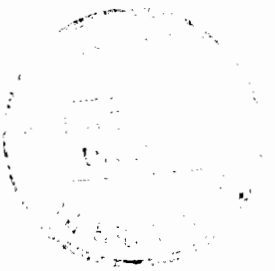
Serie D-110
LC/DEM/G.46



# CHECKEDIT SYSTEM

An interactive microcomputer program for editing and correction of demographic survey data

VERSION 1.00

Latin American Demographic Centre (CELADE)
Casilla 91 - Santiago - Chile

# CHECKEDIT SYSTEM

An Interactive Microcomputer Program for Editing and Correction of Demographic Survey Data

VERSION 1.00

Serie D, No. 110 Santiago-Chile Enero de 1987

# - INTRODUCTION

The system was developed at CELADE to CHECK a sequential file according to user specified rules, and manually EDIT its records on an interactive basis, producing an error-free file. It is a generalized program that works on an IBM/PC or a compatible hardware written in IBM PC Interpreted BASIC.

Initially the user has to establish a Dictionary consisting of four files: the characteristics of the file to be edited, a table look-up for the variables in the file, the errors to be verified, and the totals that should be recorded for statistical purposes. This is done using the dBASE III software, with user-oriented menus.

The system automatically generates a program in BASIC to check the variables for out-of-range values and for conditions based on which variables of the questionnaire are requested to have responses or not. The system also takes care of statistical reporting for error statistics, record I/O, and the "overhead" components of the program.

The user needs only to program, in BASIC, the specific rules for consistency checking between variables, and the structural checks that are dependent upon the questionnaire.

When executing the program, as errors are detected the record that caused the error, together with appropriate error messages, are displayed on the screen, giving the user the opportunity to correct the error. If it is an error that involves more than one record, the system has an option for printing the questionnaire identification and the error description to be analized and corrected later.

Very little knowledge of dBASE III is necessary (only how to work with the BROWSE command). Some previous experience in BASIC programming is required.

# II - USER'S GUIDE

The CHECK and EDIT system has four steps:

1. Create the Dictionary files.
2. Write the User specific error checking in BASIC.
3. Generate the Totalchk program.
4. Execute the Totalchk program.

The first two steps are executed interactively, until the dictionary files are compatible with the user needs and the BASIC program. The third step is executed only once (hopefully), to generate a program in BASIC that will be used in the fourth step as many times as desired.

## 2.1 Create the Dictionary files (DICT subsystem)

Here the user will define the four dictionary files (Variable, Error, Look-up and Filter) in accordance with the Input file and checks desired. Everything is done in dBASE III, using the menus provided by the subsystem. First of all, one has to enter the word DBASE when in DOS. After the prompt (.) in dBASE, enter "do checkedit".

At this moment, a general menu will appear asking which basic operations the user wants to be performed. This menu looks like:

```
            CHECKEDIT SYSTEM  -  MAIN MENU

      1 - UPDATE FILES
      2 - REPORTS
      3 - REBUILD FILE LINKS
      4 - GENERATE OUTPUT FILES

      E - EXIT TO DBASE III
      Q - QUIT TO DOS

      YOUR CHOICE
```

The only operation that needs a little knowledge of dBASE III is the first one (UPDATE FILES). It works with the BROWSE command, which has an online help using the [F1] on and off. It has a specific menu to select which file is to be updated, looking like:

OPTION 1 - FILE UPDATE

1 - UPDATE VARIABLE FILE
2 - UPDATE ERROR FILE
3 - UPDATE LOOK-UP FILE
4 - UPDATE FILTER FILE

E - EXIT

YOUR CHOICE

Choosing from 1 to 4 the user can update the four system files, and then select "E" to exit to the main menu. To add a new record to any file, position the cursor with the "down arrow" past the last record and the system will prompt whether you want to insert a new record.

For more details on how to use the BROWSE command, please refer to the dBASE III manual.

If the user selects the "E" option, the menu will bring back the dBASE III prompt (.), and he can work directly with it.

The "Q" option leaves dBASE III and returns to DOS. This option should be used when the user finishes his tasks at step 1.

Option "2" has a menu identical to option "1" but the header (REPORT OPTIONS now). The user selects one of the four files to be reported, and then the system asks for the media (Screen or Printer). After that, for some of the files, the user also may select a particular order of presentation. The reports are formatted for 80 lines per page, 8 lines per inch.

Option "3" (Rebuild File Links) is used to connect internally the system files, after the user has finished updating them. This step is needed to create the automatic error numbers in the variable file and to generate the variable limits in the look-up table. It should be exercised every time the Variable file, the Table Look-up file, and/or the Error file are updated.

Option "4" (Generate Output Files) should be executed after option "3", in order to generate an outside dBASE III image (DOS sequential file) of each of the system files. It has a menu to select the file to be written out, or a general option to write all of them. The option to select a specific file should be used when a small correction is made in only one file. The generated files have the same names as the original ones with the extention ".DAT", and are used later in step 4 (Execution of the TOTALCHK program).

Appendix "A" has an example of each one of the four system files.

## 2.2 Write the User-specific error checking in BASIC

The system automatically checks for out-of-range errors (for the variables having TYPEDIT = 1 or 2 in the Variables file). It also checks for the need to answer some variables depending upon any system filter specified (see examples in Appendix "A").

If there are some specific checks that the user wants to be performed on the records (and there always are), they have to be written in BASIC routines that can be clustered together in one module, to be merged with the system's main program. Basically, there are three different routines:

### 2.2.1 Structural checks

By structural checks we mean any error control that is done with variables from more than one record, where a record refers to the unit of observation in the survey. For example, the user wants to check that there is one and only one head of the household. Or that the marital status of the head is the same as specified for the spouse.

Generally, these errors are corrected looking at the questionnaire as a whole; for this reason the system prints them rather than displaying them on screen.

### 2.2.2 Specific checks

These are intrarecord checks, performed on variables of the same record. For example, the number of children alive should not be greater than the number of ever born children, or education has to be consistent with age. That is, they are the "consistency checks".

There are two differences between structural and specific checks: first, when creating the Error file in dBASE (see step 1 above), the variable called ERROR TYPE has the value of 4 for structure errors and 5 for specific errors. The second difference is implicit by the place, in the user program, where the check is done. If the BASIC instructions to verify the error are placed inside the structural check routine (i.e., between 12000 and 14999), it will be executed as an structural error.

### 2.2.3 End of questionnaire checks

There are some checks that can be performed only when the system detects a new questionnaire. For example, the number of persons stated in the household record has to be equal to the number of person records.

For each one of these types, the system provides an entry point (a BASIC instruction number), that will be performed by the main program as a subroutine, using the GOSUB statement. So, the user has to have a RETURN statement for each. These entry points are:

1. Structural checks - 12000
2. Specific checks - 15000
3. End of questionnaire - 17000

### 2.2.4 Remarks

1. The user program has to be between BASIC instructions numbers 11000 and 40000, and user defined variables should be initialized between 1200 and 1299. In order not to lose system and user variables, it is recommended that all user variables start with "U.".

2. If the program detects an error, it should assign the error number to a variable called NUMERR% and perform GOSUB 3070. This is the address of a system routine that will take care of the error. For example:

```
1320 IF NUMBADS% > 10 THEN NUMERR% = 21: GOSUB 3070
```

## 2.3 Generate the TOTALCHK program

After creating the screen files and writing the user checking routines, the third step is to generate the TOTALCHK BASIC program. This is the program that will be executed for each user file, and is composed of a merge of three parts:

1. The main program (CHECKEDIT.BAS). This program is provided with the system and should not need any change. However, if the need for a change arises, the user can do it by modifying the BASIC instructions at his or her own risk (keep a copy of the original!!).

2. The user program. This is the module written in the second step.

3. The support program (VARIAB.BAS). This module is generated by the system upon reading the files generated in step one. It contains three parts, one to provide the variable values for each variable name, one to produce the filters accounting, and one to flag the filters on and off to be checked later for each filtered variable.

This last module is generated automatically by the system and the user does not need to be concerned about it. The GENTOTAL procedure will produce the VARIAB.BAS file and the TOTALCHK.BAS program. This BAT also produces one screen file for each record type, which will be named SCREEN1.BAS, SCREEN2.BAS etc. to be used later by the EDIT routine.

First, the GENTOTAL procedure generates the VARIAB.BAS file. Then, it copies the CHEKEDIT.BAS file, VARIAB.BAS, and the user program, producing the TOTALCHK.BAS program. In order to do this the user must specify his program name (without the ".BAS" extension) to the GENTOTAL procedure.

```
A> GENTOTAL SAMPLE
```

where SAMPLE.BAS contains the User's own BASIC instructions.

After that, the procedure calls BASICA, which is the IBM/PC BASIC language interpreter. At this time, the user has to load the TOTALCHK program ([F3]Totalchk), save it in a compressed form with any name such as [F4]Totalchk, for example, and leave BASICA with the SYSTEM instruction. Do not use the names CHECKEDIT, CHEKEDIT nor SAMPLE, since they will erase either the main procedure or your program. Summarizing (inside BASICA):

```
LOAD "TOTALCHK
SAVE "TOTALCHK
SYSTEM
```

## 2.4 Execute the TOTALCHK program

This is the last step, and should be used for every data file. In order to simplify operations, the default drive and directory should be the same as that used for the CHECKEDIT system. For example, if you are using the CHECKEDIT disk in drive "A", be sure the DOS prompt is like "A>". If you are using a hard disk and loaded the CHECKEDIT files in a directory, position yourself inside this directory after the disk prompt. For example, "C> CD\CHECK" to change to the directory CHECK.

Then, write "BASICA TOTALCHK" in DOS mode to execute it.

### 2.4.1 Disk drive and directory

First, the program will ask for a disk drive (the default is the system drive). You can answer a disk drive, for example, "B:", or a directory where the file to be checked is, for example, "C:\SAMPDIR\" (Do not enter the double quotes).

### 2.4.2 Running option

Next, the program will ask for the running option. There are three: (1)-structural checks, (2)-consistency checks and (3)-both, a combination of the two.

In fact, the election of the running option is much more than that, because it decides also the way the file to be checked will be read. If we choose running option 1 (only structural checks), then the system will open the file as "Input" and the structural errors will be written in the LOG file (structural errors are never shown on screen). If there are no errors, CHECKEDIT automatically will produce a random file, copying the records from the input file and changing the file extension to ".FOR".

On the other hand, if running option 2 or 3 is selected, that is, consistency checks or structural AND consistency checks, the system will open the file to be CHECKEDITed as "Input/Output" ("R" in BASIC). The structural errors will still be written in the LOG file, and the consistency errors will be shown on the screen to be fixed interactively by the user. So, the selection of the running option will depend on the file type you start with.

#### 2.4.2.1 The initial file is sequential

In order for CHECKEDIT to work properly, displaying the consistency errors on the screen, it needs a "random file", hence, the very first CHECKEDIT execution MUST be with running option 1. The structural errors must be corrected out of CHECKEDIT (see item 2.4.6 below), and option 1 should be exercised repeatedly, untill there are no more structural errors left, at which time the system will produce automatically the random file.

#### 2.4.2.2 The initial file is already in a random organization

In this case, use either 2 or 3, but never option 1. 3. Our suggestion is to start by selecting option 3 (both checks) till there are no more structural errors left.

### 2.4.3 Data file

After that, the system will ask for the name of the data file to be edited. If the user chooses option 1, this file need to have the extension of ".DE", otherwise the file extension must be ".FOR". These extensions may be easily changed by modifying the first instructions of the CHEKEDIT main program.

### 2.4.4 Log file

Next, the system will ask for the name for the LOG file that will be created containing the system messages and reports. If the RETURN key is pressed, the system assumes the same name as the edited file plus a extension ".LOG".

### 2.4.5 Starting record

Finally, the system will ask for the starting record to be CHECKEDITed in the file (the default value is record number 1).

Every time the program detects an error, it takes one of two possible actions: it writes the error message and the questionnaire identification in the LOG file (if it is a structural error), or it shows the invalid record on the screen, together with the appropriate error message(s) (if they are specific errors).

### 2.4.6 Structural errors

In order to correct the structural errors, the user has to print (or TYPE) the corresponding LOG file, and then correct the errors by using any editing method he wishes, for instance, the same program used for data entry.

The reason for doing this is that generally a structural error produces several consistency errors that will be automatically fixed upon correcting the structural ones. Furthermore, a structural error can involve a need to create or delete one or more records, which the CHECKEDIT system does not allow. For example, suppose that one of the persons in a household was not entered, and this was detected by comparing the total number of persons in the household record against the number of person records actually encountered, producing an error message for this questionnaire. In order to fix this error the user has to insert the missing person's record into the file prior to running the CHECKEDIT system.

### 2.4.7 Consistency errors

These errors are showed on the screen, with a maximum of five errors per record. The variables are shown vertically in two columns, with their short name, long name, and contents. By moving the arrow keys the user can position the cursor at any variable and change its value, doing this as many times as it is needed. After that the user can ask the system to recheck the same record in order to garantee that there are no more errors, or can advance to the next record (if it was a trivial error).

There are also some Browse options to navigate through all the file's records and see (and modify, if needed) their contents. For a more detailed explanation of those options, please refer to the Appendix "C".

Appendix "C" is an example of a screen that will be shown to the user, with an invalid record. Appendix "D" has an example of a LOG file generated by applying the TOTALCHK program over a sample file.

# III - DICTIONARY SUBSYSTEM (DICT)

The dictionary part of the CHECKEDIT system is maintained and manipulated using only the dBASE III software.

Basically, the system has four files that have to be filled in by the user. They will contain control information to be used later by the CHEKEDIT program, such as the record description, look-up values for the variables, filter controls for totals, etc. These files are already defined in the DICT subsystem, the user has to invoke the dBASE on the PC and enter "do chekedit" at the prompt (.). The dBASE program executed is a menu-driven system of commands that can update or display the various files, establish file links (explained latter) and write out those files in a file format to be used by CHECKEDIT.

## FILE DESCRIPTION

### 3.1 VARIABLE file

#### 3.1.1 File definition

This file will contain all the records and variable definitions that appear on the user questionnaire. It is used to specify, for each variable of each record type, its starting point, length, type of editing and controlling total. Within dBASE, the following are the filename and index files used for indexing.

Filename = CONTVARI.
Indexed by INDTYPE and INDVARI.

#### 3.1.2 Fields

1. Variable - Variable Identification (4 characters).
   It contains the short name of the variable (which will be used to write the user BASIC routines). Generally it is composed of one letter followed by the variable number in the questionnaire. (For example, P12).

2. Typereg - Record type to which the variable belongs (1 character).
   The system supports up to 10 different record types (0-9). Record type 0 is used for variables common to all record types, like ID variables.

3. Namevar - Variable name (15 characters).
   It contains the long name of the variable, to be displayed in the system error messages.

4. Init - Variable starting point in the record (3).

5. Length - Variable length (2).

6. Typedit - Variable edit type (1).
   0 - none.
   1 - continuous range check.
   2 - non-continuous range check (table look-up). In this case the user has to inform the system about the possible values, using the TABLE LOOK-UP file.

7. Filter - Variable filter (2).
   The filter number that controls the variable and indicates when the variable should have specific values or blanks. For example, the Education variable has to be answered only by persons of 5 years or more, otherwise should be left blank. Filters are informed in the FILTER file. Variables that are not to be filtered have a ZERO in this field.

8. Initrange - Variable initial range (4).
   If the variable is to be range checked (Typedit = 1), it should contain the lowest legitimate value, otherwise a zero is assumed.

9. Finarange - Variable final range (4).
   If the variable is to be range checked, that is, Typedit = 1, it should contain the highest legitimate value, otherwise a zero is assumed.

10. Errorange - Error range number to be displayed (3).
    Automatically assigned by the system, if the variable is to be range or table look-up checked.

11. Errorfilt - Number of error filter to be displayed (3).
    Automatically assigned by the system, if the variable is filtered.

### 3.1.3 Remarks

The user has no control over fields 10 and 11. They exist and are displayed, but cannot be updated by the user.

Fields 8 and 9 are automatically filled by the system when the variable has Typedit = 2 (table look-up).

IMPORTANT: The very first variable of this file has to inform the system about general characteristics of the file to be CHECKEDITed. In order to do that, its fields have to have the following values:

1. Variable - A name that will be alphabetically first, compared with the other variable names (AAA, for example).

2. Typereg - Zero.

3. "Init" - must contain the position of the record type in the records (for example, a 1 if the first character of each record is the record type).

4. Length - The record length (it is mandatory that all records have the same length).

5. "Typedit" - It should contain the starting position of the questionnaire identification in the records (this id has to be at the same place in all records).

6. "Filter" - The questionnaire ID's length (as a system limitation, those variables belonging to the ID must be placed together in the records).

## 3.2 ERROR file

### 3.2.1 File definition

This file will contain all the user defined errors that will be specifically programmed in BASIC.

Filename = CONTERR.
Indexed by INDERR and INDERRTY.

### 3.2.2 Fields

1. Error - Error identification (3).
   It contains the error number that will be used when writing the user defined checks, identifying the error for editing and statistical purposes. It has to be unique for the whole system.

2. Errtype - Error type (1).
   There are two different types of errors:

   4 - Structural checks (for example, no Household Head is identifyed, that is, an error found trough an interrecord check). This kind of error will not be displayed on the screen, but printed acompanied by the corresponding questionnaire ID.

   5 - Specific checks (an error that involves more than one variable of the same record). These errors will be displayed on the screen.

3. Totalref - Referring total (2).
   The universe of cases that should be counted in order to obtain the relative percentage of questionnaires with this error. It can be a record type or a filter number (see FILTER file later in this chapter), depending upon the variables involved in the error checking.

4. Name - Error description (45).
   Error description to be printed or displayed.

## 3.3 LOOK-UP file

### 3.3.1 File definition

This file will contain the accepted values for those variables with "TYPEDIT" equal 2 in the VARIABLE file, that is, the variables with non-continuous range checks that use table look-up checking. The values can be informed one by one or in pairs of range values (minimun and maximun values for each set).

This file has four fields, the variable field, one value field, a type field to qualify this value field, and a position field (internal to the system). Each accepted value for the specific variable has to be given, with the variable name and the accepted value.

Filename = CONTVALU.
Indexed by INDVALU.

### 3.3.2 Fields

1. Variable - Variable name (4).

2. Type - Value type (1).

   1 - The value is a range limit (lower or upper).
   2 - The value is a unique value to be checked.

3. Value - One of the accepted values for the variable (4).

4. Position - Relative position of the value in the file (4).

### 3.3.3 Remarks

1. It is the user responsability to guarantee that it will have always an even number of type "1" values (a lower and an upper limit). There can be more than one pair of range limits for a variable (or none).

2. The user has no control over the Position field. It is used only for internal purposes.

3. The user does not need to be concerned about the order in which he enters the values for each variable. The system takes care of ordering them by variable (major order), type and value (minor order).

4. For example, if a variable called V08 has the values from 1 to 20 or 99, it should have three entries in the LOOK-UP file. If another variable, say P10, can have the values 1 to 9, 12 to 18, or 25, it should have five entries in the file. The table below is the representation of how the LOOK-UP file should look with these two variables:

| Variable | Type | Value |
|---|---|---|
| V08 | 1 | 1 |
| V08 | 1 | 20 |
| V08 | 2 | 99 |
| P10 | 1 | 1 |

| | | |
|---|---|---|
| P10 | 1 | 9 |
| P10 | 1 | 12 |
| P10 | 1 | 18 |
| P10 | 2 | 25 |

## 3.4 FILTER file

### 3.4.1 File definition

This file will contain the "filter" information for each universe of cases that should be counted separatedly. First of all, each record type is a filter in itself, that is, the system counts how many times each record type was processed, so they do not need to be indicated as filters.

A filter has to do with the structure of each record type. For example, a record containing information about persons may have fields that are answered only by the women older than 14 years. This is a filter (women over than 14 years), and the variables answered in this part of the questionnaire should be "filtered" by it. In that sense, we will have two separate sets of persons; the answering set and the non-answering one. The latter (all men plus women 14 years old or less) should have blank values for these variables. At execution time, the system will check these conditions automatically, for each variable having a filter.

Filename = CONTFILT.
Indexed by INDFILTE.

### 3.4.2 Fields

1. Filt - Filter identification (2).
   A number from 1 to 99, but different from all the record types (because each record type is itself a filter).

2. Type - Record type (1).
   The record type this filter is associated with. It must be informed just in Part 1 (see following field) for each filter, if there is more than one part.

3. Part - Filter part (1).
   A filter information can have up to 9 parts, depending upon the complexity of its specification. These parts have to be in a sequential order starting with 1.

4. Description - Filter description (40).
   A free-form description of the filter, that will be printed at the end of the program together with their respective totals. It must be provided just in Part 1 for each filter.

5. - Fld1 to Fld4 - Filter operands (4).
   These fields have the information to compose the filter expression in BASIC. Each field can have the following elements:

- variable name - it can be a system variable (defined in the VARIABLE file) or a user variable.
- constant - any numeric constant.
- operator - any logical operator (=, <>, >, <, >= and <=).
- connector - AND and/or OR.
- parenthesis - ( or ).

### 3.4.3 Remarks

With 9 parts and 4 fields (Fld1 to Fld4) in each part, the user can construct an expression with up to 36 symbols (operands).

A filter has to have at least 3 operands.

The expression is evaluated from left to right, with no order of precedence. The parenthesis will serve to establish an order of operations, if desirable.

# IV - CHECK AND EDIT SUBSYSTEM (CHECK)

The CHECK and EDIT subsystem is the main part of the CHECKEDIT system. It is composed of several BASIC modules, combined together in order to provide a powerful tool to verify the consistency of records of a file, and change its values interactively.

## 4.1 Input files

### 4.1.1 Control files

These are the files generated by the DICT subsystem.

4.1.1.1 Variable file (CONTVARI.DAT).
4.1.1.2 Error file (CONTERR.DAT).
4.1.1.3 Look-up file (CONTVALU.DAT).
4.1.1.4 Filter file (CONTFILT.DAT).

### 4.1.2 Screen files

These are the files generated by the GENTOTAL procedure through the GENBAS.BAS program (see item 2.3 of the User's Guide). There is one file for each existing record type, and they are called "SCREENx.BAS", where x is the record type number.

### 4.1.3 File to be edited

This is the file containing the questionnaires to be edited if the user started with running option 1 (see item 2.4.2 of the User's Guide). This means that the system will work with a sequential file, and after there are no structural errors, it will automatically create a random file for I/O (see file 3 below).

## 4.2 Output files

There is only one output file, which is the LOG file. It contains the following information:

### 4.2.1 Invalid questionnaires

The identification of each of the questionnaires that had at least one structural error in one of its records. The identification is printed, together with the record number that produced the error, and an accompanying error message.

### 4.2.2 Execution Diary

This has information about the program run, like date and time, names of the checked and log files, and the number of records read, edited and written, per record type. It prints also the program option used

(structural check, consistency check or both), the starting record number and the final record number.

### 4.2.3 Statistics Report

It prints the error occurences of the run, that is, how many times each error appeared in the file records, with absolute and percentage values. The errors are separated by automatic and user errors. Finaly, this part prints also the totals from which the system calculated the error percentages. They are the record type counts and the filter counts.

There is also a column named "complement", containing the diference between the total number of cases and the filtered number of cases. For example, if a filter specifies the persons greater than 10 years old, the complement is the number of cases of persons less than or equal to 10 years old.

## 4.3 Input/Output files

There is only one input/output file, which is the file to be checked and edited. It has to be a random file, without any "carriage return" or "line feed" after each record. Its definition is specified in the Variable file of the dictionary subsystem (DICT).

## 4.4 Program description

The CHEKEDIT program is divided into program routines, each one of them with a specific function. They are:

SET UP module.

- asks for : disk drive.
  running option.
  filename to be edited.
  filename of the log file.
  starting record.
- loads CONTROL files (Variable, Error, Look-up and Filter).

PROCESS module.

- error checks : structural (user)
  filter and range (automatic)
  consistency (user)
- routines : input
  output
  accounting errors
  edit (display records and accepts changes)

CLEAN UP module.

- calls the "end of questionnaire" routine
- prints diary

- calculates and prints error statistics
- prints totals
- close files

4.4.1 SET UP module

The default for the disk drive is the system default drive. The filename to be edited has an automatic extension of ".DE" if the running option is 1 (only structural checks). If not, the extension is ".FOR". The log filename has an automatic extention of ".LOG" and its default is the name of the file to be edited. Of course, all of these can be changed easily by the user.

The running option is: Structural checks (reads a sequential file and errors are printed in the log file), Consistency checks (reads and writes a random file and errors are displayed for correction), or both (Structural errors are printed and Consistency errors are displayed, also reading and writing a random file).

The starting record number is specified if the user wants to check and edit his file by parts, or if he did not finish editing it during the last session and wants to resume editing from where he stopped. The default is 1.

The Esc (Escape) key can be used any time, and it has the function of going back to the previous option. The Backspace key can be used to correct any typing error the user makes when choosing the options and filenames.

The loading of the control files is performed one by one, reading them from the system drive with their respective names of CONTVARI.DAT, CONTERR.DAT, CONTVALU.DAT and CONTFILT.DAT. They are loaded into program tables defined at the beginning. For their sizes, see Appendix "E" - System Specifications and Limits.

After loading the Control files, the program stablishes the running variables like LASTREC (last file record), IDST (starting point of the questionnaire identification), and IDLEN (id's length).

4.4.2 PROCESS module

The Structural check is a gosub 12000 (user routine), depending upon the running option. The filter and range checks are done automatically for each variable, if the running option specifies so. The Consistency check is a gosub 15000 (user routine).

Every time the program detects an error, it calls the Accounting Errors routine to update the error table.

After finishing checking the record, or if the number of encountered errors is greater than 5, the program calls the Edit routine, which displays the record on the screen and accepts user corrections for each field. This routine loads from the system drive the specific screen format of the record type to be displayed. The return to the main program is done using three function keys (F1, F9 and F10). F1 is to stop running the program (after

writing the displayed record). F9 is to recheck the same record, and F10 is to check the next record (after writing the displayed one).

The corrections to the displayed record variables are done by using the arrow keys to position the cursor at the desired field and then entering the new value. Typing errors can be corrected by using the Backspace key or by retyping the whole field. There are also other keys that can be used, like the Home, End, PgUp, PgDn and Esc keys (the first four ones have different meanings if pressed together with the Ctrl key). Basically they serve to Browse back and forth through the file, displaying records. The system accepts corrections for any displayed record, but they will not be checked unless the F9 key is selected, that is, an order to recheck the record. For a more detailed explanation on the use of the program keys, please see Appendix "C".

The input and output routines work as their names say, reading and writing one record at a time, and counting them. The input routine also establishes the KEYID$ variable (quest. id.) and loads each variable of the specific record type. It also adds the proper filters counts.

### 4.4.3 CLEAN UP module

There is an entry point for an "end of questionnaire" user written routine (17000), for errors that can only be detected after the last record of the questionnaire. This routine is called by the user at any time, and by the program in this module.

The diary of the program run is printed on the LOG file, relative error results are calculated and printed on the same file, together with the filter totals.

# Appendix A - DICT files

The following pages have an example of reports from each one of the system files created by the user in dBASE III.

## A.1 Variable file

This report is sorted by record type and was printed after the execution of the "Rebuild file links" step. This example shows the definition of an user file with four record types (1, 2, 5 and 6). Type one contains the household information, type two refers to the persons in the household, type five is used for the people that died recently, and type six has information about the persons living abroad. Record type "zero" does not appear in the file, but is used to store the information about the variables that are common to all record types, for example, the record type variable itself (I01) and the identification variables (I02 to I06).

All variables shown have a starting position in the record and their respective length. The "AAA" variable is a system requirement to store general information about the user file (please, see "DICT subsystem, item 3.1.3 - Remarks"). In this example, looking at the AAA variable, we can see that:

.the record type is at column 1 in the user file ("init" = 1).
.the record size is 60 ("length" = 60).
.the questionnaire ID starts at column 2 ("edittype" = 2), and is 11 bytes long ("filter" = 11).

For the other variables, we can see, for example:

.variable V01, ROOMS, is two bytes long starting at position 13, and is to be range checked ("edittype" = 1), with limits of 0 and 15. It is not to be filtered, that is, it has to have a valid value for all questionnaires. Those fields were specified by the user, and the system included automatically the error range number (66).

.variable P16, MARITALST, the person's marital status, is one byte long at position 34, is to be checked by table look-up values ("edittype" = 2), and is conditioned by filter number 8 ("filter" = 8). These are the fields that the user had to fill in, the others are produced automatically by the system.

The table look-up values for this variable must be entered in the Look-up file, one by one. In this example (see the Table Look-up values report below), the user indicated eight possible values for P16, starting at the 28th position in the file. This is reflected automatically in the Variable file when the system executes the Rebuild procedure, writing the Initial (28) and Final (8) range fields, to be used by the CHEKEDIT program. The fields Error Range (44) and Error Filter (45) were also recorded by the system. These are the numbers that will be shown on the screen if the variable is out of range (error 44) or should not be answered (error 45) because of the filter condition.

Filter equal 8 means that P16 (Marital Status) will be controlled by a condition numbered 8. So, there has to be a filter identified as such in the Filter file. If we look at the Filter file report, explained below, we can see that there is a filter 8 belonging to record type 2 (persons), and specifying persons greater or equal to 10 years old (P03 >= 10). That is, variable P16 was to be answered only by persons with 10 or more years of age, being blank otherwise.

## A.2 Error file

This report is sorted by error type, with the Structural errors (type = 4) first, and then the Consistency ones (type = 5). These are the only errors that should be programmed by the user (see Appendix "B"). They do not cover all the possible checks one can make on the records described in the Variable file, but just some of the most common ones. For example, error 14 is a Structural error referring to total number 1 (type 1 records, i.e., households). If it occurs, it will mean that the first person of the household is not the head. This error is checked at line 13300 of the user program (see Appendix "B").

Error 22 is a consistency check (type = 5) between the number of children ever born and the woman's age. It refers to total 10 (filter number 10 in the Filter file selects the women of 14 years old or more).

## A.3 Look-up file

This report shows the valid values for each variable that has a table look-up check. For example, the variable P06 (Religion) can have only the values 1, 2 or 9.

## A.4 Filter file

This report shows the filters to be executed for each record type. For example, filter 9 is to be applied to the type 2 records (persons), and is composed of four parts. It specifies economic active persons, that is, people greater or equal to 10 years old and with ecconomic status from 1 to 4.

```
P03 >= 10 AND (P18 >= 1 AND P18 <= 4)
```

VARIABLE FILE

(SORTED BY RECORD TYPE)

| VARIABLE | REC TYPE | NAME | INIT | LENGTH | EDIT TYPE | INITIAL RANGE | FINAL RANGE | ERROR RANGE | FILTER | ERROR FILTER |
|---|---|---|---|---|---|---|---|---|---|---|
| ** RECORD TYPE 0 | | | | | | | | | | |
| AAA | 0 | TOTAL RECORD | 1 | 60 | 2 | 0 | 0 | 0 | 11 | 0 |
| I01 | 0 | TYPEREC | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| I02 | 0 | REGION | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| I03 | 0 | MA | 4 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| I05 | 0 | EDNUMBER | 6 | 3 | 0 | 0 | 0 | 0 | 0 | 0 |
| I04 | 0 | URBAN/RURAL | 9 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| I06 | 0 | QUESTIONN | 10 | 3 | 0 | 0 | 0 | 0 | 0 | 0 |
| ** RECORD TYPE 1 | | | | | | | | | | |
| V01 | 1 | ROOMS | 13 | 2 | 1 | 0 | 15 | 66 | 0 | 0 |
| V02 | 1 | BEDROOMS | 15 | 2 | 1 | 0 | 10 | 67 | 0 | 0 |
| V03 | 1 | WATER | 17 | 1 | 1 | 1 | 7 | 68 | 0 | 0 |
| V04 | 1 | SANITARY | 18 | 1 | 1 | 1 | 5 | 69 | 0 | 0 |
| V05 | 1 | ELECTRICITY | 19 | 1 | 1 | 1 | 2 | 70 | 0 | 0 |
| V06 | 1 | DEADS | 20 | 1 | 1 | 0 | 9 | 71 | 0 | 0 |
| V07 | 1 | ABROADS | 21 | 1 | 1 | 0 | 9 | 72 | 0 | 0 |
| V08 | 1 | NUMPERSONS | 22 | 2 | 1 | 0 | 99 | 73 | 0 | 0 |
| ** RECORD TYPE 2 | | | | | | | | | | |
| P01 | 2 | PERSONUMB | 13 | 2 | 1 | 1 | 99 | 22 | 0 | 0 |
| P02 | 2 | RELATIONSHIP | 15 | 1 | 2 | 11 | 3 | 23 | 0 | 0 |
| P03 | 2 | AGE | 16 | 2 | 1 | 0 | 99 | 24 | 0 | 0 |
| P04 | 2 | SEX | 18 | 1 | 1 | 1 | 2 | 25 | 0 | 0 |
| P05 | 2 | RACE | 19 | 1 | 2 | 14 | 3 | 26 | 0 | 0 |
| P06 | 2 | RELIGION | 20 | 1 | 2 | 17 | 3 | 27 | 0 | 0 |
| P07 | 2 | NACREGION | 21 | 2 | 1 | 0 | 20 | 28 | 0 | 0 |
| P08 | 2 | NACCOUNTRY | 23 | 1 | 1 | 1 | 9 | 29 | 12 | 30 |
| P09 | 2 | YEARRIVAL | 24 | 2 | 1 | 12 | 86 | 31 | 12 | 32 |
| P10 | 2 | MOTHERALIVE | 26 | 1 | 2 | 22 | 3 | 33 | 0 | 0 |
| P11 | 2 | LIVGUYA | 27 | 1 | 2 | 25 | 3 | 34 | 13 | 35 |
| P12 | 2 | YEARDEATHMO | 28 | 2 | 1 | 12 | 86 | 36 | 14 | 37 |
| P13 | 2 | REGION5AGO | 30 | 2 | 1 | 0 | 20 | 38 | 7 | 39 |
| P14 | 2 | COUNTRY5AGO | 32 | 1 | 1 | 1 | 9 | 40 | 15 | 41 |
| P15 | 2 | EDUCATION | 33 | 1 | 1 | 1 | 9 | 42 | 7 | 43 |
| P16 | 2 | MARITALST | 34 | 1 | 2 | 28 | 8 | 44 | 8 | 45 |
| P17 | 2 | WIDOWHOOD | 35 | 1 | 2 | 36 | 3 | 46 | 8 | 47 |
| P18 | 2 | ECCONOMIC | 36 | 1 | 1 | 1 | 9 | 48 | 8 | 49 |
| P19 | 2 | OCCUPATION | 37 | 2 | 2 | 39 | 7 | 50 | 9 | 51 |
| P20 | 2 | INDUSTRY | 39 | 1 | 1 | 1 | 8 | 52 | 9 | 53 |
| P21 | 2 | OCCSTATUS | 40 | 1 | 1 | 1 | 5 | 54 | 9 | 55 |
| P22 | 2 | EVER BORN | 41 | 2 | 1 | 0 | 15 | 56 | 10 | 57 |
| P23 | 2 | ALIVE | 43 | 2 | 1 | 0 | 10 | 58 | 11 | 59 |
| P24 | 2 | CHILDABROAD | 45 | 2 | 1 | 0 | 10 | 60 | 11 | 61 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| P25A | 2 | MONTHLAST | 47 | 2 | 1 | 1 | 12 | 62 | 11 | 63 |
| P25B | 2 | YEARLAST | 49 | 2 | 1 | 12 | 86 | 64 | 11 | 65 |
| ** RECORD TYPE 5 | | | | | | | | | | |
| M00 | 5 | DEADNUMBER | 13 | 2 | 1 | 1 | 9 | 16 | 0 | 0 |
| M01A | 5 | DAYOFDEATH | 15 | 2 | 1 | 1 | 31 | 17 | 0 | 0 |
| M01B | 5 | MONTHDEATH | 17 | 2 | 1 | 1 | 12 | 18 | 0 | 0 |
| M01C | 5 | YEARDEATH | 19 | 2 | 1 | 83 | 86 | 19 | 0 | 0 |
| M02 | 5 | SEX | 21 | 1 | 1 | 1 | 2 | 20 | 0 | 0 |
| M03 | 5 | AGE | 22 | 2 | 1 | 0 | 99 | 21 | 0 | 0 |
| ** RECORD TYPE 6 | | | | | | | | | | |
| A00 | 6 | ABROADNUMB | 13 | 2 | 1 | 1 | 9 | 1 | 0 | 0 |
| A01 | 6 | RELATIONSHIP | 15 | 1 | 1 | 1 | 8 | 2 | 0 | 0 |
| A02 | 6 | SEX | 16 | 1 | 1 | 1 | 2 | 3 | 0 | 0 |
| A03 | 6 | YEARDEPART | 17 | 2 | 1 | 10 | 86 | 4 | 0 | 0 |
| A04 | 6 | AGEDEPART | 19 | 2 | 1 | 5 | 50 | 5 | 0 | 0 |
| A05 | 6 | COUNTRY | 21 | 2 | 1 | 1 | 9 | 6 | 0 | 0 |
| A06 | 6 | REASON | 23 | 1 | 1 | 0 | 5 | 7 | 0 | 0 |
| A07 | 6 | EDUCATION | 24 | 1 | 1 | 0 | 9 | 8 | 16 | 9 |
| A08 | 6 | OCCUPATION | 25 | 2 | 2 | 1 | 10 | 10 | 17 | 11 |
| A09 | 6 | INDUSTRY | 27 | 1 | 1 | 0 | 9 | 12 | 17 | 13 |

Page No. 1
05/09/86

ERROR FILE

(SORTED BY ERROR TYPE)

| ERROR | ERROR TYPE | TOTAL REFER | ERROR DESCRIPTION |
|---|---|---|---|
| ** ERROR | TYPE 4 | | |
| 1 | 4 | 1 | NUMB DEATHS DONT CORRS WITH TOTAL |
| 2 | 4 | 5 | DEATHS NOT IN SEQUENCE |
| 6 | 4 | 1 | NUMBER ABROAD DONT CORRS WITH TOTAL |
| 7 | 4 | 6 | NUMBER ABROAD NOT IN SEQUENCE |
| 8 | 4 | 1 | MORE THAN ONE HOUSEHOLD RECORD |
| 9 | 4 | 1 | THERE IS NO HOUSEHOLD RECORD |
| 11 | 4 | 1 | NUMB PERSONS DONT CORRS WITH TOTAL |
| 12 | 4 | 2 | NUMBER PERSONS NOT IN SEQUENCE |
| 13 | 4 | 1 | MORE THAN ONE HEAD OF HOUSEHOLD |
| 14 | 4 | 1 | FIRST PERSON IS NOT THE HEAD |
| 15 | 4 | 1 | MORE THAN ONE SPOUSE |
| 16 | 4 | 1 | HEAD SEX INCONSIST SPOUSE SEX |
| 21 | 4 | 8 | HEAD AND WIFE MARITAL ST INCONSIST |
| ** ERROR | TYPE 5 | | |
| 10 | 5 | 1 | BEDROOMS INCONSIST ROOMS |
| 20 | 5 | 7 | EDUCATION INCONSIST AGE |
| 21 | 5 | 8 | OCCUPATION SHOULD NOT BE ANSWERED |
| 22 | 5 | 10 | EVER BORN INCONSIST AGE |
| 24 | 5 | 11 | ALIVE INCONSIST EVER BORN |

Page No. 1
05/13/86

TABLE LOOK-UP VALUES

| POSITION | VARIABLE | TYPE | VALUE |
|---|---|---|---|
| 1 | A08 | 2 | 0 |
| 2 | A08 | 2 | 1 |
| 3 | A08 | 2 | 2 |
| 4 | A08 | 2 | 10 |
| 5 | A08 | 2 | 11 |
| 6 | A08 | 2 | 22 |
| 7 | A08 | 2 | 33 |
| 8 | A08 | 2 | 55 |
| 9 | A08 | 2 | 88 |
| 10 | A08 | 2 | 99 |
| 11 | P02 | 1 | 1 |
| 12 | P02 | 1 | 7 |
| 13 | P02 | 2 | 9 |
| 14 | P05 | 1 | 1 |
| 15 | P05 | 1 | 5 |
| 16 | P05 | 2 | 9 |
| 17 | P06 | 2 | 1 |
| 18 | P06 | 2 | 2 |
| 19 | P06 | 2 | 9 |
| 20 | P08 | 2 | 1 |
| 21 | P08 | 2 | 9 |
| 22 | P10 | 2 | 1 |
| 23 | P10 | 2 | 2 |
| 24 | P10 | 2 | 9 |
| 25 | P11 | 2 | 1 |
| 26 | P11 | 2 | 2 |
| 27 | P11 | 2 | 9 |
| 28 | P16 | 2 | 1 |
| 29 | P16 | 2 | 2 |
| 30 | P16 | 2 | 3 |
| 31 | P16 | 2 | 4 |
| 32 | P16 | 2 | 5 |
| 33 | P16 | 2 | 6 |
| 34 | P16 | 2 | 7 |
| 35 | P16 | 2 | 9 |
| 36 | P17 | 2 | 1 |
| 37 | P17 | 2 | 2 |
| 38 | P17 | 2 | 9 |
| 39 | P19 | 2 | 0 |
| 40 | P19 | 2 | 1 |
| 41 | P19 | 2 | 11 |
| 42 | P19 | 2 | 22 |
| 43 | P19 | 2 | 33 |
| 44 | P19 | 2 | 88 |
| 45 | P19 | 2 | 99 |

Page No. 1
05/09/86

FILTER FILE

```
FL T P FLD1 FLD2 FLD3 FLD4 DESCRIPTION


** RECORD TYPE 2
  7 2 1 P03  >=   5          PERSONS >= 5 YEAR OLD
  8 2 1 P03  >=   10         PERSONS >= 10 YEARS OLD
  9 2 1 P03  >=   10   AND   ECONOMIC ACTIVE PERSONS
  9 2 2 (    P18  >=   1
  9 2 3 AND  P18  <=   4
  9 2 4 )
 10 2 1 P04  =    2    AND   WOMEN >= 14 YEARS OLD
 10 2 2 P03  >=   14
 11 2 1 P22  >    0          WOMEN WITH CHILDREN
 12 2 1 P07  =    0          BORN ABROAD
 13 2 1 P10  =    1          MOTHER ALIVE
 14 2 1 P10  =    2          MOTHER DEAD
 15 2 1 P03  >=   5    AND   PERSONS >= 5 YEARS OLD  BORN ABROAD
 15 2 2 P13  =    0

** RECORD TYPE 6
 16 6 1 A04  >=   5          EDUCATION ONLY FOR OLDER 5 YEARS
 17 6 1 A04   >=  10         OCCUPATION AND INDUSTRY ONLY FOR 10 OLDER
```

## Appendix B - User program

The following program is an example of how to write the BASIC instructions to trap the errors specified in the Error file in Appendix "A". There are some points that should be emphasized:

B.1 User's data definition

The first three instructions define and load a table (U.EDTABLE) that will be used when checking education versus age (line 15250). These and other user data should be defined between 1200 and 1299.

B.2 Entry points

Instructions 12000, 15000 and 17000 are entry points to the Structural checks, Specific checks and End of questionnaire routines, respectively.

Upon detecting any error (instruction 12500 for example), the user assigns the error number to the variable NUMERR% and performs the accounting error routine through the entry point 3070.

B.3 Return to main program

After each block of errors, the control should go back to the main program, using the RETURN statement (in 15400, for example).

B.4 Error checking for each record type

After entering the Structural checks routine (or the Specific checks routine), the program should have the ON TYPEREC GOTO a, b, etc (see 12250), to execute the routine depending on the record type.

B.5 Structural checks. For example:

B.5.1 Instruction 12200 is executed always, independently of the record type.

The called routine (12300) is used to check the fields (variables) common to all record types.

B.5.2 Instructions 12650 through 12750 initialize some user variables at the beginning of each questionnaire.

B.5.3 Instruction 12900 checks for more than one head of the household (if flag U.HEAD is already "on").

B.5.4 Instruction 13350 counts persons records. This count will be checked against the total informed in the household record (see instruction 17300).

```
1200 DIM U.EDTABLE(9)
1220 FOR U.I = 1 TO 9: READ U.EDTABLE(U.I): NEXT U.I
1240 DATA 0,0,12,17,17,17,22,0,0
12000 ' structural checks
12200 GOSUB 12300
12250 ON TYPEREC GOTO 12450,12800,0,0,13450,13650
12300 ' *** checks for common fields
12350 IF KEYID$<>U.CLAVE$ AND TYPEREC<>1 THEN NUMERR%=9: GOSUB 3070
12400 U.CLAVE$=KEYID$: RETURN
12450 ' *** structural checks for rectype 1
12500 IF U.CLAVE$ = KEYID$ THEN NUMERR% = 8: GOSUB 3070
12550 GOSUB 17000     ' end of quest routine
12650 U.V06TOT=0: U.V07TOT=0: U.V08TOT=0: U.SPOUSE=0: U.HEAD=0
12670 U.P01ANT=0: U.M00ANT=0: U.A00ANT=0: U.V08ANT=V08
12700 U.HEADSEX = 0: U.HEADST = 0: U.V06ANT=V06: U.V07ANT=V07
12750 U.CLAVE$ = KEYID$: RETURN
12800 ' *** structural checks for rectype 2
12850 IF P02 <> 1 THEN 13000
12900   IF U.HEAD = 1 THEN NUMERR%=13: GOSUB 3070: GOTO 13000
12950   U.HEAD = 1: U.HEADSEX = P04: U.HEADST = P16
13000 IF P02 <> 2 THEN 13250
13050   IF U.SPOUSE = 1 THEN NUMERR% = 15: GOSUB 3070: GOTO 13250
13100   U.SPOUSE = 1
13150   IF U.HEADSEX = P04 THEN NUMERR% = 16: GOSUB 3070
13200   IF U.HEADST <> P16 THEN NUMERR% = 21: GOSUB 3070
13250 IF P01 <> U.P01ANT + 1 THEN NUMERR% = 12: GOSUB 3070 ELSE U.P01ANT=P01
13300 IF P01 = 1 AND P02 <> 1 THEN NUMERR% = 14: GOSUB 3070
13350 U.V08TOT = U.V08TOT + 1: RETURN
13450 ' *** structural checks for rectype 5
13500 IF M00 <> U.M00ANT+1 THEN NUMERR% = 2: GOSUB 3070 ELSE U.M00ANT=M00
13550 U.V06TOT = U.V06TOT + 1: RETURN
13650 ' *** structural checks for rectype 6
13700 IF A00 <> U.A00ANT+1 THEN NUMERR% = 7: GOSUB 3070 ELSE U.A00ANT = A00
13750 U.V07TOT = U.V07TOT + 1: RETURN
15000 ON TYPEREC GOTO 15100, 15200    ' *** specific checks
15075 RETURN
15100 IF V02 > V01 THEN NUMERR% = 10: GOSUB 3070  '*** specific for type 1
15150 RETURN
15200 ' *** specific checks for record type 2
15250 IF P03 < U.EDTABLE(P15) THEN NUMERR% = 20: GOSUB 3070
15300 IF P23 > P22 THEN NUMERR% = 24: GOSUB 3070
15350 IF P03 <= 14 OR P04 = 2 THEN 15400
15375    IF P03 < P23 + 12 THEN NUMERR% = 22: GOSUB 3070
15400 RETURN
17000 ' end of questionnaire
17100 IF U.V06ANT <> U.V06TOT THEN NUMERR%=1:  GOSUB 3070
17200 IF U.V07ANT <> U.V07TOT THEN NUMERR%=6:  GOSUB 3070
17300 IF U.V08ANT <> U.V08TOT THEN NUMERR%=11: GOSUB 3070
17400 RETURN
```

# Appendix C - Invalid Record Screen

There is a screen format for each record type. Basically the formats are similar, varying only where the variables for each type are displayed. This example shows the screen for the records of type 2 (persons).

The first line, common to all screen formats, is the identification line, with the record type, the sequential record number in the file and questionnaire identification.

Then follow the record fields, specific for each record format, each one with the variable short name, variable description and field contents. By using the arrow keys one can navigate through the record fields, positioning the cursor in the proper field to be edited, and make the changes in order to produce a "clean" record.

After the record fields the system displays up to five error messages, with the error number (with an "A" if it is an automatic error), and the error description.

The bottom line on screen, also common to all formats, shows the menu for the function keys. The highlighted characters in this line are shown in reversed color on the screen.

The keys work as follows:

## C.1 Editing keys

Arrows (up, down, left and right) - position the cursor at the beginning of the field in the arrow direction.
Enter (or Return) - the same as a down arrow.
Home - positions the cursor at the first field.
End - positions the cursor at the last field.

To change the value of a variable:

a. Position the cursor at the field to be changed by using the arrow keys.
b. Enter the new value by typing ALL digits, i.e., if it is a 2-byte field with a new value of "9", one has to enter "09".
c. Enter any arrow key or the Return key.

## C.2 Browsing keys

The user can see (and edit) other records than the one detected by the system. By using the appropriate keys, he or she can see the adjacent records, or can jump to any record in the file. The system will display (but not check) the elected record, even if the user changes any field of the showed record. The way to resume checking is by using the F9 and F10 keys (see next topic).

PgUp - Browse backward to the previous record. Every time this key is pressed the system will go back one record.
PgDn - Browse forward to the next record.
Ctrl - The Ctrl key has to be used together with four other keys to "jump to" or "go to" a specific record.
CtrlHome - Go to the first record in the file.
CtrlEnd - Go to the last record in the file.
CtrlPgUp - Go back n records, where n is chosen by the user after pressing both keys. The system will show a field in the first line of the screen, to be filled in with the number of records to be "jumped back". After typing the number of records, press Return.
CtrlPgDn - Advance n records in the same fashion as in CtrlPgUp.

## C.3 Resuming keys

F1 - Stop processing and go to the final routine. This key should be used if, for any reason, the user does not want to continue editing the file. All corrected records up to this point are saved (the corrections to the current record are also saved).

F9 - To recheck the record. After doing some changes, the user wants to see if the corrections applied were consistent. If the record was selected by the system, i.e., it was not a record that the user selected and displayed by means of the Browsing keys, the system will recheck the displayed record, otherwise it will recheck either the displayed record or the last checked record, whichever comes earlier in the file.

F10 - To check the next record, supposing the user made a trivial correction, or the error was a warning message, not a real error (for example, more than 15 children). If the Browsing keys were not used, the system will resume checking the record following the displayed one, otherwise it will check either the next to the displayed one or the next to the last checked one, whichever comes earlier in the file.

## C.4 Special keys

Backspace - To correct any typing error.
Esc - The Esc key will always take the cursor to the first field, not executing the command (either a change in a field or a jump to other record).

```
Record type -  2       Recno -  10              Keyid - 01010011002

P01 - PERSONUMB      - 03            P17 - WIDOWHOOD      - 2
P02 - RELATIONSHIP   - 3             P18 - ECCONOMIC      - 5
P03 - AGE            - 12            P19 - OCCUPATION     -
P04 - SEX            - 1             P20 - INDUSTRY       -
P05 - RACE           - 1             P21 - OCCSTATUS      -
P06 - RELIGION       - 1             P22 - EVER BORN      - 01
P07 - NACREGION      - 01            P23 - ALIVE          - 02
P08 - NACCOUNTRY     -               P24 - CHILDABROAD    - 03
P09 - YEARRIVAL      -               P25A- MONTHLAST      - 02
P10 - MOTHERALIVE    - 1             P25B- YEARLAST       - 02
P11 - LIVGUYA        - 1
P12 - YEARDEATHMO    -
P13 - REGION5AGO     - 01
P14 - COUNTRY5AGO    -
P15 - EDUCATION      - 2
P16 - MARITALST      - 2

 57A P22  - EVER BORN       -  SHOULD NOT BE ANSWERED
 64A P25B - YEARLAST        -  OUT OF RANGE
 24       ALIVE INCONSIST EVER BORN


Reading record     10     out of  15
F1Stop F9Rechk F10Next PgUpBrBd PgDnBrFd CtrHomeFst CtrEndLst CtrPgGoto EscExit
```

# Appendix D - LOG file

The following pages are an example of a log file produced with a CHECKEDIT execution on a sample file.

The first page lists the identification of the questionnaires having structural errors. This page is printed only if the running option specified it ("1" for Structural checks only or "3" for both checks).

After that the system prints the program execution diary, with the name of the CHECKEDITed file, the execution's date and time, and the numbers of records read, edited and written, by record type. It also shows the running option and the starting and ending record numbers.

Then, the system prints the error statistics (separated by automatic and user), with the error number, error description, number of total cases, cases failed and their percentage from the total number of cases. Note that there is an automatic error number 1 and an user error with the same number. They are independent.

At the end, it prints the totals from which the system calculated the relative error results. There are record types (whose counts are the same as showed in the diary) and filters. There is also a column with the "complement" count, that is, the difference from the universe of cases and the filter count.

STRUCTURAL ERRORS

| QUEST ID | ERROR MESSAGE |
|---|---|
| 01010011001 | MORE THAN ONE HOUSEHOLD RECORD |
| 01010011001 | NUMBER ABROAD NOT IN SECUENCE |
| 01010011001 | MORE THAN ONE HEAD OF HOUSEHOLD |
| 01010011002 | MORE THAN ONE HOUSEHOLD RECORD |
| 01010011002 | NUMB DEADS DONT CORRS WITH TOTAL |
| 01010011002 | NUMBER ABROAD DONT CORRS WITH TOTAL |
| 01010011002 | NUMB PERSONS DONT CORRS WITH TOTAL |
| 01010011002 | MORE THAN ONE HEAD OF HOUSEHOLD |
| 01010011003 | MORE THAN ONE HOUSEHOLD RECORD |
| 01010011003 | NUMB PERSONS DONT CORRS WITH TOTAL |
| 01010011003 | HEAD SEX INCONSIST SPOUSE SEX |
| 01010011003 | NUMBER PERSONS NOT IN SECUENCE |
| 01010011003 | MORE THAN ONE SPOUSE |
| 01010011003 | NUMBER PERSONS NOT IN SECUENCE |
| 01010011003 | FIRST PERSON IS NOT THE HEAD |

DIARY FOR THE EXECUTION OF THE SAMPLE FILE

LOG FILE IS SAMPLE1

DATE: 05-14-1986 TIME 16:23:55

RECORDS BY TYPE

| RECORDTYPE | READ | EDITED | WRITTEN |
|---|---|---|---|
| 1 | 3 | 0 | 0 |
| 2 | 9 | 4 | 4 |
| 3 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 |
| 5 | 1 | 1 | 1 |
| 6 | 1 | 1 | 1 |
| 7 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 |
| 9 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 |
| invalid | 1 | | |

STRUCTURAL CHECKS
CONSISTENCY CHECKS

EDIT STARTED AT RECORD 1
EDIT ENDED AT RECORD 15

CHECKEDIT SYSTEM

REPORT OF ERROR FILE

AUTOMATIC ERRORS

| error number | message | | total cases | cases failed | percentage |
|---|---|---|---|---|---|
| 1 | A00 - ABROADNUMB | - OUT OF RANGE | 1 | 1 | 100.00 |
| 13 | A09 - INDUSTRY | - SHOULD NOT BE ANSWERED | 1 | 1 | 100.00 |
| 17 | M01A- DAYOFDEATH | - OUT OF RANGE | 1 | 1 | 100.00 |
| 21 | M03 - AGE | - OUT OF RANGE | 1 | 1 | 100.00 |
| 25 | P04 - SEX | - OUT OF RANGE | 9 | 1 | 11.11 |
| 39 | P13 - REGION5AGO | - SHOULD NOT BE ANSWERED | 1 | 1 | 100.00 |
| 57 | P22 - EVER BORN | - SHOULD NOT BE ANSWERED | 3 | 1 | 33.33 |
| 64 | P25B- YEARLAST | - OUT OF RANGE | 4 | 1 | 25.00 |

USER ERRORS

| error number | message | total cases | cases failed | percentage |
|---|---|---|---|---|
| 1 | NUMB DEADS DONT CORRS WITH TOTAL | 3 | 1 | 33.33 |
| 6 | NUMBER ABROAD DONT CORRS WITH TOTAL | 3 | 1 | 33.33 |
| 7 | NUMBER ABROAD NOT IN SECUENCE | 1 | 1 | 100.00 |
| 11 | NUMB PERSONS DONT CORRS WITH TOTAL | 3 | 2 | 66.67 |
| 12 | NUMBER PERSONS NOT IN SECUENCE | 9 | 2 | 22.22 |
| 13 | MORE THAN ONE HEAD OF HOUSEHOLD | 3 | 2 | 66.67 |
| 14 | FIRST PERSON IS NOT THE HEAD | 3 | 1 | 33.33 |
| 15 | MORE THAN ONE SPOUSE | 3 | 1 | 33.33 |
| 16 | HEAD SEX INCONSIST SPOUSE SEX | 3 | 1 | 33.33 |
| 24 | ALIVE INCONSIST EVER BORN | 4 | 3 | 75.00 |

TOTALS

| number | message | total | complement |
|---|---|---|---|
| 1 | RECORD TYPE 1 | 3 | 0 |
| 2 | RECORD TYPE 2 | 9 | 0 |
| 5 | RECORD TYPE 5 | 1 | 0 |
| 6 | RECORD TYPE 6 | 1 | 0 |
| 7 | PERSONS >= 5 YEAR OLD | 8 | 1 |
| 8 | PERSONS >= 10 YEARS OLD | 8 | 1 |
| 9 | ECONOMIC ACTIVE PERSONS | 1 | 8 |
| 10 | WOMEN >= 15 YEARS OLD | 6 | 3 |
| 11 | WOMEN WITH CHILDREN | 4 | 5 |
| 12 | BORN ABROAD | 3 | 6 |
| 13 | MOTHER ALIVE | 6 | 3 |
| 14 | MOTHER DEAD | 3 | 6 |
| 16 | EDUCATION ONLY FOR OLDER 5 YEARS | 1 | 0 |

## Appendix E - System Specifications and Limits

The CHECKEDIT system was developed to work on a file with the following characteristics:

-Sequential organization (each record has a carriage return and line feed). In this case, the user must start by using the running option 1 (structural checks), and if there are no errors, the system will automatically create a Random file. If the file is already in a Random organization (without carriage return and line feed), it must be with fixed length records of no more than 999 characters.

-Only numeric variables (blanks are accepted).

-It is not permitted to delete nor include any record.

-Each questionnaire can be composed of several records. They can be of different types, each type has to have the same questionnaire identification at the same place. This Identification may be composed of various fields, but they must be continuous in the records.

-Each record must have a variable containing the record type, even if the file has only one record type.

Some system limits are imposed by dBASE III (the DICT subsystem):

-Length of variable short names (4 characters).
-Length of variable long names (15).
-Length of filter and error descriptions (45).
-Length of variables having range or table look-up checks (4).

Other limits are imposed by the CHEKEDIT program itself:

- maximun of 5 displayed errors for each record type.
- maximun of 9 record types (from 1 to 9).
- maximun of 100 automatic errors and 50 user errors for each run.
- maximun of 32 displayed variables for each record type.
- maximun of 100 variables per questionnaire.
- maximun of 30 user specified filters (they cannot have the same identification of any record type. That is, if there are three record types, say, 1, 2 and 5, these numbers are forbidden to be used as filter numbers).

There are also some restrictions when writing the BASIC routines:

- Every user variable has to start with a "U.". This is done to ensure that there are no name repetitions between the main program and the user routines.

- The user routines must be in the range of 11000 thru 39999.

- The following entry points must be used:

  . 12000 - Structural checks
  . 15000 - Consistency checks
  . 17000 - End of questionnaire routine (if needed)
  . 19000 - Initialization routine (if needed)
  . 3070 - Accounting errors routine. This routine belongs to the main program. It has only to be GOSUBed by the user program every time the user routines detect an error.

- Before GOSUBing 3070, the user must assign the error number to a variable called NUMERR%.

- As the user is writing subroutines that are performed with a GOSUB by the main program, they must return the control to it with a RETURN statement.

- There are some system variables that are available to the user, like:

  . TYPEREC - Record type.
  . NUMERR% - Error number.
  . RECNO - Record number.
  . KEYID$ - Questionnaire identification.

  IMPORTANT: They may not be changed by the user routines.

- It is suggested that each routine have an instruction like ON TYPEREC GOTO a, b, c, etc, to perform the specific record checks depending upon the record type.