BUREAU OF THE CENSUS
STATISTICAL RESEARCH DIVISION REPORT SERIES

SRD Research Report Number: CENSUS/SRD/RR-86/19

USER'S GUIDE FOR THE GENERALIZED RECORD LINKAGE PROGRAM GENERATOR (GENLINK)
SRD PROGRAM GENERATOR SYSTEM USER'S GUIDE: PART III

by

William P. LaPlant, Jr.
Statistical Research Division
U.S. Bureau of the Census
Room 3548, F.O.B. #3
Washington, D.C. 20233

(301) 763-1496

Recommended by:	Matthew A. Jaro
Report Completed:	September 1, 1986
Report Issued:	September 1, 1986

Table of Contents

Section 1. Introduction to GENLINK.

GENLINK stands for GENeralized record LINKer. GENLINK is one of a series of Bureau of the Census, Statistical Research Division, COBOL program code generators written in UNIMAC. UNIMAC is a high-level language macro processor language which itself is implemented in COBOL-74. GENLINK can be used to generate a program which creates a "pointer" file containing information that can be used to "link" corresponding records of two files based on different kinds of comparisons of user selected data fields contained on both files. This pointer file can be used together with the original data files to generate reports and new data files containing new "linked" records and unmatched and marginally matched data records or reports. Part V of this series describes a program generator which generates a program that can be used as by itself or as the basis of a more sophistocated record linkage "extractor".

This document describes how to use GENLINK.

1.1. An Example.

The following is a GENLINK program:

```
LINK
* PES+ACF TO 85-CENSUS TEST MATCH 2
INPUT
*DUMP
FILENM=PESACF
FILESPC="PESA.IN1"
FIELD=CBNABLOCK,1,9
FIELD=SOUNDEX-STREETN,196,4
FIELD=FIRST-NAME,18,18
FIELD=MIDDLE-INIT,36,1
FIELD=LAST-NAME,37,24
FIELD=RELATION,65,1
FIELD=SEX-MARITAL,178,2
FIELD=NEW-BIRTHDATE,180,6
FIELD=RACE-HISPANIC,176,2
FIELD=ACF-STREET-NAME,106,20
FIELD=ACF-HOUSE-NUM,87,7
FIELD=ACF-UNIT,134,15
FILENM=CENSUS
FILESPC="CENACF.SRT"
FIELD=CBNABLOCK,8,9
FIELD=SOUNDEX-STREETN,179,4
FIELD=LAST-NAME,19,24
FIELD=FIRST-NAME,43,18
FIELD=MIDDLE-INIT,61,1
FIELD=NEW-RELATION,162,1
FIELD=SEX-MARITAL,160,2
FIELD=NEW-BIRTHDATE,163,6
FIELD=RACE-HISPANIC,158,2
FIELD=ACF-ST,101,20
FIELD=ACF-UHN,129,7
FIELD=ACF-UAPT,137,15
```

```
PROCESS
PATTERN=BOTH
EQUATE=CBNA-BLOCK,CBNABLOCK,CBNABLOCK
VAR=CBNA-BLOCK,N,.9,.9
EQUATE=SOUNDEX,SOUNDEX-STREETN,SOUNDEX-STREETN
VAR=SOUNDEX,C,.9,.0233
ADJUST=SOUNDEX,,," "
EQUATE=LASTNAME,LAST-NAME,LAST-NAME
VAR=LASTNAME,U,.85,.0036
ADJUST=LASTNAME,700.00,," "
EQUATE=FIRSTNAME,FIRST-NAME,FIRST-NAME
VAR=FIRSTNAME,U,.9816,.0877
ADJUST=FIRSTNAME,700.00,," "
EQUATE=MIDINIT,MIDDLE-INIT,MIDDLE-INIT
VAR=MIDINIT,C,.3469,.0272
ADJUST=MIDINIT,,," "
EQUATE=RELATION,RELATION,NEW-RELATION
VAR=RELATION,N,.3881,.2051
ADJUST=RELATION,,,"0"
EQUATE=SEX-MAR,SEX-MARITAL,SEX-MARITAL
VAR=SEX-MAR,N,.8255,.2114
ADJUST=SEX-MAR,,,"0"
EQUATE=BIRTHDATE,NEW-BIRTHDATE,NEW-BIRTHDATE
VAR=BIRTHDATE,P,.9379,.0399
ADJUST=BIRTHDATE,000006,,"0"
EQUATE=RACE-HISPA,RACE-HISPANIC,RACE-HISPANIC
VAR=RACE-HISPA,N,.9043,.6748
EQUATE=HOUSENUM,ACF-HOUSE-NUM,ACF-UHN
VAR=HOUSENUM,C,.99,.0149
EQUATE=APTNUM,ACF-UNIT,ACF-UAPT
VAR=APTNUM,C,.3525,.2628
ORDER=CBNA-BLOCK,A,SOUNDEX,A,HOUSENUM,A,APTNUM,A
LAMBDA=.001
MU=.0005
OUTPUT
FILESPC="PESCEN2.OUT"
END
```

1.2. An IBM-PC GENLINK Session.

Assuming that the program in the above paragraph has been stored on the IBM-PC file "SAMPLE.GEN", the following session on an IBM-PC would cause the generation of the COBOL data standardization program shown in the next paragraph as specified in this GENLINK user program. Everything actually part of the session is in upper-case letters. What you would enter is underlined. All explanations are surrounded by curly brackets ({}).

{WARNING -- BEFORE you start a new UNIMAC session ALWAYS MAKE SURE YOU HAVE USED, COPIED, or RENAMED the UNIMAC generator output file, MACOUT.DAT. Otherwize you will lose the results of the prior session.}

{The Sample Session Begins on the next line.}

```
C>UNIMAC
*** ENTER MACRO LIBRARY NAME ***
{At this point you must enter the name of the file on your system
that contains your copy of the SRD Program Generator System.}
PCLIB.DAT

THIS IS A TEST
        BUREAU OF THE CENSUS
   STATISTICAL RESEARCH DIVISION
   RECORD LINKAGE RESEARCH STAFF

AUTOMATIC PROGRAM GENERATION SYSTEM
DATE=05/01/86   TIME=10:00:00
PLEASE SELECT ONE OF THE FOLLOWING GENERATORS:

STANDARDIZER
MATCHER
UNDUPLICATOR
*** ENTER ACCEPT FILE NAME (CON: FOR CONSOLE) ***
{At this point the user may either indicate that he is going to
enter the entire program from the keyboard by typing "CON:" or
that he wants to generate a matcher program from a previously
developed text file by entering the DOS file specification of the
text file containing his user program:}
SAMPLE.GEN
```

```
RECORD LINKAGE MODULE

USER FILE          2 * PES+ACF TO 85-CENSUS TEST MATCH 2
USER FILE          3 INPUT
USER FILE          4 *DUMP
USER FILE          5 FILENM=PESACF
USER FILE          6 FILESPC="PESA.IN1"

WARNING - TARGET NOT SET: ASSUMED TO BE "IBM-PC" BASED ON PROVISION OF
FILESPC "PESA.IN1" FOR INPUT FILE A (PESACF).
```

{The above WARNING is a message caused by line 6 of the user program. If the assumptions reported in such a warning are correct, no further action is needed on the user's part. If the assumptions are incorrect, then the resulting code should be checked or the directive "TARGET=IBM-PC" should be put into the user program before line 6 (but after line 1).}

```
USER FILE          7 FIELD=CBNABLOCK,1,9
                          .
                          .
                          .
USER FILE         14 FIELD=NEW-BIRTHDATE,180,6
                          .
                          .
                          .
USER FILE         28 FIELD=NEW-BIRTHDATE,163,6
                          .
                          .
                          .
USER FILE         55 EQUATE=BIRTHDATE,NEW-BIRTHDATE,NEW-BIRTHDATE
USER FILE         56 VAR=BIRTHDATE,P,.9379,.0399
USER FILE         57 ADJUST=BIRTHDATE,000006,,"0"
                          .
                          .
                          .
USER FILE         66 MU=.0005
USER FILE         67 OUTPUT
USER FILE         68 FILESPC="PESCEN2.OUT"
USER FILE         69 END
```

{Any errors detectable while evaluating a given line would be generated immediately following the line which caused the error. Those errors which cannot be detected until the entire user program is available will follow. The following is output by the program generator as a result of the above input. The warnings result from the "P" variable comparison <TYPE> assigned to the variable "BIRTHDATE" (LINE 56, second parameter). There are two such warnings because the NEW-BIRTHDATE fields of each of the files being compared are each defined independently. In order to avoid the warning, the user would code a COBOL PICTURE string, "9(6)", as the 4th parameter (<PIC>), in both lines 14 and 28.}

```
WARNING - FIELDs associated with BIRTHDATE must be numeric for
the calculations associated with comparison type "P".  Necessary
adjustments to NEW-BIRTHDATE in File A have been made.  Check the
generated code.
WARNING - FIELDs associated with BIRTHDATE must be numeric for
the calculations associated with comparison type "P".  Necessary
adjustments to NEW-BIRTHDATE in File B have been made.  Check the
generated code.
{The appropriate PICTURE clause will be generated as a result of
this warning, if the fields in the input files do contain just
numbers.}

{Default parameter settings are reported next if necessary:}
WARNING - MAXBLK (the maximum number of records that can be read
from either File A or File B that will satisfy the blocking
criteria) was not set --> 100 assumed.
WARNING - MAXNEGW (the "maximum negative weight" used as a dummy
weight to make the composite weight matrix square for the match
assignment algorithm) was not set --> -999 assumed.
{The remainder of the messages are used by the Record Linkage
Staff to determine the progress of COBOL code generation in this
preliminary version of the program generator:}
1
A
THE PASSED ARGMENT IS A
1
B
THE PASSED ARGMENT IS B
GENERATE WORKING STORAGE
TOTAL MATRIX ELEMENTS APPROXIMATED - 0,  10000
LINKER PROCEDURE DIVISION SETUP
GENERATE FIELD COMPARISONS
*** END OF PROCESS


RUN STATS  MET210                  69 RECORDS READ FROM MACACPT (USER) FI
{This is the number of User Program Records processed by the
Program Generator.  This number should match the number of lines
in the user program.}
RUN STATS  MET108                   0 RECORDS READ FROM MACIN FILE
RUN STATS  MET204              -22002 TOTAL INPUT RECORDS PROCESSED
RUN STATS  MET109                2723 RECORDS WRITTEN TO MACOUT FILE
{This is the number of lines of COBOL code generated.  The
"MACOUT FILE" is UNIMAC's standard output file.  On the IBM-PC,
the file specification for this file is "MACOUT.DAT".  MACOUT.DAT
should be copied or renamed to a file with the DOS extension
".COB" which causes the REALIA COBOL compiler to treat the
generated program as a standard format COBOL program.}
```

```
RUN STATS  MET110                        0 ERRORS ENCOUNTERED
{WARNINGs don't count in this count!}
RUN STATS  MET216                     4000 SYMBOL TABLE CELLS ALLOCATED
RUN STATS  MET201                       32 MACIO DATA BUFFERS ALLOCATED
RUN STATS  MET202                       32 MACIO DIRECTORY BUFFERS ALLOCATED
RUN STATS  MET105                      344 MACIO PHYSICAL IO OPERATIONS

C>
{At this point the SRD Program Generator System run is complete
and MACOUT.DAT contains a complete COBOL program which can be
compiled, linked and run against your data files.  Remember to
COPY, RENAME, or use MACOUT.DAT before you use UNIMAC again!}
```

1.3 The COBOL Program Generated.

GENLINK generates approximately 2500 lines of COBOL code in a single run-unit. Based on the GENLINK User Program entered above the following is an extract of the COBOL program is generated.

```
*$COMP-4,QUOTE
*  LINKGEN
 IDENTIFICATION DIVISION.
 PROGRAM-ID. GENLINK.
 AUTHOR. W-LAPLANT VIA UNIMAC.
 DATE-WRITTEN. 10/29/85. 19:58:54.
 DATE-COMPILED.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER. IBM-PC.
 OBJECT-COMPUTER. IBM-PC.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
      SELECT PESACF-FILE-A
         ASSIGN TO "PESA.IN1".
      SELECT CENSUS-FILE-B
         ASSIGN TO "CENACF.SRT".
      SELECT OUT-FILE ASSIGN TO "PESCEN2.OUT".
      SELECT COUNTER-FILE ASSIGN TO "COUNT.DAT".
 DATA DIVISION.
 FILE SECTION.
 FD  PESACF-FILE-A
     LABEL RECORDS STANDARD
     RECORD CONTAINS 199 CHARACTERS.
 01  A-LINK-RECORD.
            ...


 FD  CENSUS-FILE-B
     LABEL RECORDS STANDARD
     RECORD CONTAINS 182 CHARACTERS.
 01  B-LINK-RECORD.
            ...

 FD  OUT-FILE
     LABEL RECORDS STANDARD
     RECORD CONTAINS 34 CHARACTERS.
 01  INITIAL-OUTPUT-RECORD PIC X(34).
 FD  COUNTER-FILE
      LABEL RECORDS STANDARD
      RECORD CONTAINS 31 CHARACTERS.
 01  COUNTER-RECORD PIC X(31).
```

```
WORKING-STORAGE SECTION.
01  OUTPUT-RECORD-STRUCTURE.
     02 OUTPUT-RECORD-AREA PIC X(34).
     02  INITIAL-OUTPUT-RECORD REDEFINES OUTPUT-RECORD-AREA.
         05 OUTPUT-RECORD-TYPE PIC XX.
         05 FILLER PIC X.
         05 ZERO-A PIC -9(5).
         05 FILLER PIC X.
         05 ZERO-B PIC -9(5).
         05 FILLER PIC X.
         05 ZERO-W PIC -9999.9999.
         05 FILLER PIC X.
         05 ZERO-P PIC 9(6).
     02  BLOCK-COUNTER-RECORD REDEFINES OUTPUT-RECORD-AREA.
             . . .

     02  SKIP-A-RECS-RECORD REDEFINES OUTPUT-RECORD-AREA.
             . . .

     02  SKIP-B-RECS-RECORD REDEFINES OUTPUT-RECORD-AREA.
             . . .

     02  MATCHED-PAIR-RECORD REDEFINES OUTPUT-RECORD-AREA.
         05 MATCHED-PAIR-TYPE-MP PIC XX.
         05 FILLER PIC X.
         05 MATCH-A-REC-INDEX PIC -9(5).
         05 FILLER PIC X.
         05 MATCH-B-REC-INDEX PIC -9(5).
         05 FILLER PIC X.
         05 MATCHED-PAIR-WEIGHT PIC -9999.9999.
         05 FILLER PIC X.
         05 MATCH-REC-PATTERN PIC 9(6).
     02  DUPLICATE-A-RECORD REDEFINES OUTPUT-RECORD-AREA.
             . . .

     02  DUPLICATE-B-RECORD REDEFINES OUTPUT-RECORD-AREA.
             . . .

     02  UNMATCHED-A-RECORD REDEFINES OUTPUT-RECORD-AREA.
             . . .

     02  UNMATCHED-B-RECORD REDEFINES OUTPUT-RECORD-AREA.
             . . .

     02  CLERICAL-REVIEW-RECORD-PAIR REDEFINES
           OUTPUT-RECORD-AREA.
             . . .
```

```
       01  COUNTER-RECORD-STRUCTURE.
            02 COUNTER-RECORD-AREA.
                05 COUNTER-INDEX-VALUE PIC 9(5).
                05 FILLER PIC X.
                05 COUNTER-VALUE PIC 9(6).
                05 FILLER PIC X(19).
            02  COMPARISON-CNTER-VAR-NAME-STR REDEFINES
                  COUNTER-RECORD-AREA.
                05 COUNTER-VARIABLE-NAME-VALUE PIC X(15).
                05 FILLER PIC X.
                05 COUNT-PROBABILITY-MATCH PIC -9.9999.
                05 FILLER PIC X.
                05 COUNT-PROBABILITY-UNMATCH PIC -9.9999.
       01  BLOCK-COUNTER-RECORD.
            05 BLOCK-COUNTER-TYPE-BP PIC XX.
            05 FILLER PIC X.
            05 BLOCK-A-COUNT PIC -(5)9.
            05 FILLER PIC X.
            05 BLOCK-B-COUNT PIC -(5)9.
            05 FILLER PIC X.
            05 FILLER PIC ----9.9999.
            05 FILLER PIC X(7).
       01  OUTPUT-FILE-TYPES.
      *   RECORD TYPE CONSTANTS FOR OUT-FILE RECORDS.
            05 BLOCK-COUNTER-TYPE PIC XX VALUE "BP".
            05 SKIP-A-RECS-TYPE PIC XX VALUE "SA".
            05 SKIP-B-RECS-TYPE PIC XX VALUE "SB".
            05 SKIP-BLOCK-PTR-MAX-EXCEED PIC XX VALUE "SP".
            05 MATCHED-PAIR-TYPE PIC XX VALUE "MP".
            05 DUPLICATE-A-TYPE PIC XX VALUE "DA".
            05 DUPLICATE-B-TYPE PIC XX VALUE "DB".
            05 UNMATCHED-A-TYPE PIC XX VALUE "UA".
            05 UNMATCHED-B-TYPE PIC XX VALUE "UB".
            05 CLERICAL-REVIEW-PAIR-TYPE PIC XX VALUE "CP".
                ...

      *    WSLNKGEN
       01  WHOLE-B-BLOCK.
            05 B-BLOCK
               OCCURS 100 TIMES.
               10 LAST-NAME            PIC X(24).
               10 FIRST-NAME           PIC X(18).
               10 MIDDLE-INIT          PIC X(1).
               10 NEW-RELATION         PIC X(1).
               10 SEX-MARITAL          PIC X(2).
               10 NEW-BIRTHDATE        PIC 9(6) COMP.
               10 RACE-HISPANIC        PIC X(2).
       01  LAST-B-KEY.
            05 CBNABLOCK               PIC X(9).
            05 SOUNDEX-STREETN         PIC X(4).
            05 ACF-UHN                 PIC X(7).
            05 ACF-UAPT                PIC X(15).
```

```
01  B-KEY.
          ...

01  B-DATA.
          ...

01  LAST-B-DATA.
          ...

01  A-KEY.
          ...

01  A-DATA.
          ...

01  VARIOUS-BLOCKER-VALUES.
          ...

01  VARIABLE-DESCRIPTORS.
     05 INDIVIDUAL-VARIABLE-DESCRIPTOR.
        10 NAME-COMPARE-LASTNAME  PIC X(10) VALUE "LASTNAME".
        10 COMP-TYPE-LASTNAME  PIC X VALUE "U".
        10 PROB-MATCH-LASTNAME  PIC S9(4)V9999 COMP VALUE .85.
        10 OLD-PROB-MATCH-LASTNAME  PIC S9(4)V9999 COMP VALUE
    .85.
        10 PROB-UNMATCH-LASTNAME  PIC S9(4)V9999 COMP VALUE
    .0036.
        10 OLD-PROB-UNMCH-LASTNAME  PIC S9(4)V9999 COMP VALUE
    .0036.
        10 WEIGHT-MATCH-LASTNAME  PIC S9(4)V9999 COMP.
        10 OLD-WGHT-MATCH-LASTNAME  PIC S9(4)V9999 COMP.
        10 WEIGHT-UNMATCH-LASTNAME  PIC S9(4)V9999 COMP.
        10 OLD-WGHT-UNMCH-LASTNAME  PIC S9(4)V9999 COMP.
        10 PRORATN-FACTOR-LASTNAME  PIC S9(4)V9999 COMP VALUE
    700.00.
        10 MAX-VAR-PROP-LASTNAME  PIC S9(4)V9999 COMP.
        10 FILE-A-LAST-NAME-MAX-LEN  PIC S9(4) COMP VALUE 24.
        10 FILE-B-LAST-NAME-MAX-LEN  PIC S9(4) COMP VALUE 24.
        10 NAME-COMPARE-FIRSTNAME  PIC X(10) VALUE "FIRSTNAME".
          ...
          05 VARIABLE-DESCRIPTOR-SET REDEFINES
             INDIVIDUAL-VARIABLE-DESCRIPTOR.
             10 VARIABLE-BY-TYPE-INDEX OCCURS 11 TIMES.
                15 VARIABLE-NAME   PIC X(10).
                15 VARIABLE-COMPARISON-TYPE  PIC X.
                15 PROBABILITY-MATCHED  PIC S9(4)V9999 COMP.
                15 OLD-PROB-MATCHED  PIC S9(4)V9999 COMP.
                15 PROBABILITY-UNMATCHED  PIC S9(4)V9999 COMP.
                15 OLD-PROB-UNMATCHED  PIC S9(4)V9999 COMP.
                15 WEIGHT-MATCHED  PIC S9(4)V9999 COMP.
                15 OLD-WEIGHT-MATCHED  PIC S9(4)V9999 COMP.
                15 WEIGHT-UNMATCHED  PIC S9(4)V9999 COMP.
                15 OLD-WEIGHT-UNMATCHED  PIC S9(4)V9999 COMP.
                15 PRORATION-FACTOR  PIC S9(4)V9999 COMP.
                15 MAXIMUM-VARIENCE-PROPORTION  PIC S9(4)V9999 COMP.
                15 FILE-A-FIELD-MAX-LENGTH  PIC S9(4) COMP.
                15 FILE-B-FIELD-MAX-LENGTH  PIC S9(4) COMP.
*   THE FOLLOWING ARE USED IN THE UNCERTAINTY-COMPARATOR.
          05 MATRICIES-UNCERT REDEFINES
             INDIVIDUAL-VARIABLE-DESCRIPTOR.
                ...

 01   C-COST-TABLE.
      02   C-COST-MATRIX                  OCCURS 10000 TIMES.
          05   C-ELEMENT                  PIC S9(4)V9999 COMP.
      02   COMPOSITE-WEIGHT-REDEF REDEFINES C-COST-MATRIX
           OCCURS 10000 TIMES.
          05   COMPOSITE-WEIGHT           PIC S9(4)V9999 COMP.
*
 01   COLUMN-TABLE.
      02   COLUMN-VECTOR                  OCCURS 1 TO 100 TIMES
                                          DEPENDING ON N-MATRIX-ORDER.
          05   COLUMN-ELEMENT             PIC S9999 COMP.
*
 01   N-MATRIX-ORDER                      PIC 99999 COMP.
 01   M-MAX-ELEMENT                       PIC S9(4)V9999 COMP.
 01   Z-OPTIMUM-ASSIGN                    PIC S9(4)V9999 COMP.
 01   COMPARE-PATTERN-MATRIX.
       05 PATTERN-COLUMN             OCCURS 100 TIMES.
          10 PATTERN-ROW             OCCURS 100 TIMES.
             15 COMPARE-PATTERN      PIC 999999 COMP.
*  FSUNLKWS
*PROGRAM-ID. FSUNTER.
 01   F-S-PARMS.
      03 LAMBDA       PIC S9(5)V9(8) COMP VALUE .001.
      03 OLD-LAMBDA   PIC S9(5)V9(8) COMP VALUE .001.
      03 MU           PIC S9(5)V9(8) COMP VALUE .0005.
      03 OLD-MU       PIC S9(5)V9(8) COMP VALUE .0005.
      03 LOW-VAL      PIC S9(5)V9(8) COMP.
      03 HIGH-VAL     PIC S9(5)V9(8) COMP.
            ...
```

```
*
*  CMPFLKWS
            ...

*  UNCRTLWS
            ...

*     LINKPDIV
 PROCEDURE DIVISION.
 MAIN-OPEN SECTION.
 SET-UP.
      OPEN INPUT PESACF-FILE-A.
      OPEN INPUT CENSUS-FILE-B.
      OPEN OUTPUT OUT-FILE.
      PERFORM INITIALIZE-OUTPUT-RECORD.
*              ^-- GENERATED IN LKOUTPUT
      MOVE ZERO TO NUMBER-OF-COMPOSITE-COMPARES.
      PERFORM INITIALIZE-COMPARE-COUNTERS
         VARYING I FROM 1 BY 1
         UNTIL I GREATER 128.
*            GENERATED IN CMPFLDS----^
                     ...

 INITIALIZE-WEIGHTS.
      MOVE 1 TO A-COUNTER, B-COUNTER.
      MOVE LOW-VALUES TO LAST-B-KEY.
      PERFORM A-READ.
      PERFORM B-READ.
 READ-B.
      MOVE ZERO TO B-COUNT.
      PERFORM READ-B-BLOCK.
      IF B-END-SWITCH = 1
         GO TO TEST-A-B.
      PERFORM READ-B-BLOCK
         UNTIL LAST-B-KEY NOT = B-KEY.
      PERFORM SET-B-BLOCK.
      GO TO TEST-A-B.
 READ-A.
            ...

*    CMPFLDS
 MATRIX-BUILD SECTION.
 MB.
* Calculate COMPOSITE-WEIGHT (*,A) by summing the BLOCKING-PENALTY
*   and each weight generated by a comparison of pairs of corres-
*   ponding fields between the current A record and each B record
*   in the B-BLOCK.
            ...
```

On the IBM-PC, the generated program will be written on the file "MACOUT.DAT". Change or copy this file immediately because it will be overwritten by UNIMAC the next time it is used on your computer.

A more complete copy of this generated program is used in Section 4 of the Maintenance Manual for GENLINK with an explanation of how various portions are generated from the GENLINK user program and which madules of the GENLINK UNIMAC program generate which portions of the program.

Section 2. Sample GENLINK Program

In the following sample user program, only the information the user enters is shown, not the response of the GENLINK program generator. How the SRD UNIMAC program generation system is accessed and how the user's computer system accesses data are not discussed here. However, a complete sample interactive session for the IBM-PC and a complete batch session for the Sperry UNIVAC is shown in Appendix B. The program shown here has generated a successfully compiled record linkage or file matcher program that was used in a real case.

2.1. The Sample Program.

Each statement of this sample program is preceded by a number and "> ". The user statement follows. For example, the first line of the user program is "LINK" with the "L" coded in position 1 of the line. Line numbers preceeded by a plus sign (+) are explained in the following paragraph. The user enters each statement shown directly into the UNIMAC GENLINK processor ("interactively") or from a text file. The latter method is re-commended. Each system on which the generator is implemented has its own mechanism for accomplishing such entry of a precoded file (for example the "redirected standard input file," ">," of PC-DOS or MS-DOS on the IBM-PC and compatible families of computers, and the @ADD runstream command on the Sperry, Inc. UNIVAC 1100 series processors) so each mechanism is documented seperately.

```
Line no.  Statement

 +   1>   LINK
 +   2>   *PES+ACF TO 85-CENSUS TEST MATCH 2
 +   3>   INPUT
 +   4>   *DUMP
 +   5>   FILENM=PESACF
 +   6>   FILESPC="PESA.IN1"
 +   7>   FIELD=CBNABLOCK,1,9
 +   8>   FIELD=SOUNDEX-STREETN,196,4
        FIELD=FIRST-NAME,18,18
    10>   FIELD=MIDDLE-INIT,36,1
    11>   FIELD=LAST-NAME,37,24
    12>   FIELD=RELATION,65,1
    13>   FIELD=SEX-MARITAL,178,2
    14>   FIELD=NEW-BIRTHDATE,180,6
 +  15>   FIELD=RACE-HISPANIC,176,-177
    16>   FIELD=ACF-STREET-NAME,106,20
    17>   FIELD=ACF-HOUSE-NUM,87,7
    18>   FIELD=ACF-UNIT,134,15
```

Sample GENLINK Program (Continued)

```
+  19>   FILENM=CENSUS
+  20>   FILESPC="CENACF.SRT"
+  21>   FIELD=CBNABLOCK,8,9
+  22>   FIELD=SOUNDEX-STREETN,179,4
   23>   FIELD=LAST-NAME,19,24
   24>   FIELD=FIRST-NAME,43,18
   25>   FIELD=MIDDLE-INIT,61,1
   26>   FIELD=NEW-RELATION,162,1
   27>   FIELD=SEX-MARITAL,160,2
   28>   FIELD=NEW-BIRTHDATE,163,6
   29>   FIELD=RACE-HISPANIC,158,2
   30>   FIELD=ACF-ST,101,20
   31>   FIELD=ACF-UHN,129,7
   32>   FIELD=ACF-UAPT,137,15
+  33>   PROCESS
+  34>   PATTERN=BOTH
+  35>   EQUATE=CBNA-BLOCK,CBNABLOCK,CBNABLOCK
+  36>   VAR=CBNA-BLOCK,N,.9,.9
+  37>   EQUATE=SOUNDEX,SOUNDEX-STREETN,SOUNDEX-STREETN
+  38>   VAR=SOUNDEX,C,.9,.0233
+  39>   ADJUST=SOUNDEX,,," "
+  40>   EQUATE=LASTNAME,LAST-NAME,LAST-NAME
+  41>   VAR=LASTNAME,U,.85,.0036
+  42>   ADJUST=LASTNAME,700.00,," "
   43>   EQUATE=FIRSTNAME,FIRST-NAME,FIRST-NAME
   44>   VAR=FIRSTNAME,U,.9816,.0877
   45>   ADJUST=FIRSTNAME,700.00,," "
   46>   EQUATE=MIDINIT,MIDDLE-INIT,MIDDLE-INIT
   47>   VAR=MIDINIT,C,.3469,.0272
   48>   ADJUST=MIDINIT,,," "
   49>   EQUATE=RELATION,RELATION,NEW-RELATION
   50>   VAR=RELATION,N,.3881,.2051
   51>   ADJUST=RELATION,,,"0"
   52>   EQUATE=SEX-MAR,SEX-MARITAL,SEX-MARITAL
   53>   VAR=SEX-MAR,N,.8255,.2114
   54>   ADJUST=SEX-MAR,,,"0"
   55>   EQUATE=BIRTHDATE,NEW-BIRTHDATE,NEW-BIRTHDATE
   56>   VAR=BIRTHDATE,P,.9379,.0399
   57>   ADJUST=BIRTHDATE,000006,,"0"
   58>   EQUATE=RACE-HISPA,RACE-HISPANIC,RACE-HISPANIC
   59>   VAR=RACE-HISPA,N,.9043,.6748
   60>   EQUATE=HOUSENUM,ACF-HOUSE-NUM,ACF-UHN
   61>   VAR=HOUSENUM,C,.99,.0149
   62>   EQUATE=APTNUM,ACF-UNIT,ACF-UAPT
   63>   VAR=APTNUM,C,.3525,.2628
+  64>   ORDER=CBNA-BLOCK,A,SOUNDEX,A,HOUSENUM,A,APTNUM,A
+  65>   LAMBDA=.001
+  66>   MU=.0005
+  67>   OUTPUT
+  68>   FILESPC="PESCEN2.OUT"
+  69>   END
```

2.2. Explanation of Sample GENLINK Program:

Line no. Explanation

1> This line specifies which SRD UNIMAC Program Generation System Processor is to be used. The current choice, LINK, specifies the GENLINK Record Linkage Program Generator.

2> This a comment line. A comment line is any line that starts with an asterisk in the first position of the line.

3> This GENLINK header statement indicates that the following GENLINK statements are associated with INPUT data.

4> DUMP is a special direcitive which causes GENLINK to list the content of all internal tables (not recommended). This directive has been made into a comment by the insertion of an asterisk in this example.

5> The FILENM statement defines one of the two files needed for a matcher program input. This name is used internally by the generator.

6> The FILESPC statement defines the actual file name for IBM-PC users.

7> The FIELD statement defines a single field in the file name whose FILENM preceeded. The FIELD statement shown here has the name "CBNABLOCK", starts in record position 1, and is 9 characters long.

8> The FIELD statement name ("SOUNDEX-STREETN") may be up to 15 characters long and may be given in any order without regard to position in the record. Note that this field statement falls at the end of the record although it is the second one defined since it starts in character position 179 of the record (i.e. it has the highest FIELD beginning position of any of the fields defined for the record). This illustrate that the order of the FIELD statements is unimportant, but one such statement is required for each field referenced on each file.

15> The third parameter of the FIELD statement may represent an ending position by being preceded by a hyphen ("-"). This FIELD statement is two characters long since it starts in character position 176 and ends in 177.

19-22> There are two INPUT header statement FILENM statements required for a GENLINK program. The same FIELD names may be reused in this new file definition. FIELDs with the same name may be in different record positions in different files (i.e., physically following different FILENM statements).

<u>Line no.</u> <u>Explanation</u>

33> A GENLINK program requires a PROCESS header statement.

34> The PATTERN directive has three possible parameters: COUNT, OUTPUT, or BOTH. See Section 3 for an explanation.

35> The EQUATE statement defines variables that can be referenced in other statements. Since matching requires fields to be compared on two files, GENLINK needs to know which fields on each file corresponds to each matching VARiable. This statement is coded with three parameters. The first, "CBNA-BLOCK" in this case, is the "variable" name by which the pair of FIELD names specified as the second and third parameters of this statement, "CBNABLOCK", for both files in this case, are known. The second parameter is for the field of the first file defined by a FILENM statement and the third is for the field of the second file defined by a FILENM statement. In this smaple program, statement 33 specifies that a variable named "CBNA-BLOCK" is to be used to define the comparison between the FIELD named "CBNABLOCK" on the file named "PESACF" (statements 5 and 7) and the FIELD named "CBNABLOCK" on the file named "CENSUS" (statements 19 and 21). It is these two fields that will be compared as defined by additional statements some of the parameters of which are described below. The order in which the EQUATE statements are coded is the order in which comparisons are generated and the order (right to left) in which comparison success/failure is represented in the match patterns generated as a result of the PATTERN directive (see above).

36> The VAR statement provides the program generator more information about the variable first defined by a preceeding EQUATE statement (see above). The first parameter must be a variable name previously defined in an EQUATE statement. The VAR statement provides the type of comparison code, Numeric, represented by an "N", in this case, to be generated. The other possible comparison codes are: Character, Prorated, Delta-percent, and Undecided (represented by C, P, D, and U, respectively). These are represented by their respective first letters. Each comparison is explained in detail in Section 3. The third parameter is the probability of agreement between the content of the two fields given that the records in which they are contained match. The fourth parameter is the probability of agreement between the content of the two fields given that the records that they are contained in are unmatched. If any of these parameters are missing, comparison code for the FEILDs specified in the corresponding EQUATE statement will not be generated. See Section 3 for more details.

<u>Line no.</u> <u>Explanation</u>

39> The ADJUST statement provides additional information needed to generate the comparisons specified by the EQUATE and VAR statements. The second parameter provides the proration factor, the base score for the Uncertainty comparison, and the percent change for the P, U, and D variables. The third parameter is the maximum percent adjustment permitted for Bayseian adjustment. If this parameter is not present for any of the variables no Bayesian adjustment is generated. See Section 5 for a discussion of Bayesian adjustment. The fourth parameter provides the value which means "no data was entered in the field." Note that the presence of this field will cause code to be generated for both comparison variables and blocking variables.

64> The ORDER statement defines the sort key or blocking variables for the record linkage program to be generated. The parameters must be provided in pairs, the first of which is the variable name defined in a previous EQUATE statement and the second of which is the direction of the sort, either an "A" for Ascending or a "D" for Descending. The variable names listed refer to the pair of fields associated with them in previous EQUATE statements. Four order variables have been defined for this GENLINK program. When a variable is designated an ORDER variable, it no longer participates in the generation of detailed field comparisons. Note, however, that the fourth, "null data," parameter of the ADJUST statement will cause blocks whose blocking FIELDs contain that data value to be skipped. A match of the corresponding fields of all blocking (ORDER) variables is required for any pair of records to be considered for a match. Hence these are "critical" fields since failure to match any one pair of fields results in the record being unmatched.

65,66> LAMBDA and MU statements are probabilities used to generate the high and low cutoff weights for the match determination. See section 5 for a discussion of how the match decision is made. LAMBDA and MU <u>must be provided</u> for a record linkage program to be generated (it is a fatal error for either to be missing).

67> OUTPUT files are predefined.

68> Only the FILESPC statement is permitted in the OUTPUT header statement and then only in the IBMPC environment. The first occurrence is applied to the output pointer file LINKOUT.DAT and the second to the optional COUNT.DAT. The latter is generated only if PATTERN=COUNT or BOTH.

69> When the END header statement is processed GENLINK starts generating COBOL program based on previous input (directives, header statements, statements and parameters see the definitions in the next section).

Section 3. General GENLINK Language Structure.

3.1. GENLINK Components.

The GENLINK user language consists of "directives", "header statements", "statements", and "parameters". Directives, header statements, and statements must each start a new line. Each statement, including parameters, must be coded in upper-case when letters of the alphabet are used.

3.1.1. Directives.

Directives are user instructions to the GENLINK UNIMAC processor about how the processor is to function while the user statements are being evaluated. GENLINK directives may or may not be associated with specific header statements.

3.1.2. Header Statements.

Header Statements set the "state" of the UNIMAC GENLINK processor. There are four GENLINK header statements:

INPUT	describes the characteristics of the two files to be matched.
PROCESS	describes the nature and charateristics of the comparisons to be made by the generated matcher program.
OUTPUT	decribes the matcher program output file or files. Due to memory constraints, the generated matcher programs do not produce matched and other data files or reports directly but rather each produces a pointer file and an optional compare pattern counter file (described in Appendix B). [Future plans call for the addition of an ability for the user to define a simple output report program.]
END	indicates to the GENLINK processor that user's GENLINK program is finished and that code generation can begin.

3.1.3. Statements.

Each header statement has specific statements that are coded with it. Statements are GENLINK user program statements which provide the GENLINK processor with information it needs to generate a file matching program.

3.1.4. GENLINK Statement Order.

Each header statement (except END) and each statement may be repeated as often as necessary. The order of header statements and statements is unimportant except that some parameters associated with certain header statement/statement or header statement/directive combinations may require that certain information have been provided earlier. Usually only one header statement of a kind would be used in a given GENLINK program. Using this method all INPUT related FIELDs and directives would be coded, followed by all the PROCESS variable definitions, and then by any OUTPUT directives. The sample program discussed in Section 2 is an example of this style of coding. However, it is possible to use the same header statement more than once in any given GENLINK program. You might want to define the comparison variable characteristics associated with a given set of INPUT fields immediately after defining the fields. This would be accomplished by coding the following:

```
a->  INPUT
|    FILENM=A-FILE
|    FIELD=FIELD-1A,...
|    FILENM=B-FILE
|    FIELD=FIELD-1B,...
|    PROCESS
|    EQUATE=VAR-1,FIELD-1A,FIELD-1B
|    VAR=VAR-1,...
a->  ADJUST=VAR-1,...
b->  INPUT
|    FILENM=A-FILE
|    FIELD=FIELD-2A,...
|    FILENM=B-FILE
|    FIELD=FIELD-2B,...
|    PROCESS
|    EQUATE=VAR-2,FIELD-2A,FIELD-2B
|    VAR=VAR-2,...
b->  ADJUST=VAR-2,...
     INPUT
       .
       .
       .
```

The statements bracketed by the "a->" pointers define all the information needed for the VAR-1 comparison of FIELD-1A with FIELD-1B. The statements bracketed by the "b->" pointers completely defines the VAR-2 comparison characteristics. This type of coding is useful when there are a lot of FIELD definitions resulting in the comarison variable definitions being separated from the source FIELD definitions by more than a page of user code using the normal coding method. Generator efficiency is degraded using this technique.

3.1.5. Parameters, Parameter Lists and Keywords.

A parameter is the way specific information about content or choice is programmed by the user. All statements and some directives have parameters which always must be entered on the same line as their associated statement or directive. Statements and directives which have parameters are called keywords. There may be more than one parameter associated with a keyword. This set of parameters is called a parameter list.

3.1.6. Coding GENLINK Statements.

A GENLINK keyword must be followed by an equal sign (=) followed by the parameters.

Example:

```
     VAR=SOUNDEX,C,0.9,0.0036
```

For readability, you may use spaces to separate parameters and keywords. The following example is identical to the example above except that spaces have been added for readability.

Example:

```
     VAR =  SOUNDEX, C, 0.9 , 0.0036
```

If a parameter contains spaces, commas, apostrophes (') or quote marks ("), the parameter must be surrounded by quote marks. To leave out a parameter or to enter a null parameter in a parameter list, use an additional comma. The following example has four parameters: "LASTNAME", "700.00", a null parameter, and a space.

Example:

```
     ADJUST=LASTNAME,700.00,," "
```

To represent a quote mark in a parameter, use two quote marks. In the following example there are again four parameters in the parameter list: "MIDINIT", two null parameters, and one quote mark.

Example:

```
     ADJUST=MIDINIT,,,""""
```

See Appendix A for a formal description (using a modified Backus-Naur Form or BNF) of this and other GENLINK user language statements.

3.1.7. Comments.

An asterisk (*) in the first position of a line means the line is a comment and the line will be ignored by the GENLINK processor. Comment lines may be used any place (except line 1). See the Sample Program, section 2, lines 2 and 4, for examples.

3.1.8. Description Format.

In the following paragraphs, each type of GENSTAN statement or directive is illustrated by a format. This paragraph describes how that format is constructed to illustrate the coding of each type of GENSTAN statement or directive. The formats given are either general formats or examples. Those formats preceded by an "Ex: " are examples containing illustrative coding. All others are general formats. For formats that illustrate keyword expressions, the keyword precedes an equal sign ("=") and the list of formal parameters or an example of actual parameter(s), which might be entered to complete the expression, follows.

In general formats, each possible parameter type is called a "formal parameter." In this system, all keyword expression parameters are positional. This means that the program generator knows how to treat each parameter by the position of the parameter in the keyword expression. A formal parameter is represented by the name of the parameter, preceeded by "<" and followed by ">", in it's relative position in the keyword expression. A required formal parameter will be underlined in the general format. A required formal parameter is one for which the user must code something. The meaning of each formal parameter in a general format is given in the "explanation" column. In cases where the general format has a parameter position that may contain one of several choices, the possible alternatives are shown in the appropriate position and are separated by bars (|) in the general format in the expession column. Note that a formal parameter is <u>not</u> actually coded but rather represents what kind of information <u>might</u> be coded in a given parameter location.

For example, the following is a general format for a keyword expression:

```
FIELD = <NAME>, <BEG>, <LNG> | <END>, <PIC>, <REP>
<-a-> ↑ <--b->↑ <-c->↑ <-e-> ↑ <-f->↑ <-g->↑ <-h->
      i       j      j       k      j      j
                       <-----d----->
```

This is a general format for the FIELD statement. The FIELD statement is a statement type available under both INPUT and OUTPUT header statements. It is coded by entering the keyword FIELD (marked a above) separated from its parameters by an equal sign (marked i). This statement can be coded with as many as 5 actual parameters, represented by 6 formal parameters (marked b, c, e, f, g, and h). The formal parameters marked e and f are separated by a bar (|, marked k) indicating that they represent a choice in the third actual parameter (marked d). Thus, either an <END> parameter or a <LNG> parameter would be coded. The commas (marked j) are optional when actually coding but indicate here the separation between actual parameters. Only those formal parameters which are underlined must be provided.

If a group of parameters may be repeated, they will be surrounded by square brackets ([]) and followed 3 periods (...).

In examples, the keyword expression (directive or statement) in the "Expression" column is the way a the directive or statement might actually be coded. The meaning of each example is given in the "explanation" column surrounded by parentheses. Notes on the general use of the type of statement illustrated by the example would <u>not</u> surrounded by parentheses.

3.2. INPUT Header Statement

The statements and directives described below appear following the INPUT header statement.

3.2.1. The File Name Statement.

The input file name is specified by this statement. It must not be the same as the output file name and cannot be a COBOL reserved word.

Expression	Explanation
Ex: FILENM = ABCD	(The COBOL name by which this file is known is "ABCD".)

An input file name (FILENM) may be up to 6 characters long, must start with an alphabetic character and may have numeric characters and hyphens (-). This statement is required to be present, and there must exactly two unique input file names defined by it. As a short-hand, we will call the first file "File A" and the second file "File B." All statements coded following a given occurence of FILENM, until another header statement (i.e., PROCESS, OUTPUT, or END) or until a different input file name is defined, applies to the specific file defined by this statement. Switching between files is OK and is accomplished by using the other FILENM statement expression.

3.2.2. The File Specification.

The implementor defined, system specific file specification is provided by this statement.

Ex: FILESPC = "ABCDEFG.XYZ"	(The IBM-PC file specification is "ABCDEFG.XYZ".)

The FILESPC statement is used to provide additional "external" file access information needed by the "TARGET" system. Presently, this statement only applies to the IBM-PC.

3.2.3. Number of Records Per Block.

Ex: NRECS = 10	(There are 10 records per recorded block -- per block on the data storage device -- for the current file.)

Expression	Explanation

3.2.4. Number of Records to Read for Test.

Ex: TEST = 100	(Stop after processing 100 records of this file.)

This statement does not function in the current version of GENLINK, however see TEST (Paragraph 3.4.10) under the PROCESS header statement.

3.2.5. The Maximum Logical (Comparison) Block Size.

Ex: MAXBLK = 50	(The maximum matcher comparison block size expected for this file is 50.)

It is important to recognize that the maximum comparison block size is very dependent on the fields used to sort each input file. See the discussion of the steps used in determining LINK parameters (Section 5).

3.2.6. The Record Size Statement.

Ex: RECSIZE = 500	(The record size of this file is 500 characters.)

The RECSIZE value will be replaced and an appropriate warning issued if the record size specified is found to be inconsistent with subsequent FIELD or LINK data.

3.2.7. The INPUT File Device.

Ex: DEVICE = TAPE	DEVICE = TAPE | CARD | DISC (The INPUT file device is tape.)

If no DEVICE is specified, DISC is assumed.

3.2.8. General Data Charateristics of a File.

Ex: DATA = ASCII	DATA = CENIO | ASCII | EBCDIC | FIELDATA | XS3 (The character set of this file is ASCII.)

ASCII is the only parameter option currently fully implemented for this statement. ASCII is an acronym for the American Standard Code for Information Interchange. EBCDIC stands for Extended Binary Coded Decimal Information Code and is a character code developed in the late 1950's for use on IBM main-frame computer systems. EBCDIC is currently only available if the target system is the UNIVAC-1100-80. FIELDATA is a character set implemented on the UNIVAC-1100-80 to maintain compatiblity with ealier versions of Sperry UNIVAC computers. XS3 stands for the eXceSs-3 character code, a code set which was developed to support paper tape and data communications applications.

The CENIO parameter will generate all the code necessary to read or write CENIO (Census Compacted) files containing (for now) ASCII data, but buffer and record sizes may not be correct. Note that for now, CENIO only applies to the UNIVAC-1100-80.

When the CENIO parameter is used, an external UNIVAC-1100-80 COBOL file name of "10." will be used for the INPUT file.

	Expression	Explanation
3.2.9.	Define a Data Field.	

FIELD = <NAME>,<BEG>,<END>|<LNG>,<PIC>

This statement is used to define data fields for each record of the files being matched. The FIELD statement may be coded with three or four parameters.

<NAME> = Field Name

This parameter is the name of a field being defined in the current INPUT File. It may be from 1 to 15 characters in length, must start with a letter of the alphabet (A to Z) and can contain numbers (0 to 9) and dashes (-). A minus sign can't follow itself. The parameter should not be a COBOL reserved word. This last restriction is not checked by the GENLINK processor but will cause errors in the generated program when it is compiled. Occurances of <NAME> must be unique for each file, File A and File B. Thus, there may be no more than two NAME parameters the same under all INPUT header statements in any given GENLINK user program. However, we do recommend that you name the fields the same NAME that are going to be compared between the two files. See the example in Section 2.

<BEG> = Beginning Position from the leftmost position in the record (position 1)

Expression	Explanation
	<END> = Ending Position Use a Negative Number (a number with a leading minus sign)

Example:

FIELD = NAME,15,-30	A FIELD name for the current input file is "NAME" and is defined to be character positions 15 to 30.
	<LNG> = Length Use an unsigned number

Example:

FIELD = NAME,15,16	This is the equivalent FIELD statement definition to the previous example, except that the Length, <LNG>, option was used.

Note that <END> and <LNG> are mutually exclusive (if you use one in a given FIELD statement expression, you can't use the other).

	<PIC> = Standard COBOL DISPLAY PICTURE Clause (op tional)

Example:

Ex: FIELD = NAME,15,16,"X(16)"

(This is the equivalent FIELD statement definition to the previous examples, except that the COBOL PICTURE clause "X(15)" was explicitly provided.)

3.2.10. Link to GENSTAN OUTPUT File Definition.

Expression	Explanation
Ex: LINK = TRUE	(This directive indicates to GENLINK the entire definition associated with the current FILENM statement was previously defined as an OUTPUT file in a GENSTAN user program.)

See GENSTAN documentation section on the OUTPUT header statement, LINK directive for a detailed discussion of the generation of LINK UNIMAC macro subprogram. When this INPUT directive is provided, no additional statements need be coded for the associated Matcher input file. Typically, only MAXBLK and possibly FILESPC would be coded. However, additional FIELD statements would be coded if the user wanted to subdivide already defined fields on the file. The FILENM statement must immediately precede the LINK directive.

3.3. OUTPUT Header Statement

Only FILESPC is permitted as an statement associated with the OUTPUT header statement at present. A version of GENLINK is planned that will genearte report programs and matched data files.

Expression	Explanation

3.3.1. The File Specification.

The implementor defined, system specific file specification is provided by this statement.

Ex: FILESPC = "POINT.DAT"	The IBM-PC file specification for the Pointer file is: POINT.DAT

The first occurence of FILESPC after OUTPUT is associated with the pointer file generated by execution of the record linkage-program produced by GENLINK. The second use of FILESPC will be associated with the file produced by PATTERN = COUNT or BOTH under the PROCESS header statement. The name specified in this statement is an external output file name chosen by the user.

3.4. PROCESS Header Statement

The PROCESS header statement sets the state of the GENLINK processor so that it recognizes statements associated with file "blocking" and field comparison or "matching." The EQUATE, VAR and ADJUST statements define field comparison variables and the comparisons performed on them, while ORDER establishes the variables that were used to sort the two files being compared. Other statements provide match/non-match comparison cut-offs. The PATTERN directive controls match variable pattern retention. The SEQ directive causes file sequence fields to be generated, when meaningful, on the the output pointer file.

3.4.1. The EQUATE statement.

The EQUATE statement is used to associate a File A field with a File B field and both with a PROCESS variable. Thus the EQUATE statement gives a single name (called the PROCESS variable or variable name in this document) to a pair of fields -- one on each of the files being matched -- so that reference can be made to the process of comparing those fields. The File A and File B fields must already have been defined. The variable name must not have been previously defined as a PROCESS variable (i.e., as the first parameter in another EQUATE statement). The construction of a variable name follows the same rules as the INPUT FIELD <NAME> parameter. Thus, a variable name may be up to 15 characters in length. A variable name may be the same as a File A FIELD name or a File B FIELD name.

Expression	Explanation
EQUATE = <NAME>,<File_A_FIELD_NAME>,<File_B_FIELD_NAME>	
	<NAME> = Variable Name

This parameter is the name of a potential comparison variable being defined as being derived from the comparison of a File A and a File B field. Like a FIELD <NAME> parameter, a Variable <NAME> parameter may be from 1 to 15 characters in length, must start with a letter of the alphabet (A to Z) and can contain numbers (0 to 9) and hyphens (-). A hyphen can't follow itself. The parameter should not be a COBOL reserved word. This last restriction is not checked by the GENLINK processor but will cause errors in the generated program when it is compiled. Occurances of NAME must be unique for all variables. However, you may name the variables the same NAME as the File A field or File B field compared as the result of this statement. See Section 2 for an example.

Expression | Explanation

<File_A_FIELD_NAME>
<File_B_FIELD_NAME> = Name defined by the NAME parameter of a previously coded FIELD statement

Example:

```
       EQUATE = VAR-NAME,FILE-A-NAME,FILE-B-NAME
```

(The variable name "VAR-NAME" is to be used to define the comparison of the field named "FILE-A-NAME" in the record definition of the first INPUT file, File A, with the field named "FILE-B-NAME" in the record definition of the second INPUT file, File B.)

3.4.2. The Variable Comparison Definition Element.

```
        VAR = <NAME>,<TYPE>,<P(M)>,<P(U)>
```

The VAR statement is used to define the type of comparison to be done on the EQUATEd File A and File B fields.

<NAME> = Variable Name (the NAME parameter of a previously coded EQUATE statement)

<TYPE> = C | N | U | P | D
Type of field comparison

There are currently five <TYPE>s of field comparisons defined in GENLINK. These are:

"Character" comparison, indicated by a "C".

"Numeric" comparison, indicated by an "N".

The Character and Numeric comparisons are simple exact comparisons. The weights are assigned as the result of these comparisons can only be either the agreement or the disagreement weights (Wt(Agr) and Wt(Dis) -- see the discussion of the <P(M)> and <P(U)> below for derivation).

"Prorated" comparison is indicated by a "P". The prorated comparison results in a comparison weight (Comp_Wt) that is the proportional value between the agreement and disagreement weights established by the ratio that the difference between the absolute (positive) difference between the two comparison values (X1, the File A FIELD value, and X2, the File B FIELD value) between the Proration Factor, <F>(Pro), and zero:

$$\text{Comp\_Wt} = \frac{\text{Wt(Agr)} - \text{Wt(Dis)}}{0 - \text{<F>(Pro)}} * (|\ X1 - X2\ | - \text{<F>(Pro)}) + \text{Wt(Dis)}$$

The proration factor must be provided as the second parameter (<FACTOR>) of the ADJUST statement, if the a "P" is used in the <TYPE> field of the VAR statement or an error will result.

"Delta Percentage" is represented by a "D". The delta percent comparison results in a comparison weight (Comp_Wt) that is the proportional value between the agreement and disagreement weights established by the ratio of division the two comparison values (X1, the File A FIELD value, and X2, the File B FIELD value) between the Delta-Percentage Factor, <F>(D%), and zero:

$$\text{Comp\_Wt} = \frac{\text{Wt(Agr)} - \text{Wt(Dis)}}{0 - \text{<F>(D\%)}} * \left(\frac{|\ X1 - X2\ |}{X1 + X2} - \text{<F>(D\%)} \right) + \text{Wt(Dis)}$$

The delta-percent factor must be provided as the second parameter (<FACTOR>) of the ADJUST statement, if the a "D" is used in the <TYPE> field of the VAR statement or an error will result.

"Uncertainty" comparison, represented by a "U", is a string-distance comparison function, developed by Matthew A. Jaro, of the US Bureau of the Census. Given two character strings, the Jaro String-Distance Function returns a value between 900 and 0, with 900 representing a perfect match. The user may set a non-match cutoff value, <F>(Non-Mt), lower than which a string comparison result is set to the disagreement weight for the comparison variable. If the string-distance value (Str_Dist) determined by the function is higher than this cutoff, but lower than 900, the weight assigned to the comparison is the value (the comparison weight or Comp_Wt) between the agreement weight (Wt(Agr)) and the disagreement weight (Wt(Dis)) that has the same ratio between those two values as the calculated string distance bares to 900 and the non-match cutoff. This can be represented by the following equation:

$$\text{Comp\_Wt} = \frac{\text{Wt(Agr)} - \text{Wt(Dis)}}{900 - \text{<F>(Non-Mt)}} * (\text{Str\_Dist} - \text{<F>(Non-Mt)}) + \text{Wt(Dis)}$$

The non-match cutoff may be set by the user by entering a number (<F>(Non-Mt)) between 0 and 900 as the second parameter (<FACTOR>) of the ADJUST statement. If no cut-off is provided by the user, <F>(Non-Mt) = 700 is assumed.

Expression	Explanation
	P(M) = The probability that the values of the pair of fields represented by this variable agree given that the record pair being compared is a true <u>match</u>.

Expression | Explanation

P(U) = The probability that pair of fields represented by this variable the content of the fields agree given that the record being compared does not match (is actually unmatched).

The two probabilities (P(M) and P(U)) are used to calculate the weight assigned to a successful comparison (the agreement weight or Wt(Agr)) and the wieght assigned to an unsuccessful comparison (the disagreement weight or Wt(Dis)). In the following equations "ln" means "the natural log of" (i.e., log base e):

$$Wt(Agr) = \ln P(M) - \ln P(U)$$

$$Wt(Dis) = \ln (1 - P(M)) - \ln (1 - P(U))$$

Expression | Explanation

Example:

```
VAR = VAR-NAME, U, 0.8235, 0.35
```

(The variable named "VAR-NAME" will use the Uncertainty comparison (the second parameter), and a probability that the compared fields will agree given that the File A and File B records match is 0.8235 and that the compared fields will disagree given that the records they are in do not match is 0.35)

3.4.3. The Variable Adjustment Statement.

```
ADJUST = <NAME>,<FACTOR>,<Bayesian_LIMIT>,<NULL_DATA>
```

<NAME> = A previously defined variable name (See the discussion of the EQUATE statement, above)

Expression	Explanation

<FACTOR> = The specific factor required by a certain field comparison methods (specificly to D, P, and U types)

<FACTOR> applies to the Uncertainty comparison, the Delta-percent comparison, and the Prorated comparison. In the Uncertainty comparison, <FACTOR> represents the score above which the disagreement weight is assigned to the comparison. <FACTOR> is the percentage limit in the percent change comparison. Finally, <FACTOR> is the value of the proration for the prorated comparison. See the discussions above under each <TYPE>, D, P, and U for further explanations of the use of the parameter, <FACTOR>.

<Bayesian_LIMIT> = The percent limit permitted for Bayesian adjustment of the weights for this comparison variable

The <Bayesian_LIMIT> is the percentage limit by which the weights assigned to this variable comparison field are allowed to be modified by Bayesian adjustment. If this parameter is provided, Bayesian adjustment is accomplished for this field based on the amount of variation of data found in the field in each block. If this parameter is not provided, no Bayesian adjustment is attempted on the weights for this variable. If no comparison variables have this parameter, no Bayesian adjustment code is generated for this version of the program.

<NULL-DATA> = The value entered in both the File A and File B fields when no data was provided.

If this parameter is present, it applies whether this is a comparison variable or a blocking or sort variable. The presence of null-data in a blocking field of both File A and File B causes the otherwise valid block to be ignored. The presence of of null-data in a comparison variable field on both File A and File B causes a zero weight to be assigned for that comparison. Null-data in either file with valid data in the other causes the disagreement weight Wt(Dis) to be assigned to the comparison.

3.4.4. The Blocking Statement.

The ORDER or "Blocking" statement specifies those variables previously defined by EQUATE statements on which the two files to be compared are sorted. The first NAME, ORDER pair is considered to define the primary sort key, the second, the secondary key, and so forth. Up to 10 of these pairs can be specified. The keys in the two files to be matched need not be in the same relative positions. It is only important that the files be sorted in the same logical order (i.e. by respective field content) when the record linkage program is finally run against them.

Expression	Explanation
ORDER = <NAME>,<ORDER>[,<NAME-2>,<ORDER-2>] ...	
	<NAME>, <NAME-2>, etc. = A variable name previously defined by an EQUATE statement.
	<ORDER>, <ORDER-2>, ... = A | D An indicator of sort order. A = Ascending D = Descending

Only Ascending ORDER is implemented now. Future enhancements may implement Descending ORDER. This is not significant because sorting is only used to limit the region of consideration for the matcher and thus has no other intrinsic meaning.

The <NAME>s used in the ORDER statement must be given in the actual order in which the sort keys were specified for the files being matched. Subtle, hard to detect results may be generated if the ORDER <NAME>s are not in the correct order.

Ex: ORDER = SNDX-LAST-NAME,A,CBNA,A

(The files being matched are sorted on 2 fields each, pointed at by the EQUATE variables SNDX-LAST-NAME and CBNA. Both fields were sorted in ascending order.)

3.4.5. <u>LAMBDA</u> <u>and</u> <u>MU</u>.

<u>Lambda</u> is the false non-match probability. <u>Mu</u> is the false match probability. <u>Lambda</u> and <u>Mu</u> are used to calculate the HIGH-CUTOFF and LOW-CUTOFF weights for determining whether assigned record pairs match, require clerical review or are unmatched.

Expression	Explanation
Ex: LAMBDA = 0.001	(This statement indicates to GENLINK and the generated record linkage program that no more than one in 1000 false non-matches are to be permitted.)
Ex: MU = 0.0005	(This statement indicates that no more than five in 10,000 false matches are to be allowed.)

3.4.6. Comparison Pattern Generation Request.

```
PATTERN = COUNT | OUTPUT | BOTH , NULL-DATA
```

This directive indicates that field comparison patterns are to be generated and used.

COUNT causes the pattern counters to be dumped in their own output file. OUTPUT causes the pattern associated with the comparison of each tentatively assigned record pair to be output with each pointer file record associated with such pairs. The parameter BOTH causes both program options to be generated. If the PATTERN directive is not present the code to determine field comparison patterns will not be generated. If the NULL-DATA option is included, the patterns of NULL-DATA comparisons will be included (or counted) for each file.

A pattern is a binary number (or its decimal integer equivalent) which represents which comparison variables were found to match. Each bit, or binary digit, represents a particular variable, which in turn represents a pair of fields being compared (see the discussion of the EQUATE statement, above). The bits are assigned, left-most bit first, in the order in which the EQUATE statements occur in the LINKGEN user program. The number of bits used for a pattern is equal to the number of comparison variables (i.e. the number of VAR statements less the number of variables used in blocking -- appearing in the ORDER statement).

The pattern is established by the variable field comparison portion of the matcher program. Each variable comparison which is successful cause its bit in the pattern to be set to "1". Each unsuccessful comparison causes its bit to be set to "0".

Expression	Explanation
Example:	
PATTERN = BOTH	(This example will cause both a comparison pattern cumulative counter file to be generated and the individual comparison pointers to be output on the output pointer file.)
Example:	
PATTERN = COUNT	(This example will cause only the comparison pattern cumulative counter file to be produced. There will be no modification to the output pointer file.)

If the NULL-DATA option is included, each field of each file will be compared to the <NULL-DATA> for the respective comparison variable assigned as the fourth parameter of the ADJUST statement. Thus there will be two null-data comparison patterns generated, one recording the results of comparisons between each comparison field of a given record on File A with its corresponding null-data, the other recording the results of comparisons between fields on File B and their respective nulldata values. If no null-data was assigned, the comparison will be considered to have failed and the bit in that position will be set to "0". There is a null-data comparison pattern generated for each record not skipped in each block. The two additional null-data comparison patterns will be put on any pointer file record for which a comparison pattern is output.

Example:

PATTERN = OUTPUT, NULL-DATA

(This example will cause inter-file comparison patterns, File A to null-data comparison patterns and File B to null-data comparison patterns to be output with appropriate output pointer file records.)

3.4.7. <u>Generate Sequence Fields on the Output Pointer File</u>.

SEQ = <<u>NAME</u>>, <PIC>	This Directive indicates that sequence numbers will be included in the output pointer file.

The necessary logic and structures will be generated so that a pair of sequence numbers (one from each file being compared) is included in those pointer file records where such inclusion is meaningful. These fields may be any fields from the two INPUT files. It is intended that these fields be unique to individual records in each file so that they may be used in constructing match result output programs summarizing multiple passes of the same data files.

<u>Expression</u>	<u>Explanation</u>
	<NAME> = Variable Name (the NAME parameter of a previously coded EQUATE statement)

The FILE-A and FILE-B fields pointed to by the EQUATE variable <NAME> will be output on the pointer file on the "MP", "UA", "UB", and "CP" type records.

	<PIC> = Standard COBOL DISPLAY PICTURE Clause (optional)

The COBOL PICTURE clause is optional. If no PICTURE clause is provided, the PICTURE clause associated with each of the SEQ fields will be used. If either has no PICTURE clause, a PICTURE clause of X for the designated or calculated size of the indicated field will be used.

3.4.8. <u>Apply Blocking Penalty Weight</u>.

Example:

PENALTY = 1.7	(Apply a blocking "penalty weight of 1.7 to all composite comparison weights.)
PENALTY = TRUE | <WEIGHT>	TRUE = The PENALTY weight will be calculated based on the M and U probabilities assigned to the ORDER (or Blocking) VARiables by the user.
	<WEIGHT> = Any arbitrary penalty weight may be assigned by this statement.

Expression	Explanation
Example:	
PENALTY = 0	(This is the default value "PENALTY Weight" applied when none is specified by the user.)

The PENALTY Weight is the weight to be assigned if the user wants to add a weight to each comparison to compensate for the fact that the blocking variables are always matched.

3.4.9. Skip Comparison Blocks.

Example:

SKIP = 500	(SKIP the first 500 comparison blocks encountered.)

This Statement is used to generate a test version of a record linkage program that will skip a number of blocks determined by the user before actual matching begins.

3.4.10. Test a Number of Comparison Blocks.

Example:

TEST = 500	(TEST 500 blocks.)

This Statement is used to generate a test version of a record linkage program that will only compare 500 comparison blocks before terminating. What ever order coded in, TEST will always take place after SKIP has been satisfied. Thus if the Statement TEST=500 appears in a GENLINK program, then the statement SKIP=500, the resulting program will skip the first 500 comparison blocks, then consider the next 500 comparison blocks (i.e., blocks 501 to 1000) and then perform all normal and requested termination processing (see the PATTERN Statement).

3.4.11. Run-time Display Options.

This statement enables the user to control the generation of all run-time displays except the terminal counter displays. The run-time displays are used to record detailed information about the progress of a matcher run. If displays are generated, the essential content of every transaction written to the output pointer file (and to the counter file, if one is generated) is displayed on the system output device. This detailed display is the default mode when the TARGET system is the IBM-PC. The run-time display is not generated (as the default) when the TARGET system is the UNIVAC-1100-80.

Expression	Explanation
DISPLAY = ON | OFF | DEBUG	ON = Regardless of the TARGET system, run-time displays will be generated.
	OFF = Regardless of the TARGET system, NO run-time displays will be generated.
	DEBUG = Run-time displays are generated as debug lines (with a "D" in column 7).

Run-time displays are always generated if an INPUT TEST or a PROCESS TEST statement is coded. However they are generated as COBOL debug clauses if DISPLAY=OFF is used (meaning that they are in the COBOL code but are not compiled into the actual executable machine code). This is effectively the same as DISPLAY=DEBUG.

The use of debug clauses enables the user to easily generate the run-time displays if they are needed during a run without having to regenerate the matcher program but simply by adding a DEBUG directive (see paragraph 3.2.4) to the GENLINK user program or a WITH DEBUGGING MODE clause to the SOURCE-COMPUTER paragraph of the generated COBOL source program.

The WITH DEBUGGING MODE clause causes all the debug clauses to be compiled into the executable version of the program without any further modification to the program. Without the WITH DEBUGGING MODE clause, an ANS COBOL-74 conforming compiler treats all debug clauses (all lines that have a "D" in the "indicator area", or column 7) as comments. The DEBUG directive will cause GENLINK to generate this clause.

3.5. Independent Directives.

The GENLINK program generator directives which are independent of header statement state are detailed below. These GENLINK statements can appear any place in the program except as noted.

Expression	Explanation

3.5.1. GENLINK Identification Directive.

Ex: LINK	(This directive or "MATCHER" must be the first statement in a GENLINK user program.)

This directive is actually given in response to the SRD UNIMAC Program Generator System question about which generator is required. The choices currently are:

"STAN" or "STANDARDIZER" for this program, GENSTAN

"LINK" or "MATCHER" for the record linkage program generator, GENLINK

"UNDUP" or "UNDUPLICATOR" for the file unduplication program, UNDUPGEN

One of these directives must appear first in the GENLINK program or be the reponse to the initial SRD Program Generation System question if the system is being used interactively.

3.5.2. List Options.

LISTOPT = PROGRAM | NOW | OUTPUT | BOTH | ALL | OFF	PROGRAM = The GENLINK user program is generated as COBOL comments at the beginning of the generated COBOL file matcher program starting with the statement following this directive.

A GENLINK user program comment will be shifted to column 7 so that the asterisk becomes the COBOL comment indicator. All other GENLINK user statements will be shifted to the right 9 positions and the characters * and > will be put in positions 7 and 8. The asterisk will make this record a COBOL comment and the greater than sign will direntiate it from a GENLINK user program comment.

Expression	Explanation	
	NOW =	The GENLINK user pro gram is displayed as it is processed.
	OUTPUT =	The generated COBOL file matcher program is displayed to the user as it is being generated.
	ALL or BOTH =	All of the above options are activated.
	OFF =	Deactivates the currently acivated options for GENLINK user program statements following this directive.
Ex: LISTOPT = PROGRAM	(Generate a listing of this user program as comments on the generated COBOL program.)	

3.5.3. Target Machine for the Generated Program.

The TARGET directive idicates the machine upon which the COBOL program will be compiled and run. This does not have to be the machine on which SRD Program Generator is being run. If this directive is not provided, the machine on which the SRD Program Generator is operating will be considered the TARGET for the generated program.

TARGET = IBM-PC | UNIVAC-1100-80	This directive defines the tagret system of the UNIMAC SRD Program Generator System.

That means this directive is used to indicate which computer system will be used to run the file linkage program being generated by GENLINK. This is important because each COBOL implementation and each computer operating system is slightly different and the generator has to generate different code for each. In addition, some GENLINK user statements are interpreted differently for different computer target systems. For example, the FILESPC statement of the INPUT and OUTPUT header statement is meaningful, at present only for the IBM-PC and TARGET=IBM-PC will be assumed if TARGET is not specified and FILESPC is coded.

Expression	Explanation

3.5.4. The DEBUG Mode Switch.

Ex: DEBUG	(Causes the WITH DEBUGGING MODE clause to be generated on the SOURCE-COMPUTER paragraph of the ENVIRONMENT DIVISION.)

The WITH DEBUGGING MODE clause causes all the debug clauses to be compiled into the executable version of the program without any further modification to the program. Without the WITH DEBUGGING MODE clause on the SOURCE-COMPUTER paragraph of the ENVIRONMENT DIVISION, an ANS COBOL-74 conforming compiler treats all debug clauses (all lines that have a "D" in the "indicator area", or column 7) as comments. The DEBUG directive will cause GENLINK to generate this clause.

The use of COBOL debug clauses enables the user to easily compile the run-time displays that have been generated by GENLINK as a result of a PROCESS DISPLAY=DEBUG statement, if they are needed during a run without having to regenerate the matcher program but simply by adding a WITH DEBUGGING MODE clause to the generated COBOL program. The DEBUG mode switch will cause the clause to be added to the generated program icase the user is unsure of where the clause goes or is regenerating the matcher for other reasons.

3.5.5. Symbol Table Dump Directive.

DUMP	(The internal GENLINK program generator tables will be DUMPed after the END header statement but before the match program is generated.)

The DUMP directive is used to ensure that the GENLINK user statement processor is interpreting the GENLINK program correctly. It is not needed for a "production" match program generation run and since it is time consuming it is not recommended.