File Creation

Creation, correction, and updating of the MARC-format master file are accomplished through the use of the Update programs. New records are added and existing records are changed in exactly the same manner as the original master file is created.

Creation and correction capabilities

System capabilities for creating new records or changing existing records enable the user to:

* Add, delete, or replace an entire MARC-record.
* Add, delete, or replace any variable field or fields in a record.
* Change any text string in any variable field or fields.
* Change subfield codes or indicators, without re-entering field data.
* Enter the same change for a series of consecutively numbered records, for a range of contiguous records whose "did" numbers are not consecutive, or for multiple records that are not contiguous.
* Print the entire record following any change, or print only the fields affected.
* Display any other record or group of records, without changing them.
* Make systematic changes to one or more variable fields in all records in the file.
* Process transaction lines generated by the Print program (BPSPRT), without the need for an intermediate step to remove carriage control characters.

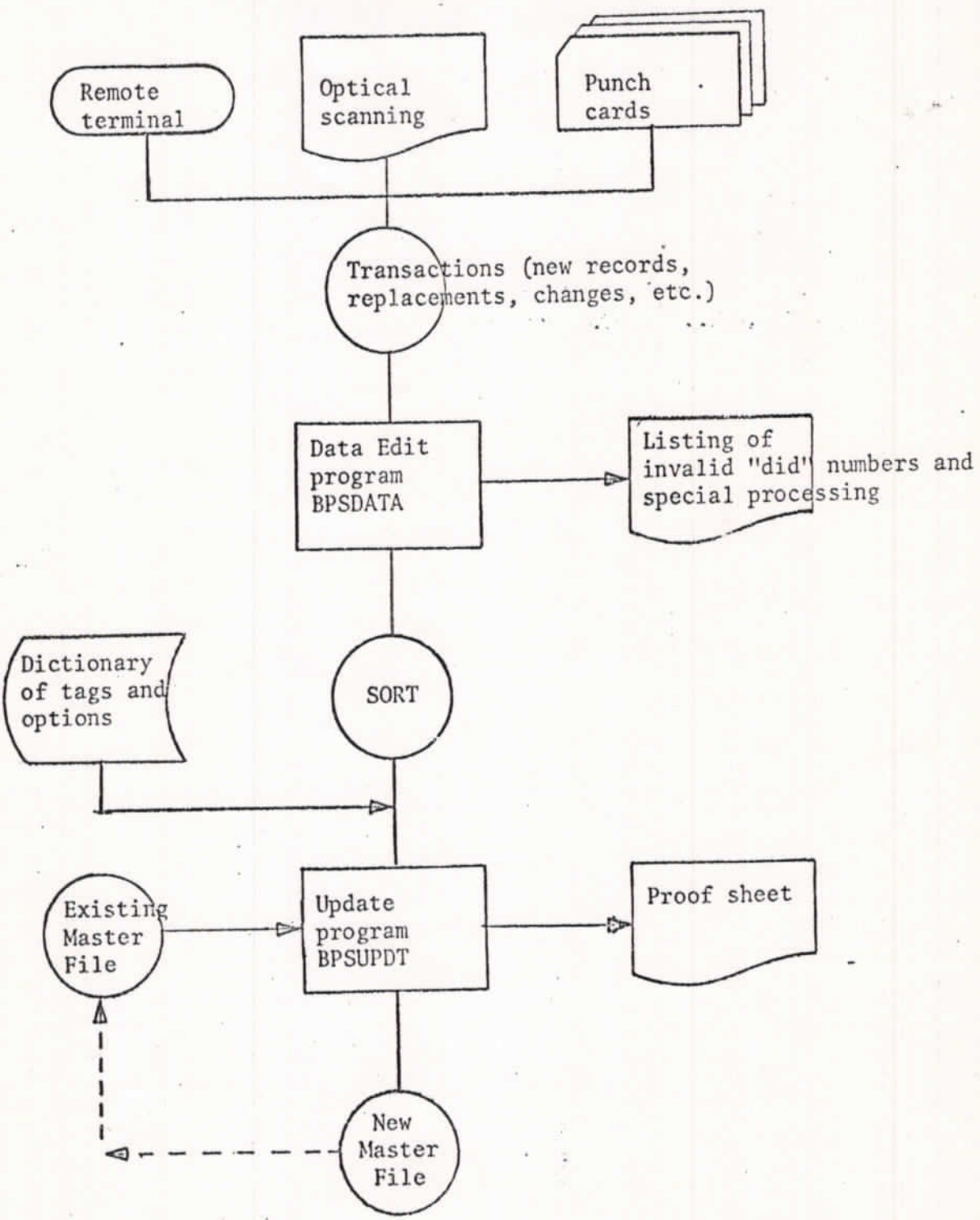
Transaction lines

Input to the Update programs consists of transaction lines, which identify the records and fields for addition or correction, along with the new data to be incorporated. The transaction lines are first read by the Data Edit program (BPSDATA). The edited transaction lines are sorted using a standard sort program (IBM Sort/Merge) and then read by the Update program (BPSUPDT) where they are matched against the master file and the indicated action is carried out.

For example, the following transaction lines could be used to create a new record:

BIBLIOGRAPHIC PROCESSING SYSTEM

UPDATE CYCLE



V. Schoenbach
6/4/76

```
did     B-63
til     The new Oregon Trail;#an account of the
        development and passage of state
        land-use legislation
        in Oregon,#by Charles E. Little.
imp     Washington, D.C.,#Conservation Foundation,#1974.
col     37 p.
pri     $1.00
ser     Conservation Foundation Series.
sup     Land development--Planning.
sup     Political activity.
sup     Legislative hearings.
sum     Conservation.
sum     Environmental policy.
nme     Little, Charles E.
```

The new record will contain a title (til), imprint (imp), collation (col), price (pri), series note (ser), three primary subject terms (sup), three minor subject terms (sum), and an author designation (nme). The particular mnemonic tags must have been included in a Tag/Option Dictionary. In this case, they are those used by the TIS-CPC Library.

Since there are no transaction codes to specify otherwise, the "did" line and the tag lines will be taken as "add" transactions. The three contiguous "sup" tags will be assigned consecutive sequence ("site") numbers (sup/1, sup/2, sup/3) as will the two "sum" tags (sum/1, sum/2).

Two fields, the title (til) and imprint (imp), contain subfield delimiters (#), which define three subfields in each of these fields. Since no subfield codes are present to identify the subfields, they are assumed to be subfields "a", "b", and "c".

Note that the author's name is entered twice, once as field "nme" and once as part of the "til" field. The reasons for this are so that it will be available in both inverted and normal forms, and also so that the statement of responsibility part of the title may have whatever form is desired without restriction on the form of the author's names for the author index.

Types of transaction lines

The four kinds of transaction lines are:

1. "did" lines
2. tag lines
3. continuation lines
4. command lines.

"did" lines

"did" lines identify the record(s) to be added, changed, deleted, or otherwise affected. A "did" line must be the first transaction line for any change to a record. Examples of "did" lines are:

File creation

```
did         A-12
did(C       B-21993
did(D       B-44-*
did         Start-multiple.
did         R-249 thru R-257
did(C       S-12-702-12:S-12-712-19
did(C       All
```

Tag lines

Tag lines identify the variable field(s) to be added, changed, deleted, or otherwise affected. Certain types of transactions (e.g., "delete record") do not involve tag lines. Examples of tag lines are:

```
nme         Lee, S.H.
nme/2(R     Goldfarb, Alvin I.
nme/5(D
til#ac      "Social demography of a new city,"#by John Curry.
til(C       'new' 'planned'
sup/*(C     / labour/ labor/ all
```

Tag lines have a tag in columns 1-3 followed by other identifying and control information.

Continuation lines

Continuation lines contain field data for a previous tag line. Continuation lines have a blank in column 1, to distinguish them from tag lines. An example of a tag line with continuation lines is:

```
til         National resources, policies and planning for
            developing countries,#by Joseph L. Fisher
            and Roger Revelle.
```

Command lines

Command lines are used to alter the mode of operation or default assumptions of the Data Edit program (BPSDATA). Command lines contain a period (.) in column 1 followed immediately by a command word, e.g.:

```
.AUTODID
```

causes all transaction lines that follow to be converted to "did" lines if either column 2 or 3 contains a hyphen (which would be necessary for a valid "did" number).

Components of transaction lines

"did" and tag lines each consist of two parts:

1. a control portion, which specifies the tag, indicators, site number, subfield codes, and transaction code
2. a data portion, which contains the field data.

Control portion

The control portion must consist of at least the mnemonic tag ("did" for a "did" line) in columns 1-3. Additional control information, if present, follows the tag without any intervening blank (the first blank or tab character signals the Update program that field data follows). Except for the indicators, which must follow the tag, additional control information may be entered in any order.

Control information may be in upper case, lower case, or mixed upper and lower case. It will be translated to the Update program conventions for the Proof Sheet:

Tag -- lower case

Indicators -- lower case

Subfield codes -- lower case

Transaction code -- upper case

Field data

The field data is separated from the control information by at least one blank or tab character (hexadecimal 05).

Subfields within the data are delimited by a number sign (#). Blanks are permitted at the end of a subfield (i.e., before the "#"), but not at the beginning of a subfield (i.e., after the "#"). If a delimiter is followed by a space, the Update program will remove it and give a warning indication (one or more asterisks preceding the tag). If two or more delimiters occur without intervening non-blank characters, only the first will be kept and any blanks between the delimiters will be removed. A warning indication will also be given in this case.

If the field data does not fit on one line, it may extend onto as many additional lines as necessary. These continuation lines must begin with a blank or tab character in column 1. One blank will be inserted between the last word of the line being continued and the first word of the continuation line, unless the line being continued ends with a subfield delimiter (#). In the latter case, a warning indication will be given.

Field data may be entered in "as-is" mode or in UPCASE mode. In "as-is" mode, no translation of alphabetic text occurs. Data may be entered in upper and lower case, or if it is entered in upper case it will be preserved in that form.

In UPCASE mode, all alphabetic text is transformed to lower case except letters that are preceded by a shift character and letters that are in words preceded by a double shift character. UPCASE mode may be specified in the Tag/Option Dictionary or the PARM field of the Update program EXEC statement. Unless another shift character has been selected (through the SHIFTCHAR option of the Update program), the dollar sign ($) is used as the shift character.

Examples of use of UPCASE mode are:

(UPCASE mode, without a SHIFTCHAR specification):

til "$$EEG $RESPONSE TO $PHOTIC $STIMULATION."

will be translated to:

```
til        "EEG Response to Photic Stimulation."

     (UPCASE mode, SHIFTCHAR=*):

til        *REPORT OF THE **U.N. POPULATION MISSION.

     will be translated to:

til        Report of the U.N. population mission.
```

The range of the double shift character includes all letters that follow until the first blank space or subfield delimiter. When the SHIFTCHAR option is specified, it is not necessary to specify UPCASE.

Order of transaction lines

The first transaction line for each record to be added, changed, deleted, printed, or otherwise affected specifies the "did" number of the record:

```
did              B-12345
```

The control portion of the did line consists of the letters "did" (upper or lower case) and an optional transaction code:

```
did(C            B-123-12
```

If no transaction code is present, it is assumed that a new record is being defined.

The did line must be followed by one or more transaction lines containing tagged fields if the transaction code of the did line implies the existence of data (new record, change record, and replace record all imply the existence of data to follow). If the transaction code precludes the existence of tagged fields (e.g., delete record), then it is an error to have tagged field transaction lines. Examples of correct transaction are:

```
did(C            R-676
nme/2(R          Villa, Miguel
gap/x            Chile.
did(D            CH-195
did(P            A-9
```

There is no restriction on the number of "did" lines for a particular master file record that may be entered during an update. The only requirements are that each did line must be followed by tag lines if its transaction code implies their presence, or that it not be followed by tag lines if its code implies their absence. Also, transactions such as change, delete, or replace a master file record must identify a record that exists in the master file.

When present, they must immediately follow the tag, e.g.:

```
nmec      United States Information Agency.
```

Indicators may be entered in upper or lower case.

Site numbers

The site number is used to differentiate among multiple occurrences of the same tag (e.g., sup/1, sup/2, sup/3). When no site number is present, then "/1" is assumed unless the field is an "add field" transaction and is preceded by an "add field" transaction for the same tag. In that case, a site number will be generated equal to the value of the site number of the previous field, plus one. Example:

```
nme       "/1" is implied
nme       "/2" is implied
nme(R     "/1" is implied, because this is not an
          "add field" transaction.
nme       "/1" is implied, because preceding tag line
          is not an "add field" transaction.
sup       "/1" is implied
sup/3     "/4" is implied
sup       "/1" is implied, because the tag is not
sum       the same as the preceding transaction.
```

The site number is introduced by a slash (/). Valid site numbers are 1 through 250, inclusive, though the maximum number of fields in any particular record should not exceed 140.

Subfield codes

Subfield codes identify the subfields that are present in the field data. Each subfield delimiter (#) identifies the beginning of a new subfield. If no subfield codes are entered, they will be assigned from the beginning of the alphabet, using one letter for each subfield in the field data.

If the intended subfield codes are anything other than this assumption, then all subfield codes for that field must be specified, in the same order as the subfields they identify occur in the field data. Example:

```
imp#ac    New York, N.Y.,#1974      (publisher missing)
```

Subfield codes are introduced by a number sign (#).

If the number of subfield codes entered is not equal to the number of subfields in the field data (calculated as the number of subfield delimiters plus one), then the Update program will adjust the subfield codes as follows. If too many codes were entered, then the extra ones will be truncated from the right. If too few were entered, then additional codes will be generated after the last subfield code that was entered. The maximum number of subfields (and subfield codes) permitted is 12. If adjustment to the subfield codes has taken place, a warning indication will be printed next to the tag, to call attention to the fact that the input has been altered. Examples:

```
imp             Boston,#Little & Brown,#1975.
                (#abc implied)

imp#b           Albion Press
                (Subfield "b" only)

til#ac          Basic statistics,#by Roger D. Nussbaum.
                (Subfields "a" and "c" only)

til#ac          "A wakeful, hypo-metabolic state."
                (Too many subfield codes)

imp#bc          London,#Sheed & Ward,#1973.
                (Too few subfield codes)
```

The Proof Sheet will print the last two transactions:

```
*****til        "A wakeful, hypo-metabolic state."
*****imp#bcc    London,#Sheet & Ward,#1973.
```

The subfield code assumption for the "til" field, that only subfield "a" is present, will probably be correct. The assumption for the "imp" field will require a correction by the input editor.

Transaction code

The transaction code specifies what action is to be performed for the record or field. If no transaction code is present, "add" is assumed. The transaction code is introduced by a left paren:

```
nme(R           Wright, Christopher.
```

The following transaction codes are available for the "did" line:

(A - Add new record

(C - Change or correct existing record(s).

(D - Delete existing record(s).

(E - Erase all previous transaction lines for this "did" number.

(R - Replace an existing record.

(P - Print existing record(s).

(V - Insert a "verification" code into an existing record. (Not currently in use.)

The following transaction codes are available for tag lines:

(A - Add this field to a new or existing master file record.

(C - Change existing field(s), by replacing the first text string with the second text string.

(D - Delete existing field(s).

(I - Replace indicators in an existing field.

(R - Replace an existing field.

(S - Replace the subfield codes and/or indicators

in an existing field.

Record-level and field-level transaction codes are described in greater detail in a later section in this chapter.

Processing of transaction lines

Validity checking carried out by the Data Edit program (EPSDATA) involves only the did number or the use of extended features related to record order. All other examination of control information and field data takes place in the Update program (BPSUPDT).

All tags to be used must be defined in a separate file that is available to program BPSUPDT. Each tag appears along with its numeric equivalent and a code to indicate whether it is allowed to appear more than once in a master file record. Both the alphabetic and numeric equivalent for each tag are inserted into the record directory entry for each field, so subsequent references to the field (except in the Update program) may employ either alphabetic or numeric tag.

Records in the Tag/Option Dictionary contain 'TAG' in columns 1-3 and one or more eight-character alphabetic-numeric pairs beginning in column 9, without intervening blanks. E.g.,

```
TAG     TIL-245
TAG     SUP-680,NME-780
```

An asterisk immediately following the numeric tag (instead of the comma, if there was one) causes the Update program to prohibit the use of this tag with a site number greater than "1". E.g.,

```
TAG     TIL-245*
TAG     COL-300*PRI-370*SET-480,SER-490
```

The Tag/Option Dictionary may also contain specifications for the Update program options to be used. These specifications are expressed by the use of keywords. Frequently used keywords are:

SPACING1	Single space of proof sheet
UPCASE	Input medium does not accomodate lower case alphabetic characters -- letters not preceded by a shift character are to be translated to lower case.
LIMPRT	Only the fields that have been affected, and other fields having the same tag, should be printed on the Proof Sheet when only one type of field is changed in a particular record.

Example

```
.OPTIONS SPACING1,LIMPRT
```

Option keywords may also be entered through the PARM field of the Update EXEC statement, e.g.:

```
//UP EXEC UPDATE,PARM.U='UPCASE,SPACING1'
```

Typically, the options that are not expected to change for

the particular application are specified in the Tag/Option Dictionary, while options to be used in a particular run are specified in the EXEC statement.

Extended features

In order to facilitate the entry of data that is common to more than one master file record and to make possible systematic editing of groups of records in the master file, several methods are available to apply the same transaction to multiple master file records. These facilities are:

Multiple tag facility applies the same tag line or lines to two or more records which do not appear together on the master file. The multiple tag facility will accept up to 30 transaction lines at any one time and generate a copy of them for each did number that follows. Example:

```
did(C           Start multiple
acq             7504
nog/x           Published under contract.

did             B-125

did             B-166

did             B-99

did             End multiple
```

The acquisition date 7504 and the general note will be placed in records B-99, B-125, and B-166. This example also illustrates the use of a site number 'x' (described later) which causes the tag to be placed after any others that may be present.

The start-multiple and end-multiple commands may be written with a hyphen or one blank between words. Upper, lower, or mixed upper and lower case is accepted. A period may follow, e.g.:

```
did             Start-Multiple.
```

"did" lines within a multiple operation may not contain tag lines.

did Series facility may be described as a delimited iterative did range. Transaction lines that are to apply to every record whose did numbers are consecutive may make use of this facility. Example:

```
did(C           B-125 thru B-129
acq             7504
```

The acquisition date will be inserted in records B-125, B-126, B-127, B-128, and B-129. A did "series" may also appear within a multiple operaton (in which case it may not contain transaction lines of its own).

"To" or "through" may appear instead of "thru". Upper, lower, or mixed upper and lower case are accepted. At least one blank must precede and follow "thru", "to", or "through".

Based range facility applies to every record whose did number has a common beginning. Though the common beginning is usually the did basis, it may be shorter or longer. Examples:

```
did(P           B-125-*

did(D           B-128*

did(C           B-1300-*
nme/*(C         'United States' 'U.S.' all
```

This example will cause record B-125 and any record subordinate to B-125 to be printed. Record B-1280 and any records following it and preceding record B-1290 will be deleted. Record B-1300 and any records subordinate to it will have the words "United States" replaced by the abbreviation "U.S." in any name (nme) field where it is found.

When using this facility, care must be taken that there are no other transactions for records included in a based range that contains tag lines.

Transaction lines for a based range will be listed on the Update program Proof Sheet, with the "did" number of the first record found in the range.

An extension of the based range facility allows any section of the master file, or the entire file, to be accessed (subject to the restriction in the preceding paragraph). E.g.:

```
did(C           S-*
sup/*(D
                or

did(C           all
sup/*(D
```

will delete every sup tagged field from all records the first part of whose did numbers is S- (records beginning SA, SB, etc. are not included) or, in the second example, from every record in the master file (this might be done to create an abbreviated file for a special use).

When the same transaction lines are to be applied to many different records, through the use of a range specification, it is generally advisable to make use of one or more of the following modes of operation for the Update run. The keywords for these options may be included in the PARM field of the EXEC statement for the Update program.

NOUNMAT specifies that a the "UNMAT" (unmatched) and "NTFOUND" (not found) error messages should not be issued even though the field to be changed, deleted, or replaced does not exist in the record. Failure to use this option may result in many spurious error messages, since ordinarily many of the records in the range (or the file) will not contain the field or text to be changed.

NOPRT specifies that all printing of records and fields on the Update program Proof Sheet should be suppressed except in the case of error conditions. Since a change to many records will produce a great deal of printout, this option or the LIMPRT option will generally be advisable when using a range "did" line with a broad extent. If desired,

the results of the update operation may be verified when the file is printed subsequently, either in whole or in part.

LIMPRT specifies that printing of the master file record following a change should be limited to the fields that were changed during this run. Fields having the same tag will also be printed. If fields of more than one type are changed, or if an error is detected, the entire record will be printed. The LIMPRT option also avoids the printing of the original version of a field that is the object of a change field transaction.

SPACING1 specifies single spacing of the Update program Proof Sheet. It is frequently desirable even when the range facility is not being used.

The following is an example of an "Edit" run, in which erroneous punctuation is being removed from the title field of any record where it may have been:

```
//FIXTIL JOB UNC.PC.F282C,MCCLURE,T=10,
//  PRTY=0,FORMS=1111,COND=(9,LT)
//*PROCLIB=UNC.PC.F282C.MCCLURE.PROCLIB
//* remove (@) and ("") from titles (file edit run)
//  EXEC UPDATE,OUTBLK=9000,
//  PARM.U='NOUNMAT,LIMPRT,SPACING1'
//D.TRANS DD DUMMY,DCB=BLKSIZE=133
//D.INPUT DD *
did(C    All
til(C    /@// all
til(C    /""/"/ all
//SORT.SORTOUT DD DSN=TRANSORT,DISP=(,PASS),UNIT=TAPE,
//  RING=IN,VOL=SER=UT6003
//U.OLDCAT DD DSN=CATFILE2,DISP=OLD,UNIT=TAPE,
//  VOL=SER=UT6001,LABEL=2
//U.NEWCAT DD DSN=CATFILE2,DISP=(,KEEP),UNIT=TAPE,
//  VOL=SER=UT6002,LABEL=2
```

Titles of records that have been changed will be printed on the Proof Sheet (after the change). Because of the LIMPRT option, the other fields in the records will not be printed. Records that do not contain a title or whose titles do not contain the erroneous punctuation (@) or ("") will not be affected and will not cause the "UNMAT" or "NTFOUND" error messages because of the NOUNMAT option.

Delimited inclusive range includes all records between a beginning and ending did number. The delimited inclusive range differs from the did series in that the records in its range need not have consecutive did numbers. Since transaction lines are not generated for each did number (as they are in a did series), there must be no other transactions for records included in a delimited inclusive range that contains tag lines. Examples:

```
did(P          B-125:B-197

did(C          R-129:S-199-704-23

acq(C          '7405' 'May 1974'
```

The first did line will cause to be printed all records between B-125 and B-197, inclusive. Subordinate records

(e.g., B-130-12) will also be printed. The second transaction will change the numeric form of the acquisition date to a textual form.

The following facilities are frequently of use when using the above:

Non specific site number (/x) causes the field to be added after the last field (if any) for that tag.

All occurrences site number (/*) causes the tag line to apply to all occurrences of the tagged field.

Tag series code (n** or nn*) causes the transaction line to be applied to any tagged field whose numeric tag digits match the digits in the tag series code.

Command lines

Command lines are identified by a period (.) in column one. They are used to invoke special processing modes or assumptions in the Data Edit program (BPSDATA). Available command lines are:

TRANSCODE

specifies a record-level transaction code that is to be inserted into all "did" lines that do not already contain a transaction code. Example:

```
.TRANSCODE (D
```

An existing TRANSCODE specification may be replaced by a subsequent TRANSCODE statement. An existing specification can be cancelled with a line:

```
.TRANSCODE
```

AUTODID

specifies that lines beginning with what is presumably a valid "did" number (the Data Edit program checks for a hyphen in column 2 or 3) should have the letters "did" inserted into columns 1-3 and the "did" number itself moved to columns 15-25. An existing AUTODID specification may be cancelled with a line:

```
.NOAUTODID
```

AUTORANGE

specifies that "did" lines should be read as based ranges (ordinarily denoted by an asterisk, e.g.,

"B-234-*"). Individual exceptions to this assumption may be written:

```
did        B-123#
```

(the number sign must immediately follow the last digit of the "did" number). "Did" series lines are not permitted while AUTORANGE is in effect. To discontinue the AUTORANGE assumption, enter a line:

```
.NOAUTORANGE
```

Record level transaction codes

A record level transaction code, where present, is preceded by a left paren ["("], which immediately follows the letters "did" of the did line specifying the record or records to which the transaction code applies. The following codes are available:

code	meaning
(A	Add the following record to the master file. (When no transaction code is specified, "(A" is assumed.)
(C	Make the following corrections to an existing record.
(D	Delete an existing record (including all its tagged fields) from the master file. The deleted record will be printed on the proof sheet.
(E	Erase and ignore all transactions for this record that have been entered previously during this update run. If the master file already contains a record with this did number, and if additional transactions are not entered subsequently, the record will remain as before. The transactions that have been "erased" will be listed on the proof sheet.
(R	Replace an existing record with the one that follows. The replaced record will be printed on the proof sheet.
(P	Print an existing master file record on the proof sheet. If changes for this record are also entered, this request will be ignored, since the resulting record is printed after changes have been made in any case.
(V	Verify an existing record (i.e., place a "verified" indication in it. (Not currently used.)

Field level transaction codes

Field level transaction codes indicate the action to be taken with respect to a particular tagged field. A field level transaction code, where present, is preceded by a left paren ["("], which immediately follows the tag, tag suffix (indicators), or any control information that accompanies them. Only one transaction code may appear on a line. The following codes are available:

Code	Meaning
(A	Add this field to the master record. (When no transaction code is specified, "(A" is assumed.)
(C	Change an existing field. When this code is used, the "data" portion of the line consists of a comparison string and a replacement string, each delimited by single quotes or by a special delimiter (see the detailed instructions for the change transaction code which follow). The comparison string is located in an existing field in an existing or new record and then replaced by the contents of the replacement string. This feature enables correction of individual letters, words, or phrases without re-entering or disturbing accompanying correct text. The original field will be printed on the proof sheet.
(D	Delete an existing field from an existing or new record. The deleted field will be printed on the proof sheet.
(I	Replace the indicators (tag suffix) of an existing field in an existing or new record. The original version of the field will be printed on the proof sheet. With this transaction code, both indicators should be specified, even if only one is being changed. With the exception of the tag, tag sequence (site) number if required, and tag suffix (indicators), no other information should accompany this transaction code.
(R	Replace an existing field in an existing or new record, including subfield codes, tag suffix (indicators), and field data. The replaced field will be printed on the proof sheet.
(S	Replace the subfield codes of an existing field in an existing or new record. Data may not accompany this transaction. The number of subfield codes that accompany this transaction must equal the number of subfields in the existing field. If it is desired to replace the tag suffix (indicators) as well, they may be included with this transaction. The original version of the field will be printed on the proof sheet.

The Change Field Transaction

The change field transaction (field-level transaction code "C") is designed to permit changes and corrections to variable fields of a MARC record without the necessity to resubmit the rest of the field data. Examples of change transactions are:

1.	Present status of field:	til#ac	"The effects of demegraphic factors," by Barbara Janowitz.
	Change transaction:	til(C	'meg' 'mog'
	Corrected status of field:	til#ac	"The effects of demographic factors," by Barbara Janowitz.
2.	Present status of field:	jsn	Demography,#10(4),#Nov. 1973, #p. 507.
	Change transaction:	jsn(C	'. ' '-515.'
	Corrected field:	jsn	Demography,#10(4),#Nov. 1973, #p. 507-515.
3.	Present status of field:	nme	anowitz, Barbara.
	Change transaction:	nme(C	'' 'J'
	Corrected field:	nme	Janowitz, Barbara.

(Note: Underlining is for purposes of these examples.)

The change transaction is modelled after the change subcommand of the QED command of TUCC TSO (described in TUCC TSO Reference Manual, document LSR-281-0). The change transaction consists of, in addition to the tag, transaction code "(C" and subfield codes as described below:

a comparison string, which defines the characters to be replaced; and

a replacement string, which defines the characters to be substituted.

The Update program scans the field data until it locates the contents of

the comparison string and then replaces those characters with the contents of the replacement string, expanding or compressing the field as required. The field that is changed through a change transaction may have been added to the record earlier during the run and may itself be the product of an earlier change transaction or transactions.

Both comparison and replacement strings must be transmitted exactly as they appear and are to appear in the field data. That is, the comparison string must match exactly the data to be replaced, including punctuation, blank spaces, subfield codes, and capitalization. If input is being entered in upper case mode, with the inclusion of shift characters to signal upper case letters, then conversion to upper and lower case must be anticipated (e.g., to replace the text "Puerto Rico" the upper case mode comparison string would be '$PUERTO $RICO'). Similarly, the replacement string is entered with appropriate shift characters if upper case mode is being used.

The comparison and replacement strings are delimited by the use of single quotes (apostrophes). At least one blank space must separate the beginning quote of the replacement string from the ending quote of the comparison string. If it is necessary to express a single quote or apostrophe as part of the comparison string or the replacement string, the quote is typed twice, e.g.:

"doesn't need" must be typed: 'doesn''t need'

A null string is specified as two single quotes, with no intervening blank: ''
If specified for the comparison string, the null string causes the replacement string to be inserted before the first character of the field data. If specified for the replacement string, the null string causes the deletion of the contents of the comparison string from the field data.

Examples of the use of the null string:

4.	Present status of field:	nme	Janowitz, Barbarara.
	Change transaction:	nme(C	'ra' ''
	Corrected field:	nme	Janowitz, Barbara.
5.	Present status of field:	nme	Barbara.
	Change transaction:	nme(C	'' 'Janowitz, '
	Corrected field:	nme	Janowitz, Barbara.

For purposes of the comparison scan, an existing field is assumed to have an unlimited number of trailing blanks, so to add data to the end of a field it is possible to specify a comparison string of several blanks (enclosed between single quotes). Regardless of the presence of leading or trailing blanks in the replacement field and the location in which it is inserted, a field following a change transaction will not contain leading or trailing blanks. For an example of the use of blanks in the comparison string, see example 2, above. An example of the removal of leading blanks from the replacement string is:

6.	Present status of field:	sup	Follow-up.
	Change transaction:	sup(C	'F' ' Lack of F'
	Corrected field:	sup	Lack of follow-up.

An alternate way of specifying the comparison and replacement strings is available through the use of a special delimiter. This method is described later in this section.

Note: Since the scan for the contents of the comparison string proceeds from left to right, care should be taken that the comparison string contains sufficient characters to avoid replacing data to the left of the desired correction.

Change Field Transaction -- continued, p. 4.

Changes to the number of subfields

It is possible, with the use of the change transaction, to add, insert, and remove subfields from an existing or new tagged field. Either or both the comparison string and the replacement string may contain subfield delimiters (#) to accomplish this. When the number of subfield delimiters in the comparison and replacement strings is different, then the subfield codes for the field should be respecified in the change transaction. Examples, involving successive changes to a field:

7. Present status of field: red#abd Interviews,#Thailanf,#damilies
   Change transaction: red(C 'f,#d' 'd,#f'
   Corrected field: red#abd Interviews,#Thailand,#families
   Change transaction: red(C#abde ' ' ',#1146.'
   Corrected field: red#abde Interviews,#Thailand,#families,#1146.
   Change transaction: red(C#ade 's,#' 'ing in '
   Corrected field: red#ade Interviewing in Thailand,#families,#1146

If a change transaction results in an increase or decrease in the number of subfields yet subfield codes were not specified on the change transaction, the resulting field will have subfield codes "abcde..." as required. If subfield codes did accompany the change transaction, they are inserted into the resulting tagged field whether or not there was a change to the number of subfields. If too many or too few subfield codes were specified, the number is adjusted to match the field and a warning indication is given.

Use of ellipis points for replacing lengthy character strings

If the length of the text to be replaced (the comparison string) is significant, the user may wish to take advantage of the following abbreviated form for specification of the comparison string contents:

8.	Present status of field:	til	Economic development in tropical Africa.
	Change transaction:	til(C	'... d' 'D'
	Corrected field:	til	Development in tropical Africa.
9.	Present status of field:	til	Economic development in tropical Africa.
	Change transaction:	til	'trop...' 'the tropics.'
	Corrected field:	til	Economic development in the tropics.
10.	Present status of field	til	Economic development in tropical Africa.
	Change transaction:	til	'ic d...cal' 'y and development in'
	Corrected field:	til	Economy and development in Africa.

In the abbreviated form, the contents of the comparison string is specified with the use of ellipsis points together with the text that begins the comparison string, the text that ends the comparison string, or both. The comparison string will include the text specified and the rest of the field (in the case of trailing ellipsis points), the contents of the field up to and including the text specified (in the case of leading ellipsis points), or the text between and including the characters on either side of the ellipsis points (as in example 10). Ellipsis points may not appear twice in the comparison string specification, and if they are used they will be treated as above (i.e., it is not possible to replace text that consists of or includes "..." using the change field transaction, since the "..." would be interpreted as invoking the abbreviated form of the comparison string.)

In the execution of the change field transaction scan, the test for ellipsis points takes precedence over the comparison to the field being changed. Thus if the field to be changed contains a period (.) in the text being replaced, it is not necessary to specify four dots to include the period in the comparison. Moreover, any additional consecutive dots beyond the required three will be ignored. Thus, it is not possible to gain additional clarification of the comparison field by attempting to include a period as the character immediately preceding or following the ellipsis points. Examples:

11. Present status of field: til Report of the U.N. Health Seminar.
    Change transaction: til(C 'N...' 'N. Population Conference.'
    Corrected field: til Report of the U.N. Population Conference
    ("..." is sufficient even though the text to be replaced contains a period)

12. Present status of field: til Report of the U.N. Health Seminar.
    Change transaction: til(C 'N....' 'N. Population Conference.'
    Corrected field: til Report of the U.N. Population Conference
    ("...." is also correct; "....." would also achieve the same result)

13. Present status of field: til USN. assessment of the U.N. Seminar.
    Change transaction: til(C 'N....' 'S. Defense Policy.' (incorrect)
    Incorrect result: til USS. Defense Policy.
    (intended result was: USN assessment of the U.S. Defense Policy.
    In this example, note that the additional period preceding the ellipsis points did not provide any additional clarification of the comparison text -- it did not cause differentiation between "N " and "N.")

Change Field Transaction -- continued, p. 7.

Special delimiter form for specifying comparison and replacement strings

Both comparison and replacement strings may be specified in the manner described and shown above (known as "quoted string notation"), or in an alternate manner, the "special delimiter notation". Examples of special delimiter notation are:

14.	Change transaction:	til(C	/comparison string/replacement string/
15.	Change transaction:	til(C	*comparison string*replacement string*
16.	Change transaction:	til(C	&IUD&Interuterine device&
17.	Change transaction:	til(C	/extraneous//
	(null replacement string)		
18.	Change transaction:	til(C	//additional/
	(null comparison string)		
19.	Change transaction:	til(C	/events.../events./
	(use of ellipsis points with special delimiter form)		

Any non-blank character may be used as the special delimiter, except for the number sign (#) and the apostrophe ('). It is recommended, however, that letters of the alphabet, numerals, and common punctuation (in particular, the period) not be used, to avoid confusion. It should also be noted that since special delimiter notation is detected by the absence of an apostrophe as the first character of the change transaction, if quoted string notation was intended and the initial apostrophe was omitted by accident, the program will attempt to interpret the change transaction as employing special delimiter notation with the first character of the comparison string as the special delimiter. The character chosen as the special delimiter may not appear in the text of the comparison string or the replacement string.

Change Field Transaction -- continued, p. 8.

The special delimiter form may be used in all situations described above as appropriate for quoted string notation. In addition, both quoted string and special delimiter forms may be used in the same update run. Any individual change transaction, however, must employ either one form or the other and may not attempt to combine the forms.

Which form to use is a matter of personal preference. The special delimiter form is particularly advantageous where either the comparison string or the replacement string contains apostrophes. Example:

20.	Present status of field:	til	A supplement to the volume ''Planning Policy!!
	Change transaction:	til(C	/''/'/
	Corrected field:	til	A supplement to the volume 'Planning Policy.'

(To achieve the same result with quoted string notation, the change transaction would be: til(C '''''' '''')

Changing all occurrences of a text string

It is possible to change all occurrences of a text string within a field by using the keyword "all" following the replacement string:

21.	Present status of field	til	U.S.A.I.D. Annual Report.
	Change transaction	til(C	'.' '' all
	Corrected field:	til	USAID Annual Report_
	[illegible]	[illegible]	[illegible]
	[illegible]	til	[illegible]
	[illegible]	[illegible]	[illegible]

Either quoted string or special delimiter form may be used. The key word "all" may be in upper or lower case, or mixed upper and lower case. There may be one or more blanks (or none) preceding the keyword "all".

The keyword "all" will be translated to an asterisk (*) and the transaction will be printed with an asterisk instead of "all".

It should be noted that the scan for each additional occurrence of the comparison field starts from the beginning of the data field. Therefore, the replacement string must not contain nor cause to be constructed the contents of the comparison string or the replacement string will itself be changed repeatedly. To prevent an infinite loop, a restriction of 16 has been placed on the number of occurrences that will be changed per field. Examples of incorrect uses of the keyword "all" are:

22.	Present status of field	til	Fertility,parity,and sex roles.
	Change transaction	til(C	',' ', ' all
	Incorrect result	til	Fertility,_____parity,and sex roles
23.	Present status of field	til	A.A.S.S. Inquiry
	Change transaction	til(C	'A.S.' '' all
	Incorrect result	til	Inquiry

MASTER FILE UPDATING SEQUENCE

1. Add new acquisitions to the In-Process file.
2. Correct new records on the In-Process file.
3. Transfer old records to be changed or inspected from the Master file to the In-Process file, using destructive select feature of the Update program, followed by merge (PL/1, listing/eliminating duplicates--old record has precedence).
4. Correct new and old records on the In-Process file.
5. Do Inter-Record Information Transfer.
   a. PL/1 production of select transaction for each child without parent on the In-Process file (each parent only once).
   b. Non-destructive select of parents from Master file.
   c. Merge of selected records (ghost parents) from Master file.
   d. IRIT, producing monthly acquisition/correction master list, followed by deletion of ghost parent records.
   e. Split In-Process file based on (a given) acq and cod into Fully-Corrected and In-Process records.
   f. Merge Fully-Corrected records with Master file (Master file has precedence in case of duplicates).
6. Correct current records (on In-Process file) based on monthly master list and indexes.
7. Go to 1.

May 10, 1976