Start on third or fourth print line A

Right edge should be at or over perforation --->

Start on third or fourth print line A

Right edge should be at or over perforation --->

IEBUPDTX User's Guide

Version 1.12

```
         *
     *       *

   *           *
     Dial-Tyme
   *           *

     *       *
         *
```

Consumers Computer Services, Inc.
2100 M Street, N.W.
Washington, D.C. 20037

(202) 387-8963

Printed on May 9, 1982

Updated on May 1, 1982

ACKOWLEDGEMENTS

IEBUPDTX is a utility program designed to facilitate updating symbolic source data representing 80 column card images. It will incorporate the source language modifications into either sequential or partitioned data sets.

The program was written at the Cornell University Computation Center in September of 1971. Robert Lent and Richard Cogger set up specifications and contributed ideas. Larry Brenner designed and implemented the program. A design goal of IEBUPDTX was to produce an updating program that was compatible with IBM's IEBUPDTE, but offered additional capabilities.

The current version incorporates modifications made by John R. Ehrman at the Stanford Linear Accelerator Center, Seymour J. Metz at the Computer Network Corporation, and Gerhard Postpischil of Expert System Programming, Inc.

The preliminary version of this document was based on Cornell's IEBUPDTX Users Guide, and edited by Anne Ashley and Marge Mueller. Version 1 is updated by S. Hsia. Additional changes made by G. Postpischil, including conversion to SCRIPT format.

This manual was produced using the SYSPUB facility of the University of Waterloo's SCRIPT package, Version 3.2, with PARM=2P.

Section 1

INTRODUCTION

IEBUPDTX is a utility program designed to facilitate updating symbolic source data representing 80 column card images. It will incorporate the source language modifications into either sequential or partitioned data sets. A design goal of IEBUPDTX was to produce an updating program that was compatible with IBM's IEBUPDTE, but offered additional features and extensions.

Following is a summary of major extensions of IEBUPDTX:

Many keyword operands may be specified as positional operands. Many keywords required by IEBUPDTE are made optional or ignored by IEBUPDTX. Two examples are NEW=PO/PS; SEQ2, NEW1, INCR on NUMBER or DELETE commands. SEQFLD is ignored. The sequence field is permanently defined as columns 73/80. A numeric digit must be found by column 80, unless there are no sequence numbers in the source. The LABEL detail statement is not supported. Keyword operands SEQFLD, COLUMN, UPDATE=INPLACE, INHDR, INTLR, OUTHDR, OUTTLR, TOTAL, LEVEL, and SOURCE are not supported.

- The DECK option

  Due to the use of available features in IEBUPDTX, the input for CHANGE commands is usually not compatible with IEBUPDTE. IEBUPDTX offers the capability of producing a copy of the current update request in IEBUPDTE compatible form. This feature requires the DECK option on a PARM, and a SYSPUNCH output file. PARM=ADD may also be specified to copy ADD requests.

- Chained Libraries

  IEBUPDTX allows the use of multiple versions of a PDS member by the CURRENT command and the VERSION option and the CHAIN PARM option. With this feature, several versions may be stored and run by use of version numbers. This is especially useful for testing updated members.

  Additionally, multiple versions of the chained library can be used as successive update decks when the CHAINUPD PARM option is specified.

- Recursive Updates

  More than one update deck may be run against an OLD MASTER in one job with this feature. Successive update decks are applied to the OLD MASTER produced by the preceding deck. Thus it is not necessary to keep the updated old master source for successive updates.

- WYLBUR Edit Format Data Sets

  WYLBUR format data sets (i.e. RECFM=U, LRECL=BLKSIZE=3156 or 3520) may be used as input or output to IEBUPDTX without using any additional control cards.

  A new PARM option, INTEGER, has been specified to indicate normal mode of operation (sequence numbers in columns 73/80). For files not having sequence numbers, the WYLBUR line numbers may be used instead, by resetting the INTEGER option (i.e. PARM=NOINT). When this option is used, it applies to the SYSUT1, SYSUT2, SYSLIB and SYSPUNCH files that are in EDIT format. Note that use of MACRO or multi-member sequential output for SYSUT2 or SYSPUNCH may produce unprocessable files (non-ascending line numbers).

  For SYSIN (DD * or DD DATA) files submitted using the WYLBUR RUN command, WYLBUR places line numbers into columns 73/80 by default, unless the command also specifies UNN or JCL. A new PARM option, WYLRUN, has been defined as the default. It changes all sequence fields which appear to be line numbers (period in column 77) to blanks; the line number is printed to the left of the card image. As an alternative, the PARM option WYLBUR may be used to indicate that all sequence fields, regardless of contents, be changed to blanks. This assumes that you will not include "data statements" in the input (cards to be inserted or replaced, with explicit sequence numbers in columns 73/80).

- TSO datasets

  IEBUPDTX supports datasets with RECFM=V (e.g. TSO CLIST), which contain the sequence numbers in columns 1 to 8. Prior to using such an input record, the card image is re-arranged so that the sequence number falls into columns 73/80; on output, the sequence number is again placed in columns 1 to 8.

- Listing Controls

  Additional listing controls are another IEBUPDTX feature. New master source is not listed unless the ADD or CHANGE specifies LIST=ALL. Old master cards renumbered as a result of an insert operation may optionally be listed, as may any deleted cards. Also optional is the listing of insertions and deletions caused by intermediate updates.

  The listing related PARM options are as follows :

  1) LISTING (default), when negated, causes listing of major control cards only, and overrides all other list options.

  2) LISTADD (default) causes listing of card images processed by an ADD or REPL command, but not CHANGE or REPRO cards.

For CHANGE/REPRO, only modified cards are listed by default.

3) LISTDEL (default) causes listing of deleted records. Cards altered by FIX, SCAN, CFIX and CSCAN are considered "deleted" for purposes of listing.

4) LISTRN causes listing of old master statements which were forcibly renumbered due to an INSERT operation.

5) LISTALL causes action of, but does not set, LISTADD, LISTDEL, LISTRN; it is offered as a convenience.

6) LISTLEV causes listing of modifications caused by intermediate update decks (refer to Sections 6.2 and 6.3).

7) LIST=ALL on a major control card causes the entire new master to be listed unless NOLISTING is set.

- Use of Library Source

The COPY/MACRO commands allow data to be entered into the update program from a separate library, rather than from SYSIN only. Note that MACRO keeps the sequence numbers of copied cards, and may thus destroy existing sequencing. It should be used for permanent output files with due caution.

TABLE 1

IEBUPDTX equivalents of IEBUPDTE commands

IEBUPDTE	IEBUPDTX
ADD	ADD. REPL depending on PARM
ALIAS	ALIAS
CHANGE	CHANGE, REPRO
data (seq in 73/80)	INSERT, REPLACE
data (seq in 73/80)	data (seq in 73/80)
DELETE	DELETE
ENDUP	ENDUP
LABEL	-
NUMBER	NUMBER/SEQUENCE
NUMBER INSERT=YES	NUMBER/INSERT
REPL	ADD, REPL
REPRO	CHANGE, REPRO

- New Commands

  Table 1 shows the IEBUPDTX equivalents of IBM IEBUPDTE commands. A number of new commands are featured in IEBUPDTX, some representing extensions to functions available in IEBUPDTE, and some invoking new features. A list of the new commands and their functions is shown in Table 2.

TABLE 2

New IEBUPDTX Commands

CFIX	like FIX, but does not set any error codes.
COPY	allows data to be entered into SYSIN from a separate library, instead of a deck.
CSCAN	like SCAN, but does not set any error codes.
CURRENT	Designates a specified version number of a PDS as the production version; a feature of chained libraries described in Section 6.1.
FIX	Replaces the FIRST occurrence of a specified string on each of one or more card images, so that it is not necessary to code an entire card.
GANG	Allows a gang-punched id code to identify a particular update card or deck.
GENALIAS	Adds an alias to an existing member in the new master, without requiring a prior ADD or CHANGE operation.
LIST	Lists the contents of a PDS member
LOAD	Allows a set of update decks to be stored in SYSUT2 under the names on their CHANGE and ADD cards.
LOCATE	Positions the output master forward to a specified sequence number. It is normally used prior to a COPY or MACRO request, which inserts at the "current" new master location.
MACRO	Similiar to COPY in function; different restrictions
NOTE	A comment card
PARM	Permits respecification of PARM options.
RENAME	Changes the name of a member of the new master PDS
RESTORE	Recovers, in limited circumstances, a member which was previously scratched.
SCAN	Like FIX, but will replace ALL occurrences of a specified string.
SCRATCH	Deletes a named member from the SYSUT2 (output) data set.
USER	Used to branch to a user supplied update exit routine.

- GANG-Punching Id

  The GANG statement allows an identification code to be inserted as though a deck has been gang-punched. Thus a series of updates may be easily distinguished from one another by their id codes. GANG may be specified as a detail statement with an ADD/CHANGE operation, or globally as a PARM option. The value specified by GANG overlays existing data on a card image unconditionally. To avoid this, the ASM PARM option may be used. ASM will cause insertion of the GANG id only in cards which have a blank in column 72, and blanks from the column preceding the id column to the column after. Note that this option will not add GANG information to a card which has a prior GANG id, with one special exception. When the length of the GANG id is five characters, and the old card contains five numerics delimited by blanks in the id column, the GANG information replaces the old data. This is intended for GANG in date format (yyddd).

- Modifying Statements at Intra-card level

  The FIX and SCAN statements allow a card image to be changed at the character level by specifying a string and, optionally, a column range on the card. When a FIX or SCAN request is not satisfied, the update is terminated for that function. A conditional FIX/SCAN capability exists using the CFIX or CSCAN detail statements. When CFIX or CSCAN is not satisfied, a warning message is printed, but the update continues.

- Numbering on ADD or CHANGE Card

  Numbering may be specified directly on the ADD or CHANGE card using the keywords INCR and NEW1.

- Replace/Insert Cards

  IEBUPDTE replaces or inserts card images when it finds a "data statement" (card image with sequence numbers in columns 73/80), or when it detects a ./ NUMBER INSERT=YES statement followed by unsequenced cards. IEBUPDTX provides a shorter form of the NUMBER command with the name INSERT, and adds the option of deleting a range of cards from the old master before doing an optional insertion, using the REPLACE statement (not to be confused with REPL). Unlike IEBUPDTE, INSERT and DELETE statements for the same range may be specified in either order. For compatibility with IEBUPDTE, the data statement with sequence numbers in columns 73/80 is permitted.

- Program Error Control

  A PARM option allows user specification of the highest severity error. If any error occurs above the limit, execution will stop.

- PDS Handling Commands

  Several commands apply only to PDS members and provide the ability to LIST the contents of a PDS member, LOAD a set of update decks under the names on their CHANGE and ADD cards, RENAME or SCRATCH a member. GENALIAS may be used to add an ALIAS to an existing member; the RESTORE statement may (in limited circumstances) permit retrieval of a member which was scratched.

- Updating of update decks

  Control cards recognized by IEBUPDTE and IEBUPDTX require a ./ in columns 1/2. This makes it difficult to maintain datasets containing update control cards. IEBUPDTX provides a CTL PARM which permits specification of the bytes which signal a control card (instead of ./).

1.1 PARM OPTIONS

The following options may be included in the PARM field of the EXEC card or in a ./ PARM command supplied as a function statement. When supplied from the EXEC PARM field, at most 71 characters are allowed. Unlike other control cards, PARM operands may be separated by either blanks or commas, but continuation and comments are not allowed. The PARM command may be used more than once in a run, but only in the form of a function statement. Negation of an option is accomplished by prefixing the option with NO, or by specifying it in the form of a keyword as "option=NO". Note that CHAIN=NO acts as NOCHAIN, and does not assign an ID of NO.

```
//step  EXEC  PGM=IEBUPDTX,PARM=(option...,'keyword=option'...)

./  PARM option, option,option...keyword=option...
```

For example, the libraries used may be changed dynamically. The new output master is SYSUT2 by default. This may be changed :

```
./ PARM  OUTDD=SYSUT3
closes SYSUT2 and opens SYSUT3.
```

The PARM options are given below with their defaults shown:

(NO)ADD When used with the DECK option, ADD will result in an IEBUPDTE compatible deck being punched for modules that were ADDed. If the ADD PARM option is omitted, decks for ADD modules will not be produced.

(NO)ASM The ASM option prevents uncontrolled overlay of text by the GANG statement. See PARM=GANG and ./ GANG for details.

(NO)CHAIN
CHAIN=NO
CHAIN=id CHAIN specifies that SYSUT2 is a CHAINed library, and thus VERSION numbers will be required on CHANGE and ADD control cards unless doing a ./ LOAD. The "id" is a two letter library identification code, which will be used to create internal names for new versions of members. Once the allocator (@LLOCATR) has been stowed in the library's directory, the id cannot be changed. If the id has been set previously, you may code CHAIN rather than CHAIN=id.

(NO)CHAINUPD Specifies whether or not chained members found as intermediate updates in PDSs are to be applied as an ordered set of recursive updates (otherwise, only the production version is used).

COND=8 Sets the maximum allowed severity error message. All IEBUPDTX error messages have associated with them a potential step return code. Default is eight, which allows both errors from which immediate recovery is possible, such as an insert operation which doesn't provide any new data to insert, and errors which require termination of the current member update, but allow the program to go on to the next update (in SYSIN). COND=4 prohibits recovery for these latter errors and might be desirable when SYSUT2 is sequential. COND=0 forces termination following any error or warning message; this is desirable when taking SNAP dumps after each error. COND=12 is not recommended. It might allow the program to recover from such things as sequential SYSLIBs, but then again...

CTL=./ By default, all IEBUPDTE and IEBUPDTX control cards are identified by "./" in columns 1/2. The CTL parameter may be used to reset all subsequent processing to recognize the new CTL string. This feature permits IEBUPDTX to manipulate update decks as though they were card images (subject to normal sequencing and other constraints). The value specified by the CTL parameter must have a length of two, neither byte may be a comma (or blank), and at least one of the two must have a value lower than "A" in the EBCDIC collating sequence. For example :

```
./ PARM CTL=@/
@/ ADD  NAME=A
./ ADD  NAME=B
card 1
card 2
./ CHANGE NAME=C
```

The above sequence adds member A to the output master; the contents of A are the update statements for members B and C.

(NO)DECK Specifies whether or not an IEBUPDTE compatible update deck equivalent (except for sequencing) for subsequent CHANGE operations is to be punched (written to SYSPUNCH). The punch file may be a sequential or partitioned dataset. If it is partitioned, the member name must appear either on the DD card, or on all ADD and CHANGE statements. If the new master is based in any way on data provided by an ADD command, no DECK will be produced for that member unless the ADD option is also used.

DECKINCR This option is no longer supported.

(NO)DECKQ This option is used with DECK to indicate that the punch output may include ./ SEQUENCE (./ Q) control cards. In some situations an IEBUPDTE compatible deck cannot be produced (e.g. insertion prior to the first card of an existing member); DECKQ would punch a SEQUENCE control card, otherwise an error message would be produced.

(NO)GANG Specifies that all cards inserted or added to SYSUT2 (or new master) as a result of commands interpreted within an update file are to have the significant portion of the SSI placed ending in column 71. This PARM option is equivalent to supplying a ./ GANG CODE=SSI statement after each CHANGE card. You may temporarily specify a different GANG operation within a member update via the GANG detail control card, but cannot cancel gang-punching with a null GANG statement. At DTS, if the significant portion of the SSI is five bytes, the code is placed into columns 66/70, not 67/71. Failure to provide SSI for the GANG operation does not cause the update to fail unless COND=0.

(NO)IMPLSEQ causes the old and new masters to be treated as unsequenced. Data contained in columns 73/80 are preserved during the update. Sequence numbers refer to implicit position of old cards (e.g. INSERT SEQ1=5 would insert after the fifth old master card). This option may not be used with SEQID or NOINTEGER.

INCR=1 Resets the default increment to be used if left unspecified on control statements. This default is used on detail control functions (such as INSERT). It is not used on function statements such as CHANGE or ADD, unless NEW1 is specified.

INDD=SYSUT1 Specifies the DDNAME of the old master file. The master input file is specified using the SYSUT1 or INDD DD card. Input files may be switched between commands by using a PARM of INDD with the desired alternate DD name. Input files may be concatenated providing all concatenated files have the same DCB parameters (other than the blocksize). When compatible files differ only in the blocksize, the dataset with the largest blocksize should be specified first, or the largest blocksize should be specified on the first DD card of the concatenation.

(NO)INSERT When INSERT is specified, columns 73/80 on INSERTed cards are treated as blank no matter what they contain. Otherwise, they are checked and, if non-blank, used as IEBUPDTE change data records ("data statements").

(NO)INTEGER
(NO)INT Specifies that sequence numbers are present in columns 73/80 (except for RECFM=V files). The negation NOINTEGER applies only to files in WYLBUR EDIT format. When NOINT is used, WYLBUR line numbers will be used as sequence numbers; and vice versa for output data sets in WYLBUR edit format.

LIBDD=SYSLIB Specifies the default for DDNAME= on ./ COPY or ./ LIST commands, and the DDNAME of the library used by ./ MACRO commands. When no SYSLIB DD is present, and a SYSUT1 PDS is present, it will be used instead; if neither is present, and SYSUT2 is a PDS, it will be used. This replacement is done only at program start time; subsequent specification of LIBDD=SYSLIB may not work.

(NO)LISTADD This option was added for compatibility with IEBUPDTE, which lists the output of ADD functions whether or not LIST=ALL is specified on the ADD statement.

(NO)LISTALL This option, when set, acts as though LISTADD, LISTDEL and LISTRN had been turned on. Unless NOLISTING is specified, all changes to a member, except those due to recursive or chained updates, are listed.

(NO)LISTDEL Causes all deleted records to be listed. Default is to list any deleted cards. If LISTLEV is not specified, only cards deleted by SYSIN controls will be listed. Old master records altered by FIX or SCAN are considered deleted records.

(NO)LISTING Specifies that a listing of update commands and data is to be provided under control of the other LISTxxx options (default is LISTING). NOLISTING suppresses all other LIST options, and will list only major commands and messages on the print file. The listing generated by the LOAD command is suppressed, and the LIST=ALL option on function statements is ignored.

(NO)LISTLEV Tells the program to list insertions (deletions, if applicable) caused by all intermediate update decks. Default is to list only changes caused by SYSIN.

(NO)LISTRN Causes listing of statements which were forcibly renumbered due to INSERT and REPLACE operations. Unless LISTLEV is also on, only renumbering triggered by insertions from SYSIN are listed.

MOD Provided for compatibility with IEBUPDTE. When MOD is specified, ADD operations (but not REPL) will fail if the specified member already exists in the new master. This option is <u>not</u> the default, PARM=NEW is. When CHAIN is specified, PARM=MOD is ignored.

(NO)NAMES Specifies that control statements have name fields (and thus the blank(s) following the "./" and preceding the command word are required). If the default (NONAMES) is used, the blanks may be omitted, but if the name field is used, NAMES must be specified.

NEW This option (the default) specifies that ADD operations will be treated as REPL; i.e. an ADD operation will replace an identically named member in the new master. The option can be negated with PARM=MOD unless PARM=CHAIN is specified also.

OUTDD=SYSUT2 Specifies the DDNAME of the New Master data set. It may be changed any time between major function commands. Any DDNAME beginning with SYS may be used. Note that the operating system imposes certain restrictions on this facility for sequential datasets. INDD and OUTDD should never specify the same sequential dataset; the program may abend or the input data could be lost. When OUTDD is used to respecify a previously used DDNAME (sequential or member of a PDS), all output from the prior use could be lost unless DISP=MOD is used on the DD (not allowed for PDS members).

(NO)RUN This option is an abbreviation of WYLRUN.

(NO)SEQFIX This option, when specified, causes too small SEQ2 fields to be filled by digit replacement from SEQ1; e.g. ./ INSERT SEQ1=123,SEQ2=5 is normally invalid (SEQ2 less than SEQ1). SEQFIX fills leading blanks in the SEQ2 value from corresponding SEQ1 positions. Thus the above case would be treated as SEQ1=123,SEQ2=125. This option is not generally recommended due to the fashion in which IEBUPDTX parses commands. When positional operands are used, IEBUPDTX detects the end of an operand as any special character lower than "A" in the collating sequence (e.g. / - . etc.). Many keyboards contain special characters in the upshifted or downshifted position on the same keys as digits; e.g. ./ INSERT 12305 could be punched as ./ INSERT 123/5 if the shift key is not depressed.

(NO)SSI
SSI=DATE
SSI=hex When SSI is not specified on ADD and CHANGE commands, it will be used from the SSI PARM option, if that was used. At DTS, SSI=DATE is the default, and generates an SSI field of the form 000yyddd. This matches the action of a WYLBUR SAVE command. Any user SSI may be specified; the value supplied must consist of one to eight hexadecimal digits. The form SSI= is treated as SSI=DATE. The SSI field is always present in a directory entry generated with CHAINing. See the SSI operand under ADD or CHANGE for details.

(NO)TIMES STOWs a time stamp in the SYSUT2 directory of all CHANGED or ADDED members in the form (4 bytes = YYDDDHH+). Unlike SSI, this option always requires an additional four bytes of directory space per member.

UPDATES= Datasets used for chained and recursive updates may have any DDname not starting with SYS and not reserved for system functions (e.g. JOBCAT, JOBLIB, STEPCAT, STEPLIB, etc.). During initialization, IEBUPDTX will attempt to open any unidentified DD for additional input, unless the file is clearly an output file (e.g. unit record printer/punch, DD DUMMY, etc.). Some file types, such as tapes, disks, terminal files (TERM=TS), etc., may be either input or output. To prevent problems, you may specify that only DD cards starting with a specific character string are to be treated as input files. The common prefix is specified on UPDATES=pfx, and may be one to eight characters long, and may not start with SYS. Before you invoke IEBUPDTX from a TSO terminal session, you should free any previously allocated files which are not to be used by IEBUPDTX. When this option is used, it must appear on the EXEC PARM.

(NO)USER
USER=name Specifies the external name of a global user exit routine, or cancels it. See Section 7.2 for details.

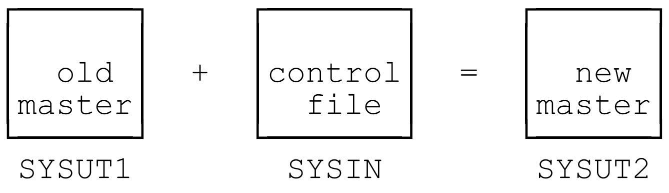
(NO)WYLBUR Allows the SYSIN control stream to have WYLBUR line numbers in columns 73/80. IEBUPDTX ignores the WYLBUR line numbers for updating purposes, but prints them at the left of the command in the output listing. Only card images from SYSIN are treated in this way. Note that any contents in columns 73/80 is ignored; hence you may not use this option when you have "data statements" (cards with explicit sequence numbers in 73/80). See WYLRUN below for an alternative. The problem may also be avoided by issuing the RUN command with the UNNumbered or JCLNUM options.

(NO)WYLRUN This option is the DTS default. It specifies that columns 73/80 of input cards are to be left as is unless they appear to contain a WYLBUR line number in EDIT format (nnnn.nnn), with leading and trailing zeroes blanked. This form of numbering is the default on the DTS RUN command. WYLRUN is ignored when WYLBUR or INSERT are used.

1.2 FILES

The source data to be updated (that is, changed in any way) is called the OLD MASTER. The set of data cards describing the update to be made is called the CONTROL FILE. The data formed as a result of the actions specified in the control file form the NEW MASTER data set. IEBUPDTX performs updating by copying data from old master to new master as directed via the control file(s). Source cards may be inserted, deleted, or edited.

For purposes of this document, it will be assumed that the control file is presented to the update program through the file described by the SYSIN Data Definition statement (OS JCL DD statement). The old master data set, wherever required, must be provided under the DDNAME SYSUT1 and the new master data set, if one is to be generated, must be described by a DD statement with DDNAME SYSUT2. This is not the case if data set associations are being dynamically controlled by the INDD and OUTDD PARM options described in Section 1.1.



The updating capabilities of IEBUPDTX apply to logically sequential old master data sets, and the program will update from SYSUT1 to SYSUT2 if they are simple sequential files. IEBUPDTX will also operate correctly if either or both of SYSUT1 and SYSUT2 are PDSs without requiring any change to the control statements in SYSIN, except possibly to specify the member name. If SYSUT1 and SYSUT2 both point to the same PDS, IEBUPDTX will automatically replace the old master with the new master if the update is completed without errors.

Note that use of MACRO may produce output that cannot be reprocessed by either IEBUPDTX or IEBUPDTE, and may result in unusable data when output is written in WYLBUR format without the INTEGER option. When output is sequential (sequential dataset or a PDS member), similar considerations apply when more than one ADD/CHANGE operation is used, or when the file appears more than once on an OUTDD PARM request (counting the implicit OUTDD=SYSUT2 from the EXEC PARM).

1.3 JOB CONTROL STATEMENTS

```
  1 //jobname JOB (acct,suba),pgmnm,CLASS= ...
  2 /*NORERUN
    /*SETUP    disks, tapes ...
  3 //step EXEC  PGM=IEBUPDTX,REGION=128K,PARM=...
  4 //SYSUT1   DD DISP=SHR,DSN=...       old master input
  5 //SYSUT2   DD DISP=OLD,DSN=...       new master output
  6 //SYSPRINT DD SYSOUT=A               listing
  7 //SYSPUNCH DD SYSOUT=B               optional IEBUPDTE deck
  8 //SYSLIB   DD DISP=SHR,DSN=...       optional COPY library
  9 //SYSUBEND DD SYSOUT=A               optional debug SNAPs
 10 //LIB1     DD DISP=SHR,DSN=...       optional intermediate
    //LIB2     ...                         update files
 11 //SYSIN    DD *                      control input
```

1 Job Card. See the DTS User's Guide for details.

2 HASP control cards. Note that updates to permanent files usually produce incorrect results when they are rerun. The DTS /*NORERUN card (system default) serves as a reminder. For updates to temporary or semi-permanent files, /*RERUN may be used when it is safe. The /*SETUP (or a /*NOSETUP) card is required to specify tape or disk volumes required by the following JCL. A /*FETCH request may be added for WYLBUR users, as may a /*NOTIFY userid.

3 EXEC PGM=IEBUPDTX - Additional information may be specified with the PARM parameter or with the ./ PARM command. See Section 1.1 for PARM options. The required REGION depends on the file blocksizes, the number of DD cards and the type of request (e.g. FIX or SCAN). Undemanding requests might run in as little as 64K; most simple updates (SYSUT1/SYSUT2 with BLKSIZE=3120) run in 80K. When REGION=128K is specified, the actual region billed is the larger of 64K and the amount used.

4 SYSUT1 - Old Master data set. Required only if CHANGE commands actually refer to it for original source records. May be sequential or partitioned, with (a) optionally blocked 80 byte fixed length records, (b) WYLBUR edit format, or (c) TSO CLIST format.

5 SYSUT2 - New Master data set. This file is generally required. SYSUT2 may be sequential or partitioned, with (a) optionally blocked 80 byte fixed length records, (b) WYLBUR edit format, or (c) TSO CLIST format. Card format is assumed when the RECFM parameter has not been set, or is F or FB. WYLBUR format is used when the RECFM is U, and the output

device is tape or disk; other device types are not supported with RECFM=U. TSO CLIST format is used when the RECFM is V or VB; a specification of VBS is treated as VB. If BLKSIZE is omitted, the program defaults the size by device type (8000 for tape, 80 for unit record, 3120 for 3330, etc.).

6 SYSPRINT - Listing data set, always required. If omitted IEBUPDTX terminates with return code 16. This file is written with DCB=(RECFM=VBM).

7 SYSPUNCH - DECK data set. Required only if the PARM DECK option is specified. SYSPUNCH may have any of the attributes valid for SYSUT2.

8 SYSLIB - COPY library. Required only if LIST, COPY, or MACRO commands reference it. Must be a PDS with 80 byte fixed length records (optionally blocked), WYLBUR edit format, or TSO CLIST format.

9 SYSUBEND - SNAP data set. If this DDNAME is present, the program will produce a SNAP dump with each error message.

10 INTERMEDIATE UPDATE FILES. An intermediate update file is provided to IEBUPDTX by using any DDNAME that doesn't begin with "SYS" to describe it. These files may be sequential or partitioned, with either optionally blocked 80 byte records, WYLBUR edit format, or TSO CLIST format. The intermediate update data sets should contain valid IEBUPDTX update decks which will be considered to apply logically between SYSUT1 and SYSIN in TIOT sequence. In such cases, SYSUT1 is updated by the intermediate update, whose output becomes "old master" to the next update or SYSIN. Also see the UPDATES PARM option.

11 SYSIN - Master control file. Generally required to provide the main source of control statements to the program, but may be omitted if the update can be performed using the last intermediate update control file in the JCL. In that case, the intermediate update file must be sequential, and is used as SYSIN. SYSIN must be sequential, with either optionally blocked fixed length 80 byte records, WYLBUR edit format, or TSO CLIST format.

Any non-SYS files are considered to be intermediate update files except the obvious ones (STEPLIB, etc.). Any other SYS DDNAME may be used for the DDNAME control on COPY operations or the OUTDD and INDD PARM options. To save storage or time, BUFNO may be specified for any file via the DCB parameter on the DD card. Note that chained scheduling is used on all non-print files.

1.4 IEBUPDTX CONTROL STATEMENTS

There are two general types of control statements in IEBUPDTX:

Function statements
Detail Statements

Function statements are used to initiate an update and may be used alone or accompanied by detail Statements. Detail statements may not be used alone but must be used in conjunction with a function statement, usually a CHANGE statement, for each data set. In this document the detail statements are covered in two sections, with basic detail statements in Section 3 and extended detail statements in Section 4. Function statements are described in Sections 2 and 5. The two function statements (ADD and CHANGE) described in Section 2 are basic statements which precede the detail statements. Table 3 shows the hierarchy of control statements.

TABLE 3

Control Statement Hierarchy

ENDUP	
CURRENT GENALIAS LIST PARM RENAME RESTORE SCRATCH	
LOAD	
ADD REPL	CHANGE REPRO
ALIAS data statement NOTE	ALIAS data statement CFIX CSCAN FIX NOTE SCAN
	DELETE INSERT LOCATE NUMBERi REPLACE
NUMBER SEQUENCE	NUMBER SEQUENCE
COPY MACRO	COPY GANG MACRO
data card	

NUMBERi represents ./ NUMBER INSERT=YES. A data statement contains a sequence number in columns 73/80. A data card is blank in columns 73/80.

1.5 COMMAND SYNTAX

IEBUPDTX commands are written in a manner similar to assembly language macro calls. The general format is:

```
./namefield   operation   positional-operands,keyword-operands
```

./ must be present in columns 1/2 (but refer to PARM=CTL)

Name field
: is not allowed unless the NAMES PARM option is selected. The NAMES option is provided for compatibility with IEBUPDTE, but is not the default. This implies that the name field is generally not to be used. Its omission need not be indicated by leaving a blank before the operation field although leaving blanks is allowed. PARM options are discussed in Section 1.1.

Operation
: contains an IEBUPDTX control statement, either a function statement (such as CHANGE), or a detail statement (such as INSERT).

Operands
: may contain both positional and keyword operands. As in assembly language, any positionally specified operands must precede any and all keywords. No embedded blanks are allowed as the first blank starts the comments field. Required operands are shown in this manual in braces ({}), while optional operands are shown in brackets ([]). Each command is presented in two formats: the first format shows all keyword operands while the second format shows all allowable positional operands. An operand that is a keyword only appears in both formats as a keyword operand. Table 4 shows which operands are allowed for which commands, and what the order of their positional operands is.

Positional Operands
: are separated by commas or by dashes to improve readability. Omitted positional parameters must be designated by a comma.

  Unlike assembly language, many operands may be specified either positionally or as keywords. For compatibility with IEBUPDTE, all operands may be specified as keywords although not all operands may be specified positionally. A list of allowed positional operands may be found in Table 4.

TABLE 4

Command Operand Summary

	Operand																			
Command	CODE	COL	COL1	COL2	DDNAME	FROMSEQ	INCR	INSERT	LIST	NAME	NEWNAME	NEW1	SEQID	SEQ1	SEQ2	SSI	TOSEQ	VERSION	STRING	TEXT
ADD	.	.	.	.	.	.	K	.	K	1̲	.	K	K	.	.	3	.	2	.	.
ALIAS	.	.	.	.	.	.	.	.	.	1̲	.	.	.	.	.	.	.	.	.	.
CFIX	.	.	3	4	.	.	.	.	.	.	.	.	.	1̲	2	.	.	.	S̲	.
CHANGE	.	.	.	.	.	.	K	.	K	1̲	K	K	K	.	.	3	.	2	.	.
COPY	.	.	.	.	K	3	.	.	.	1̲	.	.	.	.	.	.	4	2	.	.
CSCAN	.	.	3	4	.	.	.	.	.	.	.	.	.	1̲	2	.	.	.	S̲	.
CURRENT	.	.	.	.	.	.	.	.	.	1̲	.	.	.	.	.	.	.	2̲	.	.
DELETE	.	.	.	.	.	.	.	.	.	.	.	.	.	1̲	2	.	.	.	.	.
ENDUP	none.			.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
FIX	.	.	3	4	.	.	.	.	.	.	.	.	.	1̲	2	.	.	.	S̲	.
GANG	1	2	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
GENALIAS	.	.	.	.	.	.	.	.	.	1̲	.	.	.	.	.	.	.	2	.	.
INSERT	.	.	.	.	.	.	2	.	.	.	.	3	.	1̲	.	.	.	.	.	.
LIST	.	.	.	.	K	3	.	.	.	1̲	.	.	.	.	.	.	.	2	.	.
LOAD	none.			.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
LOCATE	.	.	.	.	.	.	2	.	.	.	.	3	.	1̲	.	.	.	.	.	.
MACRO	.	.	.	.	.	.	.	.	.	1̲	.	.	.	.	.	.	.	.	.	.
NOTE	comments.					.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
NUMBER	.	.	.	.	.	.	3̲	.	.	.	.	4	.	1̲	2	.	.	.	.	.
NUMBERi	.	.	.	.	.	.	3̲	K̲	.	.	.	4̲	.	1̲	2	.	.	.	.	.
PARM	see Section 1̲.1̲								.	.	.	.	.	.	.	.	.	.	.	.
RENAME	.	.	.	.	.	.	.	.	.	1̲	2̲	.	.	.	.	.	.	.	.	.
REPLACE	.	.	.	.	.	.	3	.	.	.	.	4	.	1̲	2	.	.	.	.	.
RESTORE	.	.	.	.	.	.	.	.	.	1̲	.	.	.	.	.	.	.	.	.	T̲
SCAN	.	.	3	4	.	.	.	.	.	.	.	.	.	1̲	2	.	.	.	S̲	.
SCRATCH	.	.	.	.	.	.	.	.	.	1̲	.	.	.	.	.	.	.	2	.	.
SEQUENCE	.	.	.	.	.	.	.	.	.	.	.	1̲	.	2̲	.	.	.	.	.	.
USER	4	.	.	.	.	.	.	.	.	1̲	.	.	.	2̲	3	.	.	.	.	.

Underlined items are required. At least one of NEW1 and INCR must be specified on NUMBER and SEQUENCE.

1-4	Positional operand number. May appear as keyword.
K	Keyword operand only.
S	Special string format #old#new#
T	Hexadecimal text only; no comments.

Keyword Operands

follow any positional operands and are separated by commas. Dashes may not be used as separators. All IEBUPDTX operands may be specified as keywords. No embedded blanks are allowed as the first blank starts the comment field.

Continuation

Continuation statements are indicated by following the last keyword by a comma. A statement ending with a comma followed by a blank indicates a continuation card. No continuation character is checked for in column 72. Splitting an operand at column 72 as in assembly language or IEBUPDTE is not allowed. The second card of the continuation must contain the ./ in columns 1/2, and the continued data must begin by column 16. ADD and CHANGE cards within the scope of a LOAD, and RESTORE cards may not be continued. Special rules for continuation apply to the CFIX, CSCAN, FIX and SCAN cards.

Comments may be placed following the operands by leaving a blank, except on the RESTORE statement, where comments are not allowed. In addition, a NOTE card functions as a comments card.

Line Numbers:

Line numbers may be expressed by their right-most digits. For example, SEQ1=10 is equivalent to SEQ1=00000010. SEQ1=ALL is also allowed to indicate the entire file. END can be used in any range; 100/END indicates line number 100 to the last line, inclusive. The short form 10K, 12K may be used for 10000, 12000 etc. throughout IEBUPDTX. All numeric parameters, with the exception of ALL and END, specify 8 character (maximum) decimal numbers. ALL may be used on the SEQUENCE and NUMBER commands to specify the first to last cards, inclusive.

Name

The operand NAME=name is synonymous with MEMBER=name. The MEMBER= form does not necessarily imply a PDS. It may be used for a library member name or for a sequential data set. The member name operand is optional only when all of SYSUT1, SYSUT2 and SYSPUNCH (with the DECK option) are sequential datasets.

Format:

IEBUPDTX updates data in 80 column card images. Throughout this manual, the terms "statement" and "card" are used interchangeably. "data statement" is used to describe a card which has sequence numbers in columns 73/80, and does not have control bytes "./" in columns 1/2. This type of card acts as an INSERT or REPLACE of a single card at the specified sequence number.

SSI The System Status Information field consists of four bytes of data in the variable portion of a PDS directory entry. When chaining is used, the field is always present, otherwise it is optional. The field is used by IBM for its source modules to denote modification status. With WYLBUR datasets, it is used to store the date of the last addition or change, in the form 000yyddd. At DTS, SSI is inserted by default (see PARM=SSI). Note that several utility programs provided at DTS have special provision for SSI in date format; these include PUNK and WYLVTOC.

Sequencing

Sequence numbers are 8 character decimal numbers in columns 73/80. Sequence numbers must be present in the old master card images in order to do an update, unless WYLBUR format with the NOINTEGER option is used (see Section 7.1), or the IMPLSEQ option is used. The card images may be sequenced when the old master is built, or as part of the update process. There are several ways of sequencing or resequencing source: the SEQUENCE and NUMBER statements and the INCR and NEW1 operands on the ADD or CHANGE statement may be used to sequence an entire member; the INCR and NEW1 operands on the INSERT and REPLACE commands may be used to resequence inserted or replaced portions of a deck.

It is good practice to select a relatively large increment if frequent updates are expected. This leaves "room" to insert cards without resequencing parts of the old master each time; and it allows two methods of inserting cards, both of which are described in section 3.4 under the INSERT command.

Table 5 compares the effects of sequencing by various commands. Throughout this manual, the operands INCR and NEW1 are used for sequencing:

INCR refers to the numbering increment used for sequencing the cards. The default is 1 or whatever is defined as the INCR PARM option.

NEW1 sets the sequence number for the first card to be sequenced or resequenced. The default is INCR plus the previous sequence number. If there is no previous sequence number, the default is INCR.

TABLE 5

INCR/NEW1 Sequencing by Command

Command	INCR /default	NEW1 /default	Comments
ADD	/1	/INCR	sequences an entire member. Either INCR or NEW1 must be specified or no numbering is done.
CHANGE	/1	/INCR + previous sequence number	" "
INSERT		number to be given to first inserted card. /Seq1 + INCR (Seq1 is the card in the old master which immediately precedes the inserted data cards).	numbers inserted cards and as much of the old master as is necessary to accommodate the cards inserted.
NUMBER	/1	number to be assigned to first card. /INCR + previous sequence number.	sequences an entire member. Either INCR or NEW1 must be specified or no numbering is done.
REPLACE		number to be assigned to 1st inserted card/ Seq1 (Seq1 is the number of the first card replaced)	numbers replaced cards and as much of the old master as is necessary to accommodate any additional cards inserted.
SEQUENCE		number to be assigned to first card being numbered /INCR+ previous sequence number or INCR.	may be used to number part or all of a member. At least one of INCR and NEW1 must be specified or no numbering is done.

The effect of sequencing varies with the command. If neither INCR nor NEW1 is coded on ADD or CHANGE (or on SEQUENCE or NUMBER), no numbering is done. If neither INCR nor NEW1 is coded on the REPLACE or INSERT detail statements, as much numbering as is necessary to accommodate inserted cards will be done. The defaults used will be one for both cases (since INCR defaults to 1 and NEW1 defaults to INCR).

Section 2

FUNCTION STATEMENTS

This section is concerned with two operations to be performed on a data set as a whole. At least one function statement must be provided for each PDS member or data set to be processed. The detail control statements, described later on, may not be used without a function statement, usually the CHANGE statement.

To simplify the following discussion, assume that both the new and old master data reside in a PDS. If either is a sequential data set, the update will operate accordingly. Abbreviations and synonymous commands, if any, are in parentheses following the command.

2.1 ADD (A,REPL)

Before an update can be performed, the original source data must be placed in the old master PDS as a unique member with the ADD command. The alternative would be to input the original cards every time. The data cards which are to comprise the new member should follow the ADD command in SYSIN. If all original source is provided by ADD commands, SYSUT1 need not be specified.

A REPL statement may be used instead of ADD. It is synonymous with ADD if PARM=NEW is used (default), or when chaining is in effect. When PARM=MOD is chosen, and the output is partitioned, ADD will cause termination of the update when the specified member already exists in the new master.

```
./ ADD {NAME=name}[,VERSION=version,SSI=ssi,INCR=incr,NEW1=new1
            ,SEQID=seqid,LIST=ALL]

./ ADD {name}[,,ssi,INCR=incr,NEW1=new,SEQID=seqid,LIST=ALL]
```

NAME	is the unique name to be given to the source data. MEMBER=NAME may be used instead of NAME=NAME in any command. When SYSUT2, and SYSPUNCH (with the DECK option), are sequential, the name is optional. However, it should still be coded to identify the data. The use of name as the SYSUT2 PDS member name is not the only way in which IEBUPDTX can use it.
VERSION	is described in Section 6.1.

SSI	is a code of up to 8 hex digits, assumed by IEBUPDTX to be right justified, or the word DATE which results in a value of the form 000yyddd. Specifies that the PDS directory includes four bytes of SSI data. If SYSUT2 is not a PDS, SSI is ignored. If SYSUT1 is a PDS with SSI in its directory, it is retained for use in the SYSUT2 directory. Once SSI is specified, it can not be removed by subsequent CHANGE cards, but it may be modified. The default SSI is either the SSI provided by the previous update level, the SSI specified on a PARM, or x'FF200000' if necessary.

When the original old master source data is being built, it is usually desirable to sequence the cards. If the member is to be updated later, the sequence numbers must be present on the old master card images. Sequence numbers are placed in card columns 73/80 unless the INTEGER option is not used (see Section 7.1). Sequencing may be performed by the INCR and NEW1 parameters on the ADD, CHANGE, INSERT, and REPLACE cards. It may also be done by the SEQUENCE or NUMBER detail statement described in Section 3. Specifying either INCR or NEW1 on the ADD, CHANGE, SEQUENCE or NUMBER commands results in complete resequencing of the member. If neither INCR nor NEW1 is specified, no sequencing is done. Sequencing is required when the new master will be written in WYLBUR edit format using WYLBUR line numbers (see Section 7.1).

INCR	specifies the numbering increment to be used to sequence the cards in the new master. The default is 1. If INCR is specified in the PARM options, then that value will be the default.
NEW1	specifies sequence number to be used for the first card. In most commands the default value is INCR+the previous sequence number. In this case since it is an ADD, there is no previous sequence, so the default for NEW1 is the value of INCR.
SEQID	Sequence field identification creates an alphabetic code to be placed on each card image starting in column 73. The actual sequence number then begins in column 73+length of SEQID. The SEQID field is handled automatically by IEBUPDTX. Only the first old master card in SYSUT1 is examined for a SEQID. If one is found, all old master cards from SYSUT1 have zeroes placed over the SEQID for the update process. When any card is subsequently written out to SYSUT2, the SEQID is automatically replaced starting in column 73. SEQID requires the INTEGER option (see Section 7.1).
LIST=ALL	produces a listing of the data read by the ADD function unless the NOLISTING PARM option is requested. Omission of LIST=ALL still produces a listing if PARM options LISTALL or LISTADD (default) are in effect.

2.2 CHANGE (C,CHNGE,REPRO)

To change (or update) the old master, the CHANGE card is used, followed by detail control commands which specify the update to be made. The detail control statements are: ALIAS, COPY, DELETE, FIX, GANG, INSERT, LOCATE, MACRO, NOTE, NUMBER, REPLACE, SCAN, SEQUENCE. They are fully discussed in the next section. In the absence of any detail statements, the old master is copied unchanged into the new master file. Unless the NOINTEGER PARM is used (see Section 7.1), the old master must be sequenced before the update process, or completely resequenced by the update itself. Otherwise, the job fails and no updating will be done. Complete resequencing may be done by a SEQUENCE or NUMBER statement, or by the INCR, NEW1 parameters on the ADD or CHANGE cards.

The CHANGE command is written in much the same way as the ADD command: the member name (if provided) is used to locate the old master data in SYSUT1 (if a PDS) as well as to store the new master data in SYSUT2 (if a PDS). The name is also required when the DECK option is used and SYSPUNCH is a PDS.

```
./ CHANGE [NAME=name,VERSION=version,SSI=ssi,INCR=incr,
             NEW1=new1,SEQID=seqid,LIST=ALL,NEWNAME=newname]

./ CHANGE [name,,ssi,INCR=incr,NEW1=new1]
```

NAME	is the name of the member to be changed
VERSION	is discussed in Section 6.1.
SSI	is a code of up to 8 hex digits, assumed by IEBUPDTX to be right justified, or the word DATE which results in a value of the form 000yyddd. Specifies that the PDS directory includes four bytes of SSI data. If SYSUT2 is not a PDS, SSI is ignored. If SYSUT1 is a PDS with SSI in its directory, it is retained for use in the SYSUT2 directory. Once SSI is specified, it can not be removed by subsequent CHANGE cards, but it may be modified. The default SSI is either the SSI provided by the previous update level, the SSI specified on a PARM, or x'FF200000' if necessary.

When the SSI is specified with fewer than 8 characters, it is right-justified and overlays the right-most portion of the prior SSI (or the default); e.g. SSI=1234 would default to a value of FF201234 (the FF20 is the default if SSI is not available from any other source). At DTS, the default SSI is the date in the form 000yyddd, thus the above example might result in 00081234.

INCR specifies the numbering sequence which will be used to resequence the cards. Using INCR on the CHANGE card specifies complete resequencing of a member.

NEW1 specifies the sequence number to be used for the first card. The default value is INCR+the previous sequence number. If there is no previous sequence number (such as when a member is ADDed) the default is the value of INCR.

SEQID changes the alphabetic code placed on each card by the ADD card, starting in col 73. The actual sequence number then begins in col 73+length of SEQID.

Some caution must be used when coding SEQID on a CHANGE card. The length of the SEQID may be increased but should not be made so long that it overlaps any non-zero digit of the sequence number. SEQID=0 may be coded to indicate that no sequence id is desired for the new master. This special case allows the program to remove the SEQID from the SYSUT1 cards automatically. See discussion of SEQID under the ADD command for more comments.

SEQID requires the INTEGER option (see Section 7.1).

LIST=ALL invokes a listing of the new master source data as it is written to SYSUT2. If LIST=ALL is not specified, only cards which are changed from old master to new are listed.

NEWNAME specifies the name to be applied to the new master. This is particularly helpful when SYSUT1 and SYSUT2 are the same PDS. NEWNAME= is valid only for a PDS and only in SYSIN. NEWNAME is not supported when chaining is used.

Example :

```
        ./ CHANGE HASPINIT,LIST=ALL
```

signals that changes are to be made to HASPINIT; the inserted or modified statements are to be numbered in increments of 1, the default starting with the previous sequence number + 1. If INCR is coded on the PARM field, the default increment would be that value. The new master source is to be listed. SEQID will be used if the old master had SEQID. If SEQID=0 had been coded, the SEQID from the old master would not be used.

Section 3

DETAIL STATEMENTS

Detail statements are used to modify the operation initiated by a ADD or CHANGE command, by defining the nature of the update to be performed. A range of card images may be deleted, inserted, replaced, or renumbered. Changes to character strings may be performed on a range of card images. Detail statements must be arranged in order of increasing sequence numbers of records being updated.

3.1 SEQUENCE (Q)

SEQUENCing may be specified with this SEQUENCE command, with a NUMBER command, or as a parameter on the ADD, CHANGE, INSERT, or REPLACE commands.

The first SEQUENCE card immediately follows the ADD or CHANGE card, and precedes the actual data to be added. The format of the command is:

```
./ SEQUENCE INCR=incr,NEW1=new1

./ SEQUENCE incr,new1
```

At least one of INCR or NEW1 must be coded or no renumbering is done.

INCR	specifies the numbering increment which will be used to sequence the cards. If card x has sequence number x, card x+1 will have sequence number x+INCR. The default value of INCR is 1 unless a different value has been coded for INCR in the PARM statement.
NEW1	sets the sequence number to be used for the first card. The default is the value of INCR+the previous sequence number or the value of INCR if there is no previous sequence number.

The sequencing may be dynamically modified by placing other ./ SEQUENCE commands among the source cards. Sequence numbers must be assigned in increasing order. Erroneous NEW1 values will be rejected.

It is good practice to choose a relatively large increment if one expects to do frequent updates. This leaves "room" to insert cards without resequencing parts of the old master each time. It also allows the use of two methods of inserting cards. See the description of the INSERT command in Section 3.4.

IEBUPDTX provides the ability to define the default INCR used by SEQUENCE, NUMBER, INSERT, and REPLACE detail control commands and the ADD and CHANGE function control statements. To define a default, code INCR=incr as one of the PARM options, where incr is a decimal number greater than zero. This will override the built-in default increment of one. This and other PARM options are discussed in Section 1.1 of this manual.

3.2 NUMBER (N,NUMBR)

The ./ NUMBER command is provided for IEBUPDTE compatibility. It has two forms :

- Used without INSERT=YES, it renumbers all or selected portions of the input (with ADD/REPL), or the old master (with CHANGE/REPRO). The function is similar to ./ SEQUENCE, but a start and end sequence may be specified. Used in this fashion, the first ./ NUMBER statement must immediately follow the ADD/CHANGE statement.

- When used with INSERT=YES, it acts like the ./ INSERT command.

```
./ NUMBER  {SEQ1=seq1}[,SEQ2=seq2,INCR=incr,NEW1=new1,
                INSERT=YES]
./ NUMBER  {seq1}[,seq2,incr,new1,INSERT=YES]
```

SEQ1	specifies the first card to be renumbered. ALL may be used to specify an entire member. When specifying ALL, omit seq2 but mark its place with a comma if it is not the last operand. For example, NUMBER ALL,,incr.
SEQ2	specifies the last card in the range to be renumbered.
INCR	same as in the sequence command.
NEW1	same as in the sequence command. Like the sequence command, at least one of INCR and NEW1 must be coded.
INSERT=YES	distinguishes an insert operation from a resequencing operation.

For compatibility with IEBUPDTE, the NUMBER command may be used to insert or renumber a range of the old master. That usage is not encouraged as it is better to leave the original source sequence numbers as a common base from which to work.

3.3 DELETE (D,DELET)

The DELETE detail statement is used to delete one or more cards from the old master. The cards deleted are simply not transcribed to the new master - they remain intact in the old master (unless SYSUT1=SYSUT2).

```
./ DELETE {SEQ1=seq1}[,SEQ2=seq2]

./ DELETE {seq1}[,seq2]
```

SEQ1	specifies the beginning of the range of card images to be deleted.
SEQ2	specifies the last card image to be deleted. If SEQ2 is omitted, or if SEQ2=SEQ1, a single card image, SEQ1, is deleted. SEQ2=END may be specified to delete from SEQ1 to the end of the deck.

Example 1:

```
./ DELETE 200-280 or
./ DELETE SEQ1=200,SEQ2=280
```

cause all old master records with sequence numbers between 200 and 280 to be deleted. If records 200 and 280 do not actually exist in the old master, a warning message is produced.

Example 2:

```
./ DELETE 200,200 or
./ DELETE 200
```

deletes the single card with sequence number of 200.

3.4 INSERT (I)

To add new cards to the old master, the INSERT command is used. The new cards being INSERTed will be assigned new sequence numbers by IEBUPDTX, and must not contain any punches in columns 73/80. The cards are inserted after some existing card image in the old master, whose sequence number is provided on the INSERT command.

The INSERT command must be immediately followed by the data cards to be inserted, or one of five detail statements (COPY, GANG, MACRO, NUMBER or SEQUENCE). The insert command is terminated when either a ./ statement other than the special ones, or a card with any non-blank characters in columns 73/80 is encountered.

Inserted cards are listed whether or not LIST=ALL is coded on the CHANGE card. LIST=ALL causes all new master source to be listed.

```
./ INSERT {SEQ1=seq1},[INCR=incr,NEW1=new1]

./ INSERT {seq1}[,incr,new1]
```

SEQ1	indicates where to insert the new cards in the old master. SEQ1=0 may be used to insert a card at the beginning of a deck if there is no card 0.
INCR	specifies the numbering increment. INCR defaults to 1 unless a different value of INCR is coded in the PARM statement.
NEW1	specifies the sequence number to be placed on the first inserted card. NEW1 defaults to SEQ1+INCR.

The SEQUENCE command may also be placed within the records being INSERTed to modify incr and new1 dynamically.

Example 1:

```
./ INSERT 600,10
     data cards
```

inserts the data cards into the new master, after card 600. The first card inserted is numbered 610 with subsequent cards numbered 620, 630, etc.

Single cards may be inserted without using the INSERT command by punching the sequence numbers they are to have directly in columns 73/80. For example, if the old master contains cards numbered 10, 20, 30, etc. a data card in the update deck with 00000025 punched in columns 73/80, will be inserted between cards 20 and 30 and be numbered 25. To insert in this manner, it is imperative that the number to be placed in columns 73/80 does not already exist in the old master - if it does, the old master card is DELETEd before the new card is inserted.

The INSERT command must be used whenever the number of cards inserted exceeds the "room" left for them in the old master. If the old master is sequenced in steps of 1, it is impossible to insert using the second method described. However, the INSERT command may always be used, since it automatically renumbers as much of the old master as is necessary to accommodate the cards inserted. Therefore it is a good idea to use a relatively large increment when creating old masters initially, to minimize the number of cards which need to be renumbered by subsequent insertions.

Example 2:

If the old master looks like:

```
              OLD1                                          00000100
              OLD2                                          00000101
              OLD3                                          00000200
```

and the update deck is the following:

```
 ./ CHANGE OLD
 ./ INSERT 100
              AA
              BB
              CC
              DD
```

the result would be:

```
              OLD1                                          00000100
              AA                                            00000101
              BB                                            00000102
              CC                                            00000103
              DD                                            00000104
              OLD2                                          00000105
              OLD3                                          00000200
```

where card OLD2 has been renumbered to accommodate the inserted cards.

3.5 REPLACE (R)

Two possible ways to REPLACE one or more cards in the old master with new data cards have already been described. One method is to use a combination DELETE and INSERT commands which IEBUPDTX allows in either order. Another way is to punch the data on a card in columns 1/72 with the sequence number of the card to be replaced in columns 73/80. This way can be very time consuming. The third way is to use the REPLACE command. The data to be inserted must immediately follow the REPLACE command in SYSIN and contain blanks in columns 73/80 (just as for the INSERT command).

```
./ REPLACE {SEQ1=seq1}[,SEQ2=seq2,INCR=incr,NEW1=new1]

./ REPLACE {seq1}[,seq2,incr,new1]
```

SEQ1	specifies the first card to be replaced.
SEQ2	specifies the last card in the range to be replaced. If omitted, only one card is replaced (SEQ1).
INCR	specifies the numbering increment for the inserted card. INCR defaults to 1 unless a value was coded for the INCR PARM option.
NEW1	is the sequence number to be placed on the first inserted card. The default is SEQ1.

Example 1:
```
           ./ REPLACE 200-280
     or    ./ REPLACE SEQ1=200,SEQ2=280
     or    ./ R 200,280
```

first deletes from 200 to 280 inclusive, and then inserts the new data with sequence numbers beginning at 200.

The number of cards inserted need not equal those deleted.

Example 2:
```
           ./ REPLACE 200,INCR=5
     or    ./ REPLACE 200,,5
     or    ./ REPLACE 200,,5,200
     or    ./ REPLACE SEQ1=200,INCR=5,NEW1=200
     or    ./ REPLACE 200,200,5,NEW1=200
```

replaces card 200 with the one or more cards following the REPLACE card in SYSIN. If INCR had not been specified as 5, it would have defaulted to 1, and where NEW1 was not specified, it defaulted to 200.

3.6 <u>FIX</u> (<u>F</u>) <u>CFIX</u> (<u>CF</u>)

Often it is necessary to replace an entire card simply because one word on it was misspelled, such as an assembly language op-code. (Murphy's law requires that the card with the smallest such mistake has the longest and most complicated operand field). Rather than REPLACE the entire card, the FIX command may be used to change a string located anywhere on a card or in specified columns. FIX will change the first occurrence of the string in each line. The columns (COL) option allows parts of a card such as the length of the operand to be changed while allowing the comments to start in the same column.

The syntax of the FIX command is

```
   ./ FIX {SEQ1=seq1}[,SEQ2=seq2,COL1=col1,COL2=col2]
            {#badstring#goodstring#}
   ./ FIX {seq1}[,seq2,col1,col2] {#badstring#goodstring#}
```

SEQ1 specifies the first card to be fixed.

SEQ2 specifies the last card in the range to be fixed.

COL1 specifies the first column to be searched in the old master. The default value is 1. COL1 must be less than or equal to COL2. The COL options allow definition of the subfield which is to be searched for the string.

COL2 specifies the last column which is to be searched. The default is 71. COL2 must be less than or equal to 72.

# represents any non-blank character not a member of the good or bad string.

badstring represents any string from 1 to 32 characters which is to be replaced. At least one blank must follow the range specification and precede the bad string.

goodstring represents any string from 0 to 32 characters.

The FIX operation operates from columns 1/71 by default, so that continuation characters in column 72 will not be affected. If the "good" string is shorter than the "bad" string, the right end of the resulting record is padded with blanks. If the "good" string is longer than the "bad" string, information to the right of the "bad" string is shifted to the right. If any information, other than blanks, is lost, a warning message is given.

Example 1:

```
          ./ FIX 2475 #LPR#LCR#
```

directs the update program to search columns 1/71 of the old master card 2475 for the FIRST OCCURRENCE of the string "LPR" and replace it with the string "LCR".

In the above example, the range consists of single card.

The 'bad' and 'good' strings may be of different lengths. The 'good' string may even be null. It may be necessary to specify more of the string than just the part you want replaced if there is more than one occurrence of it in a line. Only the first occurrence of the specified string on each card in the range is affected. To change every occurrence of a string, use the SCAN command.

Example 2:

```
           ./ FIX 42,,10-15    /B/BE/ or
           ./ FIX 42,COL1=10,COL2=15   /B/BE/
```

changes an assembly language mnemonic from B to BE in columns 10 through 15 of card 42. The second comma in the first FIX command marks the omitted positional parameter seq2.

It is slightly more efficient to specify columns when they are known, as this minimizes the amount of searching required by IEBUPDTX to locate the "bad" string.

The FIX command is unusual in that it doesn't interfere with other update commands. That is, you may specify more than one FIX active on a given card, or you may have a FIX active on a range of cards and still make insertions and deletions within the range. However, inserted cards will not be FIXed.

Due to the special format of the FIX cards, the cards may not be continued once the sequence/column fields are specified. When you are using long strings, continuation may be done by continuing the sequence or column fields. For example :

```
 ./      FIX   100,200,1,
 ./ 71 #very long bad string#very long good string#
```

where 1 and 71 are the default columns.

3.7 SCAN (S) CSCAN (CS)

SCAN is like FIX but SCAN replaces every occurrence of the "bad" string in the same line, not just the first. Thus, SCAN is somewhat less efficient than FIX. The syntax is the same as for FIX. When SCAN causes no substitution to be made, the update is terminated. CSCAN may be used instead of SCAN; it does not raise an error condition.

```
./ SCAN {SEQ1=seq1}[,SEQ2=seq2,COL1=col1,COL2=col2]
        {#badstring#goodstring#}
./ SCAN {seq1}[,seq2,col1,col2] {#badstring#goodstring#}
```

3.8 NOTE (*)

Comments may be placed on update commands by placing at least one blank after the last operand and following it with a comment. If you really have a lot to say, the NOTE detail command may be used:

```
./ NOTE comments

./ * comments
```

NOTE may be used only within the range of a CHANGE OR ADD command.

3.9 ENDUP

This command indicates an end of file and is provided for compatibility with IEBUPDTE. It is optional in most cases, but must not be used with the LOAD function. In that case, it would be executed and would terminate the update instead of being loaded into the library.

```
./ ENDUP
```

Section 4

ADDITIONAL DETAIL STATEMENTS

The detail statements described in Section 3 were all explicitly concerned with the update process. The detail commands presented here provide services which do not affect the basic update in progress, but may still be useful. Like all detail statements, those described here must be used with CHANGE or ADD.

4.1 ALIAS

The ALIAS command modifies the CHANGE or ADD command which it follows by specifying an additional name by which the member being stored in SYSUT2 (PDS) may be referenced. The ALIAS command is written as

```
./ ALIAS {NAME=name}

./ ALIAS {name}
```

NAME   is the additional name of the member being placed in SYSUT2.

A maximum of sixteen ALIAS cards may be used for any one ADD or CHANGE command. With the CHAIN feature, only one ALIAS is permitted, and the name must be the production member name. An ALIAS card may be placed anywhere behind the ADD/CHANGE which it modifies, preferably after the last detail card.

4.2 GANG (G)

The GANG command provides a gang-punching facility. It allows you to specify an identification code of up to 8 alphanumeric characters. This code will be placed in each data card inserted into the new master by the update deck in SYSIN, or as a result of a SCAN/FIX operation effected by SYSIN. Successive updates may have the same or a different code.

```
./ GANG [CODE=code,COL=col]

./ GANG [code,col]
```

code	is the string to be gang-punched. If code is omitted, the previous GANG operation is terminated. When GANG is specified on a PARM, a GANG with no CODE operand restores the GANG id specified on the PARM. Two special values may be supplied for CODE: CODE=SSI causes the trailing non-zero values of the current SSI to be used; e.g. if the SSI is 01234567, the GANG id is 1234567. The specification of CODE=DATE results in a 5-byte string of the form "yyddd".
COL	is the column in which the code is to begin. If omitted, the code is placed in (column 72 - code length) so that the ganged string ends in column 71.

GANG is one of the few "./" control cards which does not terminate an INSERT operation, permitting the GANG to be altered at any time.

For example: ./ GANG CODE=23GS2032
will place the code 23GS2032 on each card inserted beginning in column 64 (72-code length of 8 = 64).

The GANG CODE unconditionally overlays existing data in a card image. Such an overlay is flagged with either a G or an S in the print line for the card. To make GANG a little safer to use, the PARM option ASM may be specified. When ASM is on:

- GANG is not applied if column 72 is non-blank (ASM continuation)
- GANG is not applied unless the columns before and after the code position are blank.
- GANG is applied when either the code field is blank, or the code length is 5 bytes, and the old location is numeric. The second option is provided to permit replacing existing GANG information which uses GANG in date format (yyddd).

4.3 COPY

The COPY command provides the ability to enter data from a separate library into SYSIN (except that COPYed data is not GANGed). The data fetched as a result of a COPY command is taken from the PDS described by the DDNAME operand. The default DDNAME is SYSLIB, but the LIBDD PARM option may be used to specify another default. Copied data is not checked for ./ in columns 1/2. If such a "copy library" is not provided, IEBUPDTX will terminate with return code 12.

COPY is valid without restriction under ADD, but must be in the range of an active INSERT or REPLACE when used under CHANGE. The COPY command, like GANG, does not terminate an active INSERT. When under CHANGE, the sequence field 73/80 is automatically blanked.

```
./ COPY {NAME=name}[,VERSION=version,FROMSEQ=fromseq,
            TOSEQ=toseq,DDNAME=ddname]
./ COPY {name}[,,fromseq,toseq,DDNAME=ddname]
```

NAME	specifies a member of SYSLIB which is to logically replace the COPY command in SYSIN.
VERSION	is described in Section 6.1.
FROMSEQ	specifies the first card to copy. If omitted the COPY begins with the first card in the member. FROMSEQ=0 may be coded to explicitly request the default.
TOSEQ	specifies the last card in the range to be copied. If omitted, the COPY ends at the last card. If FROMSEQ was specified and TOSEQ is omitted, only one card is copied. TOSEQ=END may be coded to explicitly request the default.
DDNAME	specifies the ddname of the copy library to be copied from. The default is SYSLIB unless the LIBDD PARM option specifies another name. The PDS may be in WYLBUR edit format but must have sequence numbers in col 73/80 (see Section 7.1).

"./" control statements which are copied are treated as data. A CHANGE command may be used to copy from SYSUT1 to SYSUT2 and an ADD - COPY combination may be used to copy from any SYS ddname to SYSUT2.

Example :

```
./ C HASPINIT
./ R  100K
./ COPY HASPGEN
```

The COPY statement will be logically replaced in SYSIN by the SYSLIB member named HASPGEN. SYSIN will then look like:

```
./ C HASPINIT
./ R 100K
(all records of HASPGEN here)
```

4.4 MACRO

The MACRO command is similar to COPY except that under CHANGE it need not follow an INSERT and the sequence field is not blanked out. The copied data will be written to SYSUT2 at whatever point the update happens to be, and with whatever sequence information happens to be in the copied member. Data copied in this manner logically appears to have originated in SYSUT1 rather than in SYSIN.

The MACRO command is restricted in that no range may be specified (its range is always 0-END). In addition, the DDNAME of the library cannot be overridden.

```
./ MACRO {NAME=name}[,VERSION=version]

./ MACRO {name}[,version]
```

NAME	is the name of the member to be copied.
VERSION	is as described in Section 6.1

MACRO commands are ignored if not in the primary control file (SYSIN). Data inserted using MACRO is not listed unless the LIST=ALL option was selected on the CHANGE/ADD command. It is also not included in the DECK option's output.

Example :

```
./ C HASPINIT
./ MACRO  HASPGEN
```

produces the same effect as the example in the COPY command if HASPINIT contains no statement with a line number less than 100K.

4.5 <u>LOCATE</u> (<u>L</u>)

LOCATE is used to position the new master to the specified sequence number. It may be used prior to a COPY or MACRO statement to control the placement of the copied data.

```
./ LOCATE SEQ1=seq1[,INCR=incr,NEW1=new1]

./ LOCATE seq1[,incr,new1]
```

SEQ1	is the sequence number to which the new master is positioned, and after which insertions will be placed.
INCR	specifies the numbering increment. INCR defaults to 1 unless a different value of INCR is coded in the PARM statement.
NEW1	specifies the sequence number to be placed on the first inserted card. NEW1 defaults to SEQ1+INCR.

The NEW1 and INCR parameters are ignored when LOCATE precedes a MACRO request. They are used if LOCATE is followed by COPY or unsequenced data cards to be inserted.

LOCATE functions exactly as does INSERT, except that no warning message is issued when no insertion is made.

Section 5

ADDITIONAL PDS FUNCTION STATEMENTS

The following commands are useful for PDS maintenance and may be used only for PDSs. They are function statements and do not require an ADD or CHANGE card.

5.1 LIST

It is often desirable to LIST the source data in a PDS member. This could be done using the CHANGE command to update the desired member, with SYSUT2 directed to a printer and a null change-update deck, or by having SYSUT2 be the DUMMY data set, and specifying LIST=ALL. Unfortunately, this limits the listing capacity to SYSUT1, unless an ADD - COPY combination is used, and requires running a separate step to perform updates. Alternatively, it requires using INDD and OUTDD PARM options, with additional DD cards.

The LIST command provides for listing the contents of a PDS member in a more natural and efficient manner, and is written as follows:

```
./ LIST {NAME=name}[,VERSION=version,FROMSEQ=fromseq,
           TOSEQ=toseq,DDNAME=ddname]
./ LIST {name}[,version,fromseq,toseq,DDNAME=ddname]
```

NAME	is the name of the PDS member to be listed.
VERSION	is as described in Section 6.1
FROMSEQ	may be used to specify a starting sequence number to be listed. If TOSEQ is omitted, at most one card is listed.
TOSEQ	specifies the sequence number of the last card to be listed.
DDNAME	specifies another library to be listed from. If listing members from more than one library, it is most efficient to group all the LIST cards for a particular library together. The default name is SYSLIB unless the LIBDD PARM option specifies another name.

5.2 LOAD

The LOAD command creates a library of update (CHANGE and ADD) decks. Each CHANGE or ADD deck following a ./ or LOAD card in SYSIN becomes a member of the PDS. Each member is stored in SYSUT2 under the name given on its CHANGE or ADD card. The ENDUP statement should not be used with LOAD because it is executed when encountered, and the update is terminated.

```
   ./ LOAD
```

The update decks being stored must contain only detail updating commands, since a major command would be executed, not loaded. The decks should not contain any ENDUP cards. The update decks are further restricted in that their CHANGE or ADD cards may not be continued.

The LOAD function is intended to support chained libraries used with recursive or multiple version updates in a later run. The CHANGE and ADD cards will be treated as specifying VERSION=NEXT, or VERSION=0 (when the name does not already exists). Thus LOAD should not be used to store an update deck intended for another purpose. To store an update deck directly, you may use ADD or CHANGE and the CTL PARM.

5.3 GENALIAS

GENALIAS assigns an alias to an existing member in the new master. It does not require an ADD or CHANGE operation for that member.

```
   ./ GENALIAS {NAME=name,NEWNAME=newname}

   ./ GENALIAS {name,newname}
```

NAME	is the name of the existing member.
NEWNAME	is the name of the alias to be assigned to it. GENALIAS is not valid when CHAIN is in effect, and should not be used for members in a CHAIN library which have multiple versions.

5.4 RENAME

The RENAME command may be used to change the name of an existing SYSUT2 member. It is written:

```
./ RENAME {NAME=name,NEWNAME=newname}

./ RENAME {name,newname}
```

NAME	is the name of the new master PDS member to be changed.
NEWNAME	specifies the name to be given to the new member in the new master PDS.

5.5 SCRATCH (DROP)

Occasionally, it may become necessary to completely remove a member from a library (PDS). This operation may be done by using the SCRATCH command, which will delete the named member from the SYSUT2 data set. This command is written as follows:

```
./ SCRATCH {NAME=name}

./ SCRATCH {name}
```

NAME	is the PDS member to be removed from the new master.

5.6 <u>RESTORE</u> (<u>RECOVER</u>)

The RESTORE statement may be used to restore a member which has been scratched. It will work correctly only on the original library from which the member was scratched (or an FDRDSF restored version thereof), which has not been compressed in the interim.

```
./ RESTORE NAME=name hexadecimal text

./ RESTORE name hex,text
```

NAME	where NAME is the new name to be assigned to the member (it need not be the same as the original, but it must be unique in the new master). At least one blank must follow the NAME field.
hex text	is the text from a prior IEBUPDTX listing, and is copied without the labels from the OLD/NEW name ENTRY: line produced in the listing. RESTORE does not permit continuation or comments.

Example :

On a prior IEBUPDTX run, an ADD operation might produce

```
    <01> MEMBER XYZ HAS BEEN ADDED       132 RECORDS
    NEW XYZ      ENTRY: TTR=001203 INDC=04 SSI=00081355 DATA=8135522F
```

A subsequent IEBUPDTX ./ SCRATCH request would produce

```
    <01> ./  SCRATCH NAME=XYZ
    OLD XYZ      ENTRY: TTR=001203 INDC=04 SSI=00081355 DATA=8135522F
```

the corresponding RESTORE card would contain

```
         ./ RESTORE XYZ 001203,04,00081355,8135522F
```

The commas are intended for legibility and are not significant.

Caution is advised when RESTORE is used for chained members. Many CHAIN updates modify two entries in the directory (the version alias and the production alias). When the highest version alias is scratched, the production alias is rewritten with the new 'high' version number. A RESTORE for the scratched version would not update the production entry, resulting in incorrect operation of the VERSION=NEXT operand on a subsequent ADD/CHANGE.

RESTORE only performs minimal validity checking. If you enter incorrect information, the new member may be unusable or the library may become not copyable with IEBCOPY/IEHMOVE.

Section 6

ADDITIONAL UPDATING FEATURES

The features described below were deliberately omitted from the discussion in Section 2. Knowledge of these features is not required to use IEBUPDTX properly - but is required to use IEBUPDTX elegantly.

6.1 VERSION (CHAINED) LIBRARIES

Chained libraries allow multiple versions of a member to be stored in a PDS. They are convenient for the following:

- When updating an old master, the new master can be placed in the same PDS as a new version of the same member via the CHAIN=id option. After testing this new version, the CURRENT command can then be used to specify it as the production version. The older version need not be deleted and can be respecified as the production version (using CURRENT and VERSION=) if the new version should fail to perform correctly.
- Chained libraries can also be created to contain update decks. By using the CHAINUPD PARM option, chained members starting from version 0 to the production version are applied recursively to update the old master.

See Section 6.2 for example using a chained library as update.

The CHANGE and ADD commands described in Section 2 will automatically destroy an identically named member in SYSUT2. The use of VERSION on CHANGE or ADD commands avoids this possibility by automatically assigning a unique new name to the new master as it is stored in SYSUT2. With CHAINing, the first version of the member is stored with the actual name found on the CHANGE or ADD card as VERSION 0 (zero). It looks just like a member stored in any OS PDS - and hence any members which exist in a library before converting to CHAINing automatically behave like version 0 members. When storing a new member into SYSUT2 with CHAINing specified, however, it must be explicitly declared as VERSION=0.

So far, nothing special has happened. CHAINing is not implemented until a second version of the member is stored in SYSUT2. When CHAINing is implemented for a member, a member ID number is obtained from a special directory entry in SYSUT2 called the allocator (written as @LLOCATR) and assigned to the member. All versions of this member will have this member ID number in common. The PDS directory entry for version zero of the member is expanded to include, among other things, this ID number, and is re-written along with another directory entry for version zero's special "internal" name back to SYSUT2's directory.

Now there are three directory entries for the two versions of the member - one has the version zero internal name and points to (i.e. contains the TTR of) the original member, the second has the internal name representing version one, and points to the new member, and the third has the actual external member name of the member (which you put on the CHANGE card to refer to it) and still points to the version zero (original) member.

The format of an internal name is

two byte library id	three byte member id	three byte version no

Internal names may be used directly (with due caution) in most cases. Doing so will generally cause VERSION=NEXT operations to work incorrectly and possibly cause an old version to be scratched. This alternative is provided to facilitate recovery when a production alias is inadvertently scratched.

Since this third directory entry (called the "production alias" because it contains the real "production" name of the member) still points to version zero, any reference made by any OS function or program to the CHAINed member name will continue to provide the source data in version zero. Similarly, references made by IEBUPDTX as described in Section 2 will also produce only version zero. Hence your version is still invisible to the other users of your CHAINed library and will remain so until a new production version is designated with the CURRENT command.

In order to test or use another version, COPY (or MACRO) the version desired, placing it within the new master (temporary) output, and passing this to a compiler or whatever.

If someone runs an update of the member specifying version one again, version one will be replaced. To avoid this, the next member should be put in as version two (or as VERSION=NEXT). Example:

The following will load different versions of a same member into a chained library:

```
 // EXEC PGM=IEBUPDTX,PARM='CHAIN=SH'
         .
         .
 ./ ADD MASTER,VERSION=0
         .
       data
         .
 ./ ADD MASTER,VERSION=1
         .
        data
         .
 ./ ADD MASTER,VERSION=2
         .
        data
         .
```

6.1.1 CURRENT

As soon as a new version is debugged, the production alias may be rewritten to point to it instead of version zero by using the CURRENT function statement.

```
./ CURRENT {NAME=name,VERSION=version}

./ CURRENT {name,version}
```

NAME   is the name of the member to be affected.

VERSION   is the version number to become the production version.

CURRENT will set the production alias to point to the version number specified. This does not cause the previous "current" version to be lost, since it is still locatable by its internal name. Hence, the previous (working) version can always be retrieved in an emergency by using another CURRENT command.

Example:

The following inserts version 8 as the production version in a chained library of update decks (assuming that versions 0 - 7 already exist):

```
 //  EXEC PGM=IEBUPDTX,PARM='CHAIN=UP'
         .
         .
 //SYSIN DD *
 ./ LOAD
 ./ CHANGE A
 ./ INSERT 100,10
         XX
         YY
 ./ CURRENT A,VERSION=8
```

If the library is now used as an intermediate update with a PARM of 'CHAIN,CHAINUPD' (see Section 6.2), all version members from 0 - 8 will be used as recursive updates.

6.1.2 Specifying Version Numbers

Version numbers may be specified on most function statements and in two detail statements (COPY and MACRO). Table 6 lists the commands for which VERSION= is valid. Version numbers are specified as the second positional operand, or by using the VERSION= or V= keywords. VERSION=NEXT may be specified on a CHANGE or ADD card to prevent accidental deletion of a version when the user is not certain of the next version number. The output listing will tell which version number should be used to subsequently reference the version created or attempted by that run.

Version number may be designated in three ways:

```
     HASPINIT-2
     HASPINIT,2
     HASPINIT,VERSION=2
```

are all identical in meaning. All refer to version 2 of HASPINIT.

VERSION specified on a CHANGE card applies only to the new master member name. The production version will be updated. ADD-COPY and recursive updates may be used to update an arbitrary version.

To use version numbers on CHANGE/ADD commands, it is necessary to inform IEBUPDTX that SYSUT2 is to be considered a CHAINed library. This is done via an EXEC PARM, "CHAIN=ID". ID will be used as the two byte library id for the library defined by SYSUT2 if it doesn't already have one. It is a good idea to give each library a unique id so that there will be no ambiguity possible if they are concatenated later. If the library already has an id (has been run once before with CHAIN=ID specified) you may omit the id, coding simply "CHAIN".

Example 1

```
     ./ CHANGE HASPINIT-3
```

means update the current version of HASPINIT and store the result as version 3.

Example 2

To update version 2:

```
     //SYSIN DD *
     ./ CHANGE HASPINIT-3
     //OPDT1 DD *
     ./ ADD  HASPINIT
     ./ COPY HASPINIT-2,DDNAME=SYSUT1
```

Version numbers may be used on commands as shown in Table 6.

TABLE 6

VERSION operand by Command

COMMAND	With CHAIN= PARM	Without CHAIN= PARM	VERSION=NEXT allowed in command
ADD	Required ALIAS is restricted	Not valid	Yes
CHANGE	Required ALIAS is restricted NEWNAME is not allowed	Not valid	Yes
SCRATCH	Yes	Yes	No
CURRENT	Yes	Yes	No
LIST	Yes	Yes	No
COPY	Yes	Yes	No
MACRO	Yes	Yes	No

To SCRATCH any particular version of a member, after which it cannot be retrieved, simply use that version number on the SCRATCH command:

```
          ./ SCRATCH name,version
```

It is recommended that versions of a member be scratched in decreasing order of version numbers. IEBUPDTX keeps track of the next version number to be assigned, but the process only works if the highest version is scratched first. For example, if versions 3, 4, and 5 exist, VERSION NEXT is 6; if version 5 is scratched, IEBUPDTX subtracts 1 from NEXT and NEXT will equal 5. If, however, version 4 is scratched, NEXT is not affected (and remains 6) because 4 was not the highest version created.

6.2 RECURSIVE UPDATES

The use of two or more update decks at the same time to update a single member is called a recursive update. Given an original source module and the update deck necessary to go from level 0 to level 1 of this module, suppose that you want to update again to level 2. If the level 1 source module is available, it may be used, but for large modules, it may not be practical to retain complete source at every level. Given such a situation, one can either:

- create level 1 (temporary) from level 0 and update to level 2
- update the level 1 update deck and update directly from level 0 to level 2 (a very risky procedure)
- use two update decks recursively

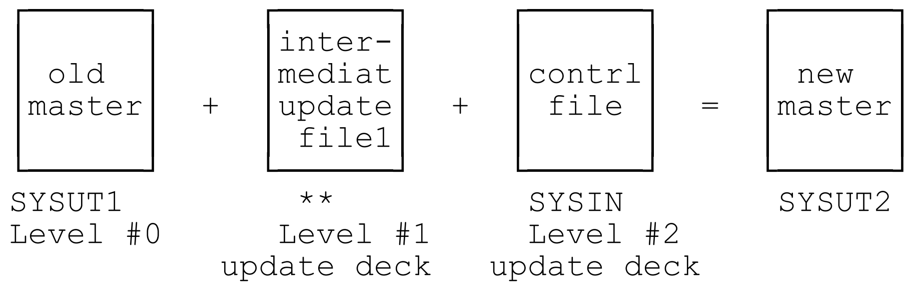
A recursive update is a "nested" update and is defined for each old master card image as:

    level 2 = update2(update1(level 0)).

To perform such an update, input:

- level 2 update deck (which updates level 1 to level 2) via the control file
- level 1 update deck via an intermediate update file

An intermediate control file is identified to IEBUPDTX by any DDNAME which does not begin with the letters SYS. If more than one intermediate update file is used, they are applied in the order of the DD JCL statements which identify and define them. SYSIN is always applied last, irrespective of the placement of the SYSIN DD statement in the JCL. It may be convenient to consider SYSUT1 as update level 0, which of course always comes first.



When using intermediate update files, the member name found on the CHANGE card in SYSIN is used to locate the update decks which correspond to the same member in the intermediate files. An intermediate update file may be sequential or partitioned. If it is sequential, the order of updates must follow the order of CHANGE cards in SYSIN; if it is partitioned, the update program can locate intermediate update decks automatically. Using partitioned intermediate update files is highly recommended for doing more than one update in a single job step.

The SEQID option, described under CHANGE and ADD, will not function well with some sequential intermediate situations. When using SEQID, INCR, or NEW1 on an intermediate CHANGE card, provide dummy CHANGE cards as place holders for members not being updated at this level. The name on the place-holder's CHANGE card must match the omitted intermediate update. It is permissible to omit an update for any SYSIN named update from an intermediate update library. It is also permissible for an intermediate update deck to specify the ADD function, in which case the source data from that member will be used as the old master.

The recursive updating technique is sometimes convenient even for a single update per member, since all the update decks could be placed in a single PDS (using the LOAD function) and the updates desired could be selected with simple CHANGE cards in SYSIN. It is also useful for updating something from a library other than the old master file defined by SYSUT1--(for instance, if SYSUT1 has an identically named member which was not to be updated)--since an ADD - COPY combination may be used in an intermediate file to provide old master source.

Also note that only updating commands are valid in intermediate update decks. The LIST, RENAME , SCRATCH, CURRENT, PARM, LOAD, ALIAS, and MACRO commands are not functional unless in the main control file. If SYSIN is omitted (not recommended), IEBUPDTX will use an intermediate file as the main control file provided that the last such file defined is sequentially organized.

6.3 CHAINED RECURSIVE UPDATES

If the CHAINUPD PARM option is specified, IEBUPDTX will check each intermediate member update fetched from a PDS to see if it is actually the production alias of a chain of updates. If this is so, each update in the chain from version 0 through the production alias will automatically be applied. If NOCHAINUPD had been specified (the default), only the production alias update would have been applied.

SEQID is a global option, and is not handled separately for each recursive update deck. Any SEQID specified by a lower level update is completely overlaid by specifying SEQID in a higher level update's CHANGE card.

If you are using a sequential intermediate update deck whose CHANGE card specifies any of SEQID, INCR, or NEW1, and if the intermediate update will not be used immediately because the previously performed update didn't update the member named on its CHANGE card, you must provide a dummy CHANGE command to precede the delayed intermediate update as a place-holder. (The name on the place-holder's CHANGE card to match the omitted intermediate update).

Section 7

ODDS AND ENDS

7.1 WYLBUR CONSIDERATIONS

Any or all data sets used in IEBUPDTX may be in WYLBUR edit format (i.e. RECFM=U, LRECL=BLKSIZE=3156 or 3520). IEBUPDTX unpresses input data set into 80 byte card images and vice versa for output data sets.

Additionally, a new PARM option, INTEGER, can be negated if WYLBUR line numbers and sequence numbers are synonymous. PARM=NOINT applies only to SYSUT1 (old master) and SYSUT2 (new master) data sets since update decks usually do not contain sequence numbers.

There are several restrictions if SYSUT2 is in WYLBUR format and the INTEGER option is turned off:

- SEQID= on ADD or CHANGE cards may not be used.
- SYSUT1, the old master, may not have an old SEQID.
- When doing an initial ADD (i.e. no SYSUT1), sequence numbers are required, and may be supplied in columns 73/80 explicitly, or by leaving columns 73/80 blank and using the NEW1/INCR options on ADD.

Advantages to not using the INTEGER option are:

- minimizing the amount of space used - 8 or more bytes are saved for every line in the data set;
- no need to put sequence numbers in columns 73/80 when inserting new lines in the data set.

Note that since WYLBUR numbers start at .001 and sequence numbers start at 1, the latter will always be 1000 times the former. Hence if the WYLBUR data set begins numbering at 1, the corresponding sequence number starts at 00001000.

7.2 WRITING USER UPDATING EXITS (USER COMMAND)

With IEBUPDTX it is possible to define your own update commands. To do this, it is necessary to write a USER update exit routine in assembly language as described below. This program is loaded from the default STEPLIB or JOBLIB and branched to as specified by the USER detail command.

This command is written as follows:

```
./ USER {NAME=name,SEQ1=seq1}[,SEQ2=seq2,CODE=code]

./ USER {name,seq1}[,seq2,code]
```

NAME	specifies the entry point name of the load module to be used as the USER exit routine.
SEQ1	specifies the sequence number of the first old master record to be USER updated.
SEQ2	specifies the sequence number of the last old master record to be USER updated. Seq2 may be omitted if only one record is to be updated.
CODE	is any string of from one to eight alphanumeric characters which will be made available to the USER exit routine, right-justified in a doubleword. The code default is eight blanks.

For each old master card found in the range defined by seq1-seq2, IEBUPDTX will branch to the entry point name of the exit routine. The USER exit routine may delete, modify, insert, or leave unchanged at or before the current old master card. This choice must be communicated to IEBUPDTX by setting the appropriate return codes in register R15, as follows:

0	Do not modify the current old master card.
4	Delete the current old master card.
8	Replace the old master card with the specified data (see below).
12	Insert specified new data before the current old master record. Branch to this routine again with the same old master record.
16	Stop this member update (severity 8)

The following registers are set at entry to the exit routine:

13-15 Standard OS linkage conventions.

1 pointer to 4 doublewords:

Offset

hex 0: The code value from the USER command. Code is right-justified in its eight byte field, blank if omitted on USER command.

hex 8: Work area: First word set to zero for each new USER command (not for each new old master card). Second word initially zero (at the beginning of each member update) but not changed by IEBUPDTX after that. The work area must be used to retain information between successive invocations of the exit routine, as following an insert. This is because the same routine may be in use by an intermediate update.

In general, the contents of seq1 and seq2 are unimportant. Seq2 may be set equal to the seq1 value to inhibit any possible future calls to the exit routine for the current USER command, except for the call which must follow an insert request. Do not set seq1 and seq2 in any manner so as to prevent this. (i.e., never set seq2 less than the sequence number of the current old master card.)

hex 10: seq1 from the USER command (EBCDIC with leading zeroes).

hex 18: seq2 from the USER command.

2 The contents of R2 + 6 point to the current old master record. (The first 6 bytes are the identification bytes, as seen in the listings -- e.g. <01>, MAC , or blanks). Do not modify any of the old master data directly.

3 Register R3 points to 6 below a 72 byte area in which to place replacement or insertion data. E.g., MVC 6(72,R3),NEWREC).

4 Register R4 points to PWA (the Print Work Area) of IEBUPDTX, and may be useful if messages are to be printed from the exit routine. Note that such messages must be printed using the XPRNTLIN macro only. XPRNTLIN is a macro found in IEBUPDTX source.

The USER exit routine must be serially re-usable, with no memory (other than the work area provided by IEBUPDTX) between invocations.

The following discussion may be helpful in using the USER feature most efficiently:

Whenever a branch to a user exit is required, IEBUPDTX compares the entry point name (saved from the USER command) against a global field which contains the name of the entry point last LOADed. If the names match, IEBUPDTX branches directly to the exit routine. If they don't match, IEBUPDTX must first issue the DELETE macro instruction for the previous routine, if any, and then the LOAD macro instruction for the new entry point name. After the new exit routine is loaded, its name and address are saved globally.

Hence it is most efficient to have a single USER exit routine per update whenever possible (one routine may perform several functions by testing the code specified on the USER command), and least efficient to have several separate load modules invoked in a mixed sequence.

If SCAN or FIX is active on the same card image as USER, USER is applied last (to allow the exit routine to detect, perhaps, if the identification field of the old master record is blank). Only one USER command can apply to an old master record, unless one uses intermediate update files.

A 'global' user exit may be specified via the PARM field. Such an exit routine applies to each card about to be written into the new master, and may only ignore, modify, or delete. In this case, modifications may be applied (by the exit routine) directly to the input record, without moving it. For compatibility, the global exit is called with R2 = R3. Furthermore, only one global exit may be active at a time and no CODE may be passed. An example of a user exit routine which may be employed either locally or globally is HERB. HERB, which inserts a minus sign into assembly language macro model statements just after the operand field (to improve readability of assembly listings), is supplied with the IEBUPDTX distribution tape.

7.3 DYNAMIC INVOCATION OF IEBUPDTX

IEBUPDTX may be invoked from Assembler programs (and others capable of providing the correct parameter lists) in a fashion similar to IEBUPDTE invocation. The interface is similar, but not identical.

- IEBUPDTX does not support the option of specifying a starting page number for SYSPRINT. The documented OPTLIST and DDNMELST entries are supported, HDNGLIST is ignored when present.
- While IEBUPDTX may be invoked via LOAD/CALL, LINK or ATTACH, the program is not re-entrant, and may not be re-usable when errors occur. The preferred form of invocation is ATTACH.
- The supported DDNMELST entries are :

1	reserved - zeroes (XL8'0')
2	reserved - zeroes
3	reserved - zeroes
4	SYSLIB replacement or zeroes
5	SYSIN replacement or zeroes
6	SYSPRINT replacement or zeroes
7	SYSPUNCH replacement or zeroes
8	SYSUT1 replacement or zeroes
9	SYSUT2 replacement or zeroes

Note that IEBUPDTX performs no validity checking on the supplied entries. The DD names for SYSIN, SYSPRINT and SYSPUNCH should always be unique; the SYSLIB name may be the same as SYSUT1 or SYSUT2. SYSUT1 and SYSUT2 should normally be unique, but could be the same when only partitioned datasets are processed.

7.4 IEBUPCHN DIRECTORY LISTING

The standard utility programs commonly used to produce directory listings are not helpful for libraries containing members created with the CHAIN option. IEBUPDTX provides a separate program which reproduces the structure of chained members. The required JCL is:

```
 ... JOB card; other control cards as required
 //step EXEC PGM=IEBUPCHN,REGION=nnnK
 //SYSPRINT  DD  SYSOUT=A    listing file
 //SYSUT2    DD  DISP=SHR,DSN=dsname   library to be processed
 //SYSPUNCH  DD  SYSOUT=B,DCB=BLKSIZE=    [optional punch file]
```

A region specification may be required for extremely large libraries. Each member, alias and version entry requires approximately 36 bytes in storage; an additional 28K should be allowed for program overhead.

When the SYSPUNCH DD is supplied, a DCB=BLKSIZE= must be specified. Output consists of one ./ CHANGE card for each member found.

The listing groups versions of the same production alias together, and indicates the current version by an arrow ->.

When the production alias entry is not found, a warning message is issued. Recovery can be effected in several ways:

- Use FDRDSF or another backup facility to recover from an old version of the library.

- Run a ./ CHANGE using the explicit high version name; e.g.
  ```
                ./ CHANGE NAME=idmmmnnn
  ```
  where "id" is the @LLOCATR (CHAIN) id; "mmm" is the member number; and "nnn" is the highest version number. This name appears on the listing. When this change is completed and has added the production alias back in, you may use ./ SCRATCH name,VERSION=LAST to delete the dummy version created by the CHANGE.

- Use ./ RESTORE with the text of the last (highest) version entry and the name of the production member, then use ./ CURRENT to assign the member to the desired production version (unless it is already the last one).

7.5 RELATED UTILITIES

The IEBCOPY utility, whether used to copy or compress, alters the directory entries for CHAINed members by changing some alias entries to main member entries. This alters directory listings produced after copying, but does not affect IEBUPDTX CHAIN processing.

IEBPDS, the DTS provided utility for listing PDS directories, and the IBM supplied utilities will not produce useful directory listings for libraries created with CHAINing. See the IEBUPCHN program in the preceding section.

PUNK, the DTS provided utility for listing, punching, and reproducing card format oriented datasets, may not produce useful output for CHAINed libraries. IEBUPDTX flags most members as ALIAS entries, which PUNK ignores by default. To ensure that all members are processed by PUNK, it should be invoked with PARM=ALIAS, which causes alias entries to be treated as members. When only the current members of chain entries are desired, the PARM field should also specify a global exclude of other chain versions, using PARM='EXCLUDE=id', where "id" is the CHAIN id of the library. Note that either the CHAIN entries or the @LLOCATR member may be excluded, but not both. Due to the form in which PUNK produces output, the output files will not produce a library in CHAIN format when reloaded (the PDS directory data are not preserved). Libraries with CHAIN entries should be copied with IEBCOPY, IEHMOVE, or FDR/FDRDSF only.

When PUNK is used to produce a SYSUPDTE file (unload in IEBUPDTE SYSIN format), any cards found within a member beginning with ./ are converted to @/. This conversion is done to prevent conflicts with ./ ADD control cards punched by PUNK for the members proper. When these members are restored, the @/ should be changed back, or the members could be processed with IEBUPDTX using the CTL=@/ PARM.

Several DTS supplied utilities, notably WYLVTOC, examine the SSI field of members. When this field is in the form 000nnnnn, it is interpreted as a date (form yyddd) and displayed on the listing. This usage matches that of the WYLBUR SAVE command, the WYLCOMP utility, and others. SSI in different form should be used with due consideration.

INDEX

TABLE OF CONTENTS

LIST OF TABLES