LBID-483

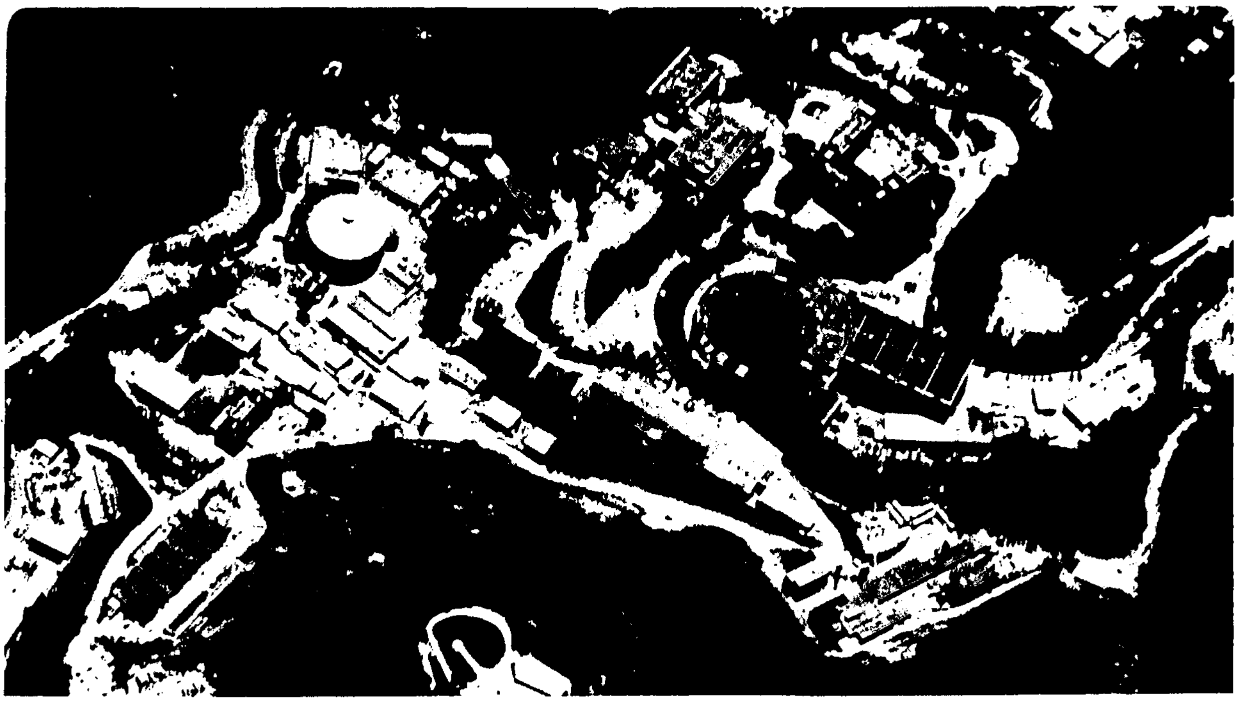
Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

Physics, Computer Science & Mathematics Division

CENFORM--CENSUS REPORT GENERATOR

R.N. Healey

February 1982



LBID-483 c.1

Prepared for the U.S. Department of Energy under Contract DE-AC03-76SF00098

LBID-483

C E N F O R M

Census Report Generator

by

R. N. Healey

Department of Computer Science and Mathematics
Lawrence Berkeley Laboratory
University of California
Berkeley, California 94720

February 1982

This work was supported in part by the Director Office of Energy Research, Office of Basic Energy Sciences, Engineering, Mathematical, and Geosciences Division of the U. S. Department of Energy under Contract No. DE-AC03-76SF00098.

LBID-483

C E N F O R M

Census Report Generator

(D R A F T - D R A F T - D R A F T)

R. N. Healey

February 24, 1982

Computer Science and Mathematics Department
Lawrence Berkeley Laboratory
University of California
Berkeley, California 94720

TABLE OF CONTENTS

Introduction

CENFORM is designed to generate reports from the 1980 Census data (or any other data that can be properly described within the confines of the program). Input includes specifications for designing a table and definitions of variables and calculations to fill the table. Designing the table may be accomplished by either using op-codes to describe the format or by entering an 'image' of the table or both.

Variables are described by either entering file name and variable name from the file, and/or by calculations involving the 'primative' variables.

Some other features include defining 'common' sections of text to be used from one table to the next, means for using a variable on more than one table, editing capabilities and direct input of an 'image' of a report.

Input, Output and Print Files

The program deals with up to three user files, an input file containing the input text and commands; an output file (optional) that will be used in the table generation program; a Print file (optional) for proof-reading of the tables. Usually the user will only have an input file and a print file for the first attempt. The proof can be checked and the input file is corrected by using a text editor such as 'ed'. When it is decided that all looks well, an output file may be specified and used as input to the generation program to acquire the final output with all the variables being replaced by data from the specified areas on the data files.

As stated above, an input file may be created using a text editor. Input may also be accepted directly from the terminal; however, the input is not saved and must be captured by editing the print file.

Input File Commands and Text .

First, a brief summary of the commands for the table--

```
newpage -- start a new table
endpage -- end of input for table

frame=<name> -- identify the table for use elsewhere

! <comment> -- everthing after ! is ignored

leftmarg=<l> -- set left margin (default is 2 )

rightmarg=<r> -- set right margin (default and limit is 130)

line<=><$><#><+><-><n> -- set to some line number and save
                          info or flag line

col=<+,-,n><c><l> -- set column number

tabset=nc#a#b#c#...# -- set tabs for columnar text/variables

text<u><b>=<$tab><string> -- enter text on line

var=<$tab>variable name -- enter variable(s)

edit<#col#skip>    --  enter text and variables to be edited in
endedit                 the generation program

roll<=nol><#skip=nsk> -- set to 'roll mode', i.e., nol areas
                         per page, skip a line after nsk lines.
endroll

newimage -- enter text to be 'image' and not edited
endimage

common=<name> -- enter text and commands to be used in current
endcom                table and in other tables
```

In addition, after the 'endpage' command, the following commands are used to define the variables entered in the page.

```
   define --  start variable definitions
   enddefine -- end of define

   file=<DDF> -- file containing the DE names of the variables

   frame   --  define variable used in othe tables
   endframe

   <var> = DE name -- name of variable on DDF file
```

```
<var> = #fra(frame,var) -- variable from another table

<var> = #equ(var) -- equivalence

<var> = #sum(v1,v2,...vn) -- sum of variables

<var> = #sub(v1,v2) -- subtract

<var> = #pct(v1,t1) -- percent

<var> = #date -- run date

<var> = #txt(v1,pa,pb) -- define part of string
                          v1(pa:pb)

<var> = #mul(v1,v2) -- multiply two values

<var> = #div(v1,v2) -- divide v1 by v2 (=0.0 if v2=0.0)

<var> = #con(kon) -- define a constant
```

Other ways of defining variables may be added as needed.

The above commands are used to create the input file. One or more tables may be entered by use of the "newpage" command.

```
newpage
frame = TAB1
   .
   .
   commands and text
   .
   .
endpage
define
   .
     variable definitions
   .
enddefine
newpage
frame = TAB2
   .
   .
endpage
define
  .
  .
enddefine
  .
   etc., etc., etc.,
```

A Closer Look at the Commands

newpage, endpage

Easy. This simply marks the beginning and end of text and command input.

frame=<ID>

This command identifies the current table so that variables used on other tables may be defined. E.g., if frame=TABLE1, and

```
   a1 = #pct(x,y)
```

is defined as a variable, then on another table we have

```
   z1 = #fra(TABLE1,a1)
```

will use the results from table 1 without having to re-calc- ulate.

leftmarg=<n>

This simply defines the lefthand margin of the table. The default is column 2.

rightmarg=<n>

This sets the righthand margin of the table. The default is column 130. The limit is also 130.

The left and right margins may be set and re-set at any point in the input.

line <=> <+,-n> <#> <$>

This command sets the current line number. 'line' by itself simply advances the line number by one.

line=n	sets the line pointer to line n
line=+n	puts the pointer on the current line plus n
line=-n	backs the point up n lines
line=#a#	where a is some identifier, advances the pointer by 1 and saves all commands to the next occurence of 'line'. The next time within the table that the same commands are needed, simply invoke line=#a# and all the saved commands will be repeated.
line=$flag	where 'flag' is some identifier, will advance the pointer by one and save the line number. The next occurence of 'line=$flag' will re-set the pointer to that line number.

col=<+,-n><c><l>

This command sets the pointer to some column number.

col=n	sets the pointer to column 'n'
col=-n	sets it to the current column minus 'n'
col=+n	sets it to the current column plus 'n'
col=c	sets the pointer to the center of the page as defined by 'leftmarg' and 'rightmarg'
col=l	set the pointer to 'leftmarg'. This is the default when the line pointer is reset.

tabset=nc#a#b#c#.....#n#

This command sets tabs at designated columns. This may be done in one of two ways. First, actual column numbers may be used, e.g.,

```
    tabset=4#l#35#65#95#r   ! set 4 tabs
```

will set four (nc=4) columns at

```
      left margin to 34
                  35 to 64
                  65 to 95
            and   95 to the right margin
```

notice the use of 'l' and 'r'. These two letters plus 'c' may be used to designate left margin, right margin and center of page.

The second method of setting tabs is to use the beginning and ending columns, e.g.,

```
    tabset=9#b=10#e=120#
```

will set nine (nc=9) columns beginning at 10 (b=10) and ending at 120 (e=120). All columns will be of equal width except, perhaps, the last one if the difference in the beginning and end is not an even multiple of the number of tabs.

Entering text and variables into the tabbed columns and justifying is done with the

```
                  text=$tab.....
     and          var=$tab......     commands.
```

text<b><u>=<$tabj>string

This command enters the text string into the page at the current line and column. E.g.,

```
   line=3
   col=c
   text=TABLE 1 -- SEX and RACE
```

will center (col=c) the string on line 3, and

```
   textb = TABLE 1 -- Sex and Race
```

will print in boldface and

```
   textu = TABLE 1 -- Sex and Race
```

will underline.

To input tabulated text, use the 'tabset' command above and then enter the text using

```
                    text=$tab#<string>#<string>#....etc
```

For example,

```
     tabset=3#b=6#e=65#
     textu=$tabc#NUMBER#MALE#FEMALE#
```

will set three tabs in columns 6-25, 26-45 and 46-65 and will center ($tab_c_) and underline (text_u_) the text.....

```
          NUMBER                MALE                FEMALE
```

The letters 'r', 'l' or 'c' may be used in the '$tab ' command to right, left or center the text in the columns.

Also, text may be repeated by the 'r=n' command within the 'text' command. For example,

```
       tabset=3#b=6#e=66#
       textu=$tabr#r=3#No.    Pct.#
```

will produce

```
                No....Pct.          No....Pct.          No....Pct.
```

The 'r=3' in the command repeats the string three times.

In the 'text=$tab' command, blank strings may be inserted by a blank between the separators,

```
     tabset=3#11#26#41#56#
     text=$tabc#Total# # #
     line
     text=$tabc#Persons#Male#Female#
```

will come out

```
             Total
            Persons          Male          Female
```

```
   var=<$tab>
   ----------
```

Two ways to enter variable positions are directly and by tabulating. For example,

```
     line
     col=5
     text=Run Date -
     col=+2
     var=date
     col=44
     text=Area -
```

```
    col=+2
    var=areaname
```

will produce the string

```
Run Date - [date]                      Area - [areaname]
```

These two variables are then defined later with the 'define' command.

The use of tabulated columns are similar to the 'text' command. For example,

```
    line
    tabset=4#b=6#e=66#
    textu=$tabr#Total#White#Black#Other#
    line
    line
    var=$tabr#t1#w1#b1#o1#
```

will result in

```
          Total          White          Black          Other

           [t1]           [w1]           [b1]           [o1]
```

edit and endedit

This command says that the following input is to be edited on the final output to fit within the columns specified. For example,

```
    line
    leftmarg=10
    rightmarg=60
    textu=Population
    line
    line
    edit
    The total population in the area in April 1980 was [p1]
    with a racial composition
    of Whites, [w1] ([wp1] percent);
       Blacks, [b1] ([bp1] percent);
    and Others, [o1] ([op1] percent).
    endedit
```

After running the data extraction, the text will look like...

```
      Population

      The total population of the area in April 1980 was
      24800 with a racial composition of Whites, 18305
      73.8 percent); Blacks, 4621 (18.6 percent); and
      Others, 1874 (7.6 percent).
```

That is, after calculating the values, the text will be edited within the specified columns (10-60).

Notice that the variables are expressed within the brackets. They must also be defined with the 'define' command after 'endpage'.
   The text may be columnized by using parameters in the edit command:

```
        edit#3#5
```

means
     edit the text into 3 columns with 5 spaces between the columns.

newimage and endimage

A table may be input as an 'image' of the output with these commands. Line, right and left margins may be specified. variable names must be put in brackets. For example,

```
  line=12
  leftmarg=10
  rightmarg=70
  newimage

                       White     Black    Asian    Other
                       -----     -----    -----    -----

  Total                 [w1]      [b1]     [a1]     [o1]

  College Grad.         [w2]      [b2]     [a2]     [o2]
  High School Grad.     [w3]      [b3]     [a3]     [o3]
  Drop Out              [w4]      [b4]     [a4]     [o4]

  endimage
```

The final output will be placed within columns 10 to 70 with the data values filled in. No other editing will be done.

roll and endroll

   Instead of printing a number of tables for each area, 'roll' mode allows one to print data for several areas on each page. The parameters for roll are number of areas per page and the number of areas between line skips, e.g.,

```
      roll=30#skip=5
```

means print 30 areas per page and skip a line after each 5 areas.

or roll=20#skip=1

means print 20 areas per page with a line skip between each area.

Text and variables may be entered as used above before the 'endroll' command.

common=<name> and endcom

Commands and text specified between 'common' and 'endcom' will be used in the current table but will be available for use in other tables as well. For example, table headings and footnotes which are used on many tables may be input once with these commands and used on all tables that required them.

```
common=headings
leftmarg=2
rightmarg=130
line=1
text=U. S. DEPARTMENT OF LABOR
line
text=EMPLOYMENT AND TRAINING ADMINISTRATION
line
text=RUN DATE -
col=+2
var=date
line
text=LAWRENCE BERKELEY LABORATORY
endcom
```

may be defined on the current table. Then, on another table invoke

```
common=headings
```

This will use all the commands defined above.

define and enddefine

After describing a table (after 'endpage'), 'define' is used to give the variables used on the table some definition. After defining the variables, 'enddefine' is used to complete the table.

The first step in defining variables is to define a file which contains a description of the data element names.

At this time, the only file usable in the program is a SEEDIS DDF file. DDF files are or will be available for the 1980 Census Data (some of the 1970 Census data files are available).

```
  file=<DDF file name>
```

Variables then may be defined by giving the DE names ('primatives') or by the use of calculations in terms of the primatives. Usually in the Census data, the DE names are by tabulation and line number, e.g., a value from tab 49, line 12 is specified as

```
      <var> = TAB49(12)
```

and area names are in the DDF as 'AREA.NAME'. Any variable not defined will appear in the output as

```
     <NA>
```

(not available). Any variable that is defined in terms of <NA> variables will produce an error message on the print file.

A range separator (;) may also be implemented for consecutive variables, i.e., instead of defining each variable, one may use, e.g., a1;a10 = tab10(11) for defining the ten consecutive primatives tab10(11) through tab10(20), and

```
     x1 = #sum(a1;a10)
```

in using calculation definitions. This command will add the ten variables, a1, a2, ..., a10.

Consecutive variables may also be used on the left side of the expression on the calculations #PCT, #SUB, #MUL and #DIV, i.e.,

```
      pc1;pc12 = #PCT(a1;a10,b1,c1,z1)
```

will calculate the 12 values as a percent of z1.

In addition, Primatives may be used in calculations without first defining them. For example,

```
  x1 = #sum(*tab10(11);10,*tab11(1),b1)
```

will sum the 10 consective vales from tab10, 11 to 20 with tab11, item 1 and b1.
Notice the asterisks (*) before the two primatives; these are important for without them they become undefined variables. Note also the range specifier (;) followed by the number of consecutive values.

The following is an example of defining variables; comments are included here with !'comment'.

```
   define
   file=sy$user:[seedis.census80]stf1.ddf  ! define DDF file
   date=#DATE                              !run date from system
```

```
areaname=#TXT(*area.name,1,30) !first 30 characters of area name
a1;a5 = tab12(01)               !5 primatives from tab12
t1 = #SUM(a1;a5)                !sum of 5 consecutive variables
pc1;pc5 = #pct(a1;a5,t1)        !5 consective percentages
frame=TAB4                      !getting variables
c1=pp1                          !defined on another table
c2=pp2
c3=pp3
c4=pp4
endframe                        !end of frame definition
d1=#fra(TAB3,pc1)               !another way to get variables
d2=#fra(TAB#,pc2)               !defined on another table
d3=#equ(t1)                     !simply equates d3 and t1
enddefine                       !end variable definition
```

this document is on sy$user:[healey]cenform.doc