DECUS
PROGRAM LIBRARY

DECUS NO.	8-556
TITLE	CHISQ CHI SQUARE PROGRAM
AUTHOR	Marjorie H. Kleinman
COMPANY	Center for Community Research New York, New York
DATE	June 19, 1972
SOURCE LANGUAGE	FORTRAN

CHISQ Chi Square Program

This program will compute up to 20 chi squares at a time on tables as large as 8 by 9. Missing data is permitted. If the ID is missing or out of range, the entire case is ignored. If any item is missing or out of range, that item for that case is ignored.

Input is in the form of identifying information, which determines in which row the subject belongs for every chi square table; and the individual's scores on up to 20 items, each score determining the column in which he belongs for the appropriate chi square table. The ID and scores can appear in any order.

Tables need not all be the same size for the same run. Items may have different ranges.

There is an option for computing the contingency coefficient for each table.

MINIMUM HARDWARE

PDP-8/I with 8K of core, with teletype, printer, and disk or DECtape.

OTHER PROGRAMS NEEDED

PS/8 Operating System with Fortran Compiler

NOTE:

The program consists of 2 chained segments. The first contains a main program and subroutine IRD, and is called CHISQ. The second is named SQUARE. Each must be compiled, loaded, and saved separately on the system device. To allow

for execution-time formatting of input data, load CHISQ and change the contents of location 4072 from 4124 to 3400 before saving in the system device. For an explanation of execution-time format statements, see the accompanying write-up.

How to run CHISQ Chi Square Program:

1) .R CHISQ (Starting Address is 02435)
2) Type in a heading to appear on output (maximum length is 36 characters).
3) Type in number of rows (ID categories). Maximum is 8.
4) Type in number of items (not including ID). Maximum is 20.
5) Type in ICORR, contingency coefficient option. (1 is yes, 0 is no).
6) Type in number of categories for each item (in 20I1 format). The number of categories can vary from item to item.
7) Type input device (1,2,3, or 4). If input is from magnetic tape, the file must reside on unit 1 with the name 'CHIN.DA'.
8) Type in which variable is ID. If it is the first, type 1. If it is last and there are 20 items, type 21, etc.
9) Type in the format statement. 2 lines of 72 characters each are permitted. (Be sure to press RETURN twice). All data must be read in integer format. The last card (or record) must have ID equal to 9.

Note: The "READ" statement is at location 04065. If there should occur a card reader error, STOP, Load Address 04065, and press START.

Execution-time Format Statements

Execution-time format statements are not permitted by the compiler. However, by changing one location in the program after it is in core, it is possible to read in a format statement (from the teletype only), and use it for reading the input. Set up the program (or subprogram) as follows:

```
      DIMENSION FMT (12)
C     This is the array where the format statement will
C     be stored.
      READ (1,15) FMT
15    FORMAT (12A6)
C     A format statement as long as 72 characters can
C     be read in here.
      .
      .
      .
20    READ (IN,25) DATA1,DATA2
C     Input data is read in from device IN
25    FORMAT (F3.0)
C     This is a dummy format that will not be used.
      .
      .
      .
      END
```

Be sure to obtain an assembly listing of the program, and also a map at execution time, so that the precise location to be changed can be determined.

Load the program and all subroutines, and begin execution. When the program pauses or waits to accept data, the appropriate location can be changed.

Now look at the assembly listing. Determine the actual address of the first word of array FMT. (See page C-4 of 8K SABR Assembler manual or PS/8 manual for computing actual addresses.) Now look through the listing for the Fortran statement where the reading of the data is done, in this case, statement 20. You will notice that in the assembled code will appear the following:

```
4Ø33  CALL  2, READ     (call to subroutine READ
                         with 2 arguments)
62Ø1  ARG   (n          (n is the device number)
DDDD                     (DDDD is the address where
                         the device number is stored.)
62Ø1  ARG   /fa         (fa is the format number, in
                         this case, 25)
FFFF                     (FFFF is the address of the
                         beginning of the format
                         statement.)
```

In our example, FFFF is the address of the first word of format statement 25. This is the location to be changed, so that it contains the address of the first word of array FMT. Therefore, compute the address where FFFF appears and change

its contents accordingly. (As a check on your computations, first compute the actual value FFFF, then load the address where FFFF appears, and see if the contents agree with your computation of FFFF. Then reload the address and deposit the address for the first word of FMT.)

Once this one location has been changed, save the core image on tape.

Altering Format Statements Already in Core

Format statements are stored 2 characters per word in packed, 6-bit ASCII code. Thus, the first half of the first word of a format statement is always a 50, the stripped ASCII code for open parenthesis. F2.0 takes 2 locations in storage, and appears as 0662 for F2, and 5660 for .Ø. Therefore, one can change all or any part of a format statement in core by determining its exact location and substituting the codes for the desired format. For any format statement that is likely to be changed in this manner, it is advisable to leave plenty of extra spaces when writing the program.