D FILFLUX.A6~~ F ~~FLUX~~

~~*D~~

*IDENT id_1

*I $id_1.n_1$

*I $id_2.n_2$, n_3

*COPY dd

*IDENT id_1

*I $dd_1.n_1$

*I $dd_2.n_2$, $.n_3$

3) L=A434 ~~[illegible]~~ are input directives listed.

*IDENT TEST

*D FLUX.~~[illegible]~~3

*D FILFLUX.6, 7

*IDENT TEST

*D ~~FLUX.3~~ EQWAVE.3

*D ~~FLUX~~ FLUX.6, .7