CYBERNET® SERVICES

UPDATE

Reference Manual

CONTROL DATA

UPDATE DIRECTIVES INDEX

Directive Format		Abbreviation	Section
*ABBREV		none	2.8.2
*ADDFILE	file, c	*AF	2.4.14
*ADDFILE	file, dname	*AF	2.4.14
*BEFORE	c	*B	2.4.3
*CALL	dname	*CA	2.7.3
*CHANGE	$idname_1$, $idname_2$, $idname_3$, $idname_4$, ..., $idname_{n-1}$, $idname_n$	*CH	2.4.7
*COMDECK	dname	*CD	2.3.2
*COMDECK	dname, NOPROP	*CD	2.3.2
*COMPILE	$dname_a$. $dname_b$	*C	2.6
*COMPILE	$dname_1$, $dname_2$, ..., $dname_n$	*C	2.6
*COPY	dname, c	*CY	2.4.6
*COPY	dname, c_a, c_b	*CY	2.4.6
*COPY	dname, c_a, c_b, file	*CY	2.4.6
*CWEOR	n	*CW	2.7.2
*DECK	dname	*DK	2.3.1
*DECLARE	dname	*DC	2.8.7
*DEFINE	$name_1$, $name_2$, ..., $name_n$	*DF	2.8.5
*DELETE	c_a, c_b	*D	2.4.4
*DELETE	c	*D	2.4.4
*DO	$idname_1$, $idname_2$, ..., $idname_n$	none	2.7.7
*DONT	$idname_1$, $idname_2$, ..., $idname_n$	*DT	2.7.7
*END		none	2.8.8
*ENDIF		*EI	2.7.5
*ENDTEXT		*ET	2.7.6
*IDENT	idname, p_1, p_2, ..., p_n	*ID	2.4.1
*IF	type, name, num	none	2.7.4
*IF	-type, name, num	none	2.7.4
*INSERT	c	*I	2.4.2
*LIMIT	n	*LT	2.8.3
*LIST		*L	2.8.4
*MOVE	$dname_1$, $dname_2$	*M	2.4.16
*NOABBREV		*NA	2.8.2
*NOLIST		*NL	2.8.4
*PULLMOD	$idname_1$, $idname_2$, ..., $idname_n$	*PM	2.8.6
*PURDECK	$dname_1$, $dname_2$, ..., $dname_n$	*PD	2.4.12
*PURDECK	$dname_a$. $dname_b$	*PD	2.4.12
*PURGE	$idname_1$, $idname_2$, ..., $idname_n$	*P	2.4.11
*PURGE	$idname_a$. $idname_b$	*P	2.4.11
*PURGE	idname, *	*P	2.4.11
*READ	file	*RD	2.5.1
*RESTORE	c	*R	2.4.5
*RESTORE	c_a, c_b	*R	2.4.5
*REWIND	file	*RW	2.5.3
*SELPURGE	$dname_1$. $idname_1$, $dname_2$. $idname_2$, ..., $dname_n$, $idname_n$	*SP	2.4.13
*SELYANK	$dname_1$. $idname_1$, $dname_2$. $idname_2$, ..., $dname_n$, $idname_n$	*SY	2.4.10
*SEQUENCE	$dname_1$, $dname_2$, ..., $dname_n$	*S	2.4.15
*SEQUENCE	$dname_a$. $dname_b$	*S	2.4.15
*SKIP	file, n	*SK	2.5.2
*TEXT		*T	2.7.6
*WEOR	n	*W	2.7.1
*YANK	$idname_1$, $idname_2$, ..., $idname_n$	*Y	2.4.8
*YANK	$idname_a$. $idname_b$	*Y	2.4.8
*YANKDECK	$dname_1$, $dname_2$, ..., $dname_n$	*YD	2.4.9
*|	comments	none	2.8.1

CYBERNET® SERVICES

UPDATE

Reference Manual

CONTROL DATA

REVISION RECORD

REVISION	DESCRIPTION
A	Initial printing
(04-15-76)	
B	To reflect changes in support of the NOS 1 operating system and an upgrading of the Common Products
(10-15-77)	through PSR level 433.

Publication No.
84000016

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

Printed in the United States of America

Address comments concerning this manual to:
CONTROL DATA CORPORATION
Publications and Graphics Division
Data Services Publications
P. O. Box 0 HQC02C
Minneapolis, MN 55440
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	REV
Cover	-
Title Page	-
Revision Record	B
iii/iv	B
v/vi	B
vii, viii	B
ix/x	B
1-1 thru 1-4	A
1-5	B
2-1	A
2-2	B
2-3 thru 2-10	A
2-11	B
2-12 thru 2-15	A
2-16	B
2-17 thru 2-19	A
2-20 thru 2-22	B
2-23, 2-24	A
2-25	B
2-26	A
2-27	B
2-28 thru 2-30	A
3-1, 3-2	A
3-3, 3-4	B
3-5 thru 3-8	A
3-9, 3-10	B
3-11 thru 3-15	A
4-1 thru 4-6	A
4-7, 4-8	B
4-9, 4-10	A
4-11	B
4-12 thru 4-22	A
4-23 thru 4-27	B
A-1	A
B-1, B-2	A
B-3	B
B-4 thru B-8	A

PAGE	REV
C-1, C-2	A
D-1, D-2	A
Index-1, Index-2	A
Index-3, Index-4	A
Comment Sheet	B
Mailer	-
Back Cover	-

PAGE	REV

PAGE	REV

PAGE	REV

PREFACE

This manual is intended primarily for CYBERNET Service users employing either the SCOPE 3.4 or NOS 1 operating systems. Its description of the UPDATE program for maintaining and updating source decks on libraries in compressed symbolic format is, however, applicable under all of the following operating systems and computer systems.

SCOPE 3.4 and NOS 1 for the CONTROL DATA® 6000 Series Computer Systems and for the CONTROL DATA® CYBER 70/Models 72, 73, and 74 Computer Systems.

SCOPE 2.1 for the CONTROL DATA® CYBER 76 Computer System.

NOS 1 for the CONTROL DATA® CYBER 170 Series Computer Systems.

It is assumed that the reader is familiar with the operating system under which his processing is being performed.

Additional manuals which the user may find useful include:

Publication Title	Publication Number
CYBERNET Interactive Service Time-Sharing Usage Reference Manual	84000029
CYBERNET Services BASIC Reference Manual	84000026
CYBERNET Services COBOL 4 Reference Manual	84000002
CYBERNET Services COMPASS 3 Reference Manual	84000003
CYBERNET Services CYBER Common Utilities Reference Manual	84000007
CYBERNET Services CYBER Record Manager Guide for Users of COBOL 4	84000006
CYBERNET Services CYBER Record Manager Guide for Users of FORTRAN Extended	84000005
CYBERNET Services CYBER Record Manager Reference Manual	84000004
CYBERNET Services CYBER Record Manager User Guide	84000020
CYBERNET Services FORM 1 Reference Manual	84000008
CYBERNET Services FORTRAN Extended Debug User Guide	84000010
CYBERNET Services FORTRAN Extended 4 Reference Manual	84000009
CYBERNET Services LOADER Reference Manual	84000014
CYBERNET Services Remote Batch Terminal User Guide	84000025
CYBERNET Services SCOPE 3.4 Reference Manual	84000021
CYBERNET Services SIFT Programming Systems Bulletin	84000012
CYBERNET Services SORT/MERGE 4 Reference Manual	84000015

CONTENTS

FIGURES

INTRODUCTION 1

UPDATE provides a means of maintaining source decks in conveniently updatable compressed format.

By using UPDATE, a user initially transfers a collection of source decks to a file known as a program library. Each card of each deck is assigned a unique identifier when it is placed on the library. This allows each card to be directly referenced during an UPDATE correction run. During correction runs, cards are inserted into or deleted from the program library according to sequence identification. However, the image of a card, even though deleted, is maintained permanently on the program library with its current status (active or inactive) and a chronological history of modifications to the status. If the history lists the card as being currently inactive, the card has been deleted and is, in effect, removed from the deck. If the status of the card is active, the card is in the deck; either it has never been deleted or has been deleted and restored. During a single UPDATE correction run, a card may undergo one or more modifications or no modifications.

If the user wishes to permanently and irrevocably remove cards from the program library, he can do so through UPDATE purge features. Once a purge has been performed on a program library, it cannot be restored to an earlier level. A set of corrections also has an identifier associated with it. Any cards affected by the correction set can be referred to, relative to the correction set. In later correction runs, all or part of a correction set can be removed (yanked). Yanking differs from purging in that it is a logical operation. The effects of a yank can be reversed.

With UPDATE directives and control card options, the user directs the process of creating a program library, correcting it, and copying the updated programs to a compile file for subsequent use by assemblers and compilers.

The compile file is a primary output of an UPDATE run and contains only the active cards of non-common decks requested by the user. A typical application is for a user to call UPDATE to update a FORTRAN source language deck maintained on an UPDATE program library, request that the modified decks be written in source language format onto the compile file, and then, in the same job call the FORTRAN compiler to read source input from the compile file.

A second type of output is a new program library. This contains updated decks requested by the user in program library format for use as an old program library in subsequent UPDATE runs. This is a required form of output during an UPDATE creation run. It may become an old program library on subsequent correction runs.

A third type of UPDATE output is in the form of an UPDATE source file. This file resembles the decks originally used to create the program library. It contains active source cards of the decks and common decks taken from the updated program library. The source file provides a means of obtaining a back-up copy of the library, of purging all inactive cards, and of resequencing the library.

A source deck can be assigned common status at the time it is first incorporated into a program library file. Common decks can be called from within other decks as they are being written on the compile file. On the compile file, UPDATE replaces the card calling a common deck with a copy of the deck provided that the call occurs within a deck or within a common deck called within a deck.

Source decks can be added to a program library during a creation UPDATE run or during a correction run provided that all common source decks precede any source decks in which they are called.

UPDATE permits two program libraries to be merged. In this mode, UPDATE alters any deck on the merge file having a name that duplicates a name already on the primary program library by assigning it a deck name that is unique.

1.1 FEATURES

Features of UPDATE include:

- Creation of a program library from source decks.
- Copying of old program libraries from sequential to random format and vice versa.
- Merging of two program library files.
- Updating of source decks by inserting, deleting, and restoring cards according to sequence in the deck or according to correction set.
- Ability to completely and permanently remove correction sets from the program library.
- Generation of a compile file containing corrected output acceptable as input to other processing programs, such as compilers and assemblers. Contents of the compile file and subdivision of the compile file into logical records and files is controlled through UPDATE directives. This file can be formatted as 80- or 90-column card images or can be in compressed form.
- Processing of directives, new text, and new source decks from a file other than the job INPUT file.
- Production of fresh source decks from the program library.
- Generation of a new, updated program library.
- Comprehensive list output noting any changes occurring during the run and status of the program library.
- Ability to change the directive card master control character.
- Recognition of abbreviated forms of directives and capability of turning off the search for the abbreviated forms to speed up processing.
- Ability to use full 64-character set, including the colon.
- Checksumming of program library.

1.2 EXECUTION

UPDATE executes as a CPU program on the CONTROL DATA® 6000 Series, 7000 series, CYBER 170 series or CYBER 70 series Computer System under control of the operating system. Certain features (section 1.3) are only available through installation assembly options.

Execution begins when the operationg system interprets an UPDATE call card on the job INPUT file control record, loads the UPDATE program from the system library, and transfers control to it. An UPDATE run is one of three primary types: creation, correction, or copy.

1.2.1 CREATION RUN

Before a user can manipulate and correct a program library, he must create one from a source file (section 3.1). The creation run provides a means of creating a program library.

The following directives can be used prior to the first DECK or COMDECK directive in either a creation run or a correction run:

ABBREV	NOABBREV
DECLARE	NOLIST
END	READ
ENDTEXT	REWIND
LIMIT	SKIP
LIST	TEXT

Any other directive changes the run from a creation run to a correction run. When UPDATE does not encounter any other directives prior to encountering a DECK or COMDECK directive, the run is a creation run and UPDATE ignores the old program library if one is present. The only directives legally encountered during a creation run are DECK, COMDECK, ABBREV, NOABBREV, file manipulation directives and compile file directives (except for DO and DONT).

A creation run consists of one or two phases. If a sequential program library is being created, two phases are required: the read source decks phase and the write program library phase. If a random program library is being created, the program library is written during the first phase.

During the read source phase, UPDATE reads the input stream and creates a scratch file (or a program library in the case of a random program library) and extracts the deck names that are to be used for the generation of the ident directory and the deck list. Each deck is preceded by a DECK directive and each common deck is preceded by a COMDECK directive. If a compile file is desired, it is created as the source decks are read from the input stream. When UPDATE encounters an end-of-record mark in the input stream, it writes the ident directory and the deck list onto the new program library. When UPDATE is creating a sequential program library (section 3.2), the scratch file is copied to the program library (phase two). On subsequent UPDATE runs, each card of each deck on this library can be referred to by deck name and sequence number. The deck directive (DECK or COMDECK) for each deck is always identified as the first card image (sequence number one).

When the library is created, UPDATE generates a deck named YANK$$$ as the first deck on the library. This deck is described further in section 3.2.

A compile file generated during a creation run contains all decks that are on the new program library. This file contains decks to be assembled or compiled.

1.2.2 CORRECTION RUN

UPDATE considers a run as a correction run when it encounters a directive other than one of the following prior to encountering a DECK or COMDECK directive:

ABBREV	NOABBREV
DECLARE	NOLIST
END	READ
ENDTEXT	REWIND
LIMIT	SKIP
LIST	TEXT
	/

A correction run consists of a read input stream phase and a correction phase. During the first phase, UPDATE reads directives and text, adds new decks, and constructs a dictionary of requested correction operations.

During the second phase, UPDATE performs the requested modifications on a deck-by-deck basis.

Correction directives cause card images to be inserted or deleted from program library decks according to card sequence number. A card can be identified by deck name and sequence number or correction set identifier and sequence number. Each new card is assigned a correction set identifier specified by the user. UPDATE sequences the new cards. All cards having the same correction identifier comprise a correction set. UPDATE permits a user to remove (yank) the effects of a correction set (or deck) and later restore the 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 been yanked. This effect on the YANK$$$ deck can be selectively controlled through DO and DONT directives in program library decks.

UPDATE also allows a complete and irreversible purging of correction sets and allows the name of a correction set to be changed.

If a compile file is desired, it is written during the correction phase. 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 end-of-file or end-of-record† marks to be written at the end of decks. The compile file directives can be in original source decks or can be inserted into program library decks during correction runs. These directives are not written on the compile file.

The compile file directives are interpreted when the compile file is written.

---

†For SCOPE 2, end-of-partition and end-of-section marks are written.

1.2.3 COPY RUN

When either the A or B modes are selected on the UPDATE control card (section 4.1.2), the only function of the UPDATE run is to perform the sequential-to-random or random-to-sequential copy of the program library file.

1.3 INSTALLATION OPTIONS

The following UPDATE features are available or unavailable through assembly options (see NOS or SCOPE Installation Handbook).

DECLKEY	Enables DECLARE directive (section 2.8.7)
CHAR64	Supports full 64-character set (see compressed text description)
PMODKEY	Enables PULLMOD card and G option (sections 2.8.6 and 4.1.2)
AUDITKEY	Allows audit functions (section 4.1.2)
EDITKEY	Allows merge and edit (section 4.1.2)
OLDPLKEY	Enables UPDATE to read both old-style and new-style old program libraries (section 3.2)
SCOPE33	Declares that interface is with SCOPE 3.3 or later systems, if SCOPE33 is defined. Otherwise, interface is with earlier versions.
EXTOVLP	Enables detection of four types of overlap involving two or more cards in a correction set. (Appendix A)
DYNAMFL	Declares dynamic table expansion. When this option is assembled, UPDATE automatically expands tables as required and dynamically requests SCOPE to change the user field length to accommodate the additional table area. At the end of the run, the field length is reduced to that requested by the user.

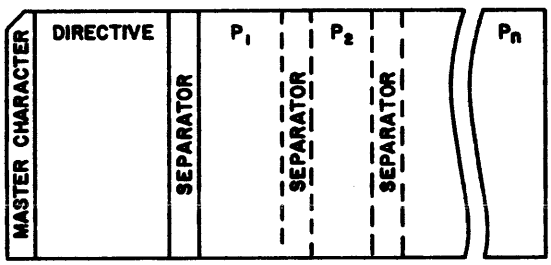
An attempt to use features when the option has not been assembled causes UPDATE to issue error messages. For example, when PMODKEY is not set, the PULLMOD card is not recognized as a legal directive.

DIRECTIVES 2

Directives allow the user to create libraries and extensively control and direct the correction and modification process. Creation directives identify text to be placed on program libraries as named decks or common decks. Correction directives identify the source of text to be inserted, set parameters of the updating process, and inform UPDATE of insertions, deletions, and other corrections. File manipulation directives allow user control of the input files. Compile file directives can be in source decks originally or can be inserted during an UPDATE run. These directives are manipulated much like source cards during the creation, updating, and correction phases but are recognized when the compile file is written.

2.1 DIRECTIVE FORMAT

A directive has the following format:



master control character

The master control character is in column one. It is defined by UPDATE to be an asterisk but may be changed through use of the * parameter on the UPDATE control card (section 4.1.2). Compile file directives are inserted with the character used on the directive. UPDATE recognizes them in a run only if the same master control character as that used on the directive has been specified on the program library header.

In this manual, all directives show asterisk as the conventional rather than the required master control character.

directive name

The directive name field starts in column two. It is terminated by a comma or one or more blanks. A blank is conventional. Most directives have a full name and an abbreviated name, e. g., the abbreviated name for ADDFILE is AF. When the NOABBRV directive is in effect, UPDATE does not recognize the abbrevia ed forms of the directive names.

separator

Parameters are separated by any character other than A through Z, 0 through 9, or + - * / ( ) $ or =. A separator has a display code value of 55_8 or greater. Some directives require specific separators. No embedded blanks are permitted within a parameter. UPDATE searches all 80 columns when interpreting its directives. Therefore, any number of blanks can be between the directive name and the first parameter.

p_i Parameter. Depending on the requirements of the directive, the directive may have no parameters or a number of parameters. Numeric parameter fields are decimal.

NOTE

If columns 73 through 80 contain information (that is, sequencing information from a previous UPDATE or comments), this information is appended to seemingly blank *WEOR, *CWEOR, or *DECLARE directives. Therefore, a null field should be specified on these directives in the following manner: *WEOR,, or *CWEOR,, or *DECLARE,,.

2.2 CARD IDENTIFICATION

The corrections to the library, that is, the newly inserted cards, replaced cards, and deleted cards make up a correction set. The IDENT directive provides a unique name to be assigned to each card inserted by this correction set and each card for which the status is changed. Each change is also assigned a sequence number beginning with one for each IDENT name.

Future corrections affecting these cards can reference these cards by their correction set identifiers.

Card identifiers assigned by UPDATE are usually permanent and can be changed only through use of the SEQUENCE directive (section 2.4.15) and the CHANGE directive (section 2.4.7).

UPDATE recognizes a full form and two short forms of identifiers.

The full form of a card identifier is:

ident.seqnum

ident 1-9 character name of a correction set or deck.

A period terminates the ident.

seqnum Decimal ordinal (1 to 131071) of the card within the correction set or deck. Any character other than 0-9 terminates the sequence number.

The shortened forms of card identifiers can be used on BEFORE, INSERT, DELETE, RESTORE, and COPY directives. Shortened forms are expanded as follows:

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.

In the short form, idname is assumed to be the last explicitly named ident 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 ident given on a BEFORE, INSERT, DELETE, RESTORE, or COPY directive that is known to be a deck name. Both of these default idents are originally set to YANK$$$ so the first directive using a card identifier must use the full form to reset the default.

All deck names are also idents. 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. If there is an explicit reference to a correction set identifier after the explicit reference to the deck name, then 281 would expand to the correction set ident while .281 would expand to EXAMPLE .281

Example:

A is a deck name and B is a correction set on an UPDATE OLDPL.

```
*ID C
*INSERT A.2
<data card>
*INSERT B.1
<data card>
*D  2,3              expands to *DELETE B.2,B.3
*D  4,.5             expands to *DELETE B.2,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 card>
*INSERT A.2
<data card>
*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
```

2.3 DECK IDENTIFYING DIRECTIVES

Each deck to be placed on a program library must be introduced into the system by a DECK or COMDECK directive during a creation or correction run. When UPDATE encounters one of these directives on the input file prior to any correction directive, the run is a creation run. When UPDATE encounters one of these directives while inserting corrections, it terminates the insert and adds the decks to the old program library following the card specified by the INSERT, DELETE, etc.

A deck is terminated by the first occurrence of a DECK or COMDECK card or by an end-of-record card. All intervening cards other than file manipulation directives comprise text. Text cards, when placed on the library, are identified with a deck name and are numerically sequenced starting with 1 for the DECK or COMDECK card, itself. File manipulation directives can be embedded within the text cards. They are valid directives and are not included in the numbering scheme. Text cards introduced as a result of a READ directive are included in the numbering scheme. Any END directives are ignored; they are accepted for compatibility with the SCOPE EDITSYM program only.

Usually, a DECK or COMDECK directive precedes each program or subprogram in a given system. However, more than one subprogram may be included in a deck, as is indicated in the following example:

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

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

Normally, a user groups two programs together if modification of one requires reassembly of both programs.

UPDATE uses the DECK and COMDECK directives while writing the compile file to delimit decks for UPDATE output. This division is meaningful during a correction run when the selective UPDATE mode is employed. Under the selective update mode only decks in which one or more cards have been changed, decks specified on COMPILE directives, and called common decks are included on the compile file. In selecting the cards to be written, UPDATE compares card activities in the current run with those on the old program library. If the status for a card has changed, the deck is considered to have been modified. Any deck that calls a common deck that has changed is also considered to be changed unless the common deck is of the nonpropagating type.

2.3.1 DECK-SOURCE DECK

Formats:

```
*DECK  dname
*DK
```

dname	1-9 character name of deck being introduced; this name must differ from any names already in the deck list. Legal characters are: A-Z 0-9 + - * / ( ) $ =

The DECK directive identifies the beginning of a new source deck. Cards up to the next DECK or COMDECK directive comprise the deck.

2.3.2 COMDECK-COMMON DECK

Formats:

```
*COMDECK  dname, NOPROP
*CD
```

dname	1-9 character name of deck being introduced; this name must differ from any names already in the deck list. Legal characters are: A-Z 0-9 + - * / ( ) $ =
NOPROP	Inclusion of this parameter specifies that if this deck is modified, decks calling this common deck are not to be considered as modified; that is, the effects of the changes are not propagated during UPDATE mode (F option not specified, section 4.1.2).

The COMDECK directive introduces a common deck. These decks are written on a compile file as a result of CALL statements encountered in regular decks while writing the compile file.

For sequential libraries, a common deck should be placed prior to any of the decks calling it. For a random library, it is possible to call a common deck from a deck that precedes the common deck in the deck list. However, to facilitate copying of libraries from random to sequential, the user is advised to place common decks prior to any decks calling them.

In a normal UPDATE run with a random or sequential old program library, a deck calling a common deck being modified that precedes the common deck is not automatically written on the compile file.

2.4 CORRECTION DIRECTIVES

Correction directives control updating of the old program library. New text is assigned a unique 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.

The following directives are used for inserting and deleting text:

INSERT	(I) Insert text after specified card
BEFORE	(B) Insert text before specified card
DELETE	(D) Deactivate card and optionally insert text in its place
RESTORE	(R) Reactivate card and optionally insert text after it
COPY	(C) Copy and insert text from specified library deck

These directives indicate to UPDATE that:

1. New text is to be inserted into the library and sequenced according to the current correction set identifier.

2. That old text is to be deleted.

While inserting, UPDATE interprets file manipulation directives (e.g., READ changes the source of insertion cards but does not terminate insertion). COPY does not terminate insertion and can be used to obtain insertion text from another deck on the library. Compile file directives (section 2.7) are inserted as if they are text; the master control character written on the program library is that specified on the directive.

Unless a TEXT directive has been encountered, UPDATE terminates an insertion when it encounters the next insertion directive or one of the following directives:

PURGE
SELPURGE
PURDECK
ADDFILE
IDENT

File manipulation directives are interpreted (and may change the source for insertion cards) but they do not terminate insertion. They are not inserted into the deck. Insertion cards can include compile file directives and directives destined for the YANK$$$ deck.

Correction directives that modify on a correction set or deck basis rather than on a card basis are the following:

YANK	(Y)	Deactivate correction sets
SELYANK	(SY)	Selectively deactivate correction sets
YANKDECK	(YD)	Deactivate all cards in decks
PURGE	(P)	Permanently remove correction sets
SELPURGE	(SP)	Permanently remove cards belonging to correction sets from specific decks
PURDECK	(PD)	Permanently remove all cards in decks
CHANGE	(CH)	Change correction set name
SEQUENCE	(S)	Resequence decks and purge all inactive cards

2.4.1 IDENT-IDENTIFY NEW CORRECTION SET

A correction set usually begins with an IDENT but need not if no new program library is being generated. In this case, UPDATE uses the default .NO.ID. for new text cards.

A PURGE, SELPURGE, PURDECK, ADDFILE, IDENT, SEQUENCE, or end-of-record† terminates a correction set.

Formats:

```
*IDENT  idname, p1, p2, ..., pn
*ID
```

idname — 1-9 character identifier to be assigned to this correction set. Legal characters are:

A-Z 0-9 + - * / ( ) $ =

This name causes a new entry in the directory. Each card inserted by this correction set and each card for which the status is changed receives a correction history byte that indexes this idname. Sequencing of new cards begins with one for this idname.

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.

This idname remains in effect until UPDATE encounters another IDENT directive or encounters PURGE, SELPURGE, PURDECK, ADDFILE, or SEQUENCE directive.

† End-of-section for SCOPE 2.

p_i Any number or none of the following parameters.

B=num Bias of num is to be added to sequence numbers. If more than one B parameter is specified, UPDATE uses the last one encountered.

K=ident The specified ident must be already in the directory for this correction set to be incorporated. If ident is unknown, UPDATE skips the correction set and resumes processing with the next IDENT, PURGE, SELPURGE, PURDECK, or ADDFILE directive. If more than one K parameter is specified, all the idents must be known or the correction set is skipped.

U=ident The specified ident must not be known for this correction set to be processed. If ident is known, UPDATE skips the correction set and resumes processing with the next IDENT, PURGE, SELPURGE, PURDECK, or ADDFILE directive. If more than one U parameter is specified, all the idents must be unknown or the correction set is skipped.

NOTE

An ident that has been yanked is still known; that is, an ident is known whether it is active or inactive. An ident must be purged to become unknown.

Example:

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

The bias of 100 (decimal) is added to all ZAP correction set card sequence numbers. That is, the first card in correction set ZAP has sequence number 101 not 1. UPDATE skips the correction set if ACE is unknown or either NON or ARF is known.

2.4.2 INSERT-INSERT CARDS AFTER

Formats:

```
*INSERT c
*I
```

c Identifies card (section 2.2) after which new cards will be inserted.

Cards to be inserted immediately follow the INSERT or I card on the input file.

2.4.3 BEFORE-INSERT CARDS BEFORE

Formats:

```
*BEFORE c
*B
```

c	Identifies card (section 2.2) before which new cards will be inserted.

Cards to be inserted immediately follow the BEFORE or B card on the input file.

2.4.4 DELETE-DELETE AND INSERT CARDS

Formats:

```
*DELETE c
*D
```

```
*DELETE ca,cb
*D
```

c	Card identifier (section 2.2) for single card to be deleted.
c_a, c_b	Card identifiers (section 2.2) of first and last cards in sequence of cards to be deleted. c_a must occur before c_b on the library. The range can include cards already deleted which are not affected by the DELETE.

With the DELETE or D directive, the user deactivates a card or block of cards and optionally replaces it with insertion cards following the DELETE directive.

A deactivated card remains on the library and retains its sequencing. It can be referred to in the same way as an active card.

A deactivated card is not included in the compile decks or source decks.

2.4.5 RESTORE-REACTIVATE CARDS

Formats:

```
*RESTORE c
*R
```

```
*RESTORE ca, cb
*R
```

c	Card identifier of single card (section 2.2) to be restored.
c_a, c_b	Card identifiers of first and last cards in sequence of cards to be restored. Any cards in the sequence that are already active are not affected by the RESTORE. c_a must occur before c_b on the library.

With the RESTORE directive a user reactivates a card or block of cards previously deactivated through a delete and optionally inserts additional cards after the restored card or block of cards. The cards to be inserted immediately follow the RESTORE card.

2.4.6 COPY-COPY TEXT

The COPY directive has two forms. The first form is used only during insertion and directs UPDATE to copy one or more active cards from a deck on the old program library and insert them as if they were text on the input stream. The second form cannot be used during insertion. It provides a means of obtaining a copy of one or more active cards from a deck on the old program library and writing them on a file specified by the user. An attempt to copy decks being introduced during the same UPDATE produces an informative message. Copying into a new deck from an existing deck is legal. UPDATE copies the cards before applying any corrections to them. Thus, the first form allows a user to move a sequence of cards by copying them and deleting the original cards in the same UPDATE run.

Format one:

```
*COPY dname, c
*CY
```

```
*COPY dname, ca, cb
*CY
```

dname	Deck on old program library that contains cards to be copied.
c	Card identifier (section 2.2) of single card to be copied.
c_a, c_b	Card identifiers of first and last cards in sequence of cards to be copied.

For this form of COPY, an INSERT, DELETE, BEFORE, or RESTORE must be in effect.

In the following example, this first form of COPY is valid because the INSERT has initiated insertion. Cards BDECK.4 through BDECK.8 are copied and inserted after the text cards. The copied cards are sequenced as part of correction set X.

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

The following text stream is not valid because insertion is not in effect and UPDATE does not know where to write the card copies:

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

Format two:

```
*COPY  dname, ca, cb, file
*CY
```

dname	Name of deck containing text to be copied. This deck must be on the old program library.
c_a, c_b	Card identifiers (section 2.2) of first and last cards in sequence of cards to be copied.
file	Name of file onto which cards are to be copied. The user is responsible for the disposition of this file. The file is a coded file that contains 80-column card images. It has one record for each COPY directive. No sequencing information is appended.

Placement in the input stream of this form of COPY is not restricted to insertion. This form of COPY is not a correction directive.

2.4.7 CHANGE-CHANGE CORRECTION SET IDENTIFIER

Format:

```
*CHANGE  idname1, idname2, idname3, idname4, . . . , idname n-1, idnamen
*CH
```

$idname_i$	Name of correction set to be changed.
$idname_{i+1}$	New correction set name, 1-9 alphaumeric characters. Legal characters are: A-Z 0-9 + - * / ( ) $ =

The CHANGE directive changes $idname_i$ in the directory to $idname_{i+1}$. As a secondary effect, changing the name of the correction set invalidates any YANK or SELYANK directives in the YANK$$$ deck that refer to the set by its previous name. A CHANGE directive goes into effect immediately. Thus, any subsequent references to the correction set must use the new name.

The CHANGE directive does not terminate insertion and need not be part of a correction set.

CHANGE cannot be used to change deck names.

2.4.8 YANK-REMOVE EFFECTS OF CORRECTION SET

Format one:

```
*YANK  idname1, idname2, . . . , idnamen
*Y
```

$idname_i$	Name of correction set previously applied to the program library. If UPDATE fails to find $idname_i$, it issues an error message.

Format two:

```
*YANK  idnamea.idnameb
*Y
```

The correction set $idname_a$ and all sets up to and including $idname_b$ are yanked. If $idname_a$ and $idname_b$ cannot be located or are in reverse order, UPDATE issues an error message.

UPDATE places the YANK directive in the YANK$$$ deck. During the modification phase, UPDATE checks each correction to see if it has been yanked. All yanked corrections are ignored. If the card was deactivated by the yanked correction set, UPDATE reactivates it. If the card was activated by the yanked correction set, UPDATE deactivates it. Thus, UPDATE changes the correction history byte for every card that changed status.

The YANK can be selectively nullified during the correction phase through the introduction of DO and DONT directives in the decks.

For an example of YANK use, refer to section 4.2.4.

A YANK must be part of a correction set.

A YANK directive does not terminate insertion.

2.4.9 YANKDECK-DEACTIVATE DECKS

The YANKDECK directive deactivates all cards within the decks specified.

Format:

```
*YANKDECK  dname1, dname2, . . . , dnamen
*YD
```

$dname_i$	Name of deck to be deactivated. All cards in the deck are deactivated regardless of the correction set to which they belong. If UPDATE is unable to find $dname_i$, it issues an error message.

The YANK$$$ deck cannot be yanked.

The YANKDECK directive must be part of a correction set.

YANKDECK does not terminate insertion.

For an example of YANKDECK, see section 4.2.4.

2.4.10 SELYANK-SELECTIVELY REMOVE EFFECTS OF CORRECTION SET

The SELYANK directive resembles the YANK directive but the effect is limited to the deck specified on the SELYANK directive.

Formats:

```
*SELYANK  dname1.idname1, dname2.idname2, . . . ,dnamen.idnamen
*SY
```

$dname_i$	Name of deck from which correction set $idname_i$ is to be removed.
$idname_i$	Correction set to which cards to be removed belong.

If UPDATE is unable to find either $dname_i$ or $idname_i$, it issues an error message.

Cards in the YANK$$$ deck can be yanked with SELYANK.

The SELYANK directive must be part of a correction set. It does not terminate insertion.

For examples of use, see section 4.2.4.

2.4.11 PURGE-PURGE CORRECTION SETS

The PURGE directive causes the permanent (irreversible) removal of a correction set or group of correction sets.

A PURGE directive can be any place in the directives input. The YANK$$$ deck cannot be purged.

Purging cannot be rescinded. See section 4.2.4 for an example of use.

The PURGE directive has three basic formats:

Format one:

```
*PURGE  idname1, idname2, ..., idnamen
*P
```

$idname_i$ Identifiers for correction sets to be purged.

Format two:

```
*PURGE idnamea.idnameb
*P
```

Correction set $idname_a$ and all sets up to and including $idname_b$ on the directory are purged. If $idname_a$ and $idname_b$ cannot be located or are in reverse order, UPDATE issues an error message.

Format three:

```
*PURGE  idname,*
*P
```

Correction set idname and all correction sets that have been introduced after idname are purged. This returns the library to an earlier level only if no PURGE, SELPURGE, PURDECK, or SEQUENCE directive has been issued previously.

If UPDATE cannot locate a specified correction set, it issues an error message. Purged idnames can be reused on subsequent correction sets provided they do not appear in the YANK$$$ deck.

2.4.12 PURDECK-PURGE DECKS

A PURDECK directive causes the permanent (irreversible) removal of a deck or group of decks from the program library. Every card in a deck is purged, regardless of the ident it belongs to. PURDECK does not purge idnames. Thus, a deck name purged as a result of PURDECK can be reused as a dname. It can be used as a new idname only if it is not already in the directory list. See section 4.2.4 for example of use.

The deck name remains in the directory until removed through use of the E option on the UPDATE control statement (section 4.12) on a subsequent UPDATE run.

The deckname can also be removed by resequencing the library, that is, by creating a source file in one UPDATE run and using the source file as input on a second run.

A PURDECK directive can be any place in the directives input. The YANK$$$ deck cannot be purged. Purging cannot be rescinded.

The PURDECK directive has two basic formats:

Format one:

```
*PURDECK  dname1, dname2, . . . , dnamen
*PD
```

$dname_i$ Deck names for decks to be purged.

Format two:

```
*PURDECK dnamea. dnameb
*PD
```

The deck named $dname_a$ and all decks up to and including $dname_b$ listed in the deck list are purged. If $dname_a$ and $dname_b$ cannot be located or are in reverse order, UPDATE issues an error message.

2.4.13 SELPURGE-SELECTIVELY PURGE CORRECTION SETS

The SELPURGE directive causes all cards in a specified deck that belong to the specified correction set to be purged. It permanently removes the effects of correction set idname in deck dname. The idname is not purged and therefore remains known to UPDATE. No deck other than dname is altered in any way.

A SELPURGE directive can be any place in the directives input.

Formats:

```
*SELPURGE  dname1.idname1, dname2.idname2, . . . , dnamen.idnamen
*SP
```

dname	Name of deck from which correction set is to be removed.
idname	Correction set to which cards to be removed belong.

If UPDATE is unable to find either dname or idname, it issues an error message.

Cards in the YANK$$$ deck can be purged.

2.4.14 ADDFILE-ADD FILE OF NEW DECKS

ADDFILE directs UPDATE to read creation directives (DECK and COMDECK) and text data from the named file and insert this information after the specified deck or card on the new program library. The first card on the file must be a DECK or COMDECK directive. UPDATE reads from the file until it encounters a 7/8/9 card and then returns to the primary input file. If the file referred to by the ADDFILE is the primary input file, UPDATE adds cards until it encounters a 7/8/9 or the next directive that is not a compile file directive or file manipulation directive. UPDATE does not reposition the file specified on the ADDFILE directive. Any repositioning must be requested through SKIP or REWIND directives. ADDFILE is illegal on an alternate input file. A READ directive is illegal during processing of the added file.

Formats:

```
*ADDFILE  file, c
*AF
```

```
*ADDFILE  file, dname
*AF
```

file	Name of file from which information is to be read. This text cannot contain correction directives. If file is omitted, it is assumed to be the UPDATE input file (INPUT or its equivalent as specified by the I parameter).
c	Identifier for card (section 2.2) after which decks are to be placed on program library.
dname	Name of deck after which new decks are to be placed on program library. If the dname 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 directive (refer to section 2.2). If no such directive precedes the ADDFILE, YANK$$$ is used.

If only one parameter is present, it is assumed to be the file name. Omission of the second parameter causes UPDATE to add the decks at the end of the library.

For example of use, see section 4.2.5.

2.4.15 SEQUENCE-RESEQUENCE DECKS

The SEQUENCE directive directs UPDATE to resequence and purge inactive cards from the specified decks on the new program library.

Formats:

```
*SEQUENCE dname1, dname2, . . . , dnamen
*S
```

```
*SEQUENCE dnamea. dnameb
*S
```

$dname_i$	All active cards in $dname_i$ are resequenced under identifier $dname_i$. All previous correction history bytes and all inactive cards are purged. In the first form, each of the decks is resequenced. In the second form, each of the decks in the deck list starting with $dname_a$ and ending with $dname_b$ is resequenced.

UPDATE normally allows deck and correction sets having the same name to co-exist on the old program library. If a deck having the same name as a correction set is resequenced and cards 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.

Only those decks explicitly mentioned on the SEQUENCE directive are resequenced. Thus, if a correction set (e.g., SET1) has been applied that affects more than one deck on a program library (e.g., DECK1 and DECK2), and only DECK1 has been subsequently resequenced through SEQUENCE, the SEQUENCE directive does not affect SET1 cards within DECK2.

SEQUENCE does not result in idnames being deleted from the library even if, as a result of resequencing, no references to an idname 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 idname in this case requires an EDIT or PURGE in a subsequent UPDATE run.

2.4.16 MOVE-MOVE DECK

Format:

```
*MOVE  dname1, dname2
*M
```

The MOVE directive enables the user to reorder decks while producing a new program library. The deck $dname_1$ is moved from its position on the old library and placed after $dname_2$ on the new library. A MOVE referencing a deck introduced in the same UPDATE run produces an informative diagnostic.

MOVE does not terminate insertion and need not be part of a correction set.

2.5 FILE MANIPULATION DIRECTIVES

File manipulation directives allow user control over files during UPDATE processing. The READ directive may be used to change the source of directives and insertion text from the input file to an alternate file. A file change while an insertion is in progress does not terminate insertion. File manipulation directives are illegal when UPDATE is reading from an alternate file (such as during ADDFILE processing). They can be on the primary input file only.

File manipulation directives include:

READ	(RD)	Read input stream from specified file
SKIP	(SK)	Skip forward specified number of records on file
REWIND	(RW)	Rewind named file

These operations cannot be performed on the following reserved files or their equivalents:

INPUT	Source of directives and input text
OUTPUT	List output
COMPILE	Compile output
SOURCE	Source output
OLDPL	Old program library
NEWPL	New program library
MERGE	Merge

UPDTSCR UPDTCDK UPDTTPL UPDTEXT UPDTAUD UPDTPMD	UPDATE scratch files

The scratch files are always reserved but the other files are reserved by their corresponding UPDATE control card options only. That is, if the S or T options do not specify a file named SOURCE, (e.g., the source file is S1) the user can have a file named SOURCE to which file manipulation directives can refer; however, he cannot manipulate the file named S1.

2.5.1 READ-READ ALTERNATE DIRECTIVES FILE

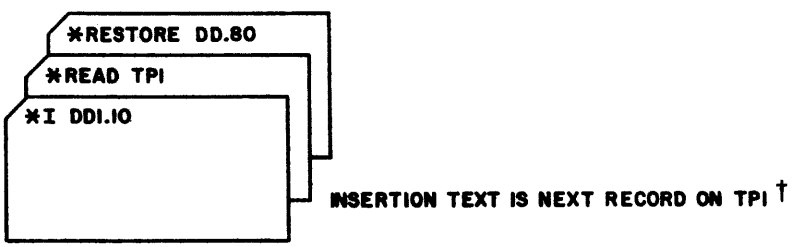
Formats:

```
*READ file
*RD
```

file	Name of file containing insertion text and/or directives; READ, SKIP, REWIND, and ADDFILE are illegal on the file.

The READ directive causes UPDATE to temporarily stop reading the input file and begin reading directives and insertion text from the named file at its current position. UPDATE reads from this alternate directives file until it encounters an end-of-record† and then resumes with the next card on the primary input file.

Example:



† End-of-section for SCOPE 2.

2.5.2 SKIP-SKIP FORWARD ON FILE

Formats:

```
*SKIP file
*SK
```

```
*SKIP file,n
*SK
```

file	Name of file to be positioned
n	Number of logical records† to be skipped in the forward direction. If n is omitted, UPDATE skips one record.

The SKIP directive repositions the named file forward one or more logical records. If an end-of-file (end-of-partition) or end-of-information is encountered before the requested number of records has been skipped, the file is positioned at the EOF, EOP, or EOI that stopped the skipping.

2.5.3 REWIND-REWIND FILE

Format:

```
*REWIND file
*RW
```

file	Name of file to be rewound

The REWIND directive repositions the file to its first logical record.†

2.6 SELECTIVE COMPILE DIRECTIVE (COMPILE)

Formats:

*COMPILE $dname_1, dname_2, \ldots, dname_n$
*C

*COMPILE $dname_a . dname_b$
*C

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

† Section(s) for SCOPE 2.

The first form of the directive requests one or more decks in any sequence on the library. The second form requests all decks in the deck list on the old program library starting with $dname_a$ through $dname_b$. If UPDATE fails to find the named deck or if the deck names are reversed, it issues a diagnostic message.

When a deck is being introduced in the same run that contains a COMPILE directive for the deck, UPDATE must not encounter the COMPILE directive before the DECK directive. Otherwise, COMPILE directives can be anywhere in the input stream.

Calling a common deck from within a deck being updated results in the common deck being updated.

2.6.1 NORMAL SELECTIVE MODE

During a normal UPDATE run (F and Q are not selected on the UPDATE control card), UPDATE writes on the compile file all decks specified on COMPILE directives as well as all decks corrected during the run. COMPILE causes a deck to be written regardless of whether it was corrected or not. In normal mode, decks are written on the compile file in the sequence encountered on the old program library. If a common deck is modified, a deck calling it that precedes the common deck is not automatically written on the compile file. The deck preceding a deck that has been purged by PURDECK and/or PURGE will be written on the compile file. By selecting the K parameter on the UPDATE control card, the user causes decks to be written in the sequence specified on COMPILE directives followed by any decks for which corrections were made that were not mentioned on COMPILE directives.

2.6.2 FULL UPDATE MODE

During a full update run (F selected on UPDATE control card), UPDATE ignores COMPILE directives. It updates all decks in the sequence encountered on the library. This sequence cannot be changed through the K option.

2.6.3 QUICK MODE

During a quick UPDATE run (Q selected), only decks specified on COMPILE directives and called common decks are written on the compile file. These decks are written in the sequence encountered on the program library unless the K option has been specified, in which case, they are written in the order specified on the COMPILE directives. For an example, see section 4.2.6.

2.7 COMPILE FILE DIRECTIVES

The directives described in this section provide user 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 directives are written on the compile file.

The user can prepare his original source deck with compile file directives embedded in it (except for DO or DONT) or he can insert compile file directives into program library decks as a part of a correction set. Compile file directives are not recognized when they are on the input file; they do not terminate insertion but are simply considered as text cards to be inserted and are sequenced accordingly.

Compile file directives include:

WEOR	(W)	Write end-of-record of specified level on compile file
CWEOR	(CW)	Write end-of-record or end-of-file on compile file if the buffer is not empty
CALL	(CA)	Write called common deck onto compile file
IF		Write text onto compile file if condition is defined
ENDIF	(EI)	Optionally used to specify end-of-conditional text
TEXT	(T)	The text following is not to be scanned for directives
ENDTEXT	(ET)	Resume scan for directives
DONT	(DT)	Do not rescind yanks of correction set applying to the following text
DO		Rescind yanks of correction sets applying to the following text

To be recognized while the compile file is being written, these directives must have the same master control character as that defined when the library was created.

2.7.1 WEOR-WRITE END-OF-RECORD OR END-OF-FILE

Format:

```
*WEOR level
*W
```

level

If level is 0-14 decimal, UPDATE (SCOPE 3.x and NOS 1) writes an end-of-record of the specified level on the compile file. UPDATE (SCOPE 2) writes an end-of-section.

If level is 15 decimal or greater, UPDATE writes a zero-level end-of-record followed by an end-of-file on the compile file (end-of-partition for SCOPE 2). If level is omitted, UPDATE writes a zero-level end-of-record.

This directive aids organization of a compile file to be used by compilers, assemblers, etc.

2.7.2 CWEOR-CONDITIONALLY WRITE END-OF-RECORD OR END-OF-FILE

Format:

```
*CWEOR level
*CW
```

level — If level is 0-14 decimal, UPDATE (SCOPE 3.x and NOS 1) writes an end-of-record of the specified level on the compile file. UPDATE (SCOPE 2) writes an end-of-section. If level is 15 or greater, UPDATE writes an end-of-file on the compile file (end-of-partition for SCOPE 2). If level is omitted, UPDATE writes an end-of-record.

If level is 15 decimal, or greater, UPDATE writes a zero-level end-of-record followed by an end-of-file on the compile file (end-of-partition for SCOPE 2). If level is omitted, UPDATE writes a zero-level end-of-record.

This directive causes UPDATE to write an end-of-record mark or end-of-file mark only if information has been placed in the output buffer since the last end-of-record (or end-of-file) was written.

2.7.3 CALL-CALL COMMON DECK

Format:

```
*CALL dname
*CA
```

dname — Name of common deck to be written on compile file.

UPDATE writes the text of a previously encountered common deck, dname, onto the compile file. Common code, such as system symbol definitions, may be declared in the common deck and used in subsequent decks or assemblies without repeating the data cards. The CALL card does not appear on the compile file. The contents of the common deck, excluding the COMDECK card, follow immediately. Common decks can call other common decks. However, to avoid circularity of calls, a common deck must not call itself or call decks that contain calls to the common deck. A CALL directive is effective only when it is within a deck.

2.7.4 IF-CONDITIONALLY WRITE TEXT

The writing of text cards on the compile file can be conditionally controlled through the IF directive. When UPDATE encounters an IF directive while it is writing on the compile file, UPDATE conditionally writes the text following the IF or skips the text.

Formats:

```
*IF  type, name, num        or        *IF  -type, name, num
```

type — Type of conditional name. 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.

Type	Name
DECK	Name is a deck name. To be known, it must be in the deck list.
IDENT	Name is a correction set identifier. To be known, it must be in the directory.
DEF	Name is defined through DEFINE directive encountered during correction phase. DEFINE directives are maintained in the YANK$$$ deck.

name — According to type, a deck name, correction set identifier, or defined name.

num — When the condition is not met, this is the number of card images that are skipped. If the condition is not met and num is omitted, UPDATE searches for an ENDIF directive and resumes processing of the deck at that point. When the condition is met, no cards are skipped.

Examples:

1. During the correction phase, UPDATE processes the following directive:

   ```
   *DEFINE ABC
   ```

   PROG2, a deck to be written on the compile file, contains the following sequence:

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

   All active text cards between IF and ENDIF are written as if they are part of PROG2. Removing the DEFINE from the YANK$$$ deck would cause these text cards to be skipped.

2. DECKA has mutually exclusive requirements depending on the availability of correction set IDC.

```
*DECK DECKA
  :
*IF IDENT,IDC,15
(15 active text cards)          Written if IDC is available
*IF -IDENT,IDC
(active text cards)             Not written if IDC is available
*ENDIF
```

3. The following example illustrates nesting of IF directives. SAM has an IF-controlled sequence containing a second IF-controlled sequence.

```
*DECK SAM
  :
*IF IDENT,JOE                   Text following is written if JOE is available.
  :
*IF IDENT,BOB                   Text following is written if both JOE and BOB are
  :                             available
  :
*ENDIF                          Terminates both IFs
```

2.7.5 ENDIF -END OF CONDITIONAL TEXT

Format:

```
*ENDIF
*EI
```

This directive is used with IF when the num parameter is omitted from the IF directive. An ENDIF indicates the end of conditional text. It is not written on the compile file.

NOTE

ENDIF should not be used if num is specified on the IF directive. If it is used, num takes precedence. The ENDIF is included in the count of active cards and is written on the compile file.

2.7.6 TEXT AND ENDTEXT-IDENTIFY TEXT

Formats:

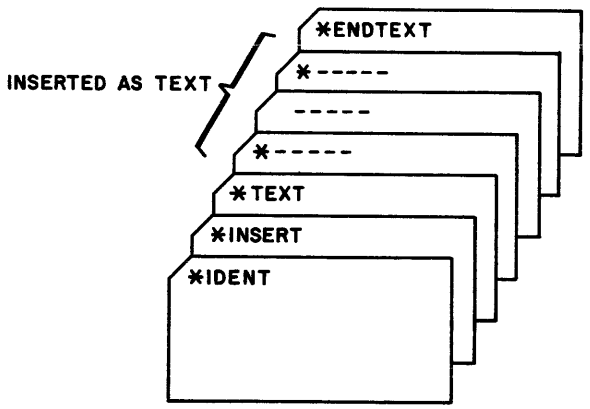
```
*TEXT
*T
```

```
*ENDTEXT
*ET
```

The TEXT (or T) and ENDTEXT (or ET) directives delimit a sequence of text card images that might be interpreted as correction directives by UPDATE. When UPDATE encounters a TEXT directive, that card and all following it up to and including the ENDTEXT directive are considered as data and are written on the program library. These cards are not checked or altered in any way. They do, however, have an identifier associated with them and may be modified and corrected as any other UPDATE text. 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 cards; they are not written on the compile file.

Any information in columns 10 - 80 is taken as a comment.

Example:



If ENDTEXT is encountered before TEXT, UPDATE issues the message UNBALANCED TEXT/ENDTEXT CARDS, LAST ENDTEXT CARD IGNORED.

2.7.7 DO AND DONT- TEMPORARILY RESCIND YANK AND SELYANK

Formats:

```
*DO  idname1, idname2, ..., idnamem

*DONT  idname1, idname2, ..., idnamem
*DT
```

$idname_i$	Name of correction set for which yanking is to be rescinded or initiated. Omitting idname results in a format error.

When UPDATE encounters a DO directive and until it encounters a DONT directive for a correction set, it checks to see if a card was deactivated by a YANK or SELYANK and reactivates it. If it was activated by the YANK or SELYANK, UPDATE deactivates the card.

If UPDATE encounters a DO for an unyanked correction set, it issues an informative message and ignores the DO.

Although a DONT is normally used to terminate a DO, the DONT directive can also be used to initiate a yank of an unyanked correction set at some place other than the beginning of the old program library. 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 deck is already yanked, UPDATE issues an informative message and ignores the DONT.

DO and DONT directives can be any place in the library. They are not written on the compile file. For an example of use, see section 4.2.4.

2.8 SPECIAL DIRECTIVES

The directives described in this section provide extended features. With the exception of PULLMOD, they can be any place in the directives file for creation or correction and primarily affect the operating features of UPDATE.

/		List Comment
NOABBREV	(NA)	Deselects the abbreviated directives feature
ABBREV		Reselects the abbreviated directives feature
LIMIT	(LT)	Changes maximum size allowed for list output file
NOLIST	(NL)	Inhibits list output
LIST	(L)	Resumes list output
DEFINE	(DF)	Defines names for subsequent use by IF directives
PULLMOD	(PM)	Recreates correction sets
DECLARE	(DC)	Restrict corrections to declared deck.

2.8.1 /-LIST COMMENTS

Format:

```
*/ comments
```

Other than being copied onto the UPDATE list output, a comment card is simply ignored. If column 3 does not contain a comma or a blank, the card is interpreted as a text card.

The comment character / can be changed through the / option on the UPDATE control card.

2.8.2 ABBREV AND NOABBREV-DO OR DO NOT CHECK FOR ABBREVIATED DIRECTIVES

Formats:

```
*ABBREV
```

```
*NOABBREV
*NA
```

Most UPDATE directive names can be written in abbreviated form. UPDATE expands the name when it reads an abbreviated form so that it is a full name on the listings, in all UPDATE output, and in the program library. Because checking for the abbreviated forms and expanding them is a time-consuming feature, the user has the option of not using abbreviations and turning off the check through the NOABBREV feature. In this mode, an abbreviated directive is not recognized but is taken as text. The ABBREV directive causes checking for abbreviations to resume.

UPDATE searches for abbreviations until a NOABBREV directive is encountered. In ABBREV mode, directives can be either abbreviated or unabbreviated.

2.8.3 LIMIT-LIMIT LIST OUTPUT

Format:

```
*LIMIT n
*LT
```

n	New line limit for list output (decimal)

The LIMIT directive changes the maximum size for the list output file from the default value of 6000 lines to n lines.

When the limit is reached, options 3 (card image, deck name, and modification key) and 4 (input stream) are turned off. Errors and control cards are still listed, however, if options 1 and 2 were selected. Options 5 (active compile file directives) to 9 (correction history resulting from list options 5, 7, and 8) are not affected. See L option, section 4.1.2.

2.8.4 LIST AND NOLIST-SELECT OR DESELECT LIST OPTION 4

Formats:

```
*NOLIST
*NL
```

```
*LIST
*L
```

NOLIST and LIST disable and enable list option 4, the list of cards in the input stream. UPDATE stops listing cards in the input stream when it encounters a NOLIST and resumes listing cards when it encounters a LIST.

Decks inserted by ADDFILE are not listed if list option 4 is selected by default but they are listed if option 4 is explicitly selected. See L parameter on UPDATE control card, section 4.1.2.

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.

2.8.5 DEFINE-DEFINE NAMES FOR USE BY IF

Format:

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

$name_i$	One or more names to be tested by IF directive during the write compile phase (section 2.7.4).

UPDATE places DEFINE directives in the YANK$$$ deck. A DEFINE directive has no purpose other than to establish a condition to be tested by IF directives. DEFINE names are unrelated to correction set identifiers or deck names.

A DEFINE directive can be placed anywhere in a correction set. It does not terminate insertion.

2.8.6 PULLMOD-RECREATE CORRECTION SETS

Format:

```
*PULLMOD idname1, idname2, ..., idnamen
*PM
```

$idname_i$	Identifiers of correction sets to be recreated.

The PULLMOD directive can be used when the PMODKEY installation option has been assembled for UPDATE. This directive requests UPDATE to search the library for all cards belonging to each of the correction sets and to reconstruct a set of directives and text that produces the same results as the original correction set. Each reconstructed correction set is written on the file specified by the G parameter on the UPDATE card. The information is contained within one record† on the file.

NOTE

It is the user's responsibility to determine whether or not the regenerated sets accurately reflect the original corrections. For example, PULLMOD is unable to determine if cards 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.

A PULLMOD directive does not affect the library.

2.8.7 DECLARE- DECLARE RESTRICTED CORRECTIONS

The DECLARE directive can be used when the DECLKEY installation option has been assembled. This directive provides a means of checking on the validity of UPDATE corrections. It protects decks other than the declared deck from being inadvertantly altered.

Format:

```
*DECLARE dname
*DC
```

dname	Name of deck to which following corrections are restricted. The restriction remains in effect until UPDATE encounters a DECLARE directive with no deck name, or another DECLARE directive with a different deck name.

†Section for SCOPE 2

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

1. PURGE and YANK directives are illegal.
2. INSERT, DELETE, RESTORE, and BEFORE directives can apply only to cards in the declared deck. If they do not, the operation is not performed and UPDATE issues an informative message.
3. 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.

2.8.6 END-END DECK

Format:

```
*END
```

UPDATE ignores an END directive if it encounters one in a source deck. It does not copy it onto the old program library.

The END directive provides compatibility with the SCOPE EDITSYM program.

FILES 3

Types of UPDATE files that concern the user are:

- Source files
- Program library files
- Input file
- Compile file
- Pullmod file
- Scratch files

3.1 SOURCE DECKS AND FILES

Before they can be updated, source decks must be converted to a program library through an UPDATE creation run or added to a library through a correction run. A source file is a group of source language decks either prepared by a user or generated by UPDATE.

The presence of the UPDATE directives generally prohibits the use of source decks as input to most language processors (e.g., COBOL). However, the COMPASS assembler treats the directives as comments (if the master control character is *) and accepts input from an UPDATE-generated source file.

3.1.1 SOURCE DECKS PREPARED BY USER AS INPUT TO UPDATE

A user prepares a source deck for input to UPDATE by placing a DECK or COMDECK directive (section 2.3.1 and 2.3.2) in front of the source language deck (Figure 3-1). At the same time, he can also insert compile file directives (section 2.7) into the source language deck to control compile file output from UPDATE. The next DECK or COMDECK directive or a 7/8/9 card terminates the deck. The DECK or COMDECK directive becomes the first card of the deck on the program library and has sequence number 1. This deck is assigned the deck name specified on the DECK or COMDECK directive. Common decks should precede non-common decks in the source file.

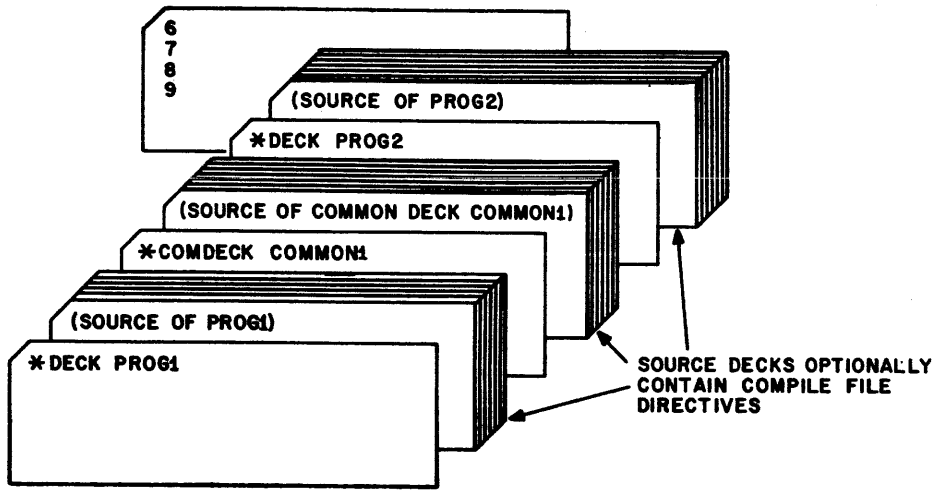


Figure 3-1. Example of Source Decks on a Source File

UPDATE source decks should not be confused with a compiler or assembler program. An UPDATE deck can contain any number of FORTRAN programs, subroutines, or functions, or COMPASS assembler IDENT statements or sets of data. Usually, each UPDATE deck contains one program for the assembler or compiler.

Source decks are placed in the input stream for UPDATE during a creation run or a correction run; they can also be on an alternate file accessed through the READ and ADDFILE directives.

3.1.2 SOURCE FILE GENERATED BY UPDATE

The source file contains a copy of all active DECK and COMDECK directives and all active cards within each deck. The source file is optional output from UPDATE (through use of the S or T options on the UPDATE control card) and once created can be used as source input on subsequent UPDATE runs. The source file is a coded file that contains 80-column images.

When Q is not selected on the UPDATE control card, the source file generated contains all cards needed to create a program library. This source file can be used as a back-up copy of the library or, providing that common decks occur first, can be used as input for an UPDATE run that produces a resequenced program library with all the inactive cards purged. When Q is selected on the UPDATE control card and the old program library is in random format, the source file contains only the decks explicitly requested on the COMPILE directives and any common decks called by those decks. The only directives that can legally appear in a source file are DECK, COMDECK, and compile file directives (section 2.7). If the old program library is in sequential format, the source file contains all decks requested on COMPILE directives, all common decks that they call, and any common decks encountered prior to processing of all of the specified decks.

3.2 PROGRAM LIBRARY FILES

A program library file is created during an UPDATE run and may be used as a primary data base for later UPDATE runs. The library provides a means of maintaining source decks in conveniently updatable compressed format. Along with each deck is a correction history for each card image.

Card images on the program library are grouped into decks. A deck consists of a DECK or COMDECK card image and all following card images either up to but not including the next DECK or COMDECK card image, or to an end-of-record.† Because DECK and COMDECK directives can be deactivated by DELETE, YANK and SELYANK, card images belonging to one deck at the beginning of an UPDATE run may belong to a different deck at the end of the run. When a DECK directive is deactivated, all card images in the deactivated deck become members of the preceding deck on the old program library and, thereafter, are affected by directives that affect the previous deck as a unit, such as PURDECK and SEQUENCE.

The UPDATE program can create and maintain symbolic library files written in two distinctly different formats: random and sequential.

3.2.1 RANDOM FORMAT

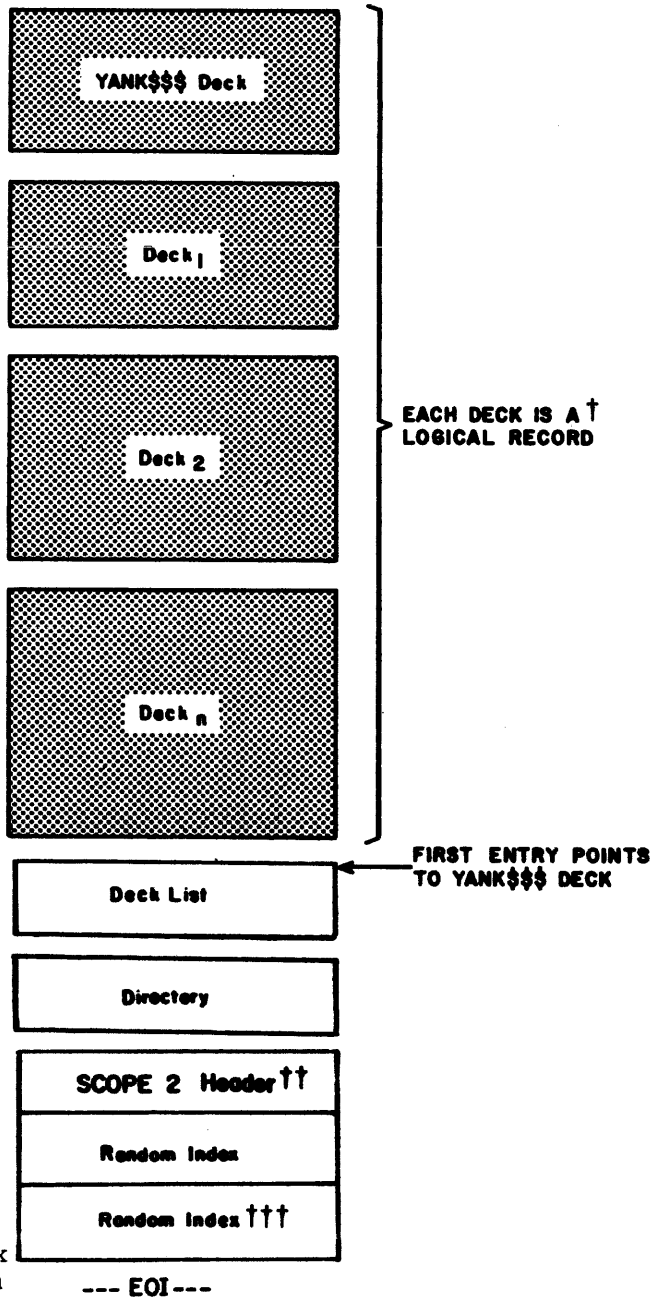
On a random format program library, each deck is a separate logical record (Figure 3-2). The deck records are followed by logical records containing the deck list, the directory, and the random index.

Using random format is substantially faster than using sequential format.

Under SCOPE 3, do not copy a random library to tape using COPYBF. The system will not recognize the format as random when you attempt to use the library on tape. Use the UPDATE B and A options, instead.

Under SCOPE 2, a random library can be copied to tape but must be copied back to mass storage on input to deblock the library. Use COPYS to copy a random program library to or from tape. Copy all sections of the program library (number of sections equals number of decks excluding the YANK$$$ deck, plus 5). Then use COPYR to copy the random index (one record).

† End-of-section for SCOPE 2.



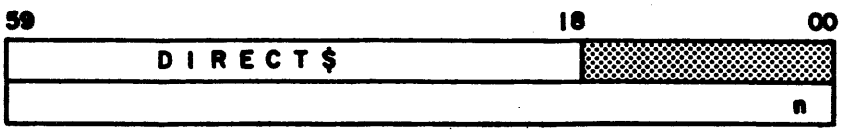
Under SCOPE 3.x, the random index record can be obtained by issuing an OPEN function on the file.

Under SCOPE 2, however, to obtain random index, skip to EOI, backspace one record, and read.

Figure 3-2. Random Program Library

† **For SCOPE 2, each deck is a section.**
†† **Header applies to SCOPE 2 only.**
††† Second copy if system is not SCOPE 3.x or NOS 1.

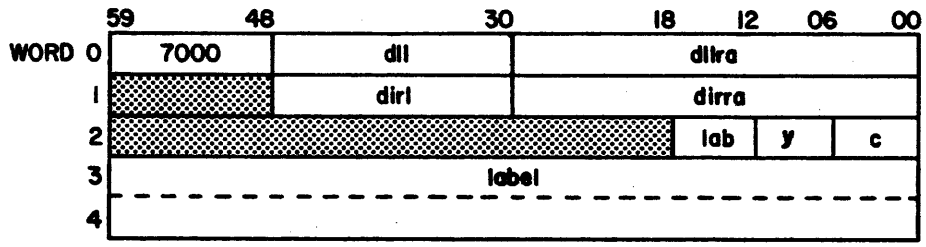
SCOPE 2 Random Index Header



n Number of words in random index; format is shown below.

Random Index Format

Depending on the operating system under which UPDATE is executing, the system generates either one or two copies of the random index. Under 7000 SCOPE 1.1, 6000 SCOPE 3.3, and 6000 KRONOS 2.0, only one copy of the index is produced. On any other system, two copies are produced because a CLOSE causes the index to be written on newer systems but not on older systems.

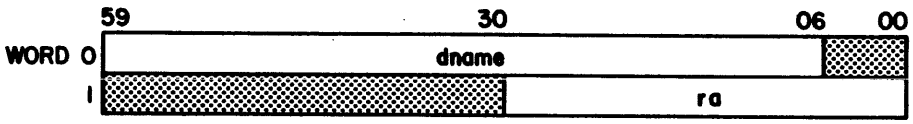


Word	Bits	Field	Description
0	59-48	7000	Identifies random directory record
0	47-30	dll	Length of the deck list in words
0	29-00	dllra	Random address of first word of deck list
1	59-48	none	Unused
1	47-30	dirl	Length of directory in words
1	29-00	dirra	Random address of first word of directory
2	59-18	none	Unused
2	17-12	lab	Label flag; if non-zero, words 3 and 4 are present in random directory and contain tape label. SCOPE 1 and SCOPE 2 do not recognize tape labels.
2	11-06	y	Type of program library; must be Y or null to indicate 64-character set.
2	05-00	c	Master control character used when the program library was generated.

Deck List Format

The deck list contains a two-word entry for each deck on the library. The first entry points to the YANK$$$ deck.

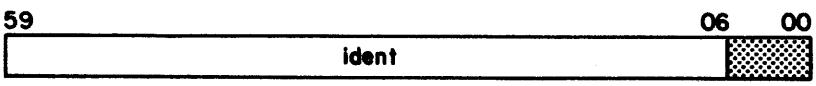
Each entry has the following format:



Word	Bits	Field	Description
0	59-06	dname	1-9 alphanumeric character deck name obtained from DECK or COMDECK directive when deck was placed on library. The first dname is YANK$$$.
0	05-00	none	Unused
1	59-30	none	Unused
1	29-00	ra	Random address of first word of compressed text for the deck.

Directory Format

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 1-9 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. An identifier that has been purged is not printed on the listable output file although table space is allocated to it. The purged identifiers are removed from the table when the E (edit) option is specified on the UPDATE control card (section 4.1). The number of identifiers in the directory is limited by the amount of central memory (or small core memory) available.



For a purged ident, bits 59-06 are zeroed and bits 05-00 contain a 20_8.

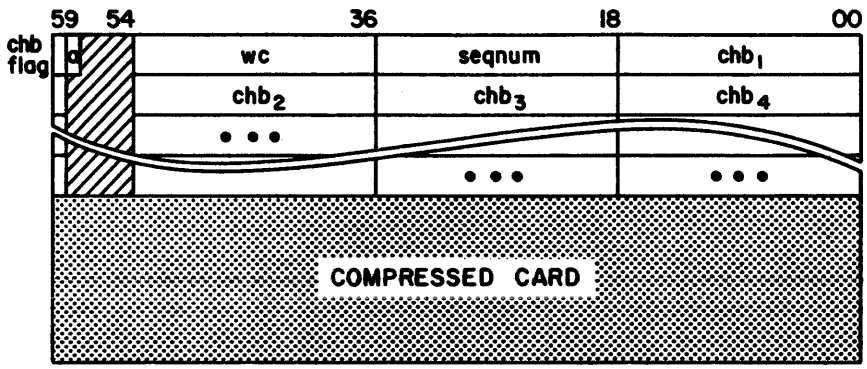
YANK$$$ Deck

The YANK$$$ deck is automatically created on a creation run as the first deck on the program library. On correction runs, UPDATE inserts into the YANK$$$ deck any YANK, SELYANK, YANKDECK, and DEFINE directives that it encounters during the read directives phase. These directives acquire identification and sequence information from the correction set from which they originate.

Although the YANK$$$ deck, as a whole, cannot be yanked or purged, cards in the deck can be deleted, yanked or purged from it. If information other than the four directive types mentioned inadvertently gets in the YANK$$$ deck, it can be purged through the E option on the UPDATE control card. On a merge, the two YANK$$$ decks are merged into a single deck. This deck does not have a DECK card as its first card image.

Compressed Text Format

Text is an indefinite number of words that contain a correction history and the compressed image of each card in the deck. Information for each card is in the following format:



Bits	Field	Description
59	chb flag	Indicates the last word containing correction history bytes. 0 Not last word 1 Last word
58	a	Activity bit for the card. 0 Card is inactive 1 Card is active
57-54	none	Unused
53-36	wc	Number of words of compressed text for this card.
35-18	seqnum	Sequence number of card (octal) according to position in deck or correction set identified by chb_1.

Bits	Field	Description
17-00 and subsequent 18-bit bytes	chb_i	Correction history byte. UPDATE creates a byte for each correction set that changes the status of the card. The format of chb_i is:

y — Yank bit

0 Card not yanked
1 Card has been yanked

a — Activity bit

0 Correction set deactivated the card
1 Correction set activated the card

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

Compressed card — The compressed image of the card 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
4 spaces replaced by 0003
5 spaces replaced by 0004
⋮ ⋮ ⋮
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.

When a space is the first character of a line, it is always represented as 55_8 even when it is part of a string of spaces.

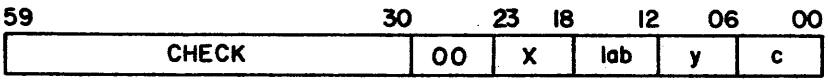
Trailing spaces are not considered as embedded and are not included in the card image. A 4-digit octal code 0000 or word count (wc) reached marks the end of the card. This is conditional on the CHAR64 option.

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

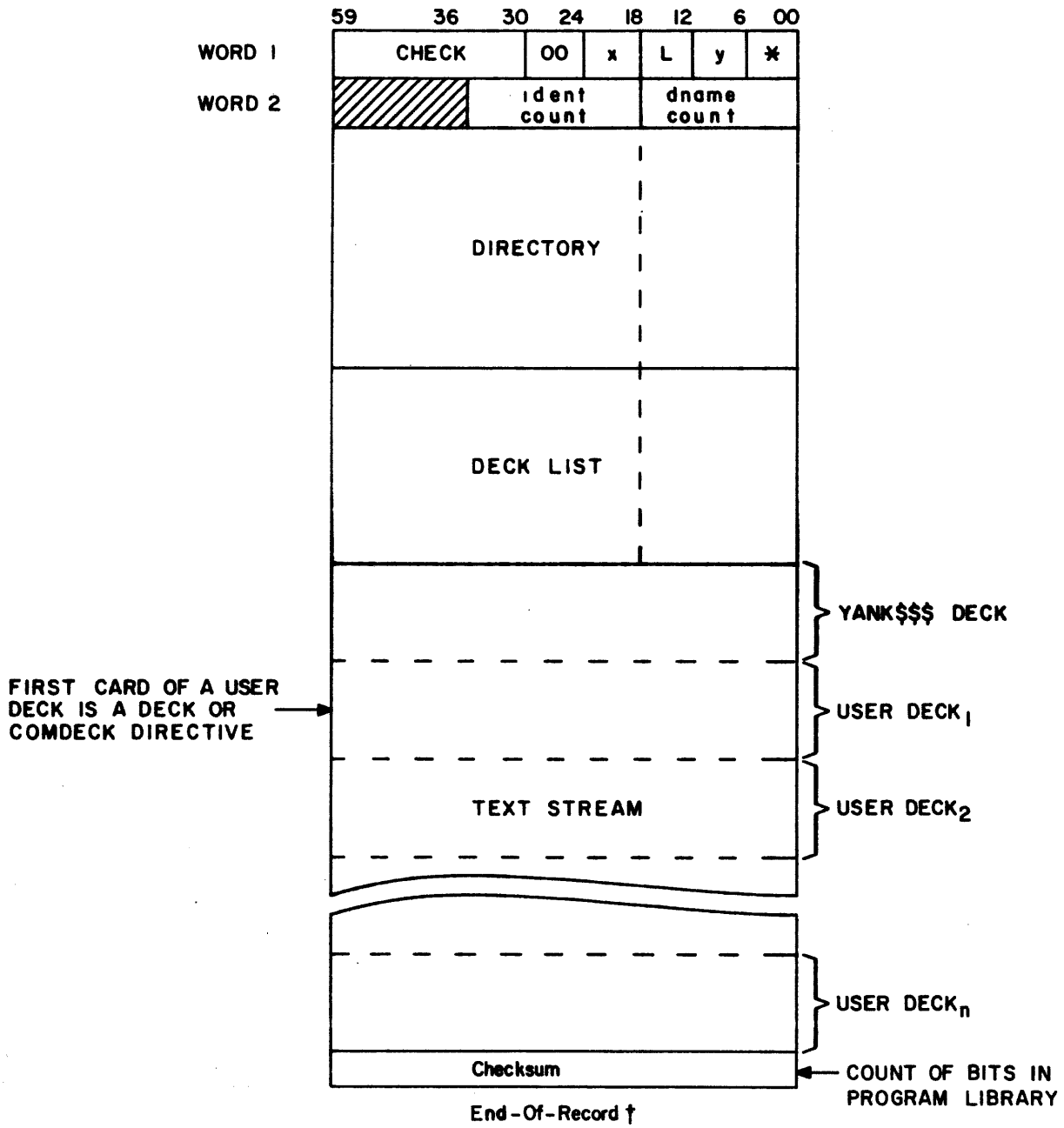
3.2.2 NEW SEQUENTIAL FORMAT

UPDATE optionally creates new library files in sequential format. On magnetic tape, sequential library files are written in SCOPE internal tape format. The entire sequential format library file is written as one binary record (Figure 3-3). The first word in the file is a display code key word; the second is a counter word containing the number of deck names in the deck list and a count of correction set identifiers in the directory. The last word in the file is a checksum.

Word One



Bits	Field	Description
59-30	CHECK	Identifies the file as being a new format sequential file. This field contains the word CHECK in display code.
29-24	none	Unused; zero
23-18	x	Character set identifier determined by IP. CSET parameter. If IP. CSET is set for a 63-character set, the field contains a 3(octal 36); if IP. CSET is set for a 64-character set, the field contains a 4 (octal 37).
17-12	lab	Presence of L indicates labeled tape. Null indicates unlabeled tape. SCOPE 2 does not recognize tape labels.
11-06	y	Indicates whether or not this library was generated using full 64-character set. Y Yes; colon is supported 00 No; colon is not supported
05-00	c	Indicates master control character in use when this library was created. * First character of directives is asterisk, the conventional master control character. See section 4.1.2. Other First character of directives is character indicated. On a correction run, if the master control character specified on the UPDATE card does not match this character, UPDATE changes the character to c. On a merge run, if the control characters for the two libraries do not match, the run is aborted.



†End of section if record type is W. Each W record is 512 words.

Figure 3-3. New Format Sequential Program Library File

Word Two

The second word of the program library file is composed of two binary counts: The count of the identifiers in the directory, and the count of the deck names in the deck list.

Directory Format

Entries in the directory are in the same format as for a random library.

Deck List Format

The deck list is a table consisting of one entry for each deck on the program library. Each entry consists of one word containing the deck name in display code, left justified, with zero fill. Decks are listed in the order in which they were introduced on the library.

Text Stream

The text stream immediately follows the deck list. Compressed text up to the first DECK or COMDECK directive comprise the YANK$$$ deck. Each subsequent deck begins with a DECK or COMDECK directive and consists of each card up to the next DECK or COMDECK directive or the end-of-record.

Compressed Text Format

Text on the sequential library is compressed in the same manner as on a random library.

3.2.3 OLD SEQUENTIAL FORMAT

UPDATE optionally accepts old library files in the old (pre-SCOPE 3.3) UPDATE sequential format (Figure 3-4). These libraries resemble the new sequential format but do not contain the CHECK word or checksum and the text format and chb's are different. Word two on the new format is the same as word one on the old format. UPDATE does not generate this obsolete sequential format.

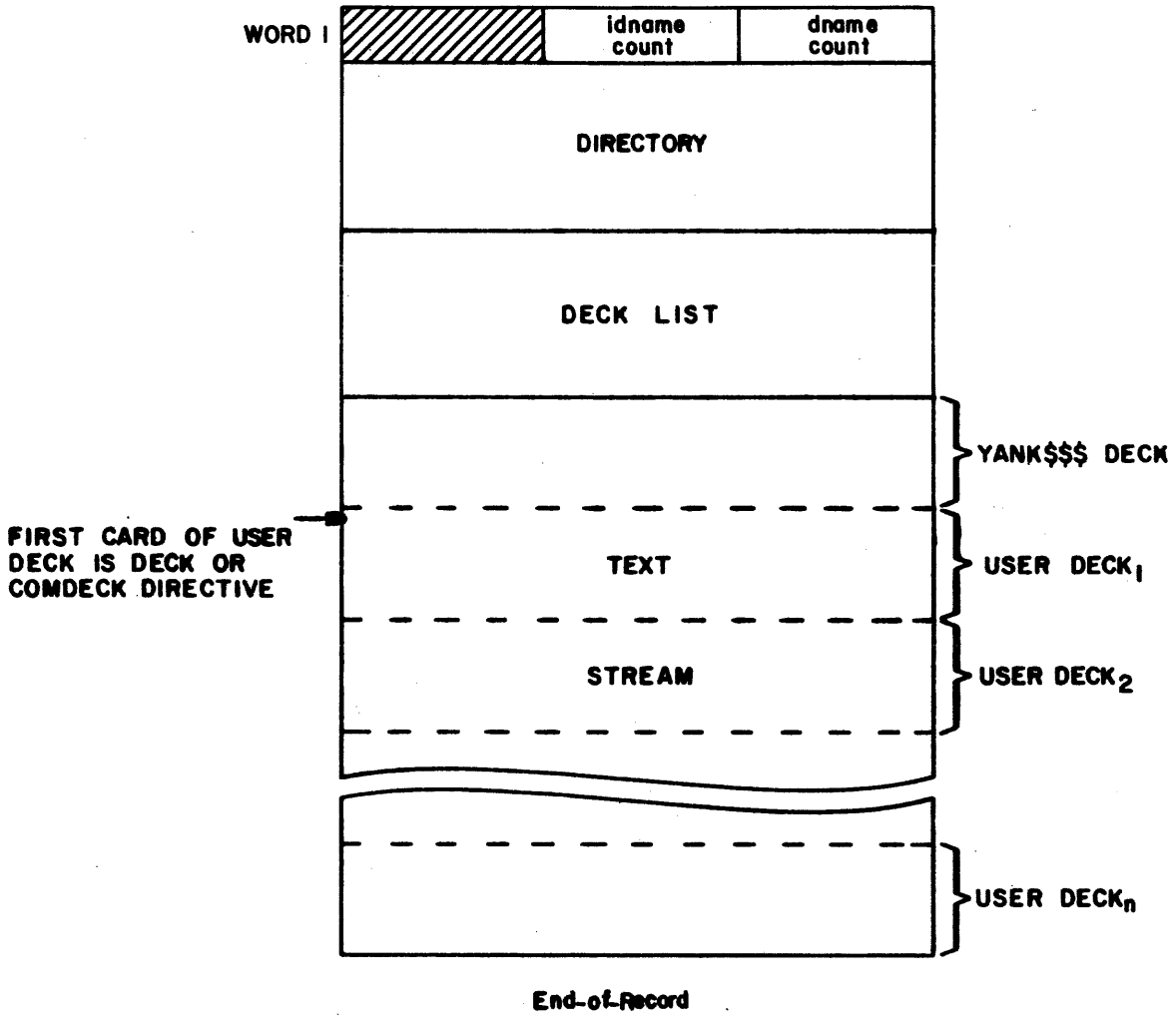


Figure 3-4. Old Sequential Program Library Format

3.3 INPUT FILES

The input file contains the UPDATE input record†. This record consists of directives and any source cards (including compile directives) to be inserted into the program library decks. The I option on the UPDATE control card (section 4.1.2) designates the file from which UPDATE reads directives. Normally, the input file is the job INPUT file. READ and ADDFILE directives can be used to direct UPDATE to stop reading directives from the primary input file named on the UPDATE card and to begin reading from some other file containing directives or insertion cards.

† Equivalent to a section for SCOPE 2

The Z option on the UPDATE control card designates that the input record is in compressed format created by a program such as the Production Control System (PCS).

3.4 COMPILE FILE

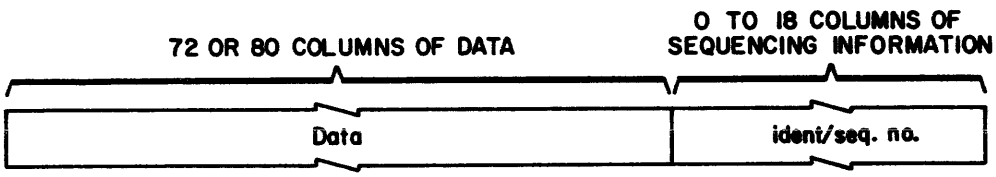
The COMPILE file is a primary form of output from UPDATE. It contains updated source card images to be assembled or compiled. The default name of the compile file is COMPILE, but the name of the file can be changed through the C option on the UPDATE control card (section 4.1.2).

During a full update run or during a creation run, the compile file contains all decks that are on the new program library. In normal mode (F and Q not selected), the compile file contains only decks updated during the current UPDATE run or decks specified on COMPILE directives. The sequence of the decks on the compile file is determined by control card options.

Through control card options, a user can specify whether the text on the file is to be compressed or expanded and sequenced or unsequenced.

The expanded compile file format for each card consists of 72 or 80 columns of data followed by 0 to 18 columns of sequence information. The maximum size of a card image is 90 columns.

Expanded card image:



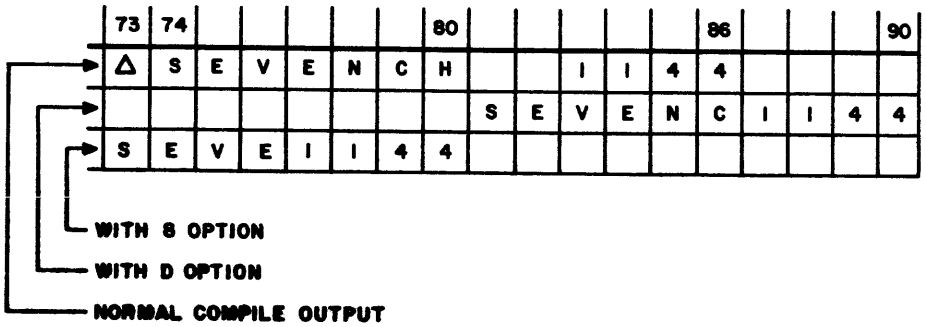
80- or 90- Column Card Image

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

When the data field is 72 and the card image is 80 columns, 8 columns are available for sequencing information. If the data field is 80 and the card image is 90, then 10 columns are available for sequencing information. In either of these cases, if the ident and sequence number exceed the field, UPDATE truncates the least significant characters of the ident leaving the sequence number intact.

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

In this example, the ident is SEVENCH, the sequence number is 1144:

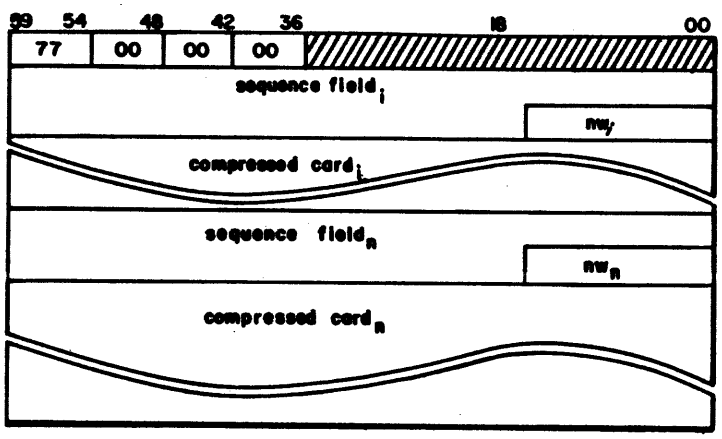


If the 80 (90) character card image on the compile file has two blanks as the last two characters, these are converted to a 0000 line terminator and the card image is 8 (or 9) words long.

If the last two columns do not contain blanks, a word containing 8 blanks and a zero byte line terminator are added, thus making the card 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 below. The first word is a loader prefix table (77_8). Compressed format is generated through the X option on the UPDATE control card.

Compressed Compile File Format:



sequence field$_i$	17 characters comprising card columns 74-90. Column 73 is always blank.
nw_i	Binary number of words in compressed card$_i$.
compressed card$_i$	Columns 1-72 of a COMPASS source card in compressed form. That is, 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_i is present, an extra all-zero word is not required to guarantee 12 zero bits.

3.5 PULLMOD FILE

A PULLMOD file is generated when the correction deck contains PULLMOD cards (section 2.8.6). The PULLMOD file is in the same format as an input file and contains directives and text of a recreated correction set.

If none of the G, S, or T options on the UPDATE control card is specified, pulled modifications are written on a file named SOURCE.

If S or T options are specified, and no specific file is defined through the G option; pulled modifications are written at the end of a source file.

3.6 SCRATCH FILES

UPDATE uses the following scratch files:

File Name	Functions	Comments
UPDTSCR	Used to make copy of decks to be written later to COMPILE file.	These files must be mass storage files. For SCOPE 3.x, NOS 1.0, and KRONOS files are evicted before, and closed and unloaded after the updating operation.
UPDTCDK	Used to hold common decks for later expansion of *CALL cards.	
UPDTTPL	Used as temporary program library.	For SCOPE 2, the files are not evicted before the operation but are closed and unloaded after the operation.
UPDTEXT	Used to copy card images to be inserted in correction run.	
UPDTAUD	Used to hold temporary audit information.	These are files that can be assigned to magnetic tape (SCOPE 3.x, NOS 1.0, and KRONOS systems).
UPDTPMD	Used to collect card images in response to PULLMOD directive.	

UPDATE EXECUTION 4

UPDATE is called from the library and placed in execution through an UPDATE call card.

4.1 CONTROL CARDS

The first record† of a job file contains a job card, the UPDATE call cards, and other optional cards described in the operating system reference manual. The record ends with an end-of-record† card. If UPDATE directives are on the job INPUT file, they usually comprise a record that follows the control card record. However, the placement of the record in the file depends on whether the job makes any calls prior to the UPDATE call.

4.1.1 JOB CARD

A job card of the following format must be the first card in the deck. The parameters following name can be in any order or can be omitted. For any omitted field, the operating system supplies a default value, which is an installation option.

Control card format:

```
jobname, Pp, Tt, CMscm, ECIcm.
```

jobname	1-7 character alphanumeric name by which the job is identified at the I/O station. The first character must be a letter.
Pp	Job priority; see operating system reference manual.

†Section for SCOPE 2.1.

Tt	CPU time limit in octal seconds ($1\text{-}7777_8$); must be sufficient to process all control cards for the job, including assembly and execution.
CMscm	Estimate of maximum amount of SCM or CM required for execution (1-6 octal digits). If the DYNAMFL installation option has been assembled, core allocation is handled dynamically and the default value for the operating system will usually suffice. If the DYNAMFL option has not been assembled, a minimum of 45000 should be specified.
ECIcm	Estimate of maximum amount of LCM/ECS in octal thousands. UPDATE has no LCM/ECS requirements.

4.1.2 UPDATE CALL CARD

The following control statement causes the UPDATE program to be loaded from the SYSTEM library and to be executed. Parameters specify modes and files for the run. A period or right parenthesis terminates the statement.

UPDATE ($p_1, p_2, \ldots, p_n$)

The word UPDATE begins in column one. The optional parameters, p_i, can be in any order within the parentheses. Generally, a parameter can be omitted or can be in one of the following forms:

option

option=filename

option=0 (valid for modes C and L only)

Option	Significance
A - Sequential-to-random copy	
omitted	No special mode
A	When this mode is selected UPDATE copies a sequential old program library to a random new program library. It performs no other UPDATE operations. The only control card options that can be used are those specifying files, and * and /. An error results if the old program library is not sequential or the new program library is not random.
B - Random-to-sequential copy	
omitted	No special mode
B	When this mode is selected UPDATE copies a random old program library to a sequential new program library. It performs no other UPDATE operations. The only other control card options that can be used, are those specifying files, and * and /. An error results if the old program library is not in random format.

Option	Significance
C - Compile file output	
omitted or C	Compile output decks will be written on file COMPILE; contents are determined by type of UPDATE (F, Q, or normal).
C=filename	UPDATE writes compile output decks on named file; contents are determined by type of UPDATE (F, Q, or normal).
C=PUNCH	This is a special form of C=filename. UPDATE writes compile output decks on PUNCH file. Decks are punched according to type of UPDATE (F, Q, or normal). This option also causes the 8 and D modes to be selected.
C=0	No compile output
D - Data width	
omitted	Compile output has 72 columns for data
D	Compile output has 80 columns for data
E - Edit; provides a means of cleaning up old program libraries †	
omitted	The old program library is not edited.
E	UPDATE rearranges the directory to reflect the actual order of decks on the program library. First, editing removes from the directory all previously purged idents and reassigns ordinals to idents in the directory according to the actual order of decks on the program library. Then editing purges the idents that exist simply as entries in the directory and have no cards or chb's associated with them. Idents purged on this run are removed from the directory in a subsequent edit run. Thus, to completely edit the library requires two edit runs. The first edit run detects unused idents and flags them as purged. The second edit run deletes the unused idents from the directory. Again, any idents purged on the second run would require yet another edit run before the directory entries would be deleted. During editing, UPDATE purges any cards other than YANK, SELYANK, and YANKDECK from the YANK$$$ deck (section 3.2.1).

†EDITKEY installation option

Option	Significance
F - Full update	
omitted	If Q is not specified, F omitted is the normal (selective) UPDATE mode. All regular decks and common decks are processed. The new program library, if specified, contains all regular and common decks, after any corrections have been made, in the sequence in which they occur on the old program library. The source file, if specified, contains all active cards with decks in deck list sequence. The compile file contains all decks corrected during this UPDATE run and all decks specified on COMPILE directives (section 2.6.1). A deck that calls a corrected common deck is also considered to be corrected unless the common deck is a NOPROP deck (see COMDECK directive) or unless the deck precedes the commom deck.
F	Source and compile files, if specified, contain all active decks in old program library sequence. The contents of the new program library are the same as if F were not specified.
G - Generate separate PULLMOD output file†	
omitted	Output from PULLMOD cards is appended to source file defined by S or T option or to the SOURCE file.
G=filename	Output from PULLMOD cards is written on named file. Any rewind option applying to the source file also applies to this file. OUTPUT is not a valid file for this option.
H - Header change	
omitted or H	UPDATE treats the old program library character set as the character set type indicated in the old program library header word.
H=3	UPDATE treats the old program library as a 63-character set program library regardless of the character set type specified in the old program library header word.
H=4	UPDATE treats the old program library as a 64-character set program library regardless of the character set type specified in the old program library header word.
I - Input	
omitted or I	Input is on job INPUT file
I=filename	Input comprises next record on named file
K - COMPILE card sequence (takes precedence over C mode)	
omitted	Output determined by C option.
K	Compile output decks to be written on file COMPILE in COMPILE directive sequence, that is, the order in which decknames are encountered on *COMPILE directives. If a deckname is mentioned more than once, its latest specification determines the deck's place within the COMPILE directive sequence.
K=filename	Compile output decks to be written on named file in COMPILE directive sequence.

†PMODKEY installation option

Option	Significance
L - List options	
Omitted	If creation run, L=A12 is automatically selected. If correction run, L=A1234 is automatically selected. If A or B mode, L=A1 is automatically selected.
$L=C_1C_2...C_n$	List options selected.

Character	Controls
A	Lists the following 1. Known DECK names 2. Known IDENT names 3. COMDECK directives that were processed (subset of 1) 4. Decks written on the compile file 5. Known definitions (see *DEFINE directive)
F	All selections other than 0.
0	Suppresses all UPDATE listing. If the digit 0 is included in the string of other list selections, the string is equivalent to L=0.
1	Lists cards in error and the associated error messages. The flag *ERROR* is appended to the left and right of each card in error.
2	Lists all active UPDATE directives encountered either on input 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 card 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 card belongs is printed in place of the five asterisks/slashes.
3	Lists all cards that changed status during this UPDATE. This listing consists of the name of the deck to which the card belongs, the card image, card identifier with sequence number, and a key as shown below:

Key	Meaning
I	Card was introduced
A	Inactive card was reactivated
D	Active card was deactivated
P	Card was purged
SEQ	Card was resequenced

If a currently active card is purged, the word ACTIVE appears to the right of the P.

Option	Significance	
L (continued)	Character	Controls
	4	Lists all non-UPDATE directives encountered in the input stream. Cards resulting from a *READ directive are marked to the right with the name of the file from which they were read. Decks inserted by ADDFILE are not listed if list option 4 is selected by default but they are listed if list option 4 is explicitly selected. Option 4 may be turned on by a LIST card (see *LIST directive) and off by a NOLIST card (see *NOLIST directive).
	5	All active compile file directives
	6	Number of active and inactive cards by deck name and correction set identifier
	7	All active cards
	8	All inactive cards
	9	Correction history of all cards listed as a result of list options 5, 7, and 8.

List options 5-9 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 output (O) file at the end of the UPDATE. When the F option is selected, options 5-9 apply to all decks on the old program library. If F is not selected, options 5-9 apply to decks listed on COMPILE directives only.

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.

Cards listed under option 7 are marked to the right of the card by the letter A (active); cards listed under option 8 are marked by the letter I (inactive).

M - Merge input ; allows two program libraries to be merged and written onto the new program library (see N option).

The old program library is considered the master file. UPDATE adds the directory and deck list from the merge file to the directory and list on the old program library. Deck names and ident names on the merge file that duplicate names on the old program library are modified to make them unique as follows:

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

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

3. If no unique name can be generated by this method, the UPDATE run is abnormally terminated.

Decks from the merge file are added to the new program library file after all decks from the old program library have been added unless the sequence is altered by MOVE directives (section 2.4.16). All deck and ident names that required modification are listed.

Option	Significance

M (continued)

UPDATE changes all DECK, COMDECK, YANK, SELYANK, YANKDECK, and CALL directives containing references to modified deck or ident names to agree with the new names.

Sequencing is unchanged. All UPDATE functions legal in a correction run are legal with the merge option, including use of the MOVE directive to specify the sequence in which merged decks are to be written on the new program library.

Use caution in including modifications in a merge run. UPDATE may change a deck or ident name to which correction cards have been applied. In this case, corrections may refer to the wrong deck or ident.

omitted	No merge file
M	Second old program library on file MERGE
M=filename	Second old program library on named file

N - New program library output

A new program library can be in random or sequential format. The format is determined by file residence (mass storage or magnetic tape) and/or by specification of the UPDATE W option as shown in the following table.

	SCOPE 3.x/NOS 1	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 card punch or W is selected.	File is staged or on-line tape or is on mass storage as record type S or record type W blocked, or W is selected or R specifies no rewind.

N	New program library to be written on file NEWPL
N=filename	New program library to be written on named file.
omitted	UPDATE does not generate a new program library.

O - List output file

omitted or O	List output is written on job OUTPUT file. This file is automatically printed.
O=filename	List output is written on named file

P - Old program library; ignored on creation run.

omitted or P	Old program library on file OLDPL
P=filename	Old program library on named file

Option	Significance

Q - Quick update (takes precedence over F)

omitted	If F is not specified, this is the normal (selective) mode. See F omitted.
Q	Only decks specified on COMPILE directives and decks added via ADDFILE directives are processed. Corrections other than ADDFILE that reference cards in decks not specified on COMPILE directives are not processed and UPDATE abnormally terminates after printing the unprocessed corrections. The compile file contains decks specified on COMPILE directives and any common decks called from decks plus those decks added via ADDFILE and their called common decks. The contents of the source file and the new program library, if specified, depends on type of old program library (random or sequential) and on deck names specified on COMPILE directives. If the old program library is sequential, the new program library contains all decks mentioned on COMPILE directives as well as all common decks they call and common decks encountered on the old program library prior to processing of all of the specified decks. The source file contains the same active cards that are written on the new program library if a new program library is selected. If the old program library is random, the new program library and the source file are the same as for a sequential with the exception that the only common decks included are those called by decks specified on COMPILE directives.

CAUTION

In Q-mode using a random old program library, a single correction IDENT containing corrections to both a DECK and a COMDECK may 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 card and/or the old program library is sequential, the sequence numbers assigned to the text cards 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

omitted	The old and new program libraries, the compile file, and the source file are rewound before and after the UPDATE run.
R	No rewinds are issued for the program libraries, compile file, or source file.
R=$c_1c_2...c_n$	Each character in string indicates a file to be rewound before and after the UPDATE run.

c	File
C	Compile
N	New program library
P	Old program library and merge library
S	Source and PULLMOD

Option	Significance

S - Source output; the contents of the source file are determined by the mode in which UPDATE is operating, the decks named on COMPILE directives, and the type of old program library in use (randon or sequential).

1. If Q is not selected (regardless of F), the source file contains all cards required to recreate the library. This recreated library is resequenced because sequencing information is not included on the source file. This file contains all currently active DECK, COMDECK, WEOR, CWEOR, CALL, TEXT, IF, ENDIF, and ENDTEXT directives in addition to all active text information. Decks are not necessarily in a sequence accepted by UPDATE for creating a new program library (section 3.1.2).

2. If Q is selected and the old program library is sequential, decks written on the source file are those named on COMPILE directives and common decks they call. All common decks encountered on the old program library before all explicitly specified decks are included on the source file.

   If Q is selected and the old program library is random, decks written on the source file are those named on COMPILE directives. The only common decks included are those called by decks named on COMPILE directives.

omitted	UPDATE does not generate a source output file unless the source output is specified by T.
S	Source output written on file source.
S=filename	Source output written on named file.

T - Source output excluding common decks (takes precedence over S)

omitted	No source output unless source output specified by S.
T	Source output excluding common decks on file SOURCE.
T=filename	Source output excluding common decks on named file.

U - Debug mode

omitted	UPDATE execution terminates upon encountering a fatal error.
U	UPDATE execution is not terminated by normally fatal errors.

W - Sequential new program library

omitted	The new program library (see N option) will be determined by characteristics of the file specified by N.
W	The new program library (see N option) will be a sequential file.

Option	Significance
X - Compressed compile file	
omitted	Compile file is not in compressed format
X	Compile file is in compressed format (section 3.4)
Z - Compressed input file	
omitted	The input file (see I option) is a normal, coded file.
Z	The input file (see I option) is assumed to be in PCS compressed format. This parameter applies to the directives input file only; it does not apply to files specified by READ directives.
8 - 80-column output on compile file	
omitted	Compile file output is composed of 90-column card images (section 3.4).
8	Compile file output is composed of 80-column card images.
* - Master control character†	
omitted	The master control character is *.
*=char	The master control character (first character of each directive) for this UPDATE run is char which can be any character having a display code octal value in the range 01-54 except for 51 and 52 (the open and close parentheses).

On a correction run, if the master control character is not the same as the character used when the old program library was created, UPDATE uses the character indicated on the old program library. On a merge run, if the master control characters for the two libraries to be marged are not the same, UPDATE aborts the run.

/ - Comment control character†	
omitted	The comment control character (section 2.8.1) is /
/=char	The comment control character for this UPDATE run is char which can be A through Z, 0 through 9 or + - * / $ = Note, however, that the character should not be changed to one of the abbreviated forms of directives unless NOABBREV is in effect.

† Under SCOPE 2, SCOPE 3.4, KRONOS 2.1, or NOS 1.0 specify a $ character as *=$$$$ or /=$$$$.

4.1.3 7/8/9 CARD

The card that separates records† in the job deck is characterized by having rows 7, 8, and 9 punched in column one. The level is assumed zero unless columns 2 and 3 contain an octal level number punched in Hollerith code. The remainder of the columns optionally contain comments.

As an example, a deck consisting of a control card record, an UPDATE input record, and a data record would include two 7/8/9 cards. The first terminates the control cards and the second terminates UPDATE input. A 6/7/8/9 card would terminate the deck.

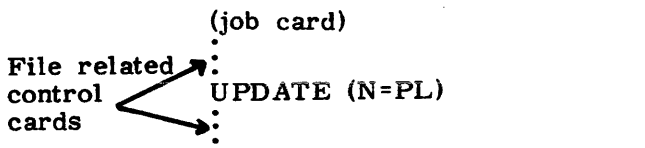
4.1.4 6/7/8/9 CARD

The card that signals the end of the job deck is characterized by having rows 6, 7, 8, and 9 punched in column one. Columns 2-80 optionally contain comments.

4.2 DECK EXAMPLES

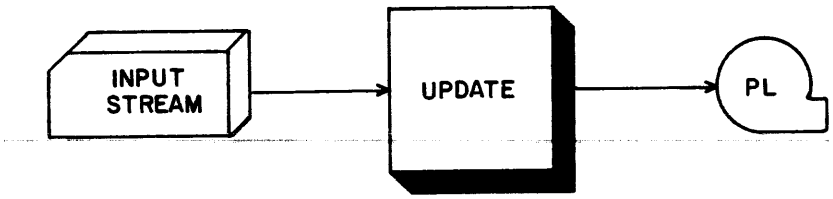
4.2.1 LIBRARY FILE CREATION

1. Creation of a program library called PL, consisting of four decks, the first two written in COMPASS, the next two in FORTRAN, and generation of a compile file to be processed by the assembler and compiler requires the following deck structure:



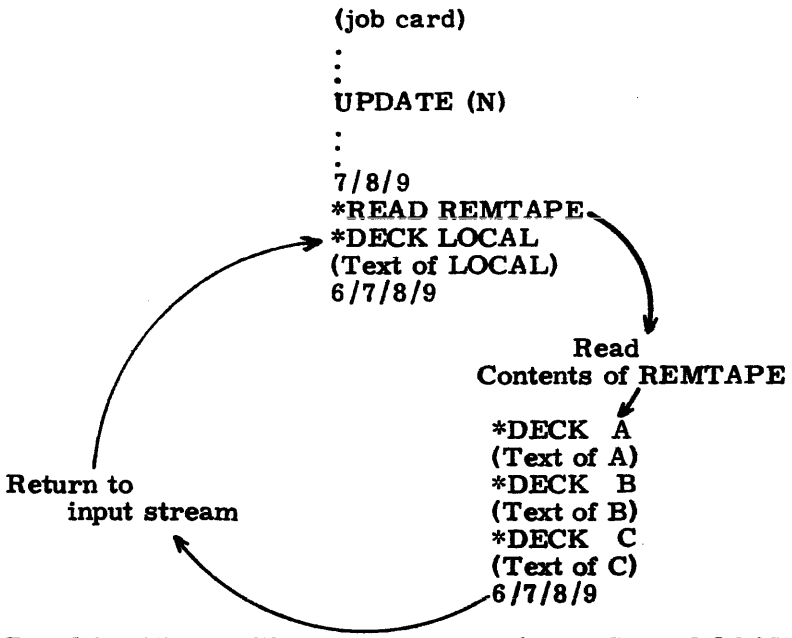

```
(job card)
.
.
.
UPDATE (N=PL)
.
.
.
7/8/9
*DECK COMPGROUP
(First COMPASS source deck)
*DECK COMPGRP1
(Second COMPASS source deck)
*WEOR
*DECK FORTGROUP
(First FORTRAN source deck)
*DECK FORTGRP1
(Second FORTRAN source deck)
6/7/8/9
```

The compile file produced by this run contains two logical records as a result of the WEOR directive. Grouping of more than one source program written in a given language is permitted following one DECK directive. Such is the case when modification of one program requires reassembly of all programs in the group.

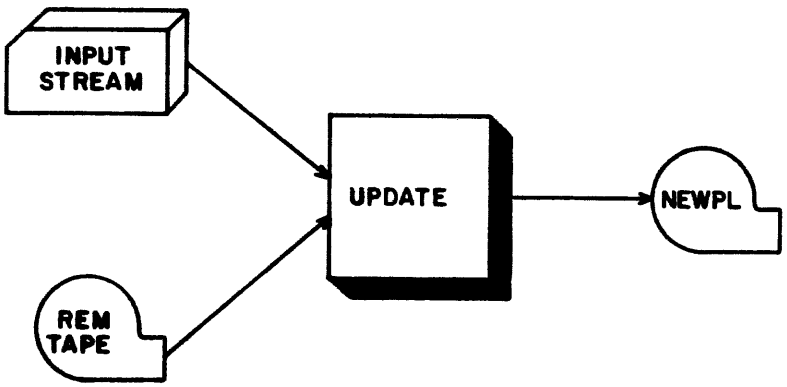


† Separates sections for SCOPE 2.

2. Read from alternate file.



Resulting library file contains decks A, B, C and LOCAL.



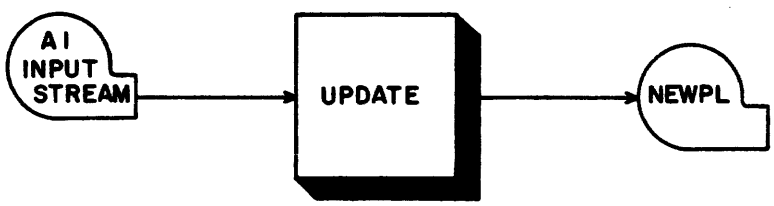
3. Create program library NEWPL containing two program decks and two common decks:

```
(job card)
.
.
.
UPDATE (N)
.
.
.
7/8/9
*COMDECK  D1
.
.
.
*COMDECK  D2
.
.
.
*DECK  XA
.
.
.
*DECK  XB
.
.
.
*CALL  D2
6/7/8/9
```

The COMPILE file that is produced by default contains decks XA and XB in that order. Deck XB has been expanded to contain common deck D2.

4.2.2 INPUT FILE NOT INPUT

Contents of File A1	Identifier given to card (does not appear on file A1):
*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
END	SET1.4
*DECK SET2	SET2.1
SUBROUTINE JIM	SET2.2
A = B - SIN(C)	SET2.3
END	SET2.4
6/7/8/9	



For the UPDATE task:

```
(job card)
.
.
.
UPDATE(I=A1,N)
.
.
.
6/7/8/9
```

The compile file contains two source decks having cards that are identified by their deck name and sequence number:

```
      PROGRAM ZIP                    SET1.2
C     A DO-NOTHING JOB               SET1.3
      END                            SET1.4
      SUBROUTINE JIM                 SET2.2
      A = B - SIN(C)                 SET2.3
      END                            SET2.4
```

4.2.3 INSERTIONS/DELETIONS/COPYING

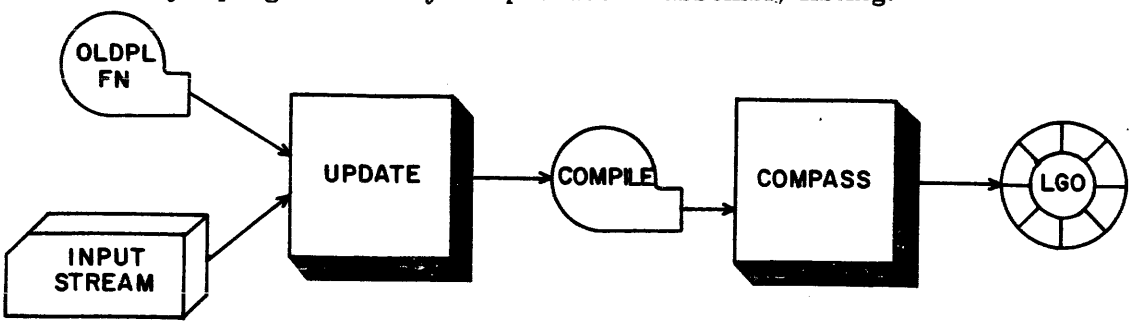
1. To alter the library created by preceding example:

```
(job card)
.
.
.
UPDATE                          (OLDPL is file produced by
7/8/9                           preceding example)
*IDENT ADD1
*DELETE SET1.3,SET1.4
*CALL CSET
  B=1.0                         } Inserted into SET1
  C=3.14159                     }
  CALL JIM                      }
*COPY SET1,SET1.4
*COPY SET2,SET2.2
*CALL CSET                      Inserted into SET1 with identification ADD1.7
*COPY SET2,SET2.3,SET2.4
6/7/8/9
```

Deck SET1 is on the COMPILE file as:

```
PROGRAM ZIP                     SET1.2
COMMON A,B,C                    CSET.2
B=1.0                           ADD1.2
C=3.14159                       ADD1.3
CALL JIM                        ADD1.4
END                             ADD1.5
SUBROUTINE JIM                  ADD1.6
COMMON A,B,C                    CSET.2
A = B - SIN(C)                  ADD1.8
END                             ADD1.9
```

2. To modify a program library and produce an assembly listing:




```
(job card)
UPDATE(P=FN)
.
.
.
COMPASS(I=COMPILE)
.
.
.
7/8/9
*IDENT CS1
*INSERT XA.1
(Insertions)
*DELETE XA.20,XA.23
6/7/8/9
```

Old program library file FN was created in a prior run.

The COMPILE file that is read by COMPASS contains deck XA since that deck was modified by UPDATE.

3. To generate a new sequential program library with corrections:

```
(job card)
.
.
.
UPDATE(P=FN1,N=FN2,W)
.
.
.
7/8/9
*IDENT XRAY
*DELETE D2.3,D2.5
*INSERT D2.8
(Insertions)
6/7/8/9
```

Old program library FN1 was created in a prior run.

4.2.4 YANKING AND PURGING

1. To reverse the effect of a correction set temporarily, but not permanently remove it from the program library LIB:

   This change may be made temporarily for testing purposes:

```
(job card)
.
.
.
UPDATE(P=LIB)
COMPASS(I=COMPILE)
7/8/9
*IDENT NEGATE
*YANK GOTTOGO
6/7/8/9
```

2. To put the preceding change onto a new program library.

```
(job card)
:
:
UPDATE(P=LIB, N=NEWLIB)
:
:
COMPASS(I=COMPILE)
7/8/9
*IDENT NEGATE
*YANK GOTTOGO
6/7/8/9
```

3. The YANK directive in the above example becomes the first card on the new library NEWLIB. The identifier for this card is NEGATE.1. The effects of YANK may be nullified in future runs (and consequently the effects of GOTTOGO restored) by specifying:

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

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

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

5. It could be removed temporarily by specifying:

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

6. Within deck ZOTS on the old program library are correction cards introduced by correction set DART. A later correction set contained a YANK directive that yanked correction set DART. Now the user wishes to nullify a portion of the YANK that affected the cards following ZOTS.19 through ZOTS.244. All other cards belonging to correction set DART are to remain yanked. Inserting a DO at ZOTS.19 and a DONT at ZOTS.244 causes UPDATE to temporarily rescind the YANK while writing decks ZOTS on the compile file. The input stream includes:

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

The DO and DONT directives are considered as text when the input stream is read.

7. 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. Cards introduced by OLDMOD are deactivated; cards deactivated by OLDMOD are reactivated.

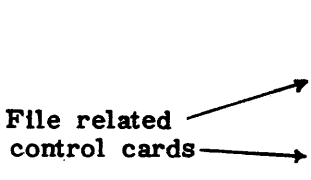
```
*SELYANK OLDDECK.OLDMOD
```

This directive accomplishes the same effect as the YANK directive above except its effect is limited to cards within the deck OLDDECK.

```
*YANKDECK OLDDECK
```

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

8. Correction sets BAD, WORSE, and WORST are no longer needed. The following job removes them from the new library (NEWPL).




```
(job card)
.
.
.
UPDATE(N, C=0, L=12)
.
.
.
7/8/9
*PURGE BAD, WORSE, WORST
6/7/8/9
```

9. Sequential program library LIBAUG has been modified periodically over a number of months. It becomes desirable to return to a previous level. The following deck sequence illustrates this use of UPDATE.

```
(job card)
.
.
.
UPDATE(N=LIBMAY, P=LIBAUG, C=0)
.
.
.
7/8/9
*PURGE JUNMOD1,*
6/7/8/9
```

LIBAUG is the most recent (August) version of the program library. This deck recreates a library modified only through May. The deck purges all modifications made after May (beginning with JUNMOD1 in the directory).

10. Deck BAD on program library LIB is no longer of use and is to be removed permanently from the program library. The following deck sequences illustrate such permanent removal:

```
(job card)
.
.
.
UPDATE(P=LIB, N=LESSBAD, C=0)
.
.
.
7/8/9
*PURDECK BAD
6/7/8/9
```

LIB is the most recent program library.

LESSBAD is the new program library with BAD purged.

*PURDECK purges all cards within deck BAD.

The deck BAD is removed from the library. If BAD is also a correction set identifier name, that idname would not be purged. *PURDECK operates so that any cards having the identifier BAD but physically located outside of the deck BAD are not purged. A *PURGE BAD directive should be added to the above run if the correction set identifier name is to be purged. If BAD was previously yanked, the *YANK BAD card must be deleted from the YANK$$$ deck.

4.2.5 ADDITION OF DECKS

A new program library, NEWPL, is to be constructed from the old program library, OLDPL, with the addition of one new common deck and two new decks. The new common deck, D1A, will be the first deck after the YANK$$$ deck; the new deck XC will follow deck XB; and the new deck SYSTEXT will be the last deck on the new program library. No compile file will be produced.

1. All three of the ADDFILEs are to be read from the main INPUT file:

```
(job card)
.
.
.
UPDATE(N,C=0)
.
.
.
7/8/9
*ADDFILE INPUT,YANK$$$ or *ADDFILE ,YANK$$$
*COMDECK D1A
.
.
.
*ADDFILE INPUT or *ADDFILE
*DECK SYSTEXT
.
.
.
*ADDFILE INPUT,XB or *ADDFILE ,XB
*DECK XC
.
.
.
6/7/8/9
```

2. All three of the ADDFILEs are to be read from the UPDATE input file; FNAME:

```
(job card)
.
.
.
UPDATE(N,C=0,I=FNAME)
.
.
.
6/7/8/9
```

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
.
.
.
```

3. Each of the three ADDFILE directives will cause UPDATE to read from a separate file, none of which is the UPDATE input file. Common deck D1A and its text are on FILEA; deck SYSTEXT and its text are on FILEB; and deck XC and its text are on FILEC.

```
(job card)
.
.
.
UPDATE(N,C=0)
.
.
.
7/8/9
*ADDFILE FILEA, YANK$$$
*ADDFILE FILEB
*ADDFILE FILEC, XB
6/7/8/9
```

4.2.6 Q OPTION

UPDATE places the deck DNAME from file FN1 on the COMPILE file. COMPASS reads deck DNAME from COMPILE file and assembles it.

```
(job card)
:
:
UPDATE(Q, P=FN1)
COMPASS(I=COMPILE)
7/8/9
*IDENT YOKE
(Correction Set for DNAME)
*COMPILE DNAME          Deck DNAME to be placed on COMPILE file for
                        COMPASS assembly.
6/7/8/9
```

4.2.7 PULLMOD OPTION

The library created by the example shown in section 4.2.2 has been altered by the following correction run:

```
(job card)
:
:
UPDATE(N=PL2)
:
:
7/8/9
*IDENT  PMEX
*DELETE SET1.3
C         THIS IS FOR PULLMOD EXAMPLE
*COMPILE  SET1
6/7/8/9
```

One of the decks on PL2 is SET1:

```
*DECK    SET1                                          SET1.1
           PROGRAM ZIP                                 SET1.2
C          THIS IS FOR PULLMOD EXAMPLE                 PMEX.1
           END                                         SET1.4
```

Pull the modification as follows:

```
(job card)
:
:
UPDATE(G=PMFILE,  P=PL2)
7/8/9
*PULLMOD  PMEX
6/7/8/9
```

File PMFILE contains:

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

4.2.8 PROGRAM LIBRARY AS SCOPE 3.4 PERMANENT FILE

1. Create program library and catalog as SCOPE 3.4 permanent file.

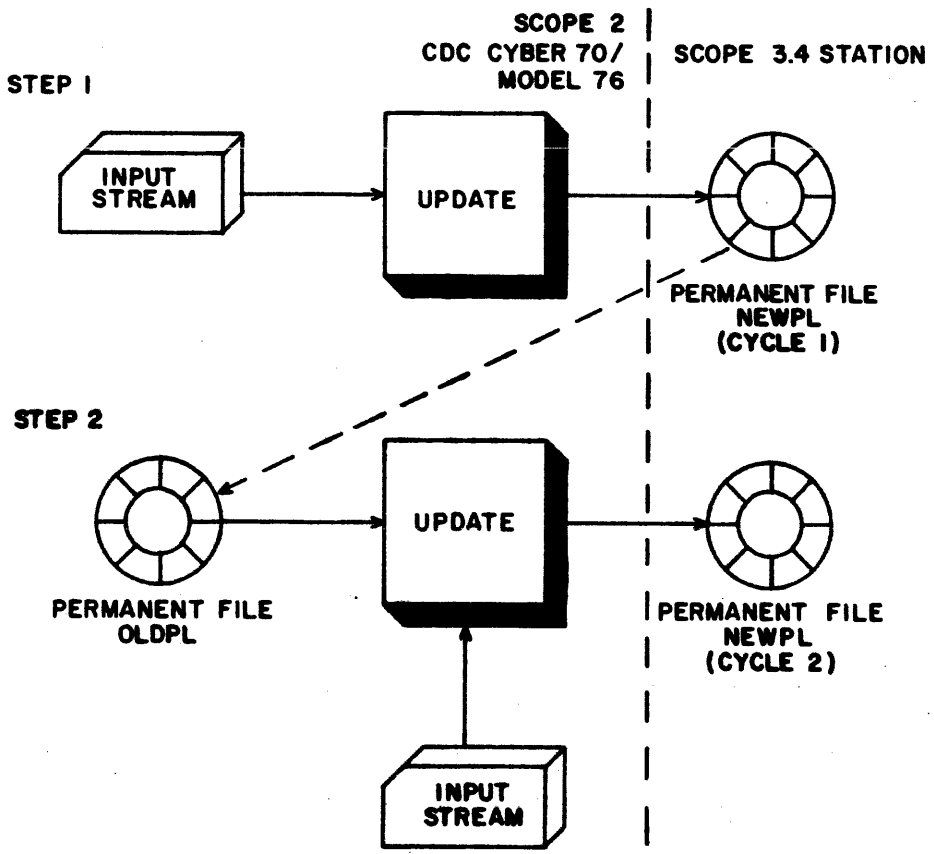
```
SCOPE 3.4 DECK

(job card)
REQUEST(PL, *PF)
UPDATE(N=PL, W, L=1234)
CATALOG(PL, PERMUPLIB, ID=JONES, PW=*****)
7/8/9
(source decks)
6/7/8/9
```

2. Update program library using UPDATE under either SCOPE 2 or SCOPE 3.4.

```
JOBCARD, CP76.
REQUEST(NEWPL, *PF)
FILE(OLDPL, RT=S)
FILE(NEWPL, RT=S)
ATTACH(OLDPL, PERMUPLIB, ST=SVL, †ID=JONES, TK=*****)
UPDATE(P=OLDPL, N=NEWPL, L=1234, W)
CATALOG(NEWPL, PERMUPLIB, ST=SVL, †ID=JONES, CY=2, PW=*****)
7/8/9
(correction deck)
6/7/8/9
```

†Station ID ignored when job is processed by SCOPE 3.4.


SCOPE 2
CDC CYBER 70/
MODEL 76
SCOPE 3.4 STATION
STEP 1
INPUT
STREAM
UPDATE
PERMANENT FILE
NEWPL
(CYCLE 1)
STEP 2
UPDATE
PERMANENT FILE
OLDPL
PERMANENT FILE
NEWPL
(CYCLE 2)
INPUT
STREAM

4.3 SAMPLE FORTRAN EXTENDED PROGRAM

This set of UPDATE examples illustrates how UPDATE can be used for maintaining a FORTRAN Extended program in program library format. The FORTRAN program is very simple. It calculates the area of a triangle from the base and height read from the data record.

1. The following job 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, FTN is called to compile the program; the source is on the COMPILE file. The LGO card 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.

```
(JOB1 CARD)
  .
  .
  .
UPDATE (P=0, N, F)
FTN(I=COMPILE)
LGO.
  .
  .
  .
7/8/9  END-OF-RECORD
*DECK ONE
      PROGRAM ONE (INPUT, OUTPUT, TAPE1)
      PRINT 5
5     FORMAT (1H1)
10    READ 100, BASE, HEIGHT, I
100   FORMAT (2F10.2, I1)
      IF (I.GT. 0) GO TO 120
      IF (BASE. LE. 0) GO TO 105
      IF (HEIGHT. LE. 0) GO TO 105
      GO TO 106
      CALL MSG
106   AREA = .5*BASE*HEIGHT
      PRINT 110, BASE, HEIGHT, AREA
110   FORMAT (///, * BASE=*F20.5, * HEIGHT=*
     I F18.5, 1, *AREA=*F20.5)
      WRITE (1) AREA
```

File related cards

Deck name

```
          GO TO 10
   120    STOP
          END
          SUBROUTINE MSA
          PRINT 400
   400    FORMAT (///,* FOLLOWING INPUT DATA NEGATIVE OR ZERO *)
          RETURN
          END
 7/8/9    END-OF-RECORD
          200.24        500.76
          300.24        600.76
          400.00        700.00
          326.32        425.36
          500.00        600.00
          000.00        150.00
          700.43        800.00
          100.00        300.00
          050.00        100.00
          150.00        200.00
                              1
6/7/8/9   END-OF-INFORMATION
```

Should be SUBROUTINE MSG

End of source deck

Data record

2. Examination of UPDATE output from the creation job reveals that the erroneous SUBROUTINE statement has card identifier ONE.20. The following job corrects the error and generates a new program library.

```
(JOB2 CARD)
   •
   •
   •
UPDATE(N,F)
FTN(I=COMPILE)
LGO.
7/8/9     END-OF-RECORD
*IDENT    MOD1
*DELETE ONE.20
          SUBROUTINE MSG
7/8/9
```

Identified as MOD1.1 on NEWPL

```
200.24      500.76
300.24      600.76
400.00      700.00
326.32      425.36
500.00      600.00
000.00      150.00
700.43      800.00
100.00      300.00
050.00      100.00
150.00      200.00
                  1
6/7/8/9   END OF INFORMATION
```

Data record

3. This job uses the same input as the first job but divides the program into two decks, ONE and MSG. Deck MSG is a common deck. A CALL MSG directive is inserted into deck ONE to assure that MSG is written on the compile file whenever deck ONE is edited.

```
(JOB3 CARD)
UPDATE(P=0, N, F)
FTN(I=COMPILE)
LGO.
  •
  •
  •
7/8/9    END-OF-RECORD
*COMDECK MSG
         SUBROUTINE MSG
         PRINT 400
400      FORMAT (///,* FOLLOWING INPUT DATA NEGATIVE OR ZERO *)
         RETURN
         END
*DECK ONE
         PROGRAM ONE (INPUT,OUTPUT,TAPE1)
         PRINT 5
5        FORMAT (1H1)
10       READ 100,BASE,HEIGHT,I
100      FORMAT (2F10.2, I1)
         IF (I.GT. 0) GO TO 120
         IF (BASE. LE. 0) GO TO 105
         IF (HEIGHT. LE. 0) GO TO 105
         GO TO 106
```

File related cards

```
105       CALL MSG
106       AREA = .5*BASE*HEIGHT
          PRINT 110, BASE, HEIGHT, AREA
110       FORMAT (///,* BASE=*F20.5,* HEIGHT=*
         I F18.5, 1, *AREA=*F20.5)
          WRITE (1) AREA
          GO TO 10
120       STOP
          END
*CALL  MSG
7/8/9     END-OF-RECORD
          200.24          500.76
          300.24          600.76
          400.00          700.00
          326.32          425.36
          500.00          600.00
          000.00          150.00
          700.43          800.00
          100.00          300.00
          050.00          100.00
          150.00          200.00
                          1
6/7/8/9   END OF INFORMATION
```

Replaced by common deck MSG on compile file

Data record

4. This example adds a deck to the library created in the previous example. Since no new program library is generated (N is omitted from UPDATE card), the addition is temporary.

```
(JOB4 CARD)
  •
  •
  •
UPDATE.
FTN(I=COMPILE)
LGO.
  •
  •
  •
7/8/9     END-OF-RECORD
```

File related cards

```
*ADDFILE, ONE
*IDENT MOD2
*INSERT ONE.20
*DECK TWO
      PROGRAM TWO(INPUT,OUTPUT)
        .
        .
        .
      END
*CALL MSG
*DELETE MSG.3
400 FORMAT (///,* FOLLOWING INPUT DATA POSITIVE *)
*COMPILE TWO
7/8/9
(DATA RECORD)
6/7/8/9
```

Replaced by common deck MSG on compile file

OVERLAPPING CORRECTIONS A

When the EXTOVLP installation option has been assembled, UPDATE detects four types of overlap involving two or more cards in a correction set.

Type 1 Two or more modifications to one card made in a single correction set.

Type 2 Attempt to activate an already active card.

Type 3 Attempt to deactivate an already inactive card.

Type 4 Insertion after a card that was inactive on the old program library.

When any of these types is detected, UPDATE prints the offending line with the words TP.n OVLOP appended on the far right. The listing of overlap lines is controlled by list option 3. If any overlap condition is encountered during a run, this dayfile message is printed: number OVERLAPPING CORRECTIONS.

Detection of an overlap does not necessarily indicate a user error. Overlap messages are advisory, and indicate conditions in which the probability of error is greater than normal.

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

Types TP.1 and TP.3 are normally detected during an UPDATE RUN and usually can be ignored.

Modifications for each correction set are performed by UPDATE in the order in which sets are introduced. The order is irrelevant if no correction is dependent on another. If a dependent relationship exists, however, order is of paramount importance.

LISTABLE OUTPUT B

Creation of a new program library produces a listing of all file manipulation and creation cards and a list of deck names and correction set names known at the end of the UPDATE run, as well as error diagnostic messages.

During a correction pass, the listings are more detailed. The first listing is a printout of the correction sets as encountered. Each IDENT (or PURGE) appears on a titled page. A printout of each card image on the input file follows. All cards resulting from the READ directive are included and identified on the right by the file from which they were read.

The second set of listings, a continuous commentary of all effective changes introduced to the file, includes all purged cards as well as cards for which the activity status changed since they were placed on the program library. Cards inserted by the ADDFILE statement are not listed. The right-hand side of the listing shows each card identifier with the idname followed by the sequence number. The deck name is to the left of the card image.

Diagnostic messages are listed as they occur. Whether or not the updating process is successful, an appropriate dayfile message appears.

When a COMPILE file is written, the locations of all CWEOR, WEOR, and CALL directives are listed. If L=0 on the UPDATE card, all listable output from UPDATE is suppressed. If L=1, the deck name list, identifier list, and continuous commentary are suppressed.

Messages on UPDATE Listing

n ERRORS IN UPDATE INPUT

Dayfile message. First pass of UPDATE processing encountered n fatal errors while reading a correction set.

x DECK STRUCTURE HAS BEEN CHANGED

Informative dayfile message.

x DECLARE ERRORS

Dayfile message

x FATAL ERRORS

Dayfile message

x NONFATAL ERRORS

Dayfile message

x OVERLAPPING CORRECTIONS

Dayfile message when correction set introduced during current run changes status of some cards more than once. Cards affected are printed. The message is informative.

x UPDATE ERRORS, JOB ABORTED

Dayfile message when errors encountered in reading the input file.

A OPTION INVALID WITH RANDOM OLDPL OR SEQUENTIAL NEWPL

Fatal dayfile message

B OPTION INVALID WITH SEQUENTIAL OLDPL

Fatal dayfile message

CREATING NEW PROGRAM LIBRARY

Informative dayfile message

COPYING INPUT TO TEMPORARY NEWPL

Informative dayfile message appearing for creation run when sequential new program library is requested.

COPYING OLDPL TO RANDOM FILE

Informative dayfile message

DECK STRUCTURE CHANGED

Informative dayfile message

FILE NAME ON UPDATE CARD GR 7 CHAR

Dayfile message

G AND O FILES CANNOT HAVE SAME FILENAME.

Fatal dayfile message

GARBAGE IN OLDPL HEADER, UPDATE ABORTED

Dayfile message

IMPROPER MASTER CHARACTER CHANGED TO char

Nonfatal dayfile message

IMPROPER UPDATE PARAMETER, UPDATE ABORTED.

Dayfile message. Unrecognizable parameter on UPDATE card.

INSUFFICIENT FIELD LENGTH, UPDATE ABORT.

Table manager ran out of room for internal tables. Dayfile message.

NO INPUT FILE, Q MODE, UPDATE ABORTED

Dayfile message

NO OLDPL, NOT CREATION RUN, UPDATE ABORT

Dayfile message

PLS HAVE DIFFERENT CONTROL CHARACTERS, ABORT

Dayfile message

READING INPUT

Informative dayfile message

STACK DEPTH EXCEEDED

Fatal dayfile message. Stack in which cards are saved becomes full while processing BEFORE or ADDFILE directive.

TABLE MANAGER LOGIC ERROR

Fatal dayfile message.

THIS UPDATE REQUIRED n WORDS OF CORE.

Informative message.

UPDATE COMPLETE

Informative dayfile message.

UPDATE CREATION RUN

Informative dayfile message.

WAITING FOR 45000B WORDS

Informative dayfile message.

deckname IS NOT A VALID DECK NAME

Fatal error

n ERRORS IN INPUT, NEWPL, COMPILE, SOURCE SUPPRESSED.

n ERRORS IN INPUT.

Informative message. Fatal errors in input stream. UPDATE continues to process corrections in order to detect further errors. The second form of the message is issued when the U option is specified on the UPDATE statement.

ADDFILE FIRST CARD MUST BE DECK OR COMDECK

Fatal error

ADDFILE CARD INVALID ON REMOTE FILE

ADDFILE directive cannot be used for file to be entered onto OLDPL with a READ directive. Fatal error.

BAD ORDER ON YANK DIRECTIVE

Identifiers separated by a period on YANK directive are in wrong order. Nonfatal error.

CARD NUMBER ZERO OR INVALID CHARACTER IN NUMERIC FIELD

Sequence number field on correction directive is erroneous. Fatal error.

CONTROL CARD INVALID OR MISSING

UPDATE detected format error on a directive, or detected directive that was unrecognizable. Illegal operation such as INSERT prior to IDENT may also have been attempted. Fatal error.

COPY TO EXTERNAL FILE NOT ALLOWED WHEN READING FROM ALTERNATE INPUT UNIT

Results in a null copy.

DECK NAME ON ABOVE CARD NOT LAST DECLARED DECK

Informative message.

DECK SPECIFIED ON MOVE OR COPY CARD NOT ON OLDPL. CARD WILL BE IGNORED

Deck not on old program library. Informative message.

DO/DONT IDENT idname is NOT YANKED/YANKED/NULL DO/DONT

A DO card to negate the effect of a YANK references an idname that has not been yanked; or a DONT card to restore a YANK references an idname that was already yanked. Informative message.

DUPLICATE DECK dname NEWPL ILLEGAL

UPDATE encountered active DECK or COMDECK card that duplicates a previous card. Condition is fatal if new program library is being created; nonfatal if new probram library is not being created.

DUPLICATE FILE NAME OF file, JOB ABORTED

Same file name has been assigned to two UPDATE FILES. Fatal error.

DUPLICATE IDENT CHANGED TO idname

Informative message.

DUPLICATE IDENT NAME

During a merge run, UPDATE encountered a duplicate idname that it could not make unique. Fatal error.

DUPLICATE IDENT NAME IN ADDFILE

Name of corrections to be added as a result of an ADDFILE directive duplicates correction set name on old program library. Fatal error.

ERRORNOT ALL MODS WERE PROCESSED***

All changes indicated in the input deck were not processed. Names specified on correction directives should corrspond to deck or identifier names on old program library. Fatal error.

FILENAME OF file IS TOO LONG, UPDATE ABORTED

A file name exceeds seven characters. Fatal error.

FILE NAME ON ABOVE CARD GREATER THAN SEVEN CHARACTERS

Fatal error.

IDENT CARD MISSING, NO NEWPL REQUESTED, DEFAULT IDENTIFIER OF .NO.ID. USED

Informative message

IDENT LONGER THAN NINE CHARACTERS idname

Fatal error.

IDENTIFIERS SEPARATED BY PERIOD IN WRONG ORDER

Identifiers on directive in wrong order. Fatal error.

ILLEGAL CONTROL CARD IN ADDFILE

ADDFILE insertions cannot contain correction directives. Fatal error.

INVALID NUMERIC FIELD

Directive does not contain required numeric field. Fatal error.

LENGTH ERROR ON OLDPL. UNUSABLE OLDPL OR HARDWARE ERROR.

Card length on old program library is greater than maximum allowed or is less than one. Fatal error.

LISTED BELOW ARE ALL IDENT NAMES WHICH WERE CHANGED DURING THE MERGE

ALL YANK, SELYANK, YANKDECK, AND CALL CARDS AFFECTED HAVE BEEN CHANGED

NEW IDENT ON CHANGE CARD IS ALREADY KNOWN

An attempt was made to change a correction set identifier to one already in existence. Fatal error.

NO ACTIVE CARDS WERE FOUND WITHIN THE COPY RANGE. NULL COPY

Nonfatal error, UPDATE continued.

NO DECK NAME ON DECK CARD

Fatal error message.

NULL ADDFILE

The first read on the file specified by ADDFILE encountered an end-of-record. Informative message.

NULL DECK NAME

During ADDFILE or creation run, a DECK or COMDECK card encountered does not contain a deck name. Fatal error.

NULL IDENT

Fatal error.

OLDPL READ ERROR - POSSIBLE LOST DATA AFTER FOLLOWING CARD
card image1
AND BEFORE THE FOLLOWING CARD
card image2

A parity error or other read error has occurred while processing an UPDATE version 1.2 old program library. As a result, UPDATE is uncertain of the position of the old program library. The first card shown is the last card on the old program library that UPDATE successfully processed. The second card is the next valid card that UPDATE is able to find following the parity error or other read error. Fatal error.

OUTPUT LINE LIMIT EXCEEDED. LIST OPTIONS 3 and 4 DEFEATED.

UPDATE output exceeds line limit specified by default or on LIMIT card. Nonfatal error.

PREMATURE END OF RECORD ON OLD PROGRAM LIBRARY

Fatal error. End of record encountered in midst of card image. If a second run of this job is not successful, the old program library probably contains irrecoverable errors.

RECURSIVE CALL ON COMDECK dname IGNORED. FATAL ERROR

A common deck has called itself or called decks that contain calls to the common deck. A circularity of calls has resulted. Fatal error.

SEQUENCE NUMBER EXCEEDS 131071

Fatal error. UPDATE immediately aborts the run. The proper range of sequence numbers is 1 through 131071.

THE ABOVE CALLED COMMON DECK WAS NOT FOUND

Fatal error.

THE ABOVE CARD AFFECTS A DECK OTHER THAN THE DECLARED DECK

Nonfatal error. The card is ignored.

THE ABOVE CARD IS ILLEGAL DURING A CREATION RUN

Fatal error.

THE ABOVE CONTROL CARD IS ILLEGAL AFTER A DECK HAS BEEN DECLARED

Nonfatal error.

THE ABOVE LISTED CARDS CANNOT EXIST IN THE YANK DECK AND HAVE BEEN PURGED DURING EDITING

Informative

THE ABOVE OPERATION IS NOT LEGAL WHEN REFERENCING THE YANK DECK

Fatal error.

THE ABOVE SPECIFIED CARD WAS NOT ENCOUNTERED

Printed as part of the printing of unprocessed modifications. Fatal error message.

THE INITIAL CARD OF THE COPY RANGE WAS NOT FOUND. NULL COPY

Nonfatal error.

THE TERMINAL CARD SPECIFIED ABOVE WAS NOT ENCOUNTERED

Printed as part of the printing of unprocessed modifications. Fatal error message.

IT MAY EXIST IN A DECK NOT MENTIONED ON A COMPILE CARD

Printed when in Q mode.

THE TERMINAL CARD OF THE COPY RANGE WAS NOT FOUND. COPY ENDS AT END OF SPECIFIED DECK

TOO MANY CHBS -- INCREASE L.CHB

Correction history bytes exceed specified limit of 100 octal for a card. Fatal error; job terminated.

UNBALANCED TEXT/ENDTEXT CARDS, LAST ENDTEXT CARD IGNORED

Nonfatal error

UNKNOWN IDENTIFIER NAME idname

A correction directive references an idname not in directory. Fatal error.

WARNINGOLDPL CHECKSUM ERROR***

Informative dayfile message. At least one updated deck from old program library is bad.

WARNING, RETURNING PRIOR NEWPL.

Informative dayfile message. Two consecutive UPDATEs are being attempted, each of which creates a random new program library of the same name, without returning the new program library created by the first UPDATE.

YANK, SELYANK, OR YANKDECK IDENT ident NOT KNOWN

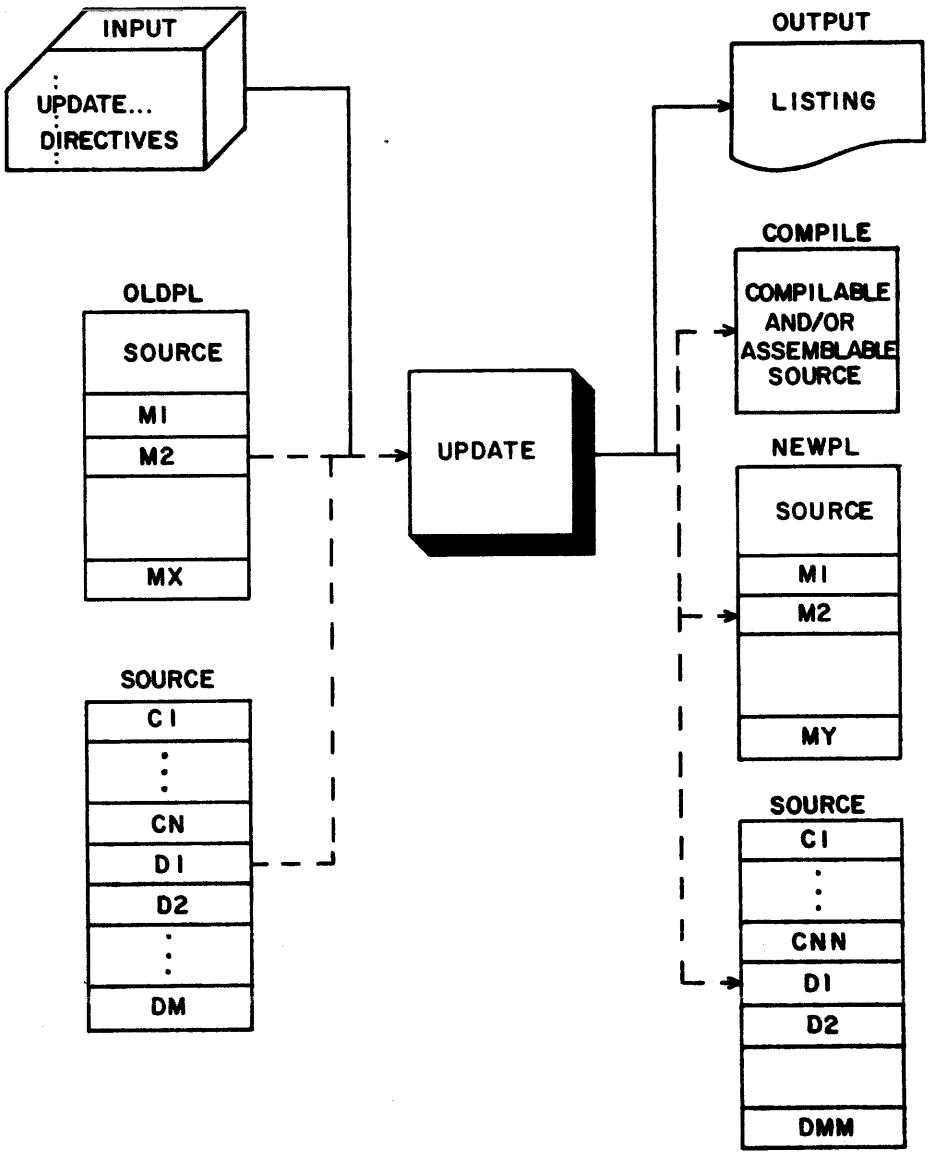
The ident referenced on a YANK or SELYANK directive has probably been purged; this applies to cards already on library. Nonfatal error.

FILE SUMMARY C

File	Function	Type	Position after UPDATE Call
Input (Directives)	Provides control information	Coded	Remains at end of record terminating UPDATE directives. Other cards may follow. If UPDATE aborts, location of input file is unpredictable.
Output	Contains listings	Coded	Current position of OUTPUT file unless O is equated to a file other than OUTPUT. File is not rewound.
Compile	Contains card images for assembly and/or compilation	Coded	Rewound before and after UPDATE operation.
Old Program Library	Contains old program library	Binary	Rewound before and after UPDATE operation.
New Program Library	Contains new program library	Binary	Rewound before and after UPDATE operation.
Source	Contains copy of all active cards on program library except YANK, SEL-YANK and YANKDECK cards. The cards contain no sequencing information. By default, any output resulting from PULLMOD is appended to the SOURCE file. The user may write this information to a different file by equating G on UPDATE control card to desired file name. Any rewind option which applies to the S (Source) file also applies to the G file.	Coded	Rewound before and after UPDATE operation.
Merge	Secondary library to be merged into new program library.	Binary	Rewound before and after UPDATE operation.

Files mentioned in ADDFILE, READ, REWIND, and SKIP operations are positioned according to the latest directive encountered on the directives file.

The files OLDPL, NEWPL, MERGE, COMPILE, and SOURCE or their equivalents normally are rewound before and after the UPDATE operation. The R parameter on the UPDATE control card specifies which files are to be rewound. Any rewind action specified on the old program library also applies to the merge file.



FILE FORMATS VS OPERATING SYSTEM USED D

The files generated and used by UPDATE have formats determined by both the operating system in use and the user. This appendix attempts to answer some of the questions concerning default file formats, allowed file formats, and the interchangeability of files among the systems.

In particular, it strives to explain when a file is random or sequential, when it must be binary or coded, which SCOPE2/SCOPE 3.4/KRONOS/NOS 1.0 record types are permitted, and when a random file can be copied to tape and read back into the system.

Sequential program libraries are interchangeable among operating systems when they are in SCOPE logical format (record manager type S records or SCOPE 1.1 I-mode records).

The below table indicates the interchangeability of random format old program libraries that have been copied onto tape. To read the table, look down the left column to determine the operating system used for generating and copying the library onto tape. Then locate the operating system with which you wish to read the tape by scanning the horizontal list. A yes indicates the tape can be read. A no indicates it cannot be read. This table does not attempt to supply procedures for using a tape when it is not readily interchangeable.

	Read Random Library From Tape				
Generate Random Library on Tape	KRONOS 2 or NOS 1.0	SCOPE 3.3	SCOPE 3.4	SCOPE 1.1	SCOPE 2
KRONOS 2 or NOS 1.0	Yes	No	No	No	No
SCOPE 3.3	Yes	No	No	No	No
SCOPE 3.4	Yes	No	No	No	No
SCOPE 1.1	No	No	No	Z,I-mode Yes X-mode No	No
SCOPE 2	No	No	No	No	Yes†

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

UPDATE FILES		SCOPE 3.x		KRONOS/NOS. 1.0/ SCOPE 1		SCOPE 2	
		Tape	Mass Storage	Tape	Mass Storage	Tape	Mass Storage
P=OLDPL	RB	Binary	Random or Sequential	Binary †	Random or Sequential	Binary, Sequential †† RT=W,S, or X	Random: RT=W unblocked Sequential: RT=W unblocked RT=W,S, or X blocked if specified through FILE card
N=NEWPL	RB	Binary	Random W-Sequential	Binary	Random W-Sequential	Binary, sequential RT=W or S	Random if unblocked Sequential if blocked or if W specified on UPDATE card RT=W unblocked by default. RT=blocked W or S; specified through FILE card
C=COMPILE	C	SCOPE 3 coded	Normal Sequential	Deter-mined by REQUEST card	Normal	RT=W, I blocked. Other types determined by FILE card	RT=W unblocked RT=S if compressed file; S specified through FILE card
I=INPUT	C	SCOPE 3 coded	Normal Sequential	Deter-mined by REQUEST card	Normal	RT=W, I blocked. Other blocking or RT=Z, FL≤100 through FILE card	RT=W unblocked RT=W blocked or RT=Z, FL≤100 through FILE card
O=OUTPUT	C	SCOPE 3 coded	Normal Sequential	Deter-mined by REQUEST card	Normal	RT=W, I blocked. Other types possi-ble through FILE card	RT=W, unblocked
S=SOURCE	C	SCOPE 3 coded	Normal Sequential	Deter-mined by REQUEST card	Normal	RT=W, I blocked. Other blocking or RT=Z, FL≤100 through FILE card	RT=W, unblocked RT=W blocked or RT=Z if specified through FILE card
*READ	C	SCOPE 3 coded	Normal Sequential	Deter-mined by REQUEST card	Normal	RT=W, I blocked. Other blocking or RT=Z, FL≤100 through FILE card	RT=W, unblocked RT=W blocked or RT=Z if specified through FILE card

† Can be random or sequential for I- or Z-mode tapes under SCOPE 1.1.
Can only be sequential for X-mode tapes.

†† Random files can be put on tape by copying the file to tape. In order 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 one of sequential file. If both tests fail, library format is unacceptable. Random format library must be unblocked, W records.

INDEX

CUT ALONG LINE

COMMENT SHEET

MANUAL TITLE CYBERNET Service UPDATE Reference Manual

PUBLICATION NO. 84000016 REVISION B October 1977

FROM:

NAME:

BUSINESS ADDRESS:

COMMENTS:

This form is not intended to be used as an order blank. Your evaluation of this manual will be welcomed by Control Data Corporation. Any errors, suggested additions or deletions, or general comments may be made below. Please include page number references.

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

STAPLE

FOLD

FOLD

FIRST CLASS
PERMIT NO. 8241

MINNEAPOLIS, MINN.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY
CONTROL DATA CORPORATION
Publications and Graphics Division
P. O. BOX 0 HQC02C
MINNEAPOLIS, MINNESOTA 55440

ATTN: DATA SERVICES PUBLICATIONS

CUT ALONG LINE

FOLD

FOLD

UPDATE CONTROL CARD PARAMETERS

UPDATE ($p_1, p_2, \ldots, p_n$)

A Sequential-to-random copy
B Random-to-sequential copy
C Compile file output; COMPILE if C or omitted. No compile if C=0. Otherwise, output to file named.
D Data width; 72 columns if no D. 80 columns if D.
E Edit. No editing if no E.
F Full update. If F omitted, corrected decks and those named on COMPILE cards are compiled. If F is specified, all decks are compiled.
G Pullmod file. If omitted, pulled modifications are appended to source file. Otherwise, output is written on named file.
H Header change. If omitted, character set is determined from old program library. If H=3, 63-character set is used regardless of header; if H=4, 64-character set is used regardless of header.
I Input. If omitted, directives and text on INPUT. Otherwise on named file.
K COMPILE card sequence. Takes precedence over C mode. If K, decks written on file COMPILE in COMPILE card sequence. If K=filename, decks written on named file.
L List options. If omitted, options A, 1, 2, 3, and 4 selected. Options A, F, and 0-9 not separated by commas. Any use of 0 suppresses listing.
M Merge input. Omitted, no merge. M, merge input on file MERGE. M=filename, input on named file.
N New program library. Omitted, no new library. N, output on NEWPL. N=filename, output to named file.
O List output file. Omitted or O, listings on OUTPUT. O=filename, output to named file.
P Old program library. Omitted or P, library on OLDPL. P=filename, library on named file.
Q Quick update. Takes precedence over F. Omitted, normal selective mode. If Q, only decks on COMPILE cards are processed.
R Rewind. Omitted, files are automatically rewound. If R, no rewinds issued. If R=$c_1c_2\ldots c_n$ then only files indicated by C, N, P, S are rewound.
S Source output. Omitted and no T, no source output. S, output on SOURCE. S=filename, output on named file.
T Source output excluding common decks (takes precedence over S). Omitted and no S, no source output. If T, output on SOURCE. T=filename, output on named file.
U Debug mode. Omitted, fatal error ends execution. If U, fatal error does not end execution.
W Sequential new program library. Omitted, new library will be random if possible. If W, new library will be sequential.
X Compressed compile file. Omitted, compile file not compressed.
Z Compressed input file. Omitted, input file is not compressed.
8 80-column output on compile file. Omitted, 90-column card images. If 8, 80-column card images.
* Master control character. Omitted, control character is *. If *=c, new character is c.
/ Comment control character. Omitted, comment character is/. If /=c, new comment character is c.

CORPORATE HEADQUARTERS
8100 34TH AVENUE SOUTH
MINNEAPOLIS, MINNESOTA
MAILING ADDRESS • BOX 0, MPLS., MINN. 55440

SALES OFFICES AND SERVICE CENTERS
IN MAJOR CITIES THROUGHOUT THE WORLD

IMPORTANT REGULATORY NOTICE

Users of Control Data services should be aware that the rules and regulations of the United States and International Telecommunications Regulatory Agencies prohibit Control Data from using communications services it leases from domestic, international and foreign communications carriers to transmit information for its users which is not part of a "single integrated" data processing service. All information transmitted must be directly related to the data processing applications or service provided by Control Data and unprocessed information shall not be allowed through the service between user terminals, either directly or on a store and forward basis. Noncompliance with these rules and regulations may force Control Data to discontinue the users' data processing service.