Oxford University Computing Laboratory

Computer Manuals

ICL 1900 Traffic Engineering
Traffic Survey Analysis

4103

TRAFFIC ENGINEERING

Traffic Survey Analysis

1900

ICL International Computers Limited

User notice

RECEIVED 19 NOV 1971
Statistical

MANUAL (NOTICE NO.) 10/3/71

4209 SURVEY ANALYSIS (1)

4103 X TRAFFIC SURVEY ANALYSIS (12)

File one copy of this notice with each of the manuals indicated.

SURVEY ANALYSIS SYSTEM

Modified Survey Analysis program #XDSB/4 (A816/1)

This issue of the program corrects all the errors in #XDSB/3F described in previous user notices and in software notices STATISTICAL AND SURVEY ANALYSIS 1,2, 3, 7 and 9.

A number of sections of the program have been rewritten and as a result:

1 The speed of processing cards containing multi-punched data has been greatly increased.

2 The amount of tape movement likely in processing table look-up expressions has been reduced.

3 The data filtering facility mentioned on page 79 of the new edition of the *Survey Analysis* manual (4209) is now available.

4 Additional checks are carried out in the processing of VECTOR statement.

FILTERING DATA

If switch 6 is on while processing primary data, no error message or print of a record is produced when it fails a REJECT test. However the count of records rejected is updated as usual. This allows large sections of data to be deliberately omitted from subsequent processing without causing a massive unwanted line printer output in the process. This facility was not available in earlier versions of #XDSB.

Issued by Technical Publications Service, 30 - 31 Friar Street, Reading, Berkshire. RG1 1JP

FORM 1/230/45(3.69)

CHECKS ON VECTOR STATEMENTS

A number of types of erroneous vector have in the past been accepted. These normally acted as intended by the user but could cause occasional unexpected results in tabulation. Users should note that:

1 All vectors are now checked for strict ascending order of their elements. Equality of two consecutive elements of a vector is only permitted if the intervening punctuation is a semi-colon.

2 No alphabetic characters may occur in numeric vectors. The unofficial practice of writing for instance [(70,90,10),9A] to avoid a three-character exclusive upper limit of 100 will no longer be permitted. This example must now be written (as on page 41 of the *Survey Analysis* manual) as [(70,90,10),>>] or even, more elegantly, [(70,>>,10)].

ICL International Computers Limited

User notice

23/6/71

MANUAL (NOTICE NO.)

4209 SURVEY ANALYSIS (2)

4103 TRAFFIC SURVEY ANALYSIS (13)

File one copy of this notice with each of the manuals indicated.

SURVEY ANALYSIS SYSTEM

Modifications to Survey Analysis program #XDSB/4A

This issue of the program corrects the errors in XDSB/4 described in Software Notices STATISTICAL AND SURVEY ANALYSIS 12 and 15 (concerning REJECT statements and range checks)

New disc overlayed program #XDSD/4A

This is the same program as XDSB/4A except that

1 #XDSD/4A is overlayed from E.D.S.

2 #XDSD/4A needs a minimum of 6016 words of core available to the user, whereas #XDSB/4A only needs 5888 words.

Notes:

1 The survey tape, data tape and tables tapes must still be magnetic tapes. Only the program is on disc.

2 The increase in core size means that #XDSD may not be used on 8K machines.

Modifications to auxiliary input program #XDSP/3

The program #XDSP reads free format paper tape data for input to #XDSB or #XDSD. This version corrects the errors in #XDSP/1 described in Software Notice STATISTICAL AND SURVEY ANALYSIS 15.

In this version of #XDSP, FORM statements containing parentheses are acceptable with or without a comma following the closing parenthesis. Thus compatibility is maintained both with the manual which demanded a comma in this position and with #XDSP/1 which would not

accept one.

Additional information

The remainder of this notice concerns three areas which are known to have caused trouble to a number of users.

RANGE OF CHECKS IN COMMON AND RECORD STATEMENTS

A set of isolated permitted values should always be expressed in the form (*a, b, c, d, e*) rather than (*a*) (*b*) (*c*) (*d*) (*e*). Likewise a set of permitted ranges should be coded as

(*a*:*b, c*:*d, e*:*f*) rather than (*a*:*b*)(*c*:*d*)(*e*:*f*)

In each case the second format is permissible but is less efficient both in processing time and core store used.

It should, however, be added that the two types of range are not permitted within the same brackets. Thus (*a*:*b, c, d*) is erroneous.

DUPLICATE TABLE NAMES

User should take care to avoid ever having two tables of the same name available to the system simultaneously. This is particularly apt to happend when a tables tape is reallocated using a TAPES statement. If this tape contains tables formed on a previous run whose names are the same as others formed on the current run the program is unlikely to give the results that the user hoped for. Some parts of the system will operate on the first of those tables to be formed and others on the most recent version.

READING DATA FROM PAPER TAPE

If primary data input and control statements are to be read from the same paper tape reader (either as separate tapes or as a single paper tape) then an extra new line should be punched following the FR-DATA block. This is mentioned in the manuals (*Survey Analysis* page 80 and *Traffic Survey Analysis* page 56), but experience has shown that it is frequently overlooked.

ICL International Computers Limited

User notice

MANUAL (NOTICE NO.) 6/10/71

4209 SURVEY ANALYSIS (3)

4103 TRAFFIC SURVEY ANALYSIS (14)

File one copy of this notice with each of the manuals indicated.

FORM 1/230/45(3.69)

SURVEY ANALYSIS SYSTEM

FRTAB and GEORGE 2

There is a conflict between **** used in Survey Analysis to denote the end of a table input with the FRTAB statement and **** used by GEORGE 2 as a document terminator. Users should avail themselves of the facility provided in GEORGE 2 for defining an alternative document terminator (e.g. ////).

Running under GEORGE 3

No general macro has been issued for running Survey Analysis under GEORGE 3 as the requirements of different users differ. There are no known restrictions on the use of Survey Analysis under GEORGE 3 beyond those already described in the Survey Analysis manual (page 80) for running under GEORGE 2.

Consistency Expressions in REJECT tests

The description given in the Survey Analysis manual pp 34,35 should be amplified to make the following points clear:

1 Variable names need not be spacefilled out to six characters.

2 The apostrophes enclosing alpha-numeric constants should be *single* quotes. Use of double quote characters will give rise to the error message "INCONSISTENT RELATIONSHIP.

Issued by Technical Publications Service, 30 - 31 Friar Street, Reading, Berkshire. RG1 1JP

3 Consistency checks for binary (B type) variables are not permitted. They are in any case meaningless.

ICL would be pleased to hear of any further items which users have found confusing in the manual, or would like to bring to our attention. If any users also have used Survey Analysis on problems which they consider unusual or otherwise interesting, please notify

R.R. Miller
Applications Support Dept
International Computers Limited,
30/31 Friar Street
Reading
Berks

Tel. Reading 581258

ICL International Computers Limited

User notice

RECEIVED 3 JAN 1972

MANUAL (NOTICE NO.) 22/12/71

4209
4103

SURVEY ANALYSIS (4)
TRAFFIC SURVEY ANALYSIS (15)

File one copy of this notice with each of the manuals indicated.

SURVEY ANALYSIS SYSTEM

The points discussed below may be of interest to users of the survey analysis system.

Magnetic tape unit numbers

Users running #XDSB or #XDSD under GEORGE 3 will need to know magnetic tape unit numbers. The information in the manual (page 80 of *Survey Analysis*) should be supplemented to read:

"The survey tape is always used as unit 6.

Tapes opened in an intial FOPEN statement are numbered sequentially from 1 (except for 6) in the order in which they are declared. When a tape is closed, its unit number becomes 'free'. If any other tapes are opened, they are given the lowest available unit number (i.e. a 'free' one if any tapes have been closed, or else a number one greater than the current highest unit number).

If an FERASE statement is used, the new survey tape is picked up as the lowest available unit number and relabelled. Copying then takes place. When copying is complete the old survey tape is unloaded and the new one is closed and then immediately reopened as unit 6."

Use of two line printers

The facilities specified in the manual for using more than one line printer were never fully implemented and should be ignored.

FORM 1/230/45(3.69)

Issued by Technical Publications Service, 30 - 31 Friar Street, Reading, Berkshire. RG1 1JP

DACONV statements

1 The *Traffic Survey Analysis* manual states (on page 24) that FREJECT statements may be used after an FDACONV statement, and the *Survey Analysis* manual may also give this impression. However, this is not the case: FREJECT may not be used in conjunction with FDACONV.

2 In the *Survey Analysis* manual, page 32, the specification of columns 15 and 16 for a table look-up expression should begin:

"Field width of result variable: this must be 06 or less for an integer table and 07 or greater for a floating-point table".

International Computers Limited

User notice

RECEIVED 14 FEB 1973

PUBLICATION (NOTICE NO.)

14/2/73

4209 SURVEY ANALYSIS (5)

4103 TRAFFIC SURVEY ANALYSIS (16)

File one copy of this notice with each of the publications indicated.

SURVEY ANALYSIS SYSTEM

Modification to programs XDSB/4B, XDSD/4B

This issue of the programs corrects the errors in the following facilities in Mk 4A:

1	Multi-punched multi-card records	(Notice 17, item 33)
2	Multi-punched fields	(Notice 19, item 36)
3	Table look-up	(Notice 22, item 40)
4	REJECT expressions	(Notice 26, item 46)
5	Constant in table operations	(Notice 26, item 47)
6	Division in DA-REC	(Notice 26, item 48)

The error described in Notice 29, item 52, concerning two consecutive title cards, has not been corrected.

New facilities

Some new facilities have been introduced to the Survey Analysis system. These facilities will be described fully in a new Edition of the Survey Analysis manual to be published later this year, but are listed below.

INPUT OF INTEGER TABLES

If a character I is punched in Column 6 of an FRTAB statement, the elements of the table that follows will be stored as integers (one word cells), and will thus not need conversion by an FT▽FIX statement before use for integer table look-up.

The cell values input must all be integers; any decimal point will cause an error. The table may have a maximum of 491 rows.

Issued by Technical Publications Service, 30 - 31 Friar Street, Reading, Berkshire. RG1 1JP

INPUT TABLE TERMINATOR

An alternative four-character terminator to tables input by FRTAB or FRABI statements may be specified by the user in place of the standard terminator ****. The final A, B or closing parenthesis is followed by a comma, which is followed in turn by the required alternative four-character terminator.

Example

```
FRTABI▽POPLTNTABLEACARDS0(2,4),####
```

SUPPRESSION OF ZERO TABLE CELLS

An additional parameter may now be specified with an FWRITE statement. It takes the form of a single character which will be printed in place of zero for all table cells that contain zero. The character is punched, preceded by a comma, immediately after the parameter *symbol* (see page 56 of the *Survey Analysis* manual) and immediately before the character [of the text specification. The use of this facility is as follows: only values exactly equal to zero will be replaced by the specified character; very small non-zero quantities will still be printed as zero.

Example

```
FWRITE▽CARTAB,F8.3,1,:,-[ ....
```

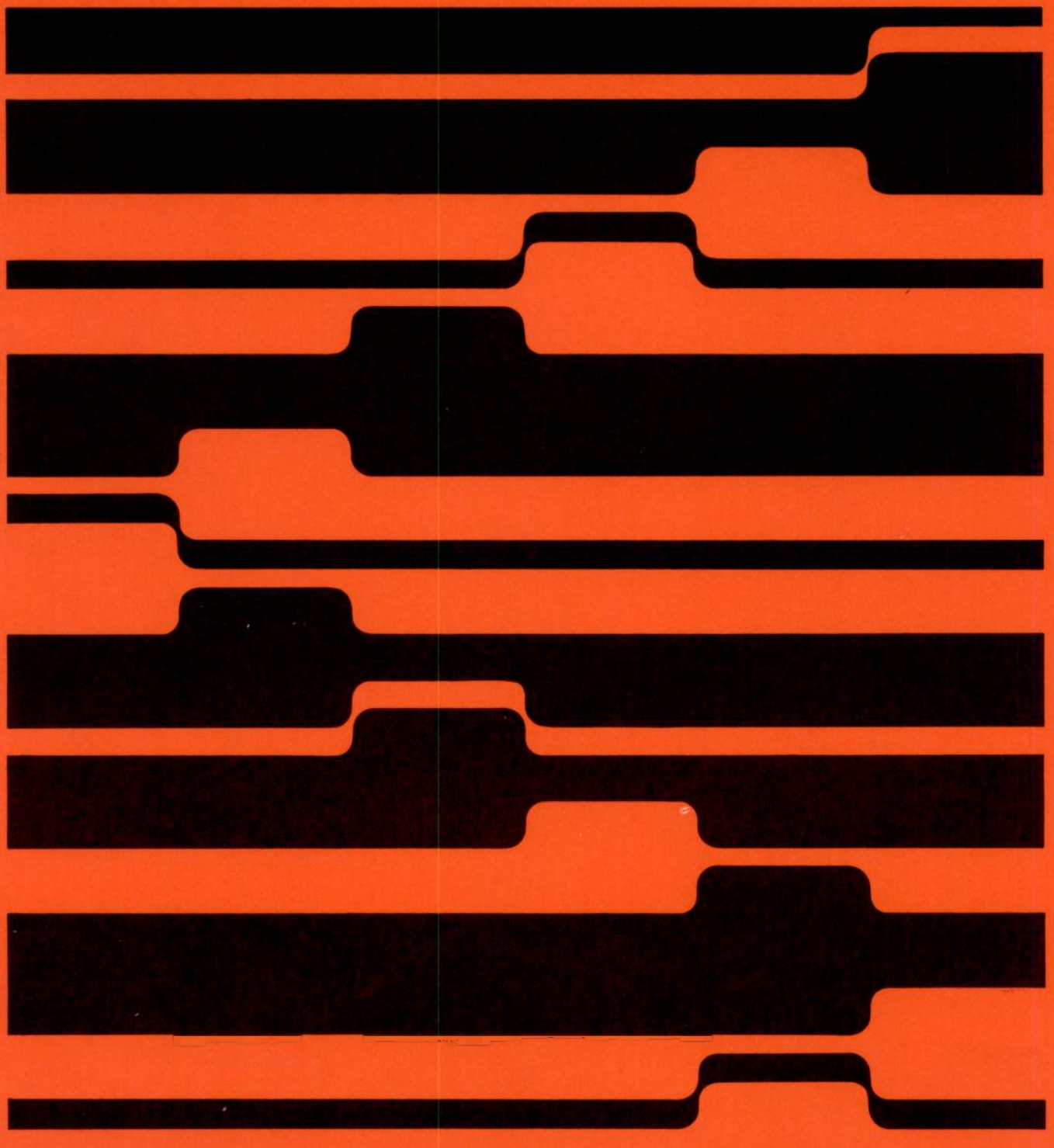
SCRATCHING MAGNETIC TAPES

A new statement FSCRTCH has been introduced in the TSET-UP category. It has as parameters a list of up to 9 six character tape names, punched as for FOPEN, FTAPES etc. All tapes mentioned will have the header SCRATCH TAPE written to them and are then closed. As this facility destroys all information on the tapes it should be used with extreme care.

ICL

Traffic survey analysis

1900 Series



ICL

Traffic survey analysis

1900 Series

With effect from 9th July 1968 the name of International Computers and Tabulators Limited has been changed to International Computers Limited.

Technical Publication 4103

First Edition August 1968

Issued by Technical Publications Service
International Computers Limited
Head Office: ICL House, Putney, London SW15
and printed in Great Britain by
ICL Printing Services, Letchworth, Hertfordshire

Preface

This manual is intended for traffic engineers and describes the use in traffic surveys of the 1900 Series Survey Analysis System (#XDSB), the general use of which is described in the manual *Survey Analysis System Mark 1*.

The minimum computer configuration required to use this system is as follows:

1 1900 Series central processor with a floating-point Executive and at least 5000 words of available core store;

4 magnetic tape decks;

1 card reader or 1 paper tape reader;

1 line printer.

These requirements exclude only the smallest versions of the 1901 and the 1902.

Special coding sheets are provided for use in writing survey analysis statements. General survey analysis coding sheets (form number 14/83) are available in pads of 100 sheets, and coding sheets for tables statements (form number 14/84) are available in pads of 50 sheets.

Chapter 1 of this manual is a general description of the system and of its relationship with other ICL traffic engineering software.

Chapter 2 discusses the forms taken by control statements and input data, and Chapters 3, 4, 5, 6, 7, 8 and 9 describe the various options in detail. Generation information on use of the facilities, operating instructions and error messages are given in Chapter 10. Appendix 1 is a list of the 1900 Series characters.

International Computers Limited wishes to thank the County Surveyor, Dorset County Council, who kindly provided the data used in the analysis shown in Appendix 2 of this manual.

Contents

Preface iii

Chapter 1 Introduction 1
THE 1900 SERIES SURVEY ANALYSIS SYSTEM 1
Phase 1 Data tape creation 1
Phase 2 Tabulation 2
VARIABLES 2
KEYS AND VECTORS 3
Phase 3 Table manipulation 4
Phase 4 Output 4
Language 4
Expansion factors 5
Recoding of variables 6
RELATIONSHIP WITH OTHER ICL TRAFFIC ENGINEERING PROGRAMS 6

Chapter 2 Input
DATA 7
Coding 7
Punching 7
CARDS 7
PAPER TAPE 10
Magnetic tape 11
Limitations on record size 11
CONTROL STATEMENTS 11
Names 12

Chapter 3 SET-UP statements 13
THE SET-UP TITLE STATEMENT 13
THE OPEN STATEMENT 13
Punching 14
THE TAPES STATEMENT 14
Punching 15
THE LIST STATEMENT 15
Punching 16

Chapter 4 FORMAT statements 17
THE FORMAT TITLE STATEMENT 17
COMMON AND RECORD STATEMENTS 17
Punching 19
THE DA-REC STATEMENT 20
Arithmetic expressions 20

Table look-up expressions 21
Punching 22
BLOCK TRANSFER METHOD 22
STATEMENTS THAT CONTAIN ARITHMETIC EXPRESSIONS 22
STATEMENTS THAT CONTAIN TABLE LOOK-UP EXPRESSIONS 23
THE DACONV STATEMENT 23
Punching 23
THE REJECT STATEMENT 24
Consistency expressions 25
Punching 25

Chapter 5 READ statements 27
THE READ TITLE STATEMENT 27
Punching 27
THE R-DATA STATEMENT 27
Punching 27
THE CONDAT STATEMENT 28
Punching 28
THE R-TAB STATEMENT 29
Table format 29
Punching 30
THE VECTOR STATEMENT 30
Vector specification 31
COLONS 31
COMMAS 31
SEMI-COLONS 31
MULTIPLE KEYS 32
Punching 32
THE TEXT STATEMENT 33
Punching 33

Chapter 6 TABLES statements 35
THE TABLES TITLE STATEMENT 35
Punching 35
THE TAB STATEMENT 35
Punching 37

Chapter 7 OPERAT statements 39
THE OPERAT TITLE STATEMENT 39
Punching 39
THE OPERAT FUNCTION STATEMENTS 39
Sub-tables 39
Punching 41
The DELETE statement 42
The ROWJN and COLJN statements 42
The ROWMRG and COLMRG statements 42
The CONADD, CONSUB, CONMPY and CONDIV statements 42

The T SQRT statement 42
The T TSP statement 42
The T ADD, T SUB, T MPY and T DIV statements 43
The T FIX statement 43
The ROWMPY, ROWDIV, COLMPY and COLDIV statements 43
The WRSUM and WCSUM statements 43

Chapter 8 OUTPUT statements 45
THE OUTPUT TITLE STATEMENT 45
Punching 45
THE WRITE STATEMENT 46
Cell value formats 46
Sections 47
Texts 47
STANDARD HEADINGS 47
TEXT SPECIFICATION 48
Punching 48

Chapter 9 XTRACT statements 53
THE XTRACT TITLE STATEMENT 53
Punching 53
THE ERASE STATEMENT 53
Punching 53
THE LIST STATEMENT 54
Punching 54

Chapter 10 Design and operation of survey analysis runs 55
DATA TAPE CREATION 55
Supplementary data tape creation runs 56
TABULATION 57
TABLE MANIPULATION 57
OUTPUT 58
LONGER RUNS 58
The calculation of expansion factors 59
RECODING OF VARIABLES 59
ERROR HALTS 59
Error halts arising from R-DATA and CONDAT statements 61
Recovery from error halts 62
ERROR MESSAGES 62
General error conditions 62
Errors associated with the SET-UP group 62
Errors associated with the FORMAT group 63
Errors associated with the READ group 64
Errors associated with the TABLES group 64.1
Errors associated with the OPERAT group 64.2
Errors associated with WRITE statements 64.3
Errors associated with the XTRACT group 64.3
POST-MORTEMS 64.4

Appendix 1 Relative sequence of 1900 Series characters 65

Appendix 2 Example analysis 67

Appendix 3 #X3GX 93

Appendix 4 #X3GV 97

Index 103

Illustrations

Figure 1	Table	2
Figure 2	Multi-level table	3
Figure 3	Combined survey form and coding sheet	8
Figure 4	(a) Multi-card records	18
	(b) Multi-record cards	18
	(c) Single-record cards	18
Figure 4.1	Survey analysis coding sheet for a re-coding run	60
Figure 5	Flow diagram of typical analysis of roadside interview survey	69

Chapter 1 Introduction

The first stage in the solution of traffic engineering problems is normally the quantitative assessment of existing conditions. Such an assessment will inevitably involve some kind of traffic survey. Before any useful information can be obtained from the survey, the data that has been collected must be analysed. Usually, the only practicable way of collecting data, of any but the most general nature, is to use questionnaires or survey forms, which are filled in either by members of the public or by trained interviewers or observers. Each form contains at least one complete set of data and the process of analysis consists of the co-ordination of all the sets of data to produce a single, overall representation of the required information. The same item, or group of items, of data is examined on each form, and identical values, or groups of values, are combined to show either the relative frequency with which the values of a single item of data occur, or the pattern of the distribution of values between two or more items. This procedure is repeated for as many items or groups of items as are required, and statistical measures such as mean values and standard deviations may also be calculated. The results of the analysis are then interpreted and applied to the solution of the original problem by the traffic engineer. It is the process of analysis and not the design, implementation or interpretation of surveys that is described in this manual.

The large amount of data collected in any truly representative traffic survey makes the task of analysis by manual or mechanical means both formidable and expensive. Electronic digital computers, however, because of their high speed of calculation and large capacity for storing data, are ideally suited to such tasks, and their use in this field not only means that surveys may be conducted in almost any degree of detail but also brings the technique of large scale surveying within the means of most, interested organizations.

THE 1900 SERIES SURVEY ANALYSIS SYSTEM

The system consists of a highly flexible program, capable of analysing the data of almost any type of survey, and a simple language, consisting of a vocabulary of statements used to specify the form of the required analysis. In the context of I.C.T. Traffic Engineering Software, this program will be used to extract information about the origins and destinations of vehicles travelling within a certain area from surveys conducted for this purpose. This information will then be used by one of the Traffic Assignment or Furness Prediction programs. However, the Survey Analysis program was not specifically designed for use with this kind of traffic survey or with traffic surveys in particular and is equally applicable to almost any kind of survey. To attempt to design the survey around this system of analysis is entirely unnecessary as the system is completely adaptable.

The system works in four phases, which are performed consecutively. The analysis itself takes place in the second and third phases, and the first and fourth are associated respectively with the input of survey data and the output of results. Analysis consists of simply the formation of *tables*. Tables are one dimensional or, usually, two dimensional arrays of numbers and show how the values of the data are distributed. For example, consider the table shown in Figure 1. At the start of the analysis the table would be blank, as shown, but at the end of the analysis, each cell of the table would contain a number, the value of which would depend on the number of survey records containing both the column and row values of that cell. Every record showing a vehicle of class 2, say a private car, with 4 occupants would contribute to the value of the contents of the shaded cell.

The four phases of the system are now described in detail.

Phase 1 Data tape creation

The first phase of analysis involves the input of the data collected in the survey. The data must first be represented in a form that is acceptable to the computer; this program will accept data on either punched cards or punched paper tape. The contents of the cards or paper tape may, however, be transferred to magnetic tape and input from there; this process is called *offlining* and may be used when there is a delay between the preparation of the cards or paper tape and the running of the analysis program. Input from magnetic tape is quicker than from cards or paper tape and is described more fully in Chapter 2, where the conversion of survey forms into punched cards or tape is also explained.

Although the fact that analysis is computer aided does not affect the design of the survey, the survey forms or questionnaires should be designed to facilitate the production of punched records, and so a preliminary study of Chapter 2 may prevent wastage of time and labour at the punching stage.

After input, the data may undergo some preliminary processing. The data may be rearranged and numerical data may be operated on arithmetically, or several items of numerical data may be arithmetically combined. Validity and range checks may also be carried out. For example, if a record were input stating, amongst other items, that a vehicle of class 1, say a motor cycle, had 9 occupants, the whole record would be rejected as invalid, and, if only 6 classes of vehicles were valid, a record stating the vehicle class to be 7 would also be rejected as it would be out of range. Rejected records are printed out on a line printer and accepted records are stored on magnetic tape for use in the second phase. Rejected records that have been corrected may be reinput and added to the records already on tape.

The tape on which records are stored is known as the *data tape* and the records are called *data records:* the records on cards or paper tape, or on an input magnetic tape are known as *primary survey records.* Note that because processing of data is possible at this stage, the data records will not necessarily be identical to the primary survey records.

Phase 2 Tabulation

During this phase tables such as the one shown in Figure 1 are compiled. The contents of each data record are examined to see what contribution they make to each table and complete tables are stored on a magnetic tape known as a *table tape.*

This phase is the basis of the analysis system and is described here in detail. The relevant terminology, which is used in the rest of the manual, is also defined.

VARIABLES

Each question on the survey form corresponds to one variable; the possible answers to the question are the values the variable may take. For example, one question may be, "How many occupants does the vehicle have?", the corresponding variable may be known as OCCUPS (the names of variables are restricted to six characters) and will take integer values from 1 to, say, 90 if public service vehicles are included. The range of a variable is the set of values it may take; in this example the range of the variable OCCUPS consists of all the integers from 1 to 90. There are three classes of variables, called *numeric, alphanumeric* and *binary.*

Numeric variables take values that consist entirely of numbers. Examples of numeric variables are: the number of occupants of a vehicle, the number of vehicles that pass a given point per hour, the length of time in minutes for which a vehicle is parked.

Alphanumeric variables take values that consist of letters or letters and numbers. Examples are: vehicle class, origin, destination. Note that the values of an alphanumeric variable may be coded numerically; if this is done for the whole range, the variable will become a numeric variable. For example, the variable for vehicle class may have the range 1, 2, 3, 4, 5, 6, where:

1 = motor cycle
2 = car
3 = light goods vehicle
4 = heavy goods vehicle
5 = coach
6 = bus

This variable would then be numeric.

		Number of occupants											
		1	2	3	4	5	6	7	8	9	10	11	12
Vehicle class	1												
	2				▨								
	3												
	4												
	5												
	6												

Figure 1 Table

Binary variables have a range of only two values, typically YES and NO. A binary field is a series of binary variables combined into one question.

Example Will your journey take you over the following bridges?

Answer YES or NO in each case.

New Bridge.....St John's Bridge.....High Street Bridge.....

East Gate Bridge.....

KEYS AND VECTORS

The variables that are tabulated in the second phase of analysis are the keys of their tables. For example, the keys of the table in Figure 1 are the variables corresponding to NUMBER OF OCCUPANTS (i.e. OCCUPS) and VEHICLE CLASS. The way in which the ranges of the keys of a table are divided into rows and columns is determined by vectors. The vectors of the table in Figure 1 are identical to the ranges of the two keys as each value corresponds to one row or column. If, however, the range of, say, OCCUPS had been 1 to 90, it would probably have been divided into groups of 10, the column vector would then specify the column ranges as 1 to 10, 11 to 20.....81 to 90. Thus a column (row) vector is the specification of a multiple test that determines to which of the columns (rows) of a table a record contributes. The values of rows and column must be specified in ascending order, according to the relative sequence of 1900 Series characters (see Appendix 1), but the arrangement of the rows and columns of tables may be altered in phase 3 of the analysis. Vectors do not, of course, apply to keys that consist of binary fields. In this case the only sensible arrangement of rows or columns is to have one row or column for each binary variable; this arrangement is always assumed.

A *multiple key* consists of more than one variable and produces a *multi-level table.* Figure 2 is an example of a multi-level table. The multiple key TRIP$\nabla\nabla$VCLASS has two main divisions, local and through traffic (TRIP$\nabla\nabla$), and each of these divisions is broken down into 6 vehicle classifications (VCLASS). If only the local part or only the through part of the table had been required. it could have been obtained by specifying, in the column vector, the inclusion of vehicles with the required destination only. The specification of vectors is described in Chapter 4. Note that the major division of the key corresponds to the first variable named in the key, i.e. TRIP$\nabla\nabla$; the subdivision is specified second. If each vehicle class were further divided into, say PURPOS, i.e. purpose of journey, the name of the key would be TRIP$\nabla\nabla$VCLASSPURPOS.

		Origin							
		1	2	3	4	5	6	7	8
TRIP VCLASS	Local 1								
	Local 2								
	Local 3								
	Local 4								
	Local 5								
	Local 6								
	Through 1								
	Through 2								
	Through 3								
	Through 4								
	Through 5								
	Through 6								

Figure 2 Multi-level table

Numeric multiple keys are combined by multiplying the value corresponding to the major division of the key by 10^a, where a is the field width corresponding to the first subdivision, adding the result to the value corresponding to this subdivision, multiplying this total by a similar factor, and so on.

If required, both of the keys of a table may be multiple keys, but it should be noted that all variables in a given key must be either type C (alphanumeric) or type D (numeric). In the case of alphanumeric

multiple keys, the maximum number of characters that may be included in the key value is 27; in the case of numeric multiple keys the recommended maximum number of significant characters is 6 but up to 11 may be specified if required.

Thus, data records consist of variables, and each variable will have a set of values, each value corresponding to one sample taken in the survey. In the tabulation phase, the keys and vectors of a table are specified and the data records are searched for the variables named in the keys. When all the required variables of one data record have been found, the value of each is tested and, by reference to the vectors, the contents of the appropriate cell, if any, are incremented. Normally the contents will be increased by one but the increment may be set at any value, including the value of a variable in the record being examined, or may be a constant or variable value multiplied by the value of another, specified variable.

This process is repeated for each record and finally the set of values of the variables named in the keys that is consistent with the specified vectors will be represented in the table. As many tables as are required may be produced.

Phase 3 Table manipulation

In this phase, the basic tables formed in phase 2 may be operated on arithmetically, either by constant numerical values or by the elements of other tables. Tables may also be merged or re-arranged. The tables produced, called *result tables,* are stored on a table tape. Examples of simple table operations are the multiplication of the elements of a table by p.c.u. factors and the averaging of the results of a weekend survey, using the estimation

$$\frac{5 \times \text{Friday} + \text{Saturday} + \text{Sunday}.}{7}$$

In some cases the basic tables produced in the second phase will be sufficient and so phase 3 is optional.

Phase 4 Output

Finally any of the tables produced in phases 2 and 3 may be printed out or, if required for input to another program, punched into cards or paper tape. Tables may be output with any required titles, row and column headings or other text.

Language

Analysis will always consist of at least phases 1, 2 and 4, phase 3 is optional. However, many options are available within each phase, and the exact course of every analysis must be precisely and completely specified. Also the names of variables, keys etc. must be provided and vectors must be defined. All of this is done by means of *control statements,* which are input to the program in the same way as data. A control statement consists of a unique statement identifier followed by the information being specified, in a fixed format. The exact format differs from statement to statement but the program recognizes the identifier and expects the appropriate format. Control statements are divided into 7 groups and are input in batches, all the statements in a batch being from the same group. Certain groups are normally used in only one phase of the analysis but others are applicable to two or more phases.

The SET-UP group of statements is used mainly to name the magnetic tapes that are to be used as data tapes or table tapes. When a tape is named it is also allocated to the analysis program and the program may not use a tape until it has been named. Similarly, tapes are released from the program by SET-UP statements. Tape names are used in other control statements to refer to particular tapes. All the tapes can be named at the start of the run or they can be named as they are needed at the start of each phase. SET-UP statements are described in Chapter 3. Note that as well as the data and table tapes, two more tapes are required. These are the *system tape,* which contains the analysis program, and the *survey tape,* which is used by the program as a work tape. SET-UP statements are not needed for the system and survey tapes.

The following five groups of statements are associated with particular phases.

Phase 1 FORMAT statements are used to describe the format of the primary survey records and the required format of the data records. Preliminary processing and validity or range checks may also be specified. FORMAT statements are described in Chapter 4.

	READ	statements are used to specify the input medium being used, and the primary survey records are input immediately after the batch of READ statements. READ statements may also be used in phase 2 to specify vectors, and in phase 4 to specify text such as table headings. Both of these may, however, be specified in the first batch of READ statements and will be stored on the survey tape. READ statements are described in Chapter 5.
Phase 2	TABLES	statements are used to obtain the basic tables and to give each table a unique name. Vectors are chosen from those specified in the READ statements and are associated with keys. Row and column sums may be obtained and the cell increment and weighting factor, if any, are specified. TABLES statements are described in Chapter 6.
Phase 3	OPERAT	statements are used to perform table operations. OPERAT statements are described in Chapter 7.
Phase 4	OUTPUT	statements are used to specify the output medium and the format of the tables that are output. Text, previously input by a READ statement, may be associated with the appropriate tables. OUTPUT statements are described in Chapter 8.

The XTRACT group may be used after the second and third phases to remove redundant tables, or tables that require alteration, from the table tapes, and after the first phase to remove incorrect vector specifications from the survey tape. Text such as table headings can also be deleted. The deletion of large quantities of unnecessary information will considerably speed up processing. XTRACT statements may also be used to obtain printed records of the vectors, tables and texts that remain on the tapes. This group of statements is described in Chapter 9.

Expansion factors

Expansion factors are multipliers, used to scale up the results of a survey to compensate for the fact that only a fraction of the population under survey is sampled. For example, if one in five vehicles were stopped in an external cordon survey, all results that were in the form of number of vehicles, or number of passenger car units, would be multiplied by five. This could be done during the tabulation or table operation phases. In practice, however, the situation is not usually so simple, and there is a number of facilities available for dealing with the calculation and incorporation of expansion factors.

Consider a more realistic example of an external cordon survey, carried out at four census stations during twelve equal time periods. The intention was to stop every fifth vehicle, but it was found to be impossible to maintain this rate, owing to periodic increases in the volume of traffic. The total number of vehicles passing through each station in each time period was also recorded (enumeration counts). A separate analysis of each time period at each station is required and so 48 different expansion factors are needed. They may be calculated in the following way.

The data, including details of stations and time periods, is input in the usual way and a table with the headings STATION and TIME PERIOD is prepared; the vectors are specified so that each cell of the table contains the number of vehicles stopped at a particular station in a particular period. A similar table is prepared giving the total number of vehicles passing through each station in each period, and this can be input as a table, using a special READ statement provided for this purpose. The first table can then be divided into the second by using a statement of the OPERAT category. When tables are divided, the contents of each cell in one table are divided into the contents of the cell in the other table that occupies the corresponding position. The result table is thus a table of expansion factors.

One of the FORMAT statements can then be used to associate the name of a new data variable with the values of the expansion factors. Each data record is thus effectively enlarged to include the expansion factor appropriate to the record, e.g. all records that contained the value 1 for time period and 1 for census station would be enlarged to contain the first element of the first row of the table (the contents of the cell for time period 1 and census station 1) as the value of the new expansion factor variable. In this way, expansion factors can be used during tabulation; the name of the expansion factor variable may be specified in TABLES statements and the contribution that each record makes to a table is then determined by the value of the expansion factor in that record.

The facilities available for the calculation and use of expansion factors are part of, or in some cases extensions of, the normal analysis facilities and are not described separately in the following chapters. The example contained in Appendix 2 includes the calculation of expansion factors from data records and enumeration counts, and the use of the factors in tabulation.

The survey analysis program contains a facility for ascribing a value to a variable on the basis of the value of another variable, when the relationship between the two values cannot be reduced to a simple algebraic expression; this facility is discussed in detail on page 59.

RELATIONSHIP WITH OTHER ICL TRAFFIC ENGINEERING PROGRAMS

The facilities described in this manual can be used to produce origin-destination tables showing the volume of traffic through the area in question travelling between each possible origin-destination pair; an example of this is given in Appendix 2, page 67.

The tables tape containing these origin-destination tables can then be input to programs which convert the tables to a form suitable for input to other traffic engineering packages. Two conversion programs are available. Program #X3GX is used to provide input suitable for the Assignment and the Furness Prediction Mark 2 Traffic engineering packages; a specification of #X3GX is given in Appendix 3, page 93. Program #X3GV is used to provide input suitable for the Capacity Restraint Traffic Assignment and Magnetic Tape File Handling for FORTRAN Data packages; a specification of #X3GV is given in Appendix 4, page 97.

Chapter 2 Input

This chapter describes how data and control statements are prepared for input to the survey analysis program.

DATA

For the sake of efficiency, the amount of data, as measured by the total number of characters that are input, should be kept as small as possible. The number of variables and the number of primary survey records will be determined by the nature of the survey and the size of the population under survey. However, there are two ways in which the total number of characters can be kept to a minimum; the first is coding and the second is the economical use of punching (described later in this chapter).

Coding

To input, store and manipulate ten characters when only one is necessary is an obvious waste of storage space and computer time and so, whenever possible, the values of variables should be coded. A previous example of this practice was given in which the values of the variable for vehicle class were 1,2,3,4,5 and 6 instead of MOTOR CYCLE, HEAVY GOODS VEHICLE etc., and most variables that would otherwise be alphanumeric can be treated in this way. Numerical rather than alphabetical codes should be used, as numbers are more easily stored and manipulated within the computer. Different variables may take identical values.

Usually, coding will be an intermediate step between the collection of data, on survey forms or questionnaires, and the punching of cards or paper tape, on which any input to a computer must be represented initially. However, if the data is of only a very simple nature and is collected by trained personnel, it may be collected in coded form. If this is done, particular attention should be given to the layout of the survey form, with the intention of making the conversion into punched form as simple, and therefore as error free, as possible. Each character of data should occupy a separate, numbered box, and the boxes should be grouped and not distributed all over the form. A group should contain the complete set of boxes for one record, i.e. for one interview, and the boxes should be in the order in which their contents are to be punched. The use of coding boxes on home interview forms completed without supervision may be confusing, and separate coding sheets are desirable with this type of survey. In all but the simplest cases, the best form layout for a survey conducted by trained personnel will be similar to that shown in Figure 3. The interviewer fills in alternate lines and, at a later date, each line is coded and the coded values entered on the line immediately underneath. Note that each sheet is numbered uniquely within each time period at each census station. This is to enable errors, detected at a later stage, to be traced back to the coding and collection stages; this will be explained fully when punching is considered. The form shown in Figure 3 has been filled in and coded; note that where two boxes have been provided for one coded value, permitting a maximum range of 0 to 99, unused boxes are filled with zero, i.e. 0,1,2,3.....9 are written as 00,01,02,03.....09. This practice minimizes punching errors and should always be used when more than one box is provided.

Punching

Data is input as a file of primary survey records on punched cards or paper tape; faster input may be obtained by transferring records from cards or paper tape to magnetic tape. The use of each of these input media is now described separately.

CARDS

Each primary survey record will consist of one set of values, obtained from one interview in the survey. A card contains 80 columns, each of which can represent one character. Each card is divided into *fields*, each field being associated with one variable and consisting of as many columns as are

OLDSHIRE COUNTY COUNCIL

ROADSIDE INTERVIEW SURVEY

Interviewer's Signature A. Brown

Coded by B. S.

Checked by J. A. W.

Census Station (1 2)	Time Period (3 4)	Sheet No. (5 6)
0 5	0 1	0 1

VEHICLE CLASS						ORIGIN	DESTINATION	TRIP PURPOSE							
Motor Cycle	Car	L. Goods	H. Goods	Bus	Coach	If origin is within town please give full address.	If destination is within town please give full address.	Work	School	Home	Business	Social	Shopping	Passenger Service	Catch Bus
✓						NEWTOWN	OLDTOWN MARKET PLACE						✓		
		1				0 3	5 5				6				
	✓					NEWTOWN	LONDON				✓				
		2				0 3	0 9				4				
			✓			NEWTOWN	OLDTOWN INDUSTRIAL ESTATE				✓				
		4				0 2									

Figure 3 Combined survey form and coding sheet

necessary to represent the largest value in the range of that variable. Fields are associated with variables in FORMAT statements.

Example The following variables, with the ranges of coded, integer values shown, would be allocated fields as follows.

Variable	*Range*	*Field*
Vehicle Class	1 to 6	1 column
Origin	01 to 60	2 columns
Destination	01 to 60	2 columns
Trip Purpose	1 to 8	1 column

Each record would therefore occupy 6 columns.

One record consists of the following values.

Vehicle Class	1
Origin	03
Destination	55
Trip Purpose	6

This record would be punched as follows.

1	0	3	5	5	6			

No separation or other distinction is made between fields on the cards, as the format of the primary survey records is described in control statements of the FORMAT category. The same layout must therefore be used for the whole file of records.

The record in the previous example consisted of numerically coded values of alphanumeric variables, and most traffic survey data will yield similar records. However, for the sake of completeness, an example of the other possible kinds of data is now given. The following example contains one uncoded value of an alphanumeric variable, a fractional numeric value and a binary field. All numerical data must be input in fixed point, decimal notation, and as the layout of records is fixed, decimal points may, and should, be omitted; the fact that the values have been raised by a factor of 10, 100 etc. should be remembered when vectors are specified and when arithmetical operation are performed. The values in binary fields will be YES and NO, which should be coded by 1 and 0. The binary field used as an example in Chapter 1 is used in the following example.

Example

Variable	*Range*	*Field*
Name of Driver	Up to 20 Characters	20 columns
Parking Duration in hours, to the nearest 1/10th	00.0 to 24.0	3 columns
New Bridge	1 or 0	4 columns
St. John's Bridge	1 or 0	
High Street Bridge	1 or 0	
East Gate Bridge	1 or 0	

One record consists of the following values.

Name of Driver	J.A. SMITH
Parking Duration in hours, to the nearest 1/10th	03.5
New Bridge	0
St. John's Bridge	0

Variable	*Range*	*Field*
High Street Bridge	1	
East Gate Bridge	1	

This record would be punched as follows:

J	.	A	.	S	M	I	T	H	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	0	3	5	0	0	1	1

There are three ways in which primary survey records may be arranged.

1 *Multi-record cards* contain several primary survey records per card. If the records do not occupy all 80 columns, the last record on each card should be followed by a terminating symbol, which may be any of the 64 characters, shown in Appendix 1 that is outside the range of the first field of each record. The symbol used is specified in a FORMAT statement.

2 *Multi-card records* are records that spread over several cards. Each card in a multi-card record should have a field reserved for a *designation value.* The designation value is different on each card and may be any of the 64 possible characters, usually a number. Each field used for a designation value should be associated with a variable by a FORMAT statement and the name of such a variable should always begin with the characters DES. Each multi-card record should begin on a new card.

3 *Single-record cards* contain one whole record only and do not require terminating symbols.

Most traffic survey data will be suitable for punching as multi-record cards; the first example of a primary survey record given in this chapter, which corresponds to the survey form shown in Figure 3, occupies only 6 columns. However, there are three variables that appear on the form in Figure 3 that were not included in the example record. These are: Census Station, Time Period, Sheet No.

These variables would require three 2 column fields, and, if included in each record, would increase the number of cards required by approximately 50%. As these variables take the same value in each record derived from this particular sheet, they may be punched as *common values.* Common values are punched only at the start of a multi-record card and are taken to apply to all the other records, called *reduced records* on the card. In the case of the form shown in Figure 3, the first 6 columns of the card would contain the common values, the next 72 columns would contain 12 reduced records, and column 79 would contain the terminating symbol; column 80 would be blank. The scope of a set of common variables does not extend over more than one multi-record card, and so each multi-record card will start with common values, when used. However, common values may also be used with multi-card records, in which case the common values apply over any required number of cards. The first card contains common values only and each set of reduced records starts on a new card.

If an error is found in any record on a card, as a result of a validity or range check, the whole of the card, including any common values, is printed out on the line printer. As errors occur most frequently at the data collection and coding stages it is useful to input the survey form sheet number as part of each record. Errors can then easily be traced back to their source.

At the end of the primary survey records one further card signifying the end of the data should be added. This card should contain only the word FINISH, in columns 1 to 6.

PAPER TAPE

The rules governing the punching of primary survey records into paper tape are similar to those for punched cards. A *block* on paper tape is analogous to a card and consists of a series of characters; each block should be followed by the newline character. Blocks, unlike cards, are not of a fixed length.

Multi-record blocks are similar to multi-record cards, but may consist of up to 1,000 characters not including the newline character, which should be punched immediately after the last record of the block. No terminating symbol is necessary. Common values, if used, must be punched for each block.

Multi-block records are punched in blocks of up to 80 characters, separated by newline characters. Each block must contain a designation value in the same manner as cards within a multi-card record. All records should start in a new block.

Single-record blocks are a special case of multi-record blocks. They may consist of up to 1,000 characters, followed by a newline character; no terminating symbol is required.

The end of data must also be indicated on paper tape by FINISH, which should follow the last newline character of the primary survey records and should itself be followed by a newline character.

Magnetic Tape

Data input to the survey analysis program on magnetic tape must consist of MTH records (see the ICL 1900 Series manual *Magnetic Tape*) in character form. The maximum number of characters in an MTH record is 1000, and the number of words in the record will be specified in a count word which is the first word of the record. MTH records may be batched into blocks of up to 512 words.

Each MTH record may contain either one survey analysis primary survey record, or an integral number of such records. The maximum size of a primary survey record is in any case 1000 characters, and a primary survey record may not extend over more than one MTH record. There is thus no need for designation values and if designation values are specified for magnetic tape input an error message will be given.

If the user has specified an end of block symbol in the FORMAT statement (see *The Format title statement*, page 17) this symbol should be used to terminate the data in each MTH record. The end of data is indicated by an end of reel, end of subfile or end of simple file sentinel, and when such a sentinel is encountered the input tape will be closed.

The file and, optionally, subfile holding the input data are specified in an IN parameter written immediately following the R-DATA statement. Any procedure may be used to create this file or subfile, so long as it is in standard MTH format. Hence standard sort merge and formatted output from FORTRAN programs (see the ICL 1900 Series manual *FORTRAN: 16K Disc Compiler*) and MTH functions may be used; among the ICL standard programs that may be found useful are #XKYA (see the ICL 1900 Series manual *Operating Systems GEORGE 1 and 2 Mark 6* and #XRMH, which both deal only with single-card records, and #XRMJ, which allows use of the batching facility. #XRMH and #XRMJ are both discussed in the ICL 1900 Series manual *Library Specifications*.

Limitations on record size

The sum of the field widths of B, C, D and ignored fields in primary survey records input to the program must not exceed 1000 characters.

In data records, each B-type field will occupy one word; each C-type field of N characters will occupy $N/3$ words, rounded up; each D-type field with a field width of less than seven characters will occupy one word; and each D-type field with a field width of seven or more characters will occupy two words. The total number of words in a data record may not exceed 498.

If either of these limits is exceeded, the error message

RECORD TOO LARGE

is output on the line printer and the program halts with the message

ERROR IN FORMAT SPECIFICATIONS

CONTROL STATEMENTS

Control statements may be input from cards or paper tape, and, as has already been explained, they are input in batches, each batch consisting of statements from one category only. The first statement of a batch is the *title statement* and identifies the category of the batch. The other statements are *function statements* and contain information for the analysis program.

The exact format used for punching each particular statement is given as part of the description of that statement in the following chapters. Punching instructions are given for cards, but paper tape is punched in the form of card images, i.e. each block corresponds to one card. Coding sheets are available for the writing of survey analysis control statements. Each line of a sheet corresponds to one punched card, and is divided into 80 spaces, each corresponding to one column of a card. Two types of coding sheet are available; one contains guide lines that divide the sheet into the fields that are common to all statements, and the other sheet is designed for TABLES statements only. The use of coding sheets for control statements not only facilitates the writing of statements but reduces the probability of the introduction of errors when the statements are punched.

Statements must always start on a new card or in a new block, and the first column always contains T for a title statement or F for a function statement. Columns 2 to 7 always contain the name of the statement; columns 8 to 74 contain the information. Columns 75 to 80 are used for card sequence numbers, but these are unnecessary on paper tape and so paper tape blocks are never more than 74 characters in length, not including the newline character, which should always follow a block. If a statement overflows onto a continuation card, the first column of this card is left blank.

The first character of a continuation block should be the space character ▽. If several consecutive function statements all have the same name, columns 2 to 7 of all but the first card of the group may be left blank (space characters on paper tape). Unused columns at the end of a card are left blank; paper tape blocks are terminated with the newline character after the last character of the statement.

Control statements are usually printed out on the line printer as they are obeyed, but this facility may be suppressed if not required (see Chapter 10). There is, however, a third type of statement, which is always printed out. This is the *comment statement*, which may be used to provide a more detailed degree of documentation of an analysis run than is provided by the normal printout of control statements. A comment statement may extend over only one card, which should contain the letter C in column 1; columns 2 to 74 may contain any required comment.

Names

Control statements are used to associate a name with each variable, vector, table, piece of text etc., and this name is used in other statements to refer to the variable, vector etc. There are four rules that apply to all names.

1 No name may consist of more than 6 characters.

2 Only letters, numbers and spaces may appear in names.

3 The first character of a name must be a letter, but a name may not start with character combination DES.

4 Names should be unique within an analysis except for the names of data variables, which may be the same as primary variable names.

Chapter 3 SET-UP statements

SET-UP statements are used to provide the program with the magnetic tapes it is to use as data tapes and table tapes, and to organize their use. There are three function statements in the SET-UP category. They are:

1 The OPEN statement, which is used to associate names with the tapes to be used as data tapes and table tapes, and to allocate these tapes to the analysis program.

2 The TAPES statement, which is used to release tapes that are not required during part of the run or to re-allocate tapes that have been released.

3 The LIST statement, which is used to obtain a printed summary of the contents of tapes.

No run may proceed if the required tapes have not been allocated and so the first batch of control statements to be input will be of the SET-UP category and will consist of at least the title statement and one OPEN statement, allocating the required data tapes. When a tape is named and allocated to the analysis program, it must be present on the tape decks, and so, if there are not enough decks available to accommodate simultaneously all the tapes that are required for an analysis, the run is split into sections that do not require more tapes than can be accommodated. At the end of each section, tapes that will not be used in the next section may be released, by a TAPES statement, and removed from the decks; their places are taken by the required tapes, which are allocated or re-allocated as necessary by OPEN and TAPES statements. There is no need to halt the run while tapes are changed; a console message indicates that a deck has been freed. The design of runs is discussed, with examples of the use of all categories of control statements, in Chapter 10.

THE SET-UP TITLE STATEMENT

The first card of any batch of SET-UP statements should contain TSET-UP in columns 1 to 7. If the batch is the first of the whole run, the following information should also be punched on this card, but not on successive SET-UP title cards.

Columns 8 to 13 — The name of the survey.

Each analysis should be associated with a name which should consist of up to six alphanumeric characters and should start with a letter, as described in Chapter 2.

Columns 14 to 18 — The number of words of core store available to the program.

The program requires access to at least 5,888 words of core store. However, if a number greater than 5,888 is specified the program will make use of the extra space; if less than 5,888 is specified or if this field is left blank the program will ignore the specification and restrict storage requirements to the minimum possible.

Example

T	S	E	T	-	U	P	T	E	S	T	∇	∇	∇	9	5	0	0

This card would be the first to be input in the analysis of a survey called TEST. 9,500 words of core store are available for the analysis. Note that the core store specification is right justified.

THE OPEN STATEMENT

Before the first time a tape is used in an analysis as a data or table tape, it must be named and allocated to the analysis program. This is done in an OPEN statement. An OPEN statement may contain up to

eight data or table tape names and for each name that appears a scratch tape will be obtained, given that name and allocated to the program (a scratch tape is one that is available for writing, its previous contents will be deleted). Any number of OPEN statements may be used at any point in the analysis run, but for every name that is specified a scratch tape must be available. Each tape should be named and allocated just once and names should be unique; before a new tape is named the program checks that no tape has already been given the same name.

Data tapes should be given names that start with DATA∇. Like all other names, these must not exceed six characters in length and so data tapes will normally be named DATA∇1, DATA∇2, etc. If necessary a data tape may extend over more than one reel of tape; the second and successive reels are automatically opened by the program, if they are available, and are given the same name as the first reel and an appropriate sequence number.

Table tapes must be given names that start with TABLE and so will usually be called TABLE1, TABLE2, etc. Table tapes may not extend over more than one reel. If a table tape becomes full, the program halts and outputs an error message (see Chapter 10) on the console typewriter and a list of the names of the tables left off the tape on the line printer.

The system tape and the survey tape do not appear in OPEN statements. The program is loaded into core store from the system tape by the computer operator and, at the start of a new analysis, the program automatically allocates itself a scratch tape which it names XXXXXXSURVEY, where XXXXXX is the survey name that appears on the first SET-UP title statement card.

Punching

The punching of an OPEN statement is as follows.

Column 1	F
Columns 2 to 7	OPEN∇∇
Columns 8 to 55	The names of the tapes to be opened. The number of tapes allocated to the program at any one time, including the system tape and the survey tape, cannot exceed the number of available tape decks.

Example

T	S	E	T	–	U	P	T	E	S	T	∇	∇	∇	9	5	0	0							
F	O	P	E	N	∇	∇	D	A	T	A	∇	1	D	A	T	A	∇	2	T	A	B	L	E	1
T	A	B	L	E	2																			

This batch of SET-UP statements would allocate two data tapes and two table tapes, with names as shown, to the program at the start of the analysis of the survey called TEST. This run would require six tape decks, providing that no tape became full.

THE TAPES STATEMENT

Because analysis takes place in four phases, there is no need for all the tapes used in an analysis to remain allocated to the program for the whole of the analysis. Table tapes are not used during the data tape creation phase, data tapes are not used during the table manipulation and output phases, and possibly certain table tapes will also not be used during one of these two phases. The TAPES statement is used to release unnecessary data tapes and table tapes from the program. These tapes can then be removed from their decks and replaced by tapes needed in the next part of the run. If the new tapes have already been allocated and then released for part of the analysis, they may be re-allocated by a TAPES statement; if they are scratch tapes, they must be allocated by an OPEN statement.

In a TAPES statement the names of tapes that are to remain allocated or are to be re-allocated are specified. When a TAPES statement is read, the program checks that all the tapes appearing in the statement are currently allocated. If any tape is missing a console message is output, requesting the operator to load the required tape. Any tape that is currently allocated but does not appear in the TAPES

statement is released. Up to eight tape names may appear in each TAPES statement and any number of TAPES statements may be used, providing that a tape deck is available for each tape that is re-allocated.

Punching

The punching of a TAPES statement is as follows.

Column 1	F
Columns 2 to 7	TAPES∇
Columns 8 to 55	The names of the data tapes and table tapes that are to remain allocated or are to be re-allocated to the analysis program.

Example Tapes DATA 1, DATA 2, TABLE1, TABLE2 are currently allocated. The following batch of statements would release the two data tapes; these are replaced by two previously un-allocated table tapes.

```
T S E T - U P
F T A P E S ∇ T A B L E 1 T A B L E 2
F O P E N ∇ ∇ T A B L E 3 T A B L E 4
```

If for any reason it were required to revert to the state of having the original data and table tapes allocated, this would be achieved by the following batch of statements.

```
T S E T - U P
F T A P E S ∇ D A T A ∇ 1 D A T A ∇ 2 T A B L E 1
T A B L E 2
```

The tapes TABLE3 and TABLE4 would be released and a console message asking for DATA 1 and DATA 2 to be loaded would be output.

THE LIST STATEMENT

The LIST statement is used to obtain a printed summary of the contents of tapes. The names of the data tapes and table tapes for which summaries are required are specified by name in the LIST statement.

The summary of a data tape consists of the following information.

1 A table, headed PRIMARY SURVEY DATA, which contains five columns, is printed first. The first column, headed FIELD, is a list of the variables whose values are represented in the primary survey records. The second column gives the type of each variable as B for binary, C for alpha-numeric or D for numeric. The third column contains C for common variables and R for variables whose values are in reduced records. The fourth column, headed SIZE, gives the number of columns, or characters, in the field of each variable and the fifth column gives the number of the column at the extreme left-hand side of each field, under the heading LHD.

2 The name of the survey and the name of the tape being listed.

3 Similar information to 1 is given for the data variables on the tape, under the heading DATA RECORD. The only difference is that the LHD column contains core store addresses.

4 The first four and last four data records. Each record occupies one line, which consists of the values in the fields, separated by spaces. If a value has become too large to be accommodated in its field, it is replaced by asterisks. At the end, the total number of data records on the tape is given.

The summary of a table tape consists of the names of all the tables on the tape.

Punching

LIST statements are punched as follows.

Column 1	F
Columns 2 to 7	LIST▽▽
Columns 8 to 13	PRINTX
	X is the unit number of the line printer to be used. If only one line printer is available X will be 0.
Columns 14 to 73	The names of the data tapes and table tapes to be listed.
	All tapes specified should be currently allocated to the program.

Example

T	S	E	T	-	U	P

F	L	I	S	T	▽	▽	P	R	I	N	T	0	T	A	B	L	E	1	D	A	T	A	▽	0

This batch of statements would produce printed summaries of the contents of the tapes TABLE1 and DATA 0.

Chapter 4 Format statements

FORMAT statements are used in the data tape creation phase of an analysis. They are concerned with the formats of primary survey records and data records and with the testing of the validity of values in both primary survey and data records.

There are two function statements used to describe the format of primary survey records. COMMON statements are used for variables whose values are common to two or more records (see Chapter 2), and RECORD statements are used for all other primary variables.

Two function statements are used with data records. DA-REC statements are used to specify data variables in terms of primary variables and DACONV statements specify new data variables in terms of existing data variables. The use of DACONV statements involves the creation of a new data tape. DA-REC and DACONV statements may also be used to incorporate elements of previously created tables into the data records.

The remaining function statement in this category is the REJECT statement. REJECT statements are used to specify validity checks and may be used with both primary and data variables. The REJECT statements that specify the tests to be performed on the values of primary variables will be input immediately after the COMMON and RECORD statements and the REJECT statements referring to data variables will follow the DA-REC statements. A further group of REJECT statements may follow the DACONV statements, if these are used.

THE FORMAT TITLE STATEMENT

The format title statement is used to introduce the batch of FORMAT statements and to specify the terminating symbol used with multi-record cards (see Chapter 2).

The statement is punched as follows.

Column 1	T
Columns 2 to 7	FORMAT
Column 8	The terminating symbol. The symbol may be any of the 64 acceptable characters that is outside the range of the variable associated with the first field of each record. If single-record cards or multi-card records are used, or if the records are on paper tape, this column should be left blank.

Example

T	F	O	R	M	A	T	;

This statement indicates that the terminating symbol is a semicolon.

COMMON AND RECORD STATEMENTS

One COMMON or RECORD statement should be input for each variable that is included in an analysis. The relationships between COMMON and RECORD statements and primary survey records on cards is shown in Figure 4; the same relationships hold true for input on paper tape.

Each COMMON or RECORD statement contains the specification of the name of the variable, the type of the variable and the size of the field used for values of the variable. If range checks are required on the values, the range is also specified. Note that if the multi-record card (or block) format is used

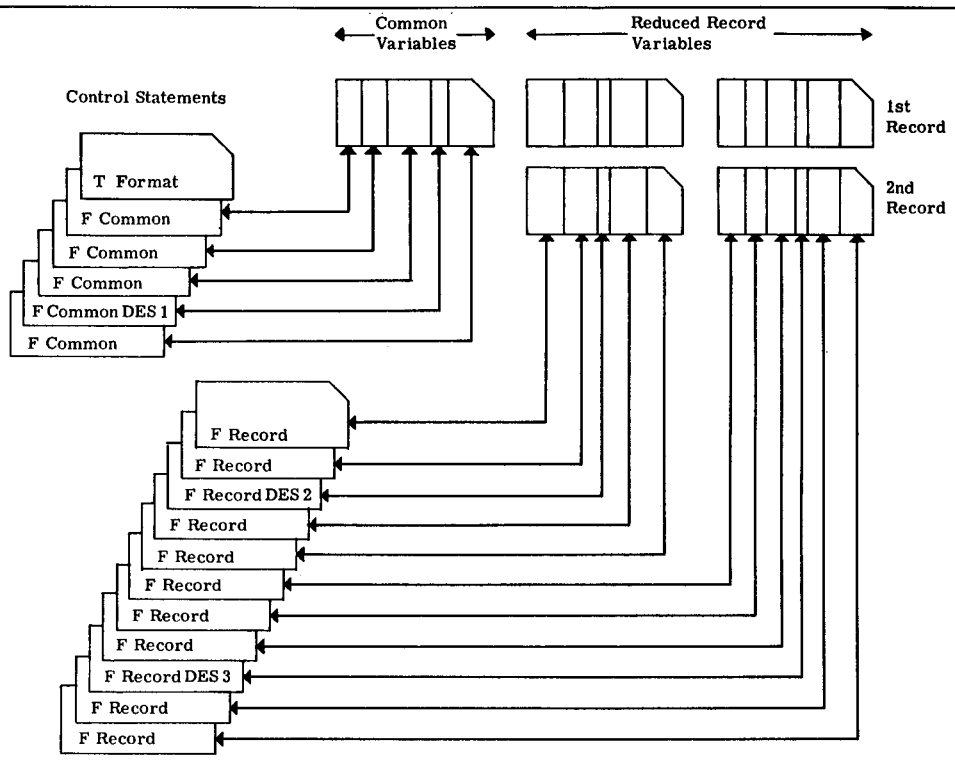


Figure 4(a) Multi-card records

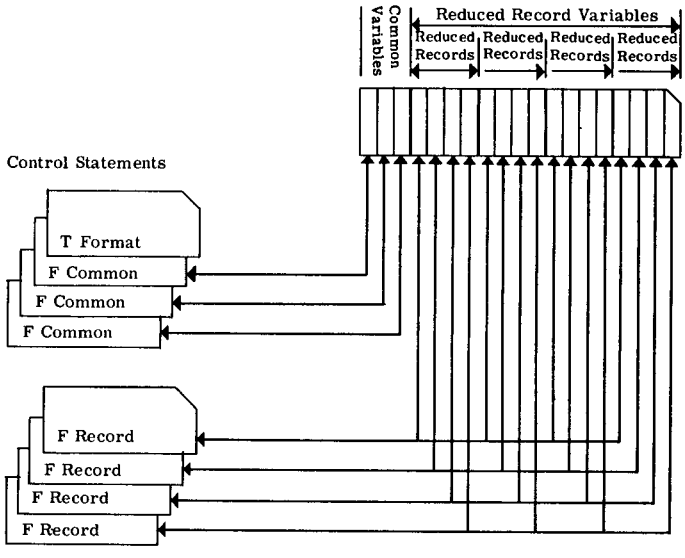


Figure 4(b) Multi-record cards

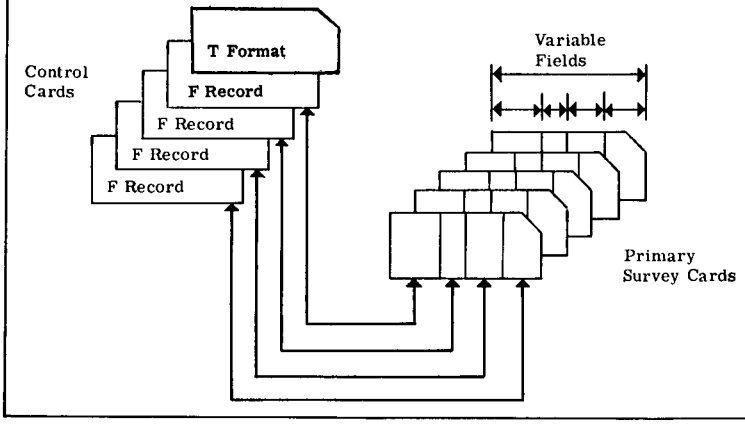


Figure 4(c) Single-record cards

when there are no common variables, one COMMON statement must be input, but no name need be specified and the field size should be specified as zero.

Punching

The statements are punched as follows.

Column 1	F
Columns 2 to 7	COMMON or RECORD These columns need be punched only for the first COMMON statement and the first RECORD statement.
Columns 8 to 13	The name of the variable. The first character must be a letter, the other five may be letters, numbers and spaces in any combination. The names of variables associated with the designation values (see Chapter 2 and Figure 4) of cards (blocks) within multi-card (block) records should begin with DES and no other variable name may start with these characters, e.g. DESTIN must not be used for destinations. Columns 8 to 13 may be left blank if the variable will not be used in the actual analysis, e.g. the survey form sheet number. The field size of such variables must still be specified (columns 15 and 16).
Column 14	B or C or D B indicates a binary variable (each variable in a binary field requires a separate statement). C indicates an alphanumeric variable. D indicates a numeric variable. Numerically coded variables should be specified as type D.
Columns 15 and 16	The number of columns (characters on paper tape) in the field used for values of the variable in the primary survey records. If the field size is less than 10 columns, a zero must be punched on column 15. The field size of a binary variable must not exceed 23 columns. The field size of a designation value is always 01.
Column 17	= or space or any character. The equals sign indicates that range checks are required on the values of this variable. If checks are not required, this column and the rest of the card should be left blank. When the variable concerned is associated with the designation value of a card (or block) in a multi-card (block) record, column 17 should be left blank but column 18 should contain the designation value. For a binary variable, this column should contain the value that is to be represented internally as 1, e.g. Y for values coded as Y and N or 1 for values coded as 1 and 0.
Columns 18 to 74	The range of valid values of the variable, if column 17 contains an equals sign. If the range is continuous, only the upper and lower limits of the range need be specified. For example, if the value x were valid if and only if $5 \leqslant x \leqslant 9$, the range would be specified as (5:9) This type of specification may be used with only numeric variables. Segmented ranges may also be expressed in this way. E.g. (1:3)(5:7)(8:9) If the range consists of only isolated values it may be expressed by specifying the complete set of values. E.g. (A1,B1,C2,D1,G6) The two methods of specification may be used together. E.g. (10:12)(20,35,60)(80:99)

Note the use of colons, commas and parentheses in the two methods of specification. Note also that all values quoted must be in ascending order, according to the relative sequence of characters shown in Appendix 1. The values of alphanumeric variables should be specified in the same number of characters as is given in columns 15 and 16 of this card; numerical values can be specified in up to and including this number of characters. Any unused columns on this card should be left blank; the range specification should always start in column 18.

If a primary survey record is found to contain a value outside the range, the record is excluded from the file of primary survey records on the survey tape. The contents of the card appear on the line printer, followed by a message indicating the name of the variable that has an invalid value.

Example

```
TFORMAT;
FCOMMONVVVVVVD02
FRECORDORIGIND02=(1:99)
```

The common variable will not be used in the analysis.

THE DA-REC STATEMENT

DA-REC statements are used to describe the data variables; this may be done in any or all of the following ways.

1 Data variables may be specified to be identical to primary variables. Primary survey records are then merely copied onto the data tape. If a block of consecutive primary variables is to be copied, one DA-REC statement may be used for the whole block. This is the most efficient way of creating data tapes as it requires less core store and takes less time than any other method. This method of specification is known as block transfer. Isolated variables must be transferred by the second method of specification.

2 If preliminary processing of the primary variables is required, data variables may be specified as arithmetic expressions that are combinations of primary variables, previously defined data variables and constant numerical values. Only D type variables may be included in arithmetic expressions, but any variable may be specified singly if it was not included in a block transfer.

3 Data variables may be specified as *table look-up expressions,* which associate variables with previously created tables. This facility is provided to enable expansion factors, previously input in tabular form (see the RTAB statement in Chapter 5), to be included in the data variables. A data variable is associated with a table, or part of a table, of expansion factors by a DA-REC statement and the variable takes the values of the elements of the table, or the part of the table specified in the statement. In the tabulation phase of the analysis, the data variable can be specified in a TABLES statement. The contents of each cell of the table being created will then be multiplied by the value of the variable that is the appropriate expansion factor. If the table of expansion factors has been created by a run of the analysis program, a new data tape will have to be created and a DACONV statement will be used to incorporate expansion factors into the data variables (see the example later in this chapter).

Arithmetic expressions

When the second method of specification is used, data variables are defined by arithmetic expressions that contain the names of the variables being combined. The following operators may be used.

Operator	*Operation*	*Remarks*
+	Addition	
-	Subtraction	
*	Multiplication	A*B = A×B
/	Division	A/B = A÷B
**	Exponentiation	A**B = A^B

The following rules govern the formation of arithmetic expressions.

1 No two operators may appear adjacent to each other, thus A*-B is invalid. Parentheses may, however, be used as separators, e.g. A*(-B).

2 The order of execution of the operations in an expression is:

1st Exponentiation

2nd Multiplication and division

3rd Addition and Subtraction

Expressions within parentheses are evaluated separately.

3 Values of expressions are rounded to the nearest value that can be expressed in the field specified for the new data variable.

4 Complete expressions should be enclosed in square brackets.

If a data variable is to be identical to a primary variable that was not included in a block transfer, the name of the primary variable may be specified, enclosed in brackets, instead of an arithmetic expression.

Example The length of time, in minutes, for which vehicles are parked is required as a data variable. No primary variable that corresponds exactly is available but the time at which parking commenced is available, in hours and minutes (24 hour clock), as the two primary variables TIM1▽H, for hours, and TIM1▽M, for minutes. Similarly, the end of the parking period is expressed as TIM2▽H and TIM2▽M. The data variable for parking duration is therefore defined in a DA-REC statement by the following expression:

[TIM2▽H*60+TIM2▽M-TIM1▽H*60-TIM1▽M]

Table look-up expressions

When the third method of specification is used, the name of a data variable is associated with the whole or part of a table. The table is referenced by an expression that contains the name and keys of the table and the names of two vectors, which determine the part of the table that is required. If the whole of the table is required, the vectors will be the same vectors that were used in the creation of the table; if different vectors are specified, they must previously have been named and defined by a READ statement, and should include the first and any number of following, consecutive columns of the table. The number of elements in the column (row) vector should not be greater than the number of columns (rows) in the table. The specification of vectors is explained in Chapter 5.

Table look-up expressions are enclosed in square brackets and contain the above information in the following order.

1 The name of the table, followed by a comma.

2 The name of the vector that defines the columns of the table that are to be associated with the data variable, followed by a comma.

3 The name of the vector that defines the rows of the table that are required.

4 Enclosed in parentheses, the name of the variable that is the column key of the table. If the table is multi-level, all the variables in the multiple key should be listed, separated by commas.

5 Enclosed in parentheses, the name of the variable that is the row key or the variables in a multiple row key.

Example See the DACONV statement later in this chapter.

The tables referenced by table look-up expressions will usually have been created from the data records of the same survey and DA-REC statements are used only when the first data tape is created. Table look-up expressions are therefore used mostly with DACONV statements, the expressions take the same form in both statements.

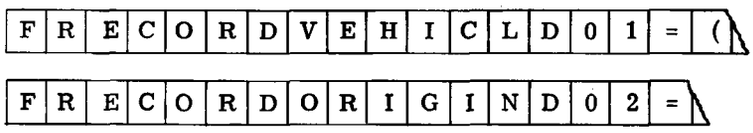
Punching

BLOCK TRANSFER METHOD

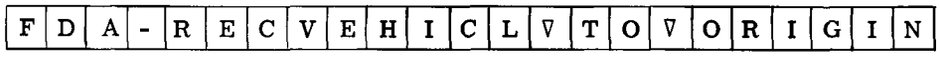
A DA-REC statement that transfers a block of primary survey records to a data tape as data records is punched as follows.

Column 1	F
Columns 2 to 7	DA-REC This is only necessary in the first DA-REC statement. Columns 2 to 7 of any following DA-REC statement cards may be left blank.
Columns 8 to 13	The name of the first primary variable in this block to be transferred.
Column 14	Blank
Columns 15 and 16	TO
Column 17	Blank
Columns 18 to 23	The name of the last primary variable in the block.

Example



If the first and last RECORD statements were as shown above, all the reduced records would be copied onto a data tape if the following DA-REC statement were input.



STATEMENTS THAT CONTAIN ARITHMETIC EXPRESSIONS

Statements that define single data variables by arithmetic expressions are punched as follows.

Column 1	F
Columns 2 to 7	DA-REC
Columns 8 to 13	The name to be given to the data variable The name must conform to the rules given in Chapter 2 and should be unique within the set of data variables; it may, however, be identical to the name of a primary variable.
Column 14	B or C or D or W B indicates that the data variable is a binary variable, C indicates alphanumeric and D indicates numeric. W indicates that the variable is not to be used for tabulation but will appear in an arithmetic expression in a following DA-REC statement.
Columns 15 and 16	The number of characters that values of the data variable will have. If this number is less than 10, column 15 must contain a zero.
Column 17	The number of decimal places in the values of a numeric variable. This column should be blank if the variable is not numeric, and should be blank or zero if the variable is to be used as a table key.

Columns 18 to 74 The arithmetical expression.

The expression must be enclosed in square brackets. Numerical constants must be expressed in normal decimal form and may not have more than six digits; fractional constants must have a zero before the decimal point. If the expression contains a name that is common to both a primary variable and a data variable, it will be taken to refer to the primary variable. Binary variables having the values 0 or 1 may be included.

The expression may be continued in columns 2 to 74 of following cards but should not occupy more than 200 columns in all. If the result of an evaluation of the expression is greater than 999,999 the value 1,000,000 will be substituted.

Example The DA-REC statement for the variable for duration of parking, given as an example of an arithmetical expression, would be punched as follows. The data variable is given the name PARKTM and will take integer values with four digits.

F	D	A	-	R	E	C	P	A	R	K	T	M	D	0	4	0	[	T	I	M	2	∇	H	*	6

0	+	T	I	M	2	∇	M	-	T	I	M	1	∇	H	*	6	0	-	T	I	M	1	∇	M	]

STATEMENTS THAT CONTAIN TABLE LOOK-UP EXPRESSIONS

DA-REC statements containing table look-up expressions are punched as follows.

Column 1 F

Columns 2 to 7 DA-REC

Columns 8 to 13 The name of the variable.

Column 14 D or W

D indicates that the variable is available for use in TABLES statements. W indicates that the variable will appear in only arithmetic expressions in following DA-REC statements.

Columns 15 and 16 The number of characters that values of the variable will have.

This should be 06 or 08. 06 is for tables of integers stored in fixed point form, and 08 is for fixed point fractions or tables stored in floating point form (see Chapter 6).

Column 17 The number of digits after the decimal point in the values of the table elements.

Columns 18 to 74 The table look-up expression.

Example See the DACONV statement.

THE DACONV STATEMENT

DACONV statements are similar to DA-REC statements but are used to describe the variables of a data tape derived from a previously created data tape. The usual reason for using DACONV statements is to incorporate expansion factors in the data variables (by table look-up expressions) when the analysis program has been used to calculate and tabulate the expansion factors. A data tape will have been created for the first run.

As well as table look-up expressions, arithmetical expressions may be used with DACONV statements, but these should not include primary variables. Block transfer may also be used and will result in a block of data variable being copied from the first data tape to the derived tape.

Punching

DACONV statements are punched in the same way as DA-REC statements except for columns 2 to 7, which contain DACONV.

Example The usual reason for re-creating the data tape is to incorporate expansion factors into the data records so that they can be used during tabulation. If the expansion factors have been calculated by a run of the analysis program, a data tape will already have been created and the new data variable for the expansion factors can only be included in the records by defining new data records in terms of the old and the additional variables. The existing variables would be transferred to the new tape by block transfer specification and the new expansion factor variable would be defined by a table look-up expression. In the following example, simple expansion factors are considered to demonstrate this use of the statement.

An external cordon survey has been conducted at four census stations during twelve equal periods of time. An unclassified (i.e. independent of vehicle class) expansion factor was required for each time period at each census station and these have been calculated by the program. A table of the number of vehicles stopped during each time period at each station was produced by TABLES statements and a similar table of the total number of vehicles passing through each station in each period was prepared and input (see the RTAB statement in Chapter 5). The second table was divided by the first by an OPERAT statement to produce a table of expansion factors. The table is called FACTOR and has the variables TIME (for time period) and STATON (for census station) as row key and column key respectively. The row vector is VEC$\nabla\nabla$0, which divides the range of TIME into twelve rows, one per period, and the column vector is VEC$\nabla\nabla$1, which divides the range of STATON into four columns, one per station. The following table look-up expression would be used to reference the whole of this table.

[FACTOR,VEC$\nabla\nabla$1,VEC$\nabla\nabla$0(STATON) (TIME)]

Note that trailing spaces in key names need not be specified.

This expression would be used in a DACONV statement to associate the name of a new data variable with this table and to incorporate the elements of the table into the data records on a new data tape. The following statement would create the new data variable EXPANS, a D type variable with a field width of 8 characters, as the elements of any table produced by an arithmetical table operation will be in floating point form. When the elements are in floating point form, column 17, the number of decimal places, is of significance only if the contents of the data tape are to be printed out; in this example, the values of EXPANS would be printed, in fixed point notation, rounded to one place of decimals, the internal representation would not be affected.

F	D	A	C	O	N	V	E	X	P	A	N	S	D	0	8	1	[	F	A	C	T	O	R	,	V
E	C	∇	∇	1	,	V	E	C	∇	∇	0	(	S	T	A	T	O	N	)	(	T	I	M	E	)	]

Each data record would be expanded to include a value of this new variable. The values of the variable EXPANS will be the elements of the table FACTOR and the value in any particular record will be the element that is the contents of the table cell defined by the values of TIME and STATON in that record. If the variable EXPANS is named in TABLES statements, the contributions that each record makes to the tables being formed will be determined by the value of EXPANS that it contains (see Chapter 6).

THE REJECT STATEMENT

REJECT statements are used to specify validity checks that are to be carried out on the primary survey records or on the values of the data variables. As previously mentioned, REJECT statements should be input after the COMMON and RECORD statements for tests on primary survey records and after DA-REC or DACONV statements if tests are required on data values. Validity checks specified in REJECT statements are independent of the range checks specified in COMMON and RECORD statements and are concerned with the consistency of values within a record. For example, a validity check could be specified that would reject any record of a solo motor cycle with more than two occupants, whereas a range check would reject the record on the grounds of too many occupants only if the number of occupants recorded were greater than that possible for any vehicle. Validity checks are specified in REJECT statements by *consistency expressions*.

Consistency expressions

Validity checks may compare the values of two variables or may compare the values of one variable with a constant. The consistency expression that specifies the check is always of the following form.

[variable name relational operator variable name or constant]

The following relational operators may be used.

Operator	*Meaning*
<	Less than
=	Equal to
>	Greater than
#	Not equal to
>=	Greater than or equal to
<=	Less than or equal to

If any record contains a value that makes the expression true, that record containing that value is rejected; it is excluded from the file of records on the data tape and is printed out with an error message (see Chapter 10) on the line printer. If a constant numerical value is specified at the right hand side of the expression, the variable at the left must be numeric. Numerical constants may be preceded by a minus sign and must have the same number of decimal places as the values of the variable. Alphanumeric values may be compared with an alphanumeric constant and in this case the constant should contain the same number of characters as the values. Alphanumeric constants must be preceded by and followed by an apostrophe.

If two variables are specified in the consistency expression, the value that each takes in any record is compared with the value of the other in the same record. The variables must be of the same type.

Compound consistency expressions may be formed by combining two or more expressions by the logical operator AND and OR, represented by an asterisk and a comma respectively. If an expression contains both logical operators, the truth or falsehood of the simple expressions combined by AND are evaluated before the effect of any OR is taken into account. However, parentheses may be used as in arithmetic expressions, to give the required meaning, e.g. [A>B,C>D*E>F] is not the same as [(A>B,C>D)*E>F]. If A is greater than B but E is not greater than F, the first expression is true but the second is false. Parentheses may be nested to a depth of three.

Punching

REJECT statements are punched as follows.

Column 1	F
Columns 2 to 7	REJECT
Columns 8 to 13	The name of the test. The name should conform to the usual rules governing names. The name of the test appears in the error message that is output on the line printer when the consistency expression is found to be true.
Columns 14 to 74	The consistency expression. This must be enclosed in square brackets.

Example The primary survey records contain values of the variable VEHICL coded numerically and the value 2 represents a motorcycle. The numeric variable OCCUPS, for number of occupants, is also included. It is required to reject all records that show the number of occupants of a motorcycle to be greater than 2. The following REJECT statement would be input immediately after the COMMON and RECORD statements. The test is given the name VTEST1.

F	R	E	J	E	C	T	V	T	E	S	T	1	[	V	E	H	I	C	L	=	2	*

O	C	C	U	P	S	>	2	]

Chapter 5 READ statements

READ statements are used whenever information is input. There are five function statements in this category.

The R-DATA statement describes the input of primary survey records and the RTAB statement enables tables to be input directly to a table tape. Both these statements specify the input medium that is to be used and the name of the tape that is to store the data records or table. The CONDAT statement is used when data records are transferred from one data tape to another.

VECTOR and TEXT statements are used to input specifications of vectors and texts such as table headings. These are then stored on the survey tape until required during the tabulation and output phases respectively. Vectors and texts are also given unique names and these names are used in TABLES and OUTPUT statements to obtain the required vectors and texts.

THE READ TITLE STATEMENT

The read title statement has no other purpose than to introduce a batch of READ function statements.

Punching

Column 1	T
Columns 2 to 5	READ

THE R-DATA STATEMENT

One R-DATA statement initiates the input of one batch of primary survey records. It specifies the input medium that will be used, the name of the data tape on which the corresponding data records are to be stored, and the unit number of the line printer to be used for error messages. The primary survey records should be input immediately after the R-DATA statement and appropriate FORMAT statements should previously have been input.

Punching

The R-DATA statement is punched as follows.

Column 1	F
Columns 2 to 7	R-DATA
Columns 8 to 13	The name of the data tape that is to hold the data records produced from the primary survey records. This tape must be currently allocated. If it becomes full, a scratch tape will automatically be opened as a continuation tape.
Columns 14 to 19	The code that indicates the medium on which the primary survey records are to be input. The codes are as follows. 1 For input from cards. CARDSX X is the unit number of the card reader to be used for the primary survey records.

2 For input from paper tape. PAPERX

X is the unit number of the tape reader to be used for the primary survey records.

3 For input from magnetic tape. MTREAD

See below.

Columns 20 to 25 PRINTX

X is the unit number of the line printer to be used for error messages (see Chapter 10).

Columns 26 to 31 FINISH

If columns 14 to 19 contain MTREAD, columns 26 to 31 may be left blank.

If the R-DATA card contains MTREAD, it should be immediately followed by a card that contains the following information, punched in column 1 and as many following, consecutive columns as are required.

IN *(filename(g/r).subfilename)*

filename is the name of the magnetic tape file that contains the primary survey records and *subfilename* is the name of the required subfile, if the file is composite (if the file does not contain subfiles the full stop and *subfilename* are omitted). g is the file generation number and r is the reel sequence number. Files and subfiles, generation numbers etc., and the general use of magnetic tape are described in the manual *Magnetic Tape*, and input from magnetic tape to the survey analysis program is also described in Part 2 of the *Statistical* manual.

Examples 1 The following batch of statements would initiate the input of primary survey records on card reader 0. The data records that will be created, according to previously input DA-REC statements, will be stored on the data tape DATA 1. If any records are rejected or if other errors occur, the rejected records and appropriate error messages will be output on line printer 0.

```
TREAD
FR-DATADATA▽1CARDS0PRINT0
FINISH
```

2 The following batch of statements would cause the magnetic tape file SURVEYDATA▽▽, generation 3, reel sequence number 0, to be searched for the subfile SURVEY▽▽1▽▽▽, which would then be read. Note that trailing spaces in file and subfile names need not be punched.

```
TREAD
FR-DATADATA▽1MTREADPRINT0
IN(SURVEYDATA(3/0).SURVEY
▽▽1)
```

THE CONDAT STATEMENT

The CONDAT statement describes how a new data tape is derived from a previously created data tape, and is used only when a previously input batch of FORMAT statements has been composed of DACONV statements. Each group of DACONV statements corresponds to one CONDAT statement, which should be input before another group of DACONV statements may be input.

Punching

The CONDAT statement is punched as follows.

Column 1	F
Columns 2 to 7	CONDAT
Columns 8 to 13	The name of the new data tape. This tape must be currently allocated to the analysis program and must be a scratch tape. If more than one reel is required, continuation reels will be opened automatically.
Columns 14 to 19	The name of the original data tape. This tape must be currently allocated to the analysis program and may extend over more than one reel.

Example The following statement would initiate the reading of data records from the tape DATA 0 and writing of new data records, as specified in DACONV statements, to the tape DATA 1.

F	C	O	N	D	A	T	D	A	T	A	∇	1	D	A	T	A	∇	0

THE RTAB STATEMENT

The RTAB statement is used to describe the direct input of a table. The name of the table, the input medium to be used, the size of the table and the name of the table tape on which the table is to be stored are specified in an RTAB statement. The table to which an RTAB statement refers is input immediately after the statements. Only cards or paper tape may be used and tables are input column by column.

The RTAB statement is part of the facilities provided to enable the analysis program to be used for the calculation of expansion factors. When a large number of expansion factors is required, the total vehicle counts may be input in tabular form. A corresponding table is produced, by a TABLES statement, from the survey data, which has already been input. The second table is then divided into the first by a statement of the OPERAT category. This produces a table of expansion factors. In the case of, say, an external cordon survey, a separate expansion factor may be required for each time period at each census station, but any required expansion factors may be calculated, provided that the keys and vectors implicit in the table of total counts are compatible with the data records. For example, an expansion factor for each time period could not be calculated if there were no data variable for time period, and the periods for which total vehicle counts are provided must either be identical to the values of this variable or must be formed by the combination of two or more values. When the table of expansion factors has been produced, DACONV and CONDAT statements may be used to associated the table or part of the table with a data variable. The original data tape may then be released and the tables used in calculating expansion factors may optionally be deleted by XTRACT statements; the analysis proper may then procede, starting with the tabulation phase. In tabulation, expansion factors may be used by specifying the name of the data variable, associated with the appropriate expansion factors, in a TABLES statement. When the table is formed, the contents of each cell will be multiplied by the correct expansion factor.

Table format

Tables must be input by columns, i.e. the first element of the first column is input first, followed by the second element of the first column, the last element of the first column is followed by the first element of the second column and so on. Note that the OUTPUT function statement WRITE outputs tables by rows, but if tables are required in a suitable format for future input by an RTAB statement, output by columns can be achieved by specifying as many table sections as there are columns (see Chapter 8).

The elements of the table to be input may be either integers or fixed point decimals. Integers may have up to 11 digits and decimals may have up to 11 digits before the decimal point and up to 11 after. Numbers may be preceded by plus or minus signs and spaces are permitted between the number and its sign but not between digits, as spaces are used to distinguish between elements; elements must be separated by at least one space character.

Tables may extend over as many cards or paper tape blocks as are required. No terminating symbols are required, but blocks on paper tape should, as always, be followed by the newline character. The maximum block length is 120 characters, not including the newline. A card containing **** in the first four columns should be input immediately after a table; on paper tape this will be a four character block.

A column total may be input with each column of the table, either after or before the column. When the table is read, the analysis program will check that the elements of each column do in fact add up to the stated total. If a discrepancy of more than 1 part in 1,000 is found in any column, an error message will be output to the line printer and the table will be rejected. The program will however, go on to check further columns. Column totals should not be used with tables that have been obtained from a run of the analysis program, as values may have been rounded and errors greater than 1 in 1,000 may have been introduced in this way.

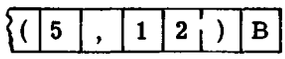
Punching

The RTAB statement is punched as follows.

Column 1	F
Columns 2 to 7	RTAB▽▽
Columns 8 to 13	The name to be given to the table.
Columns 14 to 19	The name of the table tape on which the table is to be stored.
Columns 20 to 25	CARDSX or PAPERX X should be the unit number of the card reader or paper tape reader used for the input of the table. Magnetic tape may not be used.
Columns 26 to 74	The number of rows and the number of columns in the table. The two numbers should be separated by a comma and, together, enclosed in parentheses. The number of rows should precede the number of columns. If column totals are used, the closing parenthesis should be followed by A if the totals are input at the start of their columns or B if the totals are input after their columns, i.e. as a first or last row (the row of totals should be included in the number of rows).

Example A table of twelve columns and four rows is to be input. A column total appears at the bottom of each column, making five rows altogether.

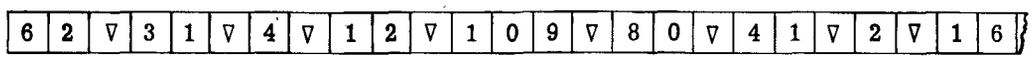
The table is input on card reader 0 and is called VTOTAL, it is to be stored on the table tape TABLE1. The following RTAB statement would be required.

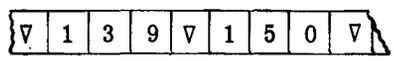


Suppose that the table VTOTAL were as follows.

62	80	150	80	20	19	17	30	51	92	30	17
31	41	90	31	12	7	4	16	30	81	19	3
4	2	1	6	3	1	3	7	2	8	4	1
12	16	8	7	12	9	6	17	31	4	7	6
109	139	249	124	47	36	30	70	114	185	60	27

The table would be punched for input in the following way.





THE VECTOR STATEMENT

When vectors are used in the formation of tables, they are not explicitly defined in the TABLES statement, but are merely specified by name. The vector and its associated name are stored on the survey tape and are retrieved when the name appears in a TABLES statement. The VECTOR statement is used to define a vector and to associate a name with it. This may be done at any convenient time before the

vector is required. In this way, the same vector can be used in the formation of any number of tables, but is defined only once. A VECTOR statement also contains a specification of the type of the vector (alphanumeric or numeric) and the field size required by the elements of the vector.

Vector specification

As was explained in the Introduction, vectors determine how the ranges of table keys are divided into rows and columns. The range of a key must contain only integer values or alphanumeric values, and columns and rows may be allocated to single values or groups of adjacent values; alphanumeric values are considered to be ordered as shown in Appendix 1. Vectors are unnecessary with binary keys as each binary variable is always allocated a column or row and whenever that variable takes the value 1, the appropriate square in the column or row is incremented.

There are basically three methods of vector specification, involving the use of colons, commas and semicolons. Colons indicate one value per column, commas indicate groups of adjacent values, and semicolons indicate that the highest value of the group allocated to one row or column is not adjacent to the lowest value in the next row or column. Vector expressions are enclosed in square brackets.

COLONS

The following vector would produce the illustrated arrangement of single values.

```
[ 2 : 4 : 6 : 7 : 8 : 9 ]
```

2	4	6	7	8

The last element in the vector, 9, is an exclusive value and could have been any integer greater than 8. Note that the elements are arranged in ascending order of magnitude; this is obligatory. The order of rows and columns may, however, be altered in the table operations phase. Alphanumeric elements must also be arranged in ascending order.

```
[ AB: BA: CA: CB: CC ]
```

8C is higher than 8B as C is higher than B. This vector would produce the following arrangement of rows or columns.

AB	BA	CA	CB

COMMAS

The element to the left of a comma is the lower inclusive limit for a column or row, and the element to the right is the upper exclusive limit.

```
[ 11, 21, 31, 41, 51 ]
```

11 to 20	21 to 30	31 to 40	41 to 50

In this example, the division of the range is regular. i.e. there is a common difference between adjacent elements. The following method of specification, which can be used only in the case of D-type variables, would have produced the same arrangement of rows or columns.

```
[ (11, 51, 10) ]
```

Note the use of parentheses. 11 is the lower inclusive limit of the whole range, 51 is the upper exclusive limit, 10 is the common difference. If the common difference is 1, it may be omitted.

```
[ (0, 5) ]
```

0	1	2	3	4

SEMI COLONS

A semicolon indicates that a range of values is to be excluded, i.e. records containing these values will not contribute to the table. The element to the left of the semicolon is the lower inclusive limit of the

excluded range and the element to the right is the upper exclusive limit. In this example, values between 30 and 51 are excluded.

```
[ 11, 21, 31; 51, 61 ]
```

The following rows or columns would be obtained.

11 to 20	21 to 30	51 to 60

Any combination of these methods of specification may be used in the same vector.

```
[1: 2: (4, 10, 2); 14, 20, 25, 31; 35, 41; 50, (60, 63)]
```

1	2	4 and 5	6 and 7	8 and 9	14 to 19	20 to 24	25 to 30	35 to 40	50 to 59

60	61	62

Negative values may appear as vector elements and should be immediately preceded by a minus sign.

Although most of the examples given in this section contain integer values, alphanumeric values may be treated in the same ways, remembering that numbers, starting with 0, precede letters, starting with the space character. Thus a column allocated to values from AB to BC would include AB AC AD AE ..... AY AZ B BA BB BC. Usually, however, it will be possible to use numerical codes for the variables encountered in traffic surveys.

MULTIPLE KEYS

Vectors for multiple keys are specified in the same ways as for simple keys. The elements of the vectors consist of the appropriate values written in the same order as the variable names appear in the key. For example, if the variable VAR$\nabla\nabla$1 had the range 1, 2 and 3 and VAR$\nabla\nabla$2 had the range 1, 2, 3 and 4, a possible vector for the key VAR$\nabla\nabla$1VAR$\nabla\nabla$2 could be:

```
[11: 12: 13: 14: 21: 22: 23: 24: 31: 32: 33: 34: 35]
```

This would produce the following arrangement of rows or columns.

VAR1 = 1				VAR1 = 2				VAR1 = 3			
VAR2				VAR2				VAR2			
1	2	3	4	1	2	3	4	1	2	3	4

The same vector could have been specified as:

```
[ (11, 15) ; (21, 25) ; (31, 35) ]
```

A similar vector for the key VAR$\nabla\nabla$2VAR$\nabla\nabla$1 would be:

```
[ (11, 14) ; (21, 24) ; (31, 34) ; (41, 44) ]
```

Punching

Vector statements are punched as follows.

Column 1	F
Columns 2 to 7	VECTOR
Columns 8 to 13	The name to be associated with the vector. This must conform to the usual rules.
Column 14	C or D C indicates an alphanumeric vector: D indicates a numeric vector.

Columns 15 and 16 The maximum field size required by elements of the vector.

This should be the same as the field size of the associated variable or, in the case of multiple keys, the sum of the field sizes of the associated variables. If this number is less than 10, column 15 must contain zero.

Columns 17 to 74 The vector expression, enclosed in square brackets.

The vector may be continued in columns 2 to 74 of following cards, if necessary. If an error is found in the format of a vector, it is rejected and an error message is output on the line printer.

Example

F	V	E	C	T	O	R	V	E	C	∇	∇	1	D	0	2	[	(	1	,	16	)	]

Note that spaces are not permitted within the square brackets of a vector specification.

THE TEXT STATEMENT

TEXT statements are used to specify items of text that will be used as table headings or row and column labels and to associate a name with each item.

The texts and their names are stored on the survey tape and are specified by name, in OUTPUT statements, when required for printing. Text may be input at any convenient time before it is required.

A table heading, which will indicate the column or row key used in the table, is treated as a continuous string of characters and is called a *line* of text; a series of row or column labels, which will indicate the range of values associated with each row or column in one table, is treated as an *array;* each label being an element of the array. One TEXT statement must be punched for each line or array of text.

Punching

TEXT statements are punched as follows.

Column 1 F

Columns 2 to 7 TEXT∇∇

Columns 8 to 13 The name to be associated with the line or array of text.

Column 14 L or A

L indicates that the statement specifies a line of text; A indicates an array.

Columns 15 to 17 The number of characters in the line or the maximum number of characters in an element of an array of text.

If an array element contains less characters than the maximum, it will be made up to this number by the addition of spaces when it is printed. Columns 15 and 16 should contain zeros if the number of characters is less than 100 or less than 10 respectively.

Columns 18 to 74 The text.

Arrays of text, but not lines, should be enclosed by square brackets, and elements in an array should be separated by commas. Any of the 64 characters shown in Appendix 1 may be used, in any combination. The maximum number of characters in a line of text will depend on the maximum line length that can be accommodated on the line printer but an array may contain up to 1900 characters.

Examples The following TEXT statement associates the name HEAD∇1 with the 11 character line DESTINATION.

F	T	E	X	T	∇	∇	H	E	A	D	∇	1	L	0	1	1	D	E	S	T	I	N	A	T	I	O	N

The following TEXT statement associates the name RLABT3 with the array shown.

```
F T E X T ∇ ∇ R L A B T 3 A 0 0 8 [ 1 ∇ T O ∇ 1 0 ,
1 1 ∇ T O ∇ 2 0 , 2 1 ∇ T O ∇ 3 0 ]
```

Chapter 6 TABLES statements

TABLES statements are used to form basic tables from the data records. The only function statement in this category is the TAB statement. Each TAB statement contains all the information required to produce one table.

THE TABLES TITLE STATEMENT

This statement introduces a batch of TAB statements and, as each batch may process the records from only one data tape, the data tape to be used must be identified in the title statement. The title statement may also nominate a line printer; this will be used for a progress report on the formation of the tables defined by the batch of statements. As each table is formed, its name, number of rows and columns and cell type (fixed point or floating point) are printed out.

Punching

The TABLES title statement is punched as follows.

Column 1	T
Columns 2 to 7	TABLES
Columns 8 to 13	PRINTX X is the unit number of the line printer to be used for the progress report. If a progress report is not required, columns 8 to 13 should be left blank (space characters on paper tape).
Column 14	The last character in the name of the data tape that is to be used by the batch of statements. This is the only unique character in the name.

Example The following statement would introduce a batch of TAB statements that would terminate the contents of the tape DATA$_{\nabla}$1. A progress report would be obtained on line printer 0.

T	T	A	B	L	E	S	P	R	I	N	T	0	1

THE TAB STATEMENT

Each TAB statement contains the following information about one table.

1 The name to be given to the table.

This name will be used in OPERAT statements if the table is used in the table manipulation phase of the analysis, or in an OUTPUT statement to obtain a printout of the basic table.

2 The name of the table tape on which the basic table is to be stored.

3 The names of the data variables to be used as the row and column keys of the table.

4 The names of the two vectors to be associated with the keys.

5 The quantity to be added to the contents of a cell of the table when data record contains values corresponding to the key values of the cell.

This quantity is called the *cell increment.*

6 The quantity by which the increment is to be multiplied.

This quantity is called the *weighting factor.*

The use of increments and weighting factors is explained later in this section; it will be seen that this system provides complete flexibility in the formation of basic tables.

7 Whether or not row and column totals are to be calculated.

8 The name of the parallel table of squares, if one is required.

A parallel table of squares is formed by the same keys and vectors, but the value of each cell increment is squared before being weighted and added to the contents of a cell. This facility may be ignored if it is not required.

Although the way in which a table is defined has been described in Chapter 1 and in the section on the VECTOR statement, the following generalized example is given here to illustrate, in precise terms, the process of tabulation.

Let the value of the data variable or variables that constitutes or constitute the column key of a table be x_i in the ith data record; x_i may be an integer or an alphanumeric value, or, in the case of a multiple key, may be an ordered pair or group of such values. Binary keys are a special case, previously described.

Let the vector associated with this key be of the following form.

$$X_0,X_1,X_2,\ldots\ldots,X_p;X_{p+1}:\ldots\ldots:X_{q-1}:X_q$$

This vector defines q columns.

Consider the following 5 cases.

1 $x_i < X_0$

2 $X_{p-1} \leqslant x_i < X_p$

3 $X_p \leqslant x_i < X_{p+1}$

4 $X_{q-1} \leqslant x_i < X_q$

5 $X_q \leqslant x_i$

In case 1 no contribution to the table will be made by the ith record.

In case 2 there may be a contribution to one of the cells in the pth column.

In case 3 there will be no contribution, as x_i belongs to an excluded range.

In case 4 there may be a contribution to one of the cells in the qth column if $x_i = X_{q-1}$.

In case 5 there will be no contribution.

There will be contributions in cases 2 and 4 only if the ith record also satisfies the conditions defined in the vector associated with the row key of the table.

When a record satisfies both the row and column vectors of a table, the value of the contents of the appropriate cell is increased by the quantity bw, where b and w are respectively the cell increment and the weighting factor as specified in the TAB statement for that table. If w is not specified, only b will be added to the cell contents.

b may be specified as a numerical constant or as a D type data variable; w may be a D type data variable or may be unspecified. When a data variable is specified, the value of the contribution that each record makes to the table is determined by the value of the variable in that record. For example, if b were specified as 2 and w as the data variable VAR$\nabla\nabla$7, when a data record contained the value 5 for VAR$\nabla\nabla$7 the contents of the appropriate table cell (if any) would be increased by 10. This facility enables expansion factors held as a data variable to be used as either the increment or the weighting factor.

In most cases it should be possible to avoid using weighting factors. The variable specified as the weighting factor could usually have been combined with the constant or variable increment at the data

tape creation stage, by means of an arithmetic expression in a DA-REC or DACONV statement. The result variable named in the DA-REC or DACONV statement would then be used as the cell increment. Whenever a weighting factor is used, the cell contents will be stored in floating point form, i.e. in the form $a.e^{x}$: the results of a DA-REC or DACONV arithmetic expression will be stored in fixed point form, provided that the field width specified in the DA-REC or DACONV statement is 06 or less. A table element held in floating point form always occupies two words of core store, even if the number is an integer. In fixed point form, fractions also occupy two words, but integers occupy only one word.

The number of words occupied by the table elements may be of importance when a large table is to be produced. The whole table is formed in core store before being stored on a table tape, and if the table is too large to be held in the available core store, it must be formed in two or more sections, described in separate TAB statements. The number of words of core store that are in use during the formation of a table is given by the following formula:

$$4{,}500 + (r + 2)c \quad e$$

r is the number of rows in the table and c is the number of columns. e is 2 when the elements are stored as floating point numbers or fixed point fractions and 1 when the elements are stored in integer form.

If the table is to be formed in one section, i.e. by only one TAB statement, the value of this expression must be less than the number of words specified in the first SET-UP title statement, i.e. in columns 14 to 18 of the first card.

Punching

The TAB statement is punched as follows.

Column 1	F
Columns 2 to 7	$TAB_{\nabla\nabla\nabla}$
Columns 8 to 13	The name to be given to the table.
Column 14	The last character in the name of the table tape on which the table is to be stored.
Columns 15 to 20	The cell increment. If a constant value is specified, it must be right-justified and any unused columns should contain zeros.
Columns 21 to 26	The name of the data variable that is to be the weighting factor. If no weighting factor is to be used, these columns may be left blank (space characters on paper tape).
Column 27	A or B A indicates that a row of column totals is required and is to be the first row of the table; B indicates that column totals are required as the last row. If column totals are not required, this column is left blank (space character on paper tape).
Column 28	L or R L indicates that row totals are required as the first column of the table, i.e. at the left hand side; R indicates that row totals are required and should be stored as the last column of the table, i.e. at the right hand side. If row totals are not required, this column is left blank.
Columns 29 to 34	The name to be given to the parallel table of squares. If this table is not required, no name should be specified. If a name is given, the table of squares is formed at the same time as the basic table, with row and column totals as indicated in columns 27 and 28 of this card. Tables of squares are always stored in floating point form.
Columns 35 to 40	The name of the vector to be used to define the arrangement of the columns of the table.
Columns 41 to 46	The name of the row vector. If either the row or column key is a binary field, the corresponding vector specification is left blank (space filled on paper tape).

Columns 47 to 74 The names of the data variables that are to be the column and row keys of the table.

Each key is enclosed in parentheses and the column key precedes the row key, e.g. (TERMIN) (VEHICLORIGIN). In this case the column key is the variable. TERMIN, i.e. destination, and the row key is the multiple key VEHICLORIGIN, which will produce a multi-level table with the major row divisions of vehicle class and secondary division according to origin.

The key specifications may be contained in columns 2 to 74 of one continuation card only. All 6 characters, including spaces, of the names used for keys must be specified.

Example The following statement would form a table with the name TAB$_{\nabla\nabla}$1. The table elements are unweighted but the increment is the data variable EXPANS. No row totals are required but column totals are specified as the last row.The column vector is VEC$_{\nabla\nabla}$1; the row vector is VEC$_{\nabla\nabla}$2. The column and row keys are as given in the example of key specification. There is to be no parallel table of squares. The table will be stored on the tape TABLE1.

F	T	A	B	∇	∇	∇	T	A	B	∇	∇	1	1	E	X	P	A	N	S	∇	∇	∇	∇	∇	∇
B	∇	∇	∇	∇	∇	∇	∇	V	E	C	∇	∇	1	V	E	C	∇	∇	2	(	T	E	R	M	I
N	)	(	V	E	H	I	C	L	O	R	I	G	I	N	)										

Chapter 7 OPERAT statements

OPERAT statements are used to form new tables by manipulation of the basic tables. There are 21 function statements in this category; they are used to perform such operations as the merging of two tables or parts of tables, and the multiplication, division, addition or subtraction of the elements of two tables. Many of the function statements are similar and are described in 9 groups in this chapter.

THE OPERAT TITLE STATEMENT

All the new tables produced by a batch of OPERAT statements must be stored on the same table tape. This tape is specified in the title statement of a batch. If a progress report on the formation of the tables is required, the line printer to be used must also be specified in this statement. The names and number of rows and columns of the tables are contained in the progress report, which is printed as the tables are formed. The names of the function statement and the basic tables that were used for each new table are also printed.

Punching

The OPERAT title statement is punched as follows.

Column 1	T
Columns 2 to 7	OPERAT
Columns 8 to 13	PRINTX X is the unit number of the line printer to be used for the progress report. If a report is not required, these columns should be left blank (space characters on paper tape).
Column 14	The last character in the name of the table tape that is to hold the tables produced by this batch of statements.

Example

T	O	P	E	R	A	T	∇	∇	∇	∇	∇	∇	2

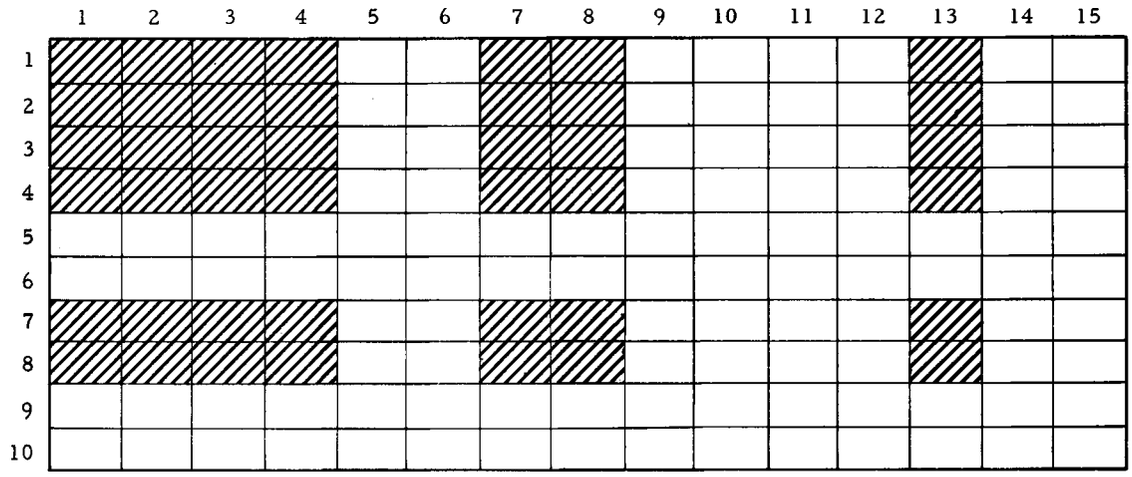
No progress report would be produced; the tables would all be stored on the tape TABLE2.

THE OPERAT FUNCTION STATEMENTS

All function statements in this category specify the name of the statement, the name to be given to the new table, and the names of one or two *operands* and any other items, such as constants, that are to be used in the operation. An operand is an existing table, or sub-table (see below), that is used to form the new table. Each function statement produces one table, by one operation.

Sub-tables

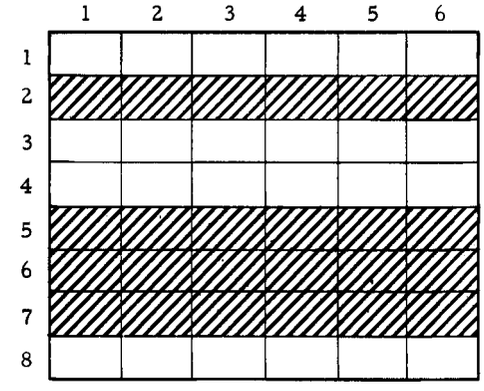
Any sub-set of the elements of a table forms a sub-table. A sub-table may be specified as an operand and is defined by the rows and columns that bound the areas of the table that form the sub-table. For example, the sub-table formed by the shaded areas of the table TAB∇10, shown in the following diagram would be specified in an OPERAT statement, by the expression given below.



TAB$_{\nabla}$10(ROW$_{\nabla\nabla}$1:ROW$_{\nabla\nabla}$4)(ROW$_{\nabla\nabla}$7:ROW$_{\nabla\nabla}$8),(COL$_{\nabla\nabla}$1:COL$_{\nabla\nabla}$4)(COL$_{\nabla\nabla}$7:COL$_{\nabla\nabla}$8)(COL$_{\nabla}$13)

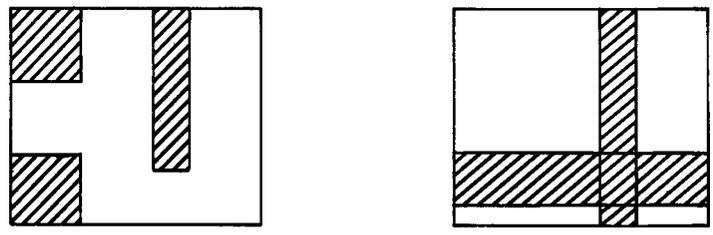
If a sub-table contains the whole of a row, or group of rows, no columns need be specified and, similarly, if whole columns are included no rows need be specified. For example, the following expression defines the sub-table consisting of the shaded areas of table FREQ$_{\nabla}$1 shown in the diagram.

FREQ$_{\nabla}$1 (ROW$_{\nabla\nabla}$2)(ROW$_{\nabla\nabla}$5:ROW$_{\nabla\nabla}$7)



Note that columns and rows are always specified in ascending order.

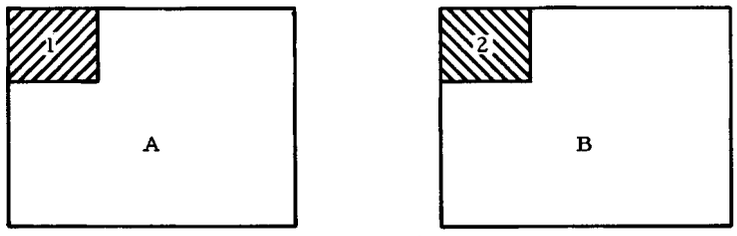
The available methods of sub-table specification exclude sub-tables of the following two forms.



All elements of a table that are not included in the sub-table must lie in rows or columns that are completely excluded from the sub-table.

If two sub-tables are specified in an OPERAT function statement, the table operation will involve only the defined sub-tables. However, the complement of either of the tables from which the sub-tables were derived may be included, unchanged, in the new table. The complement included, if any, is determined by the character punched in column 14 of the function statement card. For example, consider the two tables,

TAB▽▽1 and TAB▽▽2, shown in the diagram. The two sub-tables are indicated by 1 and 2 and the complements by A and B.

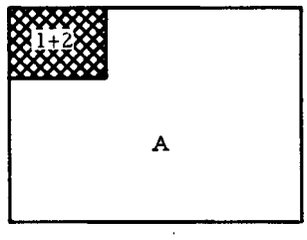


TAB▽▽1 TAB▽▽2

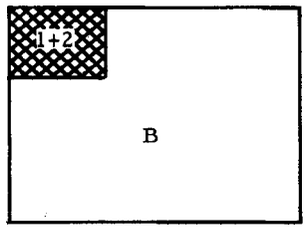
Suppose that the sub-tables 1 and 2 are specified, in that order, as the operands of a function statement that adds corresponding elements of operands. There are 3 possibilities.

1 If column 14 of the card is blank, the new table will consist of only the sums of the elements of the sub-tables.

2 If column 14 of the card contains A, the new table will consist of the sums of the elements of the sub-tables with the complement of the first operand.



3 If column 14 of the card contains B, the new table will consist of the sums of the elements of the sub-tables with the complement of the second operand.



Punching

All OPERAT function statements except WRSUM and WCSUM are punched in the same format as follows.

Column 1	F
Columns 2 to 7	The name of the function.
Columns 8 to 13	The name to be given to the new table.
Column 14	A or B or ▽ See the previous section of this chapter. This column is always left blank for the functions ROWMRG,COLMRG,ROWJN, COLJN,DELETE,T SQRT, T TSP, T FIX,WRSUM and WCSUM.

Columns 15 to 74 The operands and any other items involved in the operation (see individual statements).

The name of each table, or each expression defining a sub-table, that is an operand, must be enclosed in square brackets. This field may be continued in columns 2 to 74 of following cards. All 6 characters, including spaces, of the table names must be specified.

The DELETE statement

This statement creates and names a new table that is merely a sub-table of an existing table.

Example

F	D	E	L	E	T	E	N	T	A	B	∇	1	∇	[	O	T	A	B	∇	1	(	C	O	L	∇	∇

1	:	C	O	L	∇	1	0	)	]

A new table is created from the first 10 columns of the table OTAB∇1 and is given the name NTAB∇1.

The ROWJN and COLJN statements

The ROWJN statement forms a new table by joining two operands; the last row of the first operand is followed by the first row of the second operand. The operands must have the same number of columns.

The COLJN statement similarly joins two operands but the last column of the first operand is followed by the first column of the second operand. The two operands must have the same number of rows.

The ROWMRG and COLMRG statements

The ROWMRG statement forms a new table by merging two operands. The first row of the new table is the first row of the first operand; the second row of the new table is the first row of the second operand; the third row of the new table is the second row of the first operand and so on.

The COLMRG statement merges the two operands by columns. In both cases the two operands must have the same dimensions, i.e. the same number of rows and the same number of columns.

The CONADD, CONSUB, CONMPY and CONDIV statements

In each of these statements one operand and a numerical constant are specified. The CONADD statement forms a new table by adding the constant to each element of the operand, the CONSUB statement subtracts the constant from each element, the CONMPY statement multiplies the elements by the constant and the CONDIV statement divides the elements by the constant.

The constant may be a positive or negative number or an element of an existing table. A table element is specified as a sub-table of one row and one column; a negative constant is preceded by a minus sign and a constant of value less than 1 and greater than -1 must start with 0.

Example

F	C	O	N	M	P	Y	O	D	N	T	A	B	∇	[	O	D	∇	T	A	B	]	1	.	5

The table ODNTAB is formed by multiplying all the elements of the table OD∇TAB by 1.5.

The T SQRT statement

One operand is specified in columns 15 and onwards of the statement card; each element of the new table will be the square root of the corresponding element in the operand.

The name of the statement is punched as T∇SQRT.

The T TSP statement

A single operand is specified; the rows of the new table are the columns of the operand, i.e. the rows and columns are transposed, the first row of the new table is the first column of the operand and the first column of the new table is the first row of the operand, and so on.

The statement name is punched as T∇TSP∇.

The T ADD, T SUB, T MPY and T DIV statements

In each of these statements two operands that have the same dimensions are specified; the new table is formed by an arithmetical operation between the contents of corresponding cells of the operands. T ADD causes the addition of corresponding elements and T MPY the multiplication. T SUB causes the subtraction of the elements of the operand specified second from the corresponding elements of the other, and T DIV causes the division of the first,by the second.

The names of the statements are punched as T∇ADD∇ etc.

The T FIX statement

In this statement only one operand is specified, and the name of the statement is punched as T∇FIX∇. A T FIX statement will cause the specified operand table to be converted from floating-point to fixed-point form. The following tables are stored in floating-point form by the Survey Analysis program, and may require conversion:

1 Tables in which the contribution of a record is a floating-point variable

2 Tables in which the contributions are weighted (see page 36)

3 Tables of squares

4 Tables read in by an RTAB statement (see page 29)

5 Tables formed by arithmetic operations.

Non-integer values will be rounded down to the next integer; that is, towards zero if they are positive and away from zero if they are negative. Any number too large to be converted to a single-length signed integer will be stored as the largest number which can be so converted, that is 8388607. If the operand is already in fixed-point form it will first be floated and then fixed as above.

The ROWMPY, ROWDIV, COLMPY and COLDIV statements

In each of these statements one operand and one row or column of the operand are specified. The new table is formed by the multiplication or division of each element of the operand by the corresponding element of the specified row or column.

Examples

1

```
FROWDIVNEWTAB∇[OLDTAB]
{ROW∇∇1
```

This statement would create the new table as shown.

OLDTAB				NEWTAB			
1	2	3	4	1	1	1	1
5	6	6	4	5	3	2	1
2	10	12	12	2	5	4	3

2

```
FCOLMPYTAB∇15∇[TAB∇∇5(ROW
{∇∇2:ROW∇∇4)]COL∇∇3
```

TAB 5					TAB 15				
1	3	6	2	9					
4	8	2	2	6	8	16	4	4	12
10	3	1	5	6	10	3	1	5	6
3	9	3	6	12	9	27	9	18	36
1	2	7	5	12					

The WRSUM and WCSUM statements

In these statements one operand is specified together with the name of a *weight vector.*

A weight vector consists of a series of numerical values, separated by commas and enclosed in square brackets, and must previously have been defined by a VECTOR statement. Column 14 of a card defining a weight vector should contain W. The WRSUM statement produces a table that contains the rows of the operand, each of which has been weighted by the corresponding element of the vector, i.e. the first row is multiplied by the first element of the vector, the second row by the second element and so on. Rows of column sums may be inserted where required.

The WCSUM statement produces a table of weighted columns and columns of row sums.

In WCSUM and WRSUM statements the contents of columns 1 to 13 are as in the other OPERAT statements, column 14 is blank, and succeeding columns are punched as follows.

Columns 15 to 20	The name of the operand table.
Column 21	[
Columns 22 onwards	(a) The rows (WRSUM) or columns (WCSUM) that are to be weighted to form the first group of the new table.
	(b) The next column contains A or B or ∇ in a WRSUM statement or L or R or ∇ in a WCSUM statement. This is the way of specifying column or row totals for the group of rows or columns specified in (a), and follows the usual conventions. Rows or columns of sums will appear in the table, immediately adjacent to the group of rows or columns to which they apply.
	(c) The next six columns contain the name of the weight vector.
	The sequence (a), (b), (c) can then be repeated until all the groups of rows or columns that are to make up the new table have been specified with their weight vectors and sum specifications. If the same vector is to be used for consecutive groups, it should be specified with only the first group; if the vector is respecified, weighting will start again with the first element. When all the groups have been specified, a closing square bracket is punched and, if the weighted rows or columns are not to be complete rows or columns of the operand table, the bounding columns or rows are specified in another set of brackets, in the usual way.

Example The following WCSUM statement extends over two cards.

Card 1		*Card 2*	
Column 1	F	Column 1	Blank
Columns 2 to 7	WCSUM∇	Columns 2 to 19	8)(ROW∇14:ROW∇25)]
Columns 8 to 13	BA∇102		
Column 14	∇		
Columns 15 to 20	TAB∇80		
Columns 21 to 36	[(COL∇∇4:COL∇15)		
Column 37	R		
Columns 38 to 43	WTV∇13		
Columns 44 to 60	(COL∇22:COL∇27)R]		
Columns 61 to 74	[(ROW∇∇1:ROW∇		

The result table is called BA 102 and its first twelve columns consist of columns 4 to 15 of table TAB 80 (rows 1 to 8 and 14 to 25 only) weighted by elements 1 to 12 of vector WTV 13. The thirteenth column of BA 102 consists of the sums of the rows of the preceding columns. The next six columns are columns 22 to 27 of TAB 80 (rows 1 to 8 and 14 to 25 only) weighted by elements 13 to 18 of vector WTV 13. The last column consists of the sums of the rows of the preceding six columns.

Chapter 8 OUTPUT statements

OUTPUT statements are used to output the tables that are the results of an analysis. There is one function statement in this category, the WRITE statement, which is used to specify the formats in which tables are to be output and the headings that are to accompany tables. Standard headings are available and may be used to avoid the need to input texts (by TEXT statements) and then specify them in WRITE statements.

THE OUTPUT TITLE STATEMENT

Output will usually be in printed form, but may alternatively be obtained in punched form if required for input to another run of the analysis or other program. The OUTPUT title statement is used to specify the type and unit number of the output peripheral to be used and the overall layout of the output obtained by the batch of WRITE statements.

Punching

Column 1	T
Columns 2 to 7	OUTPUT
Columns 8 to 13	PRINTX or TAPEPX or CARDPX X is the unit number of the output peripheral. PRINTX obtains output on a line printer, TAPEPX on a paper tape punch and CARDPX on a card punch. If no peripheral is specified, PRINTO will be assumed.

The remaining two fields are not of fixed lengths, but should be separated from each other and from the previous field by commas.

Field 1 The line length, in characters, of the output to be obtained by the batch of statements.

If a line printer is used, the line length must not be greater than the number of available print positions. If cards are used, the maximum line length is 80 characters and the maximum line length on paper tape is 120 characters.

If this field is left blank, line lengths of 120 character on a line printer or tape punch or 80 characters on a card punch will be obtained.

Field 2 The number of lines per page of output.

On cards, one blank card is output to signify the start of a new page and on paper tape, any unused lines are indicated by a series of newline characters. Any headings that apply to the contents of more than one page are automatically repeated at the tops of the appropriate pages.

If this field is left blank, 60 lines per page will be obtained. Field 2 must always be blank when no line length has been specified.

Examples

T	O	U	T	P	U	T	T	A	P	E	P	0	,	1	0	0	,	4	0

This statement indicates that output is to be on tape punch 0, the line length is to be 100 characters and pages are to contain 40 lines.

T	O	U	T	P	U	T	1	6	0

This statement indicates that output is to be on line printer 0; this is not explicitly specified. Note that when no peripheral specification is given, the line length specification may start in column 8 and the preceding comma may be omitted. A line length of 160 characters is specified in this statement and page lengths of 60 lines will be assumed.

THE WRITE STATEMENT

Each table that is output requires one WRITE statement. This statement specifies:

1 The name of the table to be output.

2 The format in which the numbers that are the elements, or *cell values* of the table are to be printed.

3 The number of sections in which the table is to be printed. If a table is too wide to be accommodated in the line length specified in the OUTPUT title statement, it may be split, between columns, into sections.

4 The spacing of the columns.

5 The names of the texts that are to be used as table and section headings and row and column labels, and the layout of these texts, unless standard headings are used.

Cell value formats

The cell values of a table may be printed in one of three formats. These are:

I format, for integers, e.g. 1026.

F format, for fixed point fractions e.g. -12.75.

E format, i.e. mantissa/exponent format, for any number, e.g. .753E+4 which is equivalent to $.753\times10^{4}$.

E format is useful when the values in a table vary widely in magnitude and will be used automatically when a value cannot be accommodated in the number of characters specified for values in I or F formats, unless six or less characters have been specified, when asterisks will be printed to indicate that a value is too large. Values in E format take the following form.

$$m\ \mathrm{E}\ \pm\ e$$

This is equivalent to $m \times 10^{\pm e}$. m can take values between -1 and -0.1, when it will be preceded by a minus sign, and between 0.1 and 1, when it will be preceded by a space; e can take any integer value.

In the WRITE statement, the specification of the cell value format to be used for a table consists of E, I or F, followed by a number, which is the number of characters to be used for printing the values.

With E formats, this number should include the E and any signs and an initial space if m is positive; the exponent is always expressed in two characters and is always preceded by a plus or minus sign. At least 7 characters must be specified for both positive and negative numbers output in this form. Numbers greater than 10^{19} can be represented in E format only.

Examples For a calculated value of 12345.6789:

E15 would give .123456789E+05

E7 would give .1E+05

For a calculated value of -12345.6789:

E15 would give -.123456789E+05

E7 would give -.1E+05

For a calculated value of 0.02538:

E9 would give .254E-01

With I formats, the number of characters will be the maximum number of digits that a positive value can have; the maximum number of digits in a negative number will be one less, as the first character will be a minus sign.

Examples For a calculated value of 12345:

I5 would give 12345

I6 would give 12345

For a calculated value of -12345:

I6 would give -12345

I5 would give *****

The value cannot be represented, in any form, in 5 characters.

For a calculated value of 12345678:

I8 would give 12345678

I7 would give .1E+08

I6 would give ******

The value cannot be represented in integer form in less than 8 characters and E format requires at least 7 characters.

With F formats, the total number of characters is followed by a decimal point and the number of digits in the fractional part of the value.

Examples For a calculated value of 12.76:

F4.2 would give 12.76

F5.3 would give 12.76

F4.1 would give 12.8

For a calculated value of -12345.678:

F10.3 would give -12345.678

F9.3 would give -.123E+05

If no format specification is given I9 is assumed.

Sections

Tables may be output in sections. The number of sections is specified in the write statement. If too few sections are specified and each section is too wide to be accommodated in the line length being used, the program automatically divides the table into the minimum number of sections necessary. Sections consist of any number of whole columns, and if a table is to be input to another run of the analysis program, it should be output, in punched form, in as many sections as there are columns; the table will then be in the form required by the RTAB statement.

Texts

Any texts previously input by TEXT statements may be output with any table, by specifying the names of the lines or arrays of text in a WRITE statement; the layout of the texts is also specified. Alternatively, standard headings may be used; these require no specification.

STANDARD HEADINGS

If no text specification is given in a WRITE statement, the following arrangement of headings will be output with the table.

The name of the table, as specified in the TAB or OPERAT statement that created the table.

Two blank lines.

SECTION 1 SECTION 2 etc., above each section of the table.

Three blank lines.

COL. 1 COL. 2 etc. up to COL. 999, above the appropriate columns.

Two blank lines.

The table.

One blank line separates each row of the table and the labels ROW 1 etc., up to ROW 999 are printed to the left of the appropriate rows. If the standard 60 line page is used, 25 rows of the table will be printed per page and if the standard I9 cell format is used with a 120 character line length, sections or whole tables of up to 10 columns can be accommodated.

TEXT SPECIFICATION

A text specification consists of four sets of text names. Each set is enclosed by square brackets and the four sets correspond, in the following order, to the table heading, the section headings, the column headings, and the row labels. The name of the text to appear on each line is enclosed by parentheses within the square brackets and blank lines are indicated by numbers between named lines. If more texts than one are to appear on the same line, the names are all specified, in order, within the same parentheses and are separated by commas. Any of the standard headings can be obtained by leaving the appropriate set of square brackets empty, i.e. [ ].

Example [(TEXT$_{\nabla}$1)3(TEXT$_{\nabla}$2,TEXT$_{\nabla}$3)(TEXT$_{\nabla}$4)]

If this specification were the first group of names in a WRITE statement, a table heading would be output with the text named TEXT$_{\nabla}$1 on the first line, followed by three blank lines. The texts named TEXT$_{\nabla}$2 and TEXT$_{\nabla}$3 would occupy the next line, followed by the text named TEXT$_{\nabla}$4 on the sixth line of the heading.

Texts specified in headings will usually be lines of text, but if an array is specified, only the first element will be printed.

Arrays of text will be specified for section and column headings and row labels; consecutive elements are automatically assigned to consecutive sections, columns or rows. If two or more names are specified in one set of parentheses, the first array will be used until all the elements have been printed, when the next array will be used and so on. Lines of text may appear in such specifications and will be regarded as arrays of one element. The elements of arrays specified as row labels will be placed automatically on consecutive lines, but may be separated by any number of blank lines by specifying the number after the closing parenthesis of the specification; the rows of the table will then be spaced by the same amount. Two or more labels may also be associated with each row. Column headings should not occupy more than the number of characters specified in the cell value formats. All printed lines will automatically be centralized in the line width specified in the OUTPUT title statement.

Example [(TEXT$_{\nabla}$8)1(TEXT12)2]

If this specification were for row labels, the labels would be printed as follows.

First label	First element of TEXT$_{\nabla}$8
	Blank line
	First element of TEXT 12 and first row of table
	Blank line
	Blank line
Second label	Second element of TEXT$_{\nabla}$8
	Blank line
	Second element of TEXT 12 and second row of table

Punching

The WRITE statement is punched as follows.

Column 1	F
Columns 2 to 7	WRITE$_{\nabla}$
Columns 8 to 13	The name of the table to be output.
Columns 14 and onwards	The following four fields, which are separated by commas and may be of any length.

Field 1 The number of sections.

If the table is to be printed without division, i.e. as one section, this field should be omitted. If too few sections are specified for the page width being used, the table will automatically be divided into the minimum number of sections necessary to enable the table to be printed.

Field 2 The cell value format specification.

If this field is omitted the format I9 will be assumed.

Field 3 The number of spaces to be inserted between columns of the table.

If this field is omitted, one space will be inserted. Column headings should not extend into inter-column spaces.

Field 4 Any character in the I.C.T. 64 character code (see Appendix 1).

This character will be printed to the left of each cell value as a separator. If this field is omitted, an extra space will be inserted between columns.

The text specification follows field 4. If no specification is present, the standard headings will be obtained. If the table is to be output without division into sections, the section name specification should contain only a number, which will initate the output of that number of blank lines (this may be zero, i.e. [0]).

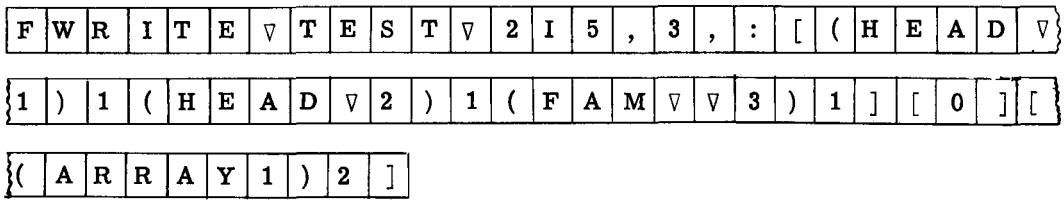
Example

```
READ
FTEXT▽▽HEAD▽1L019TEST▽TRAF
FIC▽SURV EY
FTEXT▽▽HEAD▽2L029HOME▽INTE
RVIEW▽HOUSEHOLD▽DATA
FTEXT▽▽FAM▽▽1L016NUMBER▽IN
▽FAMILY
FTEXT▽▽FAM▽▽3L009-PERSONS-
FTEXT▽▽ARRAY1A005[▽▽1,▽▽2,
▽▽3,▽▽4,▽▽5,▽▽6,▽▽7,▽▽8+,T
OTAL]
FTEXT▽▽ARRAY2A008[▽Z O▽▽▽02
,▽NE▽▽▽04,▽▽▽TOTAL]
```

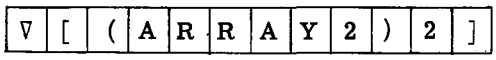
Given the above batch of previously input TEXT statements the following WRITE statement would produce the table shown. The statement extends over two cards; note that the statement is split between two sets of brackets and not inside a heading specification.

```
TOUTPUT
```

1st card



2nd card



The name of the table is TEST∇2; I5 cell value format is used. No specification is made for division into sections as the whole table can be accommodated in the standard page width; consequently the text specification for the section headings is [0].

4103 (7.68)

TEST TRAFFIC SURVEY

HOME INTERVIEW HOUSEHOLD DATA

NUMBER IN FAMILY
- PERSONS -

		1	2	3	4	5	6	7	8+	TOTAL
ZO	02	24	80	72	124	55	48	14	52	469
NE	04	2	34	57	172	55	24	0	8	352
TOTAL		26	114	129	296	110	72	14	60	821

Chapter 9 XTRACT statements

The statements of the XTRACT category are not essential to an analysis but can often be used to minimise the machine time needed. There are two function statements in this category. The ERASE statement is used to remove tables from table tapes and vectors and texts from the survey tape and the LIST statement is used to obtain a printed summary of the tables etc. that remain on the tapes.

When the basic tables have been formed, the vectors will not be needed again and so can be deleted; basic tables that have been used in table operations but are not to be printed out can be removed, and large items of text can be removed when no longer required. By removing redundant items from tapes, the time taken to find any of the remaining items is reduced. The use of an ERASE statement involves the copying of the contents of the tape involved to a scratch tape, and this process takes some time. ERASE statements should therefore not be used unless the search time that will be saved is certain to be greater than the time taken to copy the tape. However, another, convenient use of the ERASE statement is to correct any mistakes in vectors. If such an error is detected after it has reached the survey tape, the vector can be erased and a correct one, with the same name, can be input by a VECTOR statement.

THE XTRACT TITLE STATEMENT

Punching

Column 1	T
Columns 2 to 7	XTRACT
Columns 8 to 13	SCRTCH

This specification results in all tapes that are copied, i.e. the survey tape and any table tape from which tables are erased, to become scratch tapes before being released. Each tape that has been copied can then be used to receive the contents of another tape. In this way, any batch of XTRACT statements requires only one scratch tape, which will receive the contents of the first tape to be copied. If this facility is not required, this field should be left blank; released tapes will then remain named.

THE ERASE STATEMENT

Any batch that contains an ERASE statement will result in the creation of a new survey tape. If tables are erased, then the contents of any table tape involved will also be copied to a scratch tape, with the exclusion of the erased tables.

The old survey tape and any copied table tapes will be released from the analysis program.

Punching

The ERASE statement is punched as follows.

Column 1	F
Columns 2 to 6	ERASE
Column 7	T or V or X

T indicates that tables are to be erased,

V indicates that vectors are to be erased, and

X indicates that items of text are to be erased.

Columns 8 to 74 — The names of the tables, vectors or texts to be erased, separated by commas.

Each ERASE statement may contain names of one type only, i.e. only vectors or only texts or only tables. If required, the statement may be continued in columns 2 to 74 of following cards.

If an item named in an ERASE statement cannot be found, an error message is output.

THE LIST STATEMENT

This statement produces a printed list of the names of tables, vectors or texts remaining on the tapes allocated to the program. Each LIST statement can be used for only vectors or only texts or only tables. This statement should not be confused with the LIST▽▽ statement in the SET-UP category.

Punching

The LIST statement is punched as follows.

Column 1 — F

Columns 2 to 5 — LIST

Column 6 — T or V or X

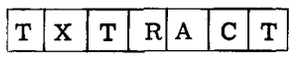
T indicates that the names of the remaining tables are to be listed, V indicates vectors and X indicates texts.

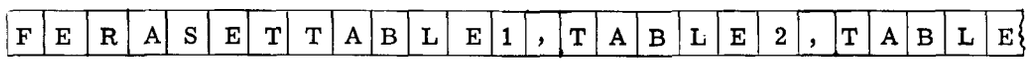
Column 7 — ▽

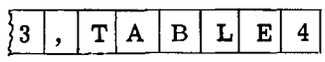
Columns 8 to 13 — PRINTX

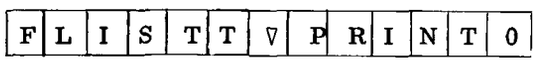
X is the unit number of the line printer on which the lists are required.

Example The tables TABLE1, TABLE2, TABLE3, and TABLE4 are no longer required, and are to be erased. A list of the remaining tables is required, on line printer 0. The following batch of statements would be used.









Chapter 10 Design and operation of survey analysis runs

The control statements described in Chapters 3 to 9 provide the language that is used to direct the course of an analysis. The way in which these statements and, consequently, the analysis program are used is entirely at the discretion of the user, provided that the rules set out in Chapters 3 to 9 are obeyed. However, most traffic survey analyses will be of a straightforward nature and this chapter describes the standard use of the analysis program. The sequence of the batches of statements that are basic to an analysis is given with the necessary operator messages. For the sake of clarity, the four phases of analysis are considered separately and the calculation of expansion factors is not included. The system of calculating and using expansion factors is superimposed on the basic analysis system and a description of how this may be done is given later in this chapter, followed by a list of the error conditions that are detectable.

The design of an analysis run is determined by the number of magnetic tape decks available and the number of tapes required. At least four decks are required for any analysis. The system and survey tapes are required during the whole analysis and data and table tapes are both required during tabulation. If more than one table tape is required and only four decks are available tapes will have to be removed from their decks when they are not being used, and replaced by currently essential tapes; if the data tape extends over more than one reel, more than four decks will be required. The analysis described in this Chapter is split into several runs and in each run only the tapes to be used are allocated to the program. A batch of SET-UP statements appears at the start of each run and, in all but the first run, this contains a TAPES statement that will name all the previously opened tapes that are to be used in the run. Only the names of statements are given in the examples in this chapter but a complete example of an actual analysis is given in Appendix 2.

DATA TAPE CREATION

In this run, a data tape is named and allocated, the formats of the primary survey and data records are described, the data is input and the data records are created. Texts and vectors are also input for future use; this is not essential at this stage but must be done before the vectors and texts are required.

Control statements

TSET-UP	
FOPEN	The data tape is named and allocated.
TFORMAT	
FCOMMON	The number of statements in this batch will depend on the number of primary variables (see Figure 4), and data variables. REJECT statements may also be used.
FRECORD	
FDA-REC	
TREAD	
FVECTOR	
.	Vectors are specified, named and stored on the survey tape.
.	
.	
FTEXT	
.	Texts are specified, named and stored on the survey tape.
.	

FR-DATA — The primary survey records are input after this statement and are followed by the card or block that contains FINISH.

Blank card — This terminates the run; the program halts with the console message END OF RUN, which is also printed on the line printer. The paper tape equivalent of a blank card is a block of one or more space characters only.

Operating instructions

This run would require the system tape, which must always be on unit 0, and at least two scratch tapes with write-permit rings; one scratch tape would become the survey tape and the other the data tape. A large survey may require further scratch tapes as continuation data tapes. The following messages would be typed on the console typewriter for this run.

FI #XDSB

ON #XDSB 0 — This indicates that a scratch tape is to be opened as the survey tape.

ON #XDSB 1 — This message is required only if the control statements are to be input on paper tape.

ON #XDSB 2 — This message is required only if the printout of control statements is to be suppressed. In this case only the first control statement and those containing errors will be printed out.

ON #XDSB 3 — This message is necessary only if a printout of the primary survey records and any tables input by RTAB statements is required, in which case the message is necessary even when the data follows the R-DATA statement on the same piece of paper tape or in the same pack of cards.

GO #XDSB 20

If the same input peripheral is to be used for both the control statements and the primary survey records, one of the following procedures should be adopted. With cards, the pack of primary survey records should be inserted in the pack of control statements immediately after the R-DATA statement. With paper tape, the R-DATA statement should be followed by an additional newline character, i.e. an empty block. When this is read, it will cause the tape reader to be released and the primary survey records can then be loaded. When the FINISH block is reached a FIX message for this peripheral will be obtained and the remaining control statements can be loaded.

Supplementary data tape creation runs

It is possible to add records to the data tape created in the first run. If a significant number of primary survey records has been rejected and contains correctable errors, the corrected records can be input and the data records added to the data tape. There is no need to create a new data tape.

Control statements

TSET-UP

FTAPES — The data tape is re-allocated.

TREAD

FR-DATA — The corrected or additional records are input after this statement and must be followed by a FINISH card or block. Note that FORMAT statements are not required.

Blank card or equivalent.

Operating instructions

The previously created data tape and the survey tape, with write permit rings, are required, as well as the system tape. The following messages should be typed.

FI #XDSB

ON #XDSB 1
ON #XDSB 2
ON #XDSB 3
} These messages may not be required; see the previous run.

ON #XDSB 4 This statement is required only if the data tape extends over more than one reel.

GO #XDSB 20

If the data tape extends over more than one reel, the program will halt with the message ALTER X4 TO RSN when the FR-DATA statement has been read, and the operator should then type the following messages.

AL #XDSB 4 X X is the reel sequence number of the last reel of the data tape that was opened in the previous run.

GO #XDSB This restarts the run.

TABULATION

In this run, the table tapes are allocated and named, and the basic tables are formed and stored.

Control statements

TSET-UP

FTAPES The data tape is re-allocated.

FOPEN One or more table tapes are named and allocated. Note that this statement follows the TAPES statement; in this way the table tapes do not appear in the TAPES statement.

TTABLES

FTAB The tables are created.

.

.

.

Blank card or equivalent.

Operating instructions

This run requires the system and data tapes and the survey tape, with a write-permit ring, in addition to one scratch tape, with write-permit ring, for each table tape that is to be opened.

The console messages are:

FI #XDSB

ON #XDSB 1
ON #XDSB 2 } See the data tape creation run.

GO #XDSB 20

TABLE MANIPULATION

This run is necessary only when operations are to be performed on the tables created in the previous run. Only the table tapes that contain tables to be used in operations need remain allocated to the program but new table tapes may be opened to receive the result tables if necessary (usually there will be room on the existing table tape or tapes and an error message will be output if an attempt is made to write on a full tape).

Control statements

TSET-UP

FTAPES The required table tape(s) are re-allocated.

FOPEN This statement is necessary only if new table tapes are required.

TOPERAT

F *operations statement* The new tables are created.

.

.

.

Blank card or equivalent.

Operating instructions

This run requires the survey and system tapes and the appropriate table tape or tapes. If only one table tape is being used, a scratch tape will also be required for certain table operations.

The console messages are as for the tabulation run.

OUTPUT

Only the tapes that contain the tables to be output need remain allocated.

Control statements

TSET-UP

FTAPES The required table tapes are re-allocated.

TOUTPUT

FWRITE

.

.

.

Blank card or equivalent.

Operating instructions

This run requires the survey and system tapes and the appropriate table tape(s).

Console messages are as for the tabulation run.

LONGER RUNS

The four runs so far described are the only four types of run of which the system is capable and, performed consecutively, form the simplest type of analysis. The four runs, or any two or three consecutive runs, may be performed without a halt by removing the appropriate blank cards. If only one table tape and a data tape of one reel are used, the whole analysis can be run without changing tapes, and TAPES statements are unnecessary if the run is not halted. However, on a multi-programming processor it is advisable to free magnetic tape units, by releasing unnecessary tapes from the analysis program, as soon as possible. Released tapes can also be replaced by any additional tapes required in the analysis. It is not necessary to halt the run by a blank card to change tapes; the normal console messages will indicate when tapes should be loaded.

In the example runs given in this chapter, only the essential statements are indicated. Statements such as REJECT, LIST and ERASE are not shown, as the use of such statements depends on the nature of the survey and the particular requirements of the traffic engineer. The statements RTAB, DACONV and CONDAT are also absent from the simple examples. These statements are primarily intended for use in the calculation and incorporation of expansion factors, which involves a more complex exploitation of the four basic types of run.

The calculation of expansion factors

The first run is the normal data tape creation run except that the enumeration counts (total number of vehicles) may be input, in tabular form to a newly opened table tape by means of an RTAB statement, in the same way that data is input by an R-DATA statement. However, even in the simplest analysis, some primary survey records will be rejected and the table tape can alternatively be opened and the table input during the supplementary data tape creation run when corrected primary survey records are input.

The next run will be a tabulation run to form, from the data records, a table that corresponds to the table already input.

The next run is a table manipulation run in which the newly formed table is divided into the input table, by means of a T DIV statement, to produce a table of expansion factors. This is followed by another data tape creation run in which the existing data records are transferred to a newly opened data tape by DACONV statement, using block transfer. A DACONV statement with a table look-up expression is also used to expand each new data record to include an expansion factor value. The READ statements in this run will then contain the function statement CONDAT.

The analysis will then proceed in the normal way, starting with the normal tabulation run, which should use the new data records; the old data tape should be released.

An example of this type of analysis is given in Appendix 2.

RE-CODING OF VARIABLES

During survey analysis, it is often necessary to convert a set of values of a variable into a new set of values which cannot be derived from the original values by any simple algebraic expression; in traffic surveys this problem arises, for example, if traffic zones are to be renumbered and combined in a non-systematic manner as in the example below:

Original zone numbers	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
New zone numbers required	1	1	2	2	1	3	3	3	4	4	4	2	2	5	5	6	7	8	8	8	8	7	7	6	9	9	9

This procedure is known as *re-coding* of variables, and may be carried out either when a data tape is created from the primary survey records or when a new data tape is derived from it.

The original values (or ranges of values) of the variable must be input as a vector by use of a VECTOR statement, and the new values correcponding to them should be input as a single-column or single-row table with the same number of cells as there are intervals in the vector. A second, dummy vector must be specified with one interval corresponding to all possible values of the variable, and the table must be converted to fixed-point form by use of a T FIX statement.

Either a DA-REC statement or a DACONV statement, depending on the stage at which the re-coding is required, may then be used to re-code the variable.

The statements required for the re-coding of zone numbers discussed above are shown on the survey analysis coding sheet on page 60.

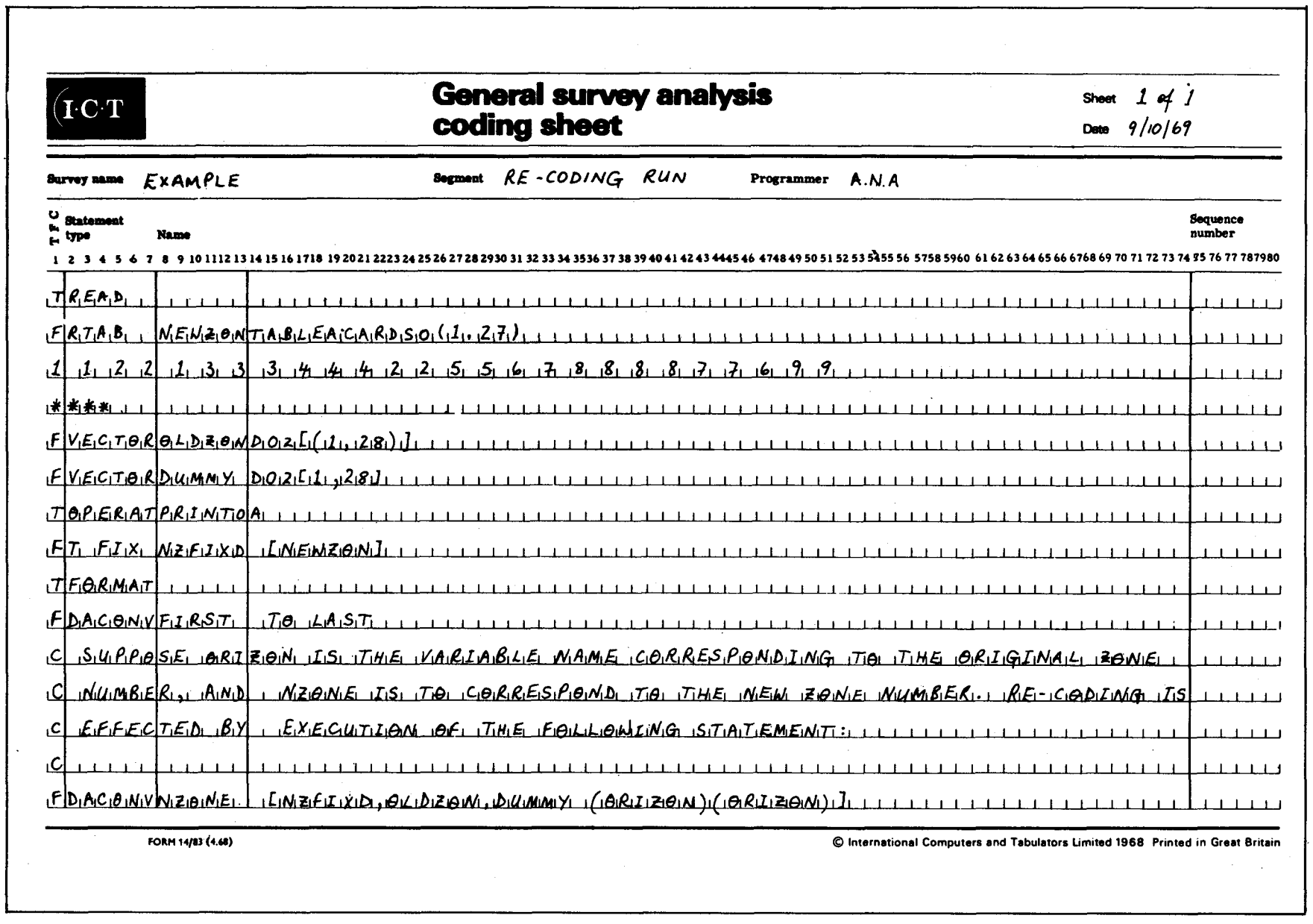
Clearly, the table look-up facility can also be used for multi-level re-coding, that is for the non-algebraic derivation of a value for a variable from the values of two or more variables.

ERROR HALTS

On the occurrence of certain error conditions, the program is unable to continue and halts after typing an error halt message on the console typewriter. Error halt messages are of the following form.

HALTED:- *message*

The following messages may be encountered. They have the meanings and require the operator actions as shown on page 61.



I·C·T

General survey analysis coding sheet

Sheet 1 of 1
Date 9/10/69

Survey name EXAMPLE Segment RE-CODING RUN Programmer A.N.A

Ft.	Statement type	Name	Sequence number
1	2 3 4 5 6	7 8 9 10 11 12 13	14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
T	READ		
F	RTAB	NEWZONTABLEACARDS01(1,27)	
1	1 2 2	1 3 3	3 4 4 4 2 2 5 5 6 7 8 8 8 8 7 7 6 9 9
*	***		
F	VECTOR	OLDZONDO2[(1,28)]	
F	VECTOR	DUMMY1 DO2[1,28]	
T	OPERAT	PRINTOA	
F	T FIX	NZFIXD [NEWZON]	
T	FORMAT		
F	DACONV	FIRST TO LAST	
C	SUPPOSE ORIZON IS THE VARIABLE NAME CORRESPONDING TO THE ORIGINAL ZONE		
C	NUMBER, AND NZONE IS TO CORRESPOND TO THE NEW ZONE NUMBER. RE-CODING IS		
C	EFFECTED BY EXECUTION OF THE FOLLOWING STATEMENT:		
C			
F	DACONV	NZONE [NZFIXD,OLDZON,DUMMY1 (ORIZON)(ORIZON)]	

FORM 14/83 (4.68)

Figure 4.1 Survey Analysis coding sheet for a re-coding run

Message	*Cause*		*Required operator action*
LP	Line printer	not available	Make the required peripheral available and restart the program by GO #XDSB Note: if a second line printer is asked for by HALTED LP but is not available, the program will continue and use only one printer after GO #XDSB.
TR	Paper tape reader		
CR	Card reader		
TP	Tape punch		
CP	Card punch		
NP	Card reader or paper tape reader to be used for primary survey records		
FIRST CARD NOT SET-UP	The first card or paper tape block does not contain TSET-UP.		Restack the cards correctly and restart the program by GO #XDSB.
SS	The program requires more core store.		Make more store available and restart by GO #XDSB. On a single programming machine the analysis must be reformulated.
ERROR IN FORMAT SPECIFICATION	An error in the format specifications has made the creation of the data tape impossible.		Abandon the run and return the control statements to the originator.
0#XDSB HALTED:- RE	There is an error in an R-DATA or CONDAT statement that makes the creation of a data tape impossible. This console message will be accompanied by one of the line printer messages discussed below.		Abandon the run and return the control statements to the originator.

Errors halts arising from R-DATA and CONDAT statements

The program halt causing an error message 0#XDSB HALTED:- RE results from an error in an R-DATA statement or a CONDAT statement, and in either case may be accompanied by one of the line printer messages below:

Message	*Cause*
FUNCTION NAME ERROR	The first letter of the statement name is not R, T or V and no TEXT or VECTOR statements have been read.
INPUT UNIT NAME ERROR	The input specified is not CARDS, PAPER or MTREAD.
DATA TAPE NOT OPEN	The specified data tape has not been opened by an OPEN statement or a TAPES statement.
FORMAT BLOCK MISSING	Some of the required format information cannot be found on the survey tape.
SUBFILE NOT FOUND	The subfile specified in an IN parameter cannot be found.
DES. WITH M.T. INPUT	Designation variables have been previously specified in a COMMON or RECORD statement but input is from magnetic tape.
INPUT SPEC. OVER 80 COLS	The RECORD statement specified more than 80 columns for single card record input.

In the case of a CONDAT statement, the following messages may also appear:

Message	*Cause*
INPUT DATA TAPE EMPTY	The data tape specified in Columns 14 to 19 of the CONDAT statement has no data on it.
OUTPUT TAPE NOT EMPTY	The output tape specified in Columns 8 to 13 of the CONDAT statement already has data on it.

Recovery from error halts

If errors are punched in COMMON, RECORD, DA-REC or REJECT statements and result in an error halt with the message ERROR IN FORMAT SPECIFICATIONS, the run must be abandoned and a new survey tape used for the corrected run. If this error halt results from errors in a DACONV statement, the survey tape already created may be used in the corrected run, which should start with the following input:

1 TSET-UP *xxxxxxnnnnn*, where *xxxxxx* is the survey name and *nnnnn* the core store allocated.

2 FTAPESV*x* . . . *x*, where *x* . . . *x* are the names of the tapes to be re-opened.

3 TFORMAT

4 The complete set of DACONV statements.

If errors are punched in R-DATA or CONDAT statements and result in an error halt with the message RE, the type of error will be listed after the statement on the line printer and the statement can be corrected before the next run.

ERROR MESSAGES

The program tests for various error conditions that do not necessarily cause halts. Error messages are output on the line printer and are printed below and to the right of the printout of the card or block that has caused the condition. Even if the printing of control statements has been suppressed by ON #XDSB 2, erroneous statements and error messages will be printed. At least one line printer must be available to the program throughout the run for the printing of error messages. The same printer will normally be used for all printed output, but if two or more line printers are being used, error messages will be output to the one with the lowest unit number.

General error conditions

Message	*Cause*
CARD SEQUENCE ERROR	The sequence number of a card is less than that of the previous card. The card has been accepted nevertheless.
LONG BLOCK EXCESS IGNORED	A paper tape block that contains a control statement has exceeded 74 characters in length. The excess characters have been ignored.
TITLE CARD ERROR	A title statement has not been recognized, The program has ignored this statement and any association function statements and has skipped to the next title statement;
UNRECOGNIZED CARD IGNORED	A card (block) does not contain C, F, T or V in the first column (as the first character). It has been ignored by the program and the next card read.

Errors associated with the SET-UP group

Message	*Cause*
ERROR IN FUNCTION NAME	Columns 2 to 7 of a card are incorrectly punched. The card has been ignored and the next card read.
TAPE ALREADY OPEN	A tape named in an OPEN statement is already allocated to the program.
ERROR IN TAPE NAME	A tape name in an OPEN or TAPES statement is unacceptable. The statement has been ignored.

Message	*Cause*
9 TAPES ALREADY OPEN	The program is not capable of handling more than 9 data and table tapes simultaneously.
FORMAT ERROR	A statement is incorrectly punched in columns 8 to 13.

LIST STATEMENTS

Message	*Cause*
FORMAT BLOCK MISSING	The required format information cannot be found and will not be output.

Errors associated with the FORMAT group

COMMON and RECORD statements

Message	*Cause*
FUNCTION NAME ERROR	Columns 2 to 7 are incorrectly punched.
RANGE ERROR	A value specified in a range check is less than the previous value.
FORMAT ERROR	There is a punching error in columns 8 and onwards. This also covers the case where two variables beginning with DES occur in the description of a single card or block.
RECORD TOO LARGE	A primary survey record occupies more than 1000 characters.
NO CONTINUATION CARD	A statement is incomplete.

DA-REC and DACONV statements

Message	*Cause*
FUNCTION NAME ERROR	Columns 2 to 7 are incorrectly punched.
FIELD NOT IN LIST OF DATA NAMES	A variable, named on the right hand side of an expression, cannot be found. The statement has been ignored.
INVALID VARIABLE	An alphanumeric or binary variable that is longer than one character is included in an arithmetic expression and the statement has been ignored.
TABLE NOT FOUND	A table, named in a table look-up expression, cannot be found.
TABLES TAPE NOT AVAILABLE	The required table tape has not been opened.
VECTOR NOT FOUND	A vector, named in a table look-up expression cannot be found.
INVALID VECTOR KEY COMBINATION	The vector and the key are of different types, or the field widths of the vector elements are not equal to the field widths of the values of the key, or variables of different types have been specified in a multiple key, or the values of the key are not integers.
FORMAT BLOCK MISSING	A DA-REC statement has been input before any RECORD statements or a DACONV statement has been input before any DA-REC statements.
FORMAT ERROR	There is a punching error in columns 8 and onwards.
RECORD TOO LARGE	A data record occupies more than 498 words.
VECTOR TOO LARGE	A vector specified in a table look-up expression has more elements than there are rows or columns in the table.

REJECT STATEMENTS

Message	*Cause*
ERROR IN FUNCTION NAME	Columns 2 to 7 are incorrectly punched.
PUNCTUATION ERROR	Mis-punching in an expression, or column 14 is incorrectly punched.
VARIABLE *xxxxxx* NOT DEFINED	A variable name cannot be found.
INCONSISTENT RELATIONSHIP	Two variables in an expression are incompatible.
PRIMARY AND DATA VARIABLES MIXED	A variable defined in a COMMON or RECORD statement is in the same REJECT statement as one defined in a DA-REC statement.
FIELD NAME ERROR	A variable name is punched with more than six characters.
STATEMENT EXCEEDS 200 CHARACTERS	The REJECT statement is too long.
CONTINUATION CARD NEEDED	The REJECT statement is incomplete.

Errors associated with the READ group

VECTOR statements

Message	*Cause*
FUNCTION NAME ERROR	Columns 2 to 7 are incorrectly punched.
VECTOR TYPE NOT C OR D	Column 14 is incorrectly punched.
VECTOR FORMAT ERROR	There is a punching error in columns 15 and onwards.
VECTOR TOO LARGE	The vector has too many elements to be stored in one block on the survey tape.
RANGE ERROR	The values in the vector specification are not in ascending order.

R-DATA statements

The following error conditions arise from primary survey records and do not cause the program to halt. Any record in which an error is found will be printed out above and to the left of the error message, even if a printout of the primary survey records has not been obtained by the operator message ON #XDSB 3. If a card (block) contains more than one record, the whole card (block) is printed, but only the erroneous record is excluded from the survey tape. If there is an error in the common part of a group of records, all the records in the group are excluded.

Message	*Cause*
NUMERIC FIELD ERROR COLX	The Xth column is part of a numeric field but contains a non-numeric character.
FIELD XXXXXX INVALID	A value of the variable named in the message (XXXXXX) has failed the range check.
BLOCK NO. XX TOO LONG – NOT PROCESSED	A block of more than 1,000 characters on paper tape has been encountered. The whole block has been rejected.
LAST RECORD INCOMPLETE	The end of a multi-record card (block) has occurred in the middle of a record. No further processing of the final record has been attempted.
DES. ERROR. CARD NOT NO. X IN SET	The expected designation value has not been found in a multi-card (multi-block) record. The record has not been processed and input has continued with the next record or reduced record.
REJECTED ON TEST *xxxxxx*	A record has failed a REJECT test.

RTAB statements

The following error conditions cause the RTAB statement to be ignored. Hence the table is not read.

Message	*Cause*
FUNCTION NAME ERROR	There is a punching error in columns 2 to 7.
RTAB FORMAT ERROR	There is a punching error in columns 8 and onwards.
TOO MANY ROWS	The table has more than the maximum acceptable number of 245 rows.
TABLES TAPE NOT OPEN	The required table has not been opened.

The following error conditions cause the input of the table to be abandoned.

TOO LITTLE DATA	The terminator **** has been encountered before the specified number of columns has been read.
TOTAL WRONG COL. XXX	The column sum for column XXX (the number of the column) does not agree with the elements read. Subsequent columns will be read before the table is abandoned.
TABLES TAPE FULL	The table tape specified in the RTAB statement is full.

The following error conditions do not cause the input of the table to be abandoned.

LONG PAPER TAPE BLOCK	There are more than 120 characters between newlines. The excess characters have been ignored and the first 110 characters are printed on the line that follows this message.
TOO MUCH DATA	Further numbers have been encountered after the table, as specified in the RTAB statement, has been completed and before the terminator **** has been met. These numbers have been ignored.

TEXT STATEMENTS

Message	*Cause*
TEXT *xxxxxx* HAS NOT BEEN DEFINED AS A LINE OR AN ARRAY	Column 14 has been incorrectly punched.
CONTINUATION CARD WANTED FOR *xxxxxx*	A TEXT statement is incomplete. If the item is a line it will be space-filled and stored, if it is an array it will not be stored.
ERROR IN COLUMNS 14 TO 18	Either column 14 is invalid, or columns 15 to 17 contain non-numeric characters, or, in the case of an array, column 18 is not [.
MORE THAN 1968 CHARACTERS DEFINED	The array being defined has too many elements.

Errors associated with the TABLES group

Message	*Cause*
NO DATA TAPE FOR TABULATION	The data specified in column 14 of the TABLES statement is not open.

TAB STATEMENTS

In each case, the TAB statement in error is ignored and the program continues to the next statement.

Message	*Cause*
TABLES TAPE NOT AVAILABLE	The required table tape has not been opened.
WRONG FUNCTION ON CARD	Columns 2 to 7 are incorrectly punched.

Message	*Cause*
INVALID TOTALS SYMBOL	Column 27 does not contain A or B or $_{\nabla}$ and/or column 28 does not contain L or R or $_{\nabla}$.
KEY BRACKETS IN INCORRECT POSITION	The left hand bracket of a key specification is in the wrong position, or is missing.
INVALID KEY/VECTOR COMBINATION	The vector and the key are of different types, or the field widths of the vector elements are not equal to the field widths of the values of the key, or variables of different types have been specified in a multiple key, or the values of the key are not integers.
STATEMENT TOO LONG	The statement extends over a second continuation card.
TABLE TOO LARGE	The table, as defined cannot be formed in the core store available to the program. The table should have been defined in two or more sections by a series of TAB statements.
VECTOR NOT ON TAPE	The vector, named in the TAB statement, cannot be found on the survey tape.
QUANTITY NOT ON TAPE	A variable (cell increment, weighting factor or key), named in the TAB statement, cannot be found on the survey tape
INVALID QUANTITY/WEIGHT FIELD	A numeric field contains non-numeric characters; or columns 21 to 28 contain a numeric field.

Errors associated with the OPERAT group

Message	*Cause*
RESULTS TAPE IS NOT AVAILABLE	The table tape named in the title statement is not currently allocated to the program. The program has continued from the next title statement.
TABLE NOT FOUND	A table, or part of a table, named in the function statement, cannot be found. The program has continued from the next function statement.
SUBTABLE DIMENSIONS DO NOT MATCH	The number of rows or columns of two operand tables are not equal. The program has continued with the next function statement.
END OF TAPE ON UNIT X	The end of tape marker has been deleted. The result table has also been deleted and the rest of the batch of statements ignored; the program has continued with the next title statement.
TAPE HOLDING OPERAND TABLE IS NOT AVAILABLE	The tape holding the specified operand is not currently allocated to the program. The program has continued with the next function statement.
INCORRECT TABLE SPECIFICATION	Columns have been specified as the operand of a statement that operates on rows, or rows have been specified for a column operation. The program has continued with the next function statement.
ERROR IN COLUMN X	Column X is incorrectly punched. The program has continued with the next function statement.
CONSTANT UNACCEPTABLE	The operand is not followed immediately by a number, a minus sign or a one cell sub-table. The program has continued with the next function statement.
INVALID FUNCTION CODE	This function does not exist. The program has continued with the next function statement.
TOO MANY ROWS	The result table would have more than the maximum number of 245 rows for floating point or fixed point fractional elements or 491 rows for integer elements.

Message	*Cause*
NO WORK TAPE AVAILABLE	The program cannot cope with this operation without a work tape, and no such tape has been made available.

WRSUM AND WCSUM STATEMENTS

Message	*Cause*
SHORT OF SPACE	Insufficient core store is available.
WEIGHT VECTOR *xxxxxx* NOT FOUND	The specified weight vector is not defined.
CONTINUATION CARD NOT FOUND	The WRSUM or WCSUM statement is incomplete.
INCOMPATIBLE DIMENSIONS	The weight vector has too few elements.
RESULTING TABLE IS TOO LARGE	The result table has more than 248 rows. (This will usually arise because of the conversion of the table to floating-point form.)

Errors associated with WRITE statements

After any of the following error messages, the table will not be output.

Message	*Cause*	
TABLE NOT FOUND *n* (*n* is a digit in the range 1 to 5)	*n* = 1	The table specified has not been formed.
	n = 2	The tape containing the table is not open.
	n = 3	The tape does not appear to contain the table.
	n = 4 *n* = 5	These values of n indicate table format errors, and the error condition will always be due to program or machine error.
ROW LABEL TOO LONG		The length of the row labels specified is greater than the page width.
HEADING HAS TOO MANY LINES		There are more lines in the table headings that there are lines specified on the page.

If the program has not enough space to store all the text specified, or if some text cannot be found, a list of the names of the omitted text items is printed along with one of the following messages:

TEXT ITEMS OMITTED (NO ROOM)

TEXT ITEMS OMITTED (NOT FOUND)

Errors associated with the XTRACT group

Message	*Cause*
FUNCTION NAME ERROR	Columns 2 to 7 of a statement are incorrectly punched.
PUNCHING ERROR COL. *x*	Column *x* is incorrectly punched.

ERASE STATEMENTS

Message	*Cause*
INSUFFICIENT STORE AVAILABLE NAMES AFTER *xxxxxx* IGNORED	Not enough core store is available to hold all the elements to be erased. All names after *xxxxxx* have been ignored.
END OF REEL on TAPE TABLE *n* LAST TABLE FORMED:- *xxxxxx*	An end of reel marker has been detected on an output tables tape, and the last table copied across was *xxxxxx*.
9 TAPES ALREADY OPEN	The program is designed to handle a maximum of nine data and table tapes simultaneously.

Message	*Cause*
EROR IN TAPE PARAMETER	The OUT parameter has been incorrectly punched, and the program will halt to enable the erroneous card to be corrected.

POST-MORTEMS

If the analysis program goes illegal, or loops, or if any error is suspected in the program post-mortem information can be obtained by the following operator message.

GO #XDSB 28

The end of the post-mortem will be indicated by the following console message.

HALTED:- PM

If spurious error messages seem to be produced, place a blank card after the offending card and run the analysis again, if possible. When the run halts with the message END OF RUN, i.e. when the blank card is encountered, perform a post-mortem.

The post-mortem output consists of 5 to 20 line printer pages, and should be sent for analysis together with the console typewriter output, to

Software Errors,
Management Services Branch,
International Computers Limited,
30/31 Friar Street,
Reading, Berkshire.
RG1 1JP

The program can be restarted after a post-mortem by arranging for the next control statement to be input to be a title statement and by typing the following message.

GO #XDSB 26

Appendix 1 Relative sequence of 1900 Series characters

The relative sequence, in ascending order of magnitude, of all the 1900 Series characters is reproduced in the table below, together with the binary pattern and value of each character. This sequence must be observed when compiling keys and vectors to specify table formats.

Character	*Binary pattern*	*Equivalent numerical value*	*Character*	*Binary pattern*	*Equivalent numerical value*
0	000000	0	R	110010	50
1	000001	1	S	110011	51
2	000010	2	T	110100	52
3	000011	3	U	110101	53
4	000100	4	V	110110	54
5	000101	5	W	110111	55
6	000110	6	X	111000	56
7	000111	7	Y	111001	57
8	001000	8	Z	111010	58
9	001001	9	[	111011	59
:	001010	10	$	111100	60
;	001011	11	]	111101	61
<	001100	12	↑	111110	62
=	001101	13	←	111111	63
>	001110	14			
?	001111	15			
∇	010000	16			
!	010001	17			
"	010010	18			
#	010011	19			
£	010100	20			
%	010101	21			
&	010110	22			
'	010111	23			
(	011000	24			
)	011001	25			
*	011010	26			
+	011011	27			
,	011100	28			
-	011101	29			
.	011110	30			
/	011111	31			
@	100000	32			
A	100001	33			
B	100010	34			
C	100011	35			
D	100100	36			
E	100101	37			
F	100110	38			
G	100111	39			
H	101000	40			
I	101001	41			
J	101010	42			
K	101011	43			
L	101100	44			
M	101101	45			
N	101110	46			
O	101111	47			
P	110000	48			
Q	110001	49			

Appendix 2 Example analysis

Shown in this appendix are the printouts from an analysis of a roadside interview survey. The survey was carried out at four census stations over 3 days (Saturday, Sunday and Monday). Fourteen time periods were taken each day and 16,664 records, each consisting of 8 values, were obtained. Classified enumeration counts were also taken.

The object of the analysis was to produce 2 origin – destination tables. One table was to show the total numbers of vehicles travelling between the given origins and destinations; the other table was to give the equivalent information in rural passenger car units.

The analysis was performed in 3 runs, as shown in Figure 5. Twelve thousand words of the core store of a 1905 processor were allocated to the analysis program and four 1973 tape decks were used. The primary survey records were punched into approximately 1,400 cards. The whole analysis took approximately 12 minutes, which were divided between the runs as follows.

Run 1		3 minutes
Run 2		$2\frac{1}{2}$ minutes
Run 3	part (a)	2 minutes
	part (b)	$4\frac{1}{2}$ minutes

RUN 1

The survey is named and 12,000 words of core store are allocated to the analysis program.

The terminating character for multi-record cards is &.

Columns 2 to 7 are blank; the contents of these columns in the previous statement are implied.

No range is specified for SHEET as values will be used only for the tracing of errors.
Each value is the rural p.c.u. of that class of vehicles; this variable is to be used in run 3 as a weighting factor.

Block transfer is used; the variable SHEET is not needed in the analysis.

This is the first card to contain rejected records; although only 2 records were rejected, the whole card has been printed.

As there is an error in a common value, all records on this card have been rejected.

This is a summary of the data tape creation; 29 of the records printed above have been rejected.

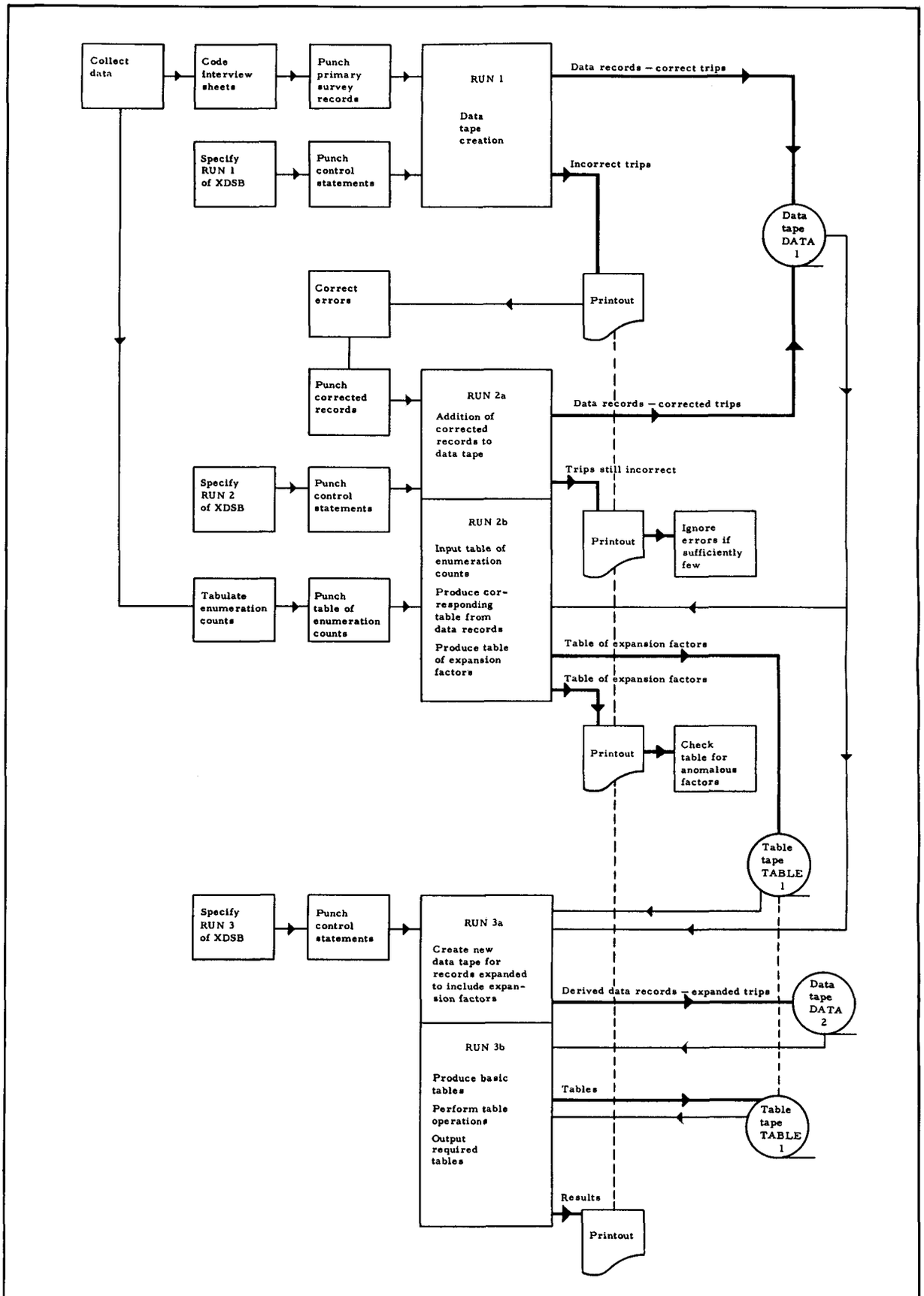


Figure 5 Flow diagram of typical analysis of roadside interview survey

```
   SURVEY ANALYSIS BY #XDSB/2A    DATE  19/04/68  TIME  15/44/41


TSET=UPTEST  12000
FOPEN  DATA 1
TFORMAT&
FCOMMONCENSUSD01=(1,2,3,4)
F       TIME   D02=(08121)
F       DAY    D01=(1,6,7)
F       SHEET  D02
FREGORDVEHICLD01=(1,3)
F       ORIGIND02=(01127)
F       TERMIND02=(01127)
F       PURPOSD01=(1,2)
FDA=RECCENSUS TO DAY
F       VEHICL TO PURPOS
TREAD
FR=DATADATA 1CARDSOPRINTOFINISH
3127021210412012311010811020812020811010811010811011011010811010811011011021918
        FIELD   VEHICL  INVALID
        FIELD   VEHICL  INVALID
2157011080411080011080111080421080411080111230111080111080111080111081921080118
        FIELD   TERMIN  INVALID
2157031080211080011080421080411080411080111080221080411080111080111080111080118
        FIELD   TERMIN  INVALID
2157071080111080211080111081921080111080611080111080111000011081921080111160118
        FIELD   ORIGIN  INVALID
        FIELD   TERMIN  INVALID
2157093080411080111060111081921080411080121080111080111000011080111080111080118
        FIELD   ORIGIN  INVALID
        FIELD   TERMIN  INVALID
2167011080111081921080411080121080111080111080211080011080111080411080411080128
        FIELD   TERMIN  INVALID
2177041080111060021080121231011080111080411060421081011080411080111060121080118
        FIELD   TERMIN  INVALID
5107073060111080111060112060111040112060811040813080111080111021110801111008188
        FIELD   CENSUS  INVALID
        NUMERIC FIELD ERROR COL.  25
        FIELD   VEHICL  INVALID
        NUMERIC FIELD ERROR COL.  37
        FIELD   VEHICL  INVALID
        NUMERIC FIELD ERROR COL.  63
        FIELD   ORIGIN  INVALID
3116053010811221911010811221911220521010811010411010611010414041921010611010618
        FIELD   VEHICL  INVALID
3116061011921010611011921011921011921010611010411011921010623011011010824010418
        FIELD   VEHICL  INVALID
3116061010621010811010811020622021921010821010623010611010811010611010811010618
        FIELD   VEHICL  INVALID
3116071010814010513010611010611010811011011010813011921021921010611010823010818
        FIELD   VEHICL  INVALID
3126051011911012311020611010811010811080811020411011014021921010612010611010618
        FIELD   VEHICL  INVALID
        FIELD   VEHICL  INVALID
3126063010821040823021924021921010812010811010811010611010811011921010611010818
        FIELD   VEHICL  INVALID
        FIELD   VEHICL  INVALID
3126061010611010411010811010811010415020811011921010811010818
        FIELD   VEHICL  INVALID

    NO. OF RECORDS ACCEPTED  16635
    NO. OF RECORDS REJECTED     29
    NO. OF RECORDS ON TAPE   16635

    END OF RUN
```

RUN 2

The data tape is retained and a table tape opened.

The corrected records are input. For this run the operator message ON #XDSB 3 was given and so all the primary survey records being input are printed.

All errors have been successfully corrected.

The table of enumeration counts is input and is printed out in the same way as the primary survey records. The table has been input by rows, instead of by columns, but will be transposed later. A table that is in the wrong form is more easily rearranged after input than before.

```
   SURVEY ANALYSIS BY #XDSB/2A    DATE  19/04/68  TIME  15/51/04

TSET=UPTEST  12000
FTAPES DATA 1
FOPEN  TABLE1
TREAD
FR=DATADATA 1CARDSOPRINTOFINISH
41070730601110801110601110601110401110608110408130801110801110601110801111008 1&
3127023012313020818
2157011080118
2157031080218
2157071010318
2157091020418
2167011080518
2177041060618
3116051041928
31160630104110219 28
3116073010518
3126051021923010618
3126061021923010811020818
FINISH

    NO. OF RECORDS ACCEPTED      29
    NO. OF RECORDS REJECTED       0
    NO. OF RECORDS ON TAPE    16664

FRTAB  ENUMCTTABLE1CARDS0( 6,56)
                42          4          36          5          23          1
                44          4          64          6          35          0
                45          1          79         19          72          3
                62          6          82         12          93          0
                63          3          70          1          68          0
                44         11          57          0          48          2
                48          4          81          4          81          0
                35          6          56          3          69          0
                57          7          52          1          48          1
                72          8          57          0          39          1
                62          2          73          1          67          0
                65          2          65          0          65          0
                48          1          71          1          66          4
                52          0          60          0          67          1
               294         39         155         28          84          3
               277         33         252         30         184          6
               365         43         390         37         364         16
               505         58         392         34         578         28
               414         40         438         14         495         14
               296         30         340         17         322          7
               429         51         404         17         526          5
               373         37         496          8         640         14
               316         30         398          6         302          1
               346         15         322          8         248          5
               315         12         260          6         257          3
               255         12         324          7         322          4
               165          3         279          3         255          3
               149          2         170          3         209          2
               122         20         138         18          39          2
               156         21         320         20          87          9
               288         23         472         17         149          0
               240         20         341         23         120          5
               190         30         215         17         145          5
               192         19         210         13         117          8
               324         13         241          3         210          8
               373         33         225         13         240          1
               596         27         256         10         352          7
```

Test is input for the table of expansion factors; some of this text will also be used in run 3.

Vectors for the table of interviews are input; these vectors will also be used in the table look-up expression in part (a) of run 3.
The table of interviews is formed; there are to be no row or column totals.

Progress report on tabulation.

The table of enumeration counts is transposed.

The table of expansion factors is the result of this operation.

The table of expansion factors is output for checking; 65 lines are specified so that the whole table can be printed on one page.

```
                477          32          359           5          703          20
                369          27          384          21          876          25
                238           5          279           3          760           3
                165           8          221           2          584           5
                122           0          252           2          372           2
                 66          18           79          17           16           3
                101          21          149          17           48           3
                224          26          249          18          157           7
                281          26          236          19          202          12
                238          22          268          20          174           4
                227          27          179           8          112           4
                255          25          213          12          182           5
                364          23          199          14          236           4
                388          42          222          11          247           6
                614          38          178           4          269           4
                268          19          209           4          302           3
                184           7          172           2          252           9
                129           8          189           4          258           4
                109           2           99           0          199           0
****
FTEXT  HEAD 1L019TEST TRAFFIC SURVEY
F      H2    L026TABLE OF EXPANSION FACTORS
F      A1    A005[DAY 1,DAY 1,DAY 6,DAY 6,DAY 7,DAY 7]
F      A2    A005[V.C 1,V.C 3,V.C 1,V.C 3,V.C 1,V.C 3]
F      A3    A006[    08,C   09,E   10,N   11,S T 12,U I 13,S M 14,  E 15,
 P   16,T   17,    18,1   19,    20,    21,    08,C   09,E   10,N   11,S T
  12,U I 13,S M 14,  E 15,P   16,T   17,    18,2   19,    20,    21,    08
 ,C   09,E   10,N   11,S T 12,U I 13,S M 14,  E 15,P   16,T   17,    18,3
   19,    20,    21,    08,C   09,E   10,N   11,S T 12,U I 13,S M 14,  E 1
 5,P   16,T   17,    18,4   19,    20,    21]
FVECTORVECT01D02[11:13:61:63:71:73:74]
F      VECT02D03[(108,122)](208,222)](308,322)](408,422)]
TTABLESPRINT01
FTAB   TABLX11000001              VECT01VECT02(DAY   VEHICL)(CENSUSTIME  )
TOPERATPRINT01

    NAME OF TABLE :- TABLX1
    NUMBER OF ROWS :- 0056
    NUMBER OF COLUMNS :- 0006
    CELL TYPE :- 1 WORD
FT TSP TENMCT [ENUMCT]

    NAME OF RESULT TABLE :- TENMCT
    NUMBER OF ROWS :- 56
    NUMBER OF COLUMNS :-  6
    FUNCTION :- T TSP
    NAME OF FIRST OPERAND TABLE :-  ENUMCT

FT DIV TABEXP [TENMCT][TABLX1]

    NAME OF RESULT TABLE :- TABEXP
    NUMBER OF ROWS :- 56
    NUMBER OF COLUMNS :-  6
    FUNCTION :- T DIV
    NAME OF FIRST OPERAND TABLE :-  TENMCT
    NAME OF SECOND OPERAND TABLE :- TABLX1

TOUTPUT120,65
FWRITE TABEXPF5,2,3,][(HEAD 1) (H2)1][0][(A1) (A2)1][(A3) ]
```

This table contains the greatest number of expansion factors that could be used in this analysis. It would have been easier and quicker to produce, say, one expansion factor for each day at each census point, but the results of the analysis would have been less accurate.

TEST TRAFFIC SURVEY
TABLE OF EXPANSION FACTORS

		DAY 1 V.C 1	DAY 1 V.C 3	DAY 6 V.C 1	DAY 6 V.C 3	DAY 7 V.C 1	DAY 7 V.C 3
	08	1.17	0.80	1.33	1.67	1.53	0.00
C	09	1.63	2.00	1.64	1.00	1.06	0.00
E	10	1.32	0.50	2.03	3.80	1.41	1.50
N	11	1.48	1.50	1.78	2.40	1.58	0.00
S T	12	2.03	1.00	2.00	0.00	1.51	0.00
U I	13	1.52	1.57	1.63	0.00	1.20	2.00
S M	14	1.41	0.50	1.53	0.00	1.23	0.00
E	15	1.59	1.50	1.75	0.00	1.41	0.00
P	16	1.90	3.50	1.53	0.00	1.92	0.00
T	17	3.60	4.00	1.63	0.00	1.34	0.00
	18	1.88	1.00	1.52	1.00	1.86	0.00
1	19	1.86	0.00	1.35	0.00	1.51	0.00
	20	1.60	0.00	1.58	1.00	1.47	1.00
	21	8.67	0.00	8.57	0.00	6.09	0.00
	08	2.75	7.80	1.58	2.33	1.45	3.00
C	09	3.64	5.50	2.23	3.33	1.79	1.20
E	10	3.17	7.17	2.95	2.85	2.96	2.67
N	11	3.18	4.46	2.97	2.62	5.16	5.60
S T	12	3.94	5.71	2.98	1.56	3.59	7.00
U I	13	2.47	5.00	3.04	2.43	1.91	2.33
S M	14	2.96	12.75	2.83	17.00	3.44	1.25
E	15	2.94	2.64	3.67	8.00	4.32	14.00
P	16	2.93	7.50	2.84	3.00	2.03	0.00
T	17	3.49	3.75	2.18	2.67	1.77	2.50
	18	3.62	6.00	2.45	3.00	2.82	3.00
2	19	5.43	4.00	2.72	3.50	3.58	4.00
	20	2.50	3.00	2.94	3.00	3.59	3.00
	21	10.64	0.00	8.95	0.00	9.50	0.00
	08	1.31	2.50	2.94	2.57	1.15	1.00
C	09	1.23	1.40	5.16	2.00	1.43	2.25
E	10	2.04	2.87	4.63	2.83	1.38	0.00
N	11	1.54	1.43	2.84	1.44	1.14	1.00
S T	12	1.30	2.00	2.01	1.70	1.31	1.25
U I	13	1.30	1.46	2.12	3.25	1.11	1.60
S M	14	1.72	1.30	1.65	3.00	1.38	2.67
E	15	1.94	2.06	1.62	1.62	1.35	0.00
P	16	2.45	3.37	1.37	2.00	1.68	1.75
T	17	2.08	1.52	1.81	1.25	2.97	4.00
	18	1.90	2.25	2.46	2.10	3.41	2.78
3	19	1.32	0.83	1.87	3.00	3.58	0.75
	20	1.21	1.33	1.71	0.00	2.39	5.00
	21	2.44	0.00	7.00	0.00	5.72	0.00
	08	3.14	3.60	1.88	1.89	2.00	3.00
C	09	2.97	10.50	2.07	2.83	2.29	0.00
E	10	2.36	3.71	2.96	3.00	1.30	1.40
N	11	2.28	4.33	2.31	1.73	2.13	1.20
S T	12	1.65	3.67	2.09	2.00	2.29	2.00
U I	13	4.83	4.50	2.32	8.00	1.81	2.00
S M	14	1.72	1.92	3.44	2.00	2.25	5.00
E	15	1.99	2.30	1.81	2.80	2.29	1.33
P	16	3.40	4.67	2.44	11.00	2.42	3.00
T	17	2.43	2.11	2.34	0.00	2.77	4.00
	18	2.11	1.58	2.35	2.00	3.18	0.00
4	19	2.63	3.50	1.45	2.00	1.47	3.00
	20	1.42	1.60	3.15	2.00	1.61	2.00
	21	2.37	2.00	3.30	0.00	2.84	0.00

END OF RUN

RUN 3

The table tape is required by the table look-up expression in part (a) of this run.

Block transfer is specified for the existing data records.
The whole of the table is referenced by the table look-up expression.

The table tape may now be released as all the required information has been extracted, although not yet output to the new data tape. Only four decks are available.

Vectors and texts are stored for use later.

The original data tape is no longer required and must be released to allow the table tape to be re-allocated.

Origin-destination tables, incorporating expansion factors, are formed for Monday (TABLE 1), Saturday (TABLE 2) and Sunday (TABLE 3).

Similar tables are formed, using VEHICL as a weighting factor; these tables will thus give rural p.c.u. values (see run 1).

The data tape will not be required again and so is released.

The next four statements calculate the average daily flows in vehicles.

The next four statements calculate the average daily flows in passenger car units.

The standard layout is used for the output of the tables.
The first table is output in 3 sections.

```
   SURVEY ANALYSIS BY #XDSB/2A    DATE  19/04/68  TIME  15/54/10


TSET-UPTEST  12000
FTAPES TABLE1
TFORMAT
FDACONVCENSUS TO PURPOS
FDACONVEXPFACD082[TABEXP,VECT01,VECT02(DAY,VEHICL)(CENSUS,TIME)]
TSET-UP
FTAPES DATA 1
FOPEN  DATA 2
TREAD
FCONDATDATA 2DATA 1

    NO. OF RECORDS ACCEPTED  16664
    NO. OF RECORDS REJECTED      0
    NO. OF RECORDS ON TAPE   16664

FVECTORVECT03D03[(101,128)]
F       VECT04D03[(601,628)]
F       VECT05D03[(701,728)]
F       VECT06D02[(01,28)]
FTEXT  H3    L024ORIGIN-DESTINATION TABLE
F      H4    L021   AVERAGE DAILY FLOW
F      H5    L008VEHICLES
F      H6    L013RURAL P.C.U'S
F      S1    A011[DESTINATION,DESTINATION,DESTINATION]
F      A4    A005[   01,   02,   03,   04,   05,   06,   07,   08,   09,
  10,   11,   12,   13,   14,   15,   16,   17,   18,   19,   20,   21,
 22,   23,   24,   25,   26,   27,TOTAL]
TSET-UP
FTAPES DATA 2TABLE1
TTABLES      2
FTAB   TABLE11EXPFAC      BR      VECT03VECT06(DAY   TERMIN)(ORIGIN)
F      TABLE21EXPFAC      BR      VECT04VECT06(DAY   TERMIN)(ORIGIN)
F      TABLE31EXPFAC      BR      VECT05VECT06(DAY   TERMIN)(ORIGIN)
F      TABLE41EXPFACVEHICLBR      VECT03VECT06(DAY   TERMIN)(ORIGIN)
F      TABLE51EXPFACVEHICLBR      VECT04VECT06(DAY   TERMIN)(ORIGIN)
F      TABLE61EXPFACVEHICLBR      VECT05VECT06(DAY   TERMIN)(ORIGIN)
TSET-UP
FTAPES TABLE1
TOPERAT      1
FCONMPYTABLE7 [TABLE1]5
FT ADD TABLE8 [TABLE2][TABLE3]
FT ADD TABLE9 [TABLE7][TABLE8]
FCONDIVTABL10 [TABLE9]7
FCONMPYTABL11 [TABLE4]5
FT ADD TABL12 [TABLE5][TABLE6]
FT ADD TABL13 [TABL11][TABL12]
FCONDIVTABL14 [TABL13]7
TOUTPUT
FWRITE TABL103,I5,3,1[(HEAD 1)1(H3,H4)1(H5)1][[(S1)1][(A4)1][(A4) ]
```

TEST TRAFFIC SURVEY

ORIGIN-DESTINATION TABLE AVERAGE DAILY FLOW

VEHICLES

DESTINATION

	01	02	03	04	05	06	07	08	09	10
01	2	10	2	774	106	348	13	1680	1	152
02	1	3	1	26	4	3	1	63	1	4
03	0	0	0	0	0	0	2	1	0	0
04	625	12	0	43	10	38	13	1452	0	9
05	106	3	0	23	0	1	0	56	0	1
06	368	12	0	40	0	3	1	23	0	1
07	12	2	0	17	0	0	0	6	0	0
08	1885	82	3	1210	16	1	4	32	0	83
09	5	0	0	0	0	0	0	0	0	0
10	217	6	0	12	1	1	0	130	0	2
11	0	0	0	0	0	0	0	0	0	0
12	0	0	0	3	0	0	2	2	0	0
13	18	12	0	0	2	2	15	3	0	0
14	0	0	0	0	0	0	0	2	0	0
15	0	0	0	0	0	0	0	0	0	0
16	19	1	2	14	0	0	0	0	0	2
17	29	1	1	31	0	3	0	66	0	4
18	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	4	0	0
20	0	0	0	0	0	0	0	0	0	0
21	0	0	0	18	0	5	0	17	0	0
22	0	0	0	7	1	1	0	8	0	0
23	87	9	0	105	25	0	1	2	0	3
24	12	1	0	7	0	0	0	0	0	0
25	9	3	0	14	0	0	0	0	0	0
26	7	1	0	6	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0
TOTAL	3402	157	9	2349	166	407	53	3547	2	261

TEST TRAFFIC SURVEY

ORIGIN-DESTINATION TABLE AVERAGE DAILY FLOW

VEHICLES

DESTINATION

	11	12	13	14	15	16	17	18	19	20
01	0	0	0	0	0	31	0	0	438	0
02	1	0	0	0	0	2	0	0	83	0
03	0	0	1	0	0	0	0	0	0	0
04	0	0	0	0	2	7	0	0	492	0
05	0	0	0	0	0	3	0	0	157	0
06	0	0	0	0	0	0	0	0	67	0
07	0	0	0	0	0	0	0	0	24	0
08	0	0	0	0	0	0	1	0	639	0
09	0	0	0	0	0	0	0	0	5	0
10	0	0	0	0	0	0	0	0	69	0
11	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	42	0
17	0	0	0	0	0	6	0	0	144	0
18	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	4	0
20	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	1	0	0	33	0
22	0	0	0	0	0	0	0	0	20	0
23	0	0	0	0	0	0	0	0	246	0
24	0	0	0	0	0	0	0	0	8	0
25	0	0	0	0	0	0	0	0	16	0
26	0	0	0	0	0	0	0	0	13	0
27	0	0	0	0	0	0	0	0	0	0
TOTAL	1	0	1	1	2	52	1	0	2501	0

TEST TRAFFIC SURVEY

ORIGIN-DESTINATION TABLE AVERAGE DAILY FLOW

VEHICLES

DESTINATION

	21	22	23	24	25	26	27	TOTAL
01	17	8	53	4	11	131	0	3781
02	3	0	3	0	2	3	0	205
03	0	0	0	0	0	0	0	5
04	28	0	70	6	11	11	0	2831
05	3	0	24	0	2	2	0	383
06	3	1	1	0	0	0	0	522
07	1	0	0	0	0	1	0	63
08	33	1	0	0	0	1	0	3992
09	0	0	0	0	0	0	0	10
10	0	0	7	1	0	0	0	446
11	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	7
13	0	0	0	0	0	0	0	52
14	0	0	0	0	0	0	0	2
15	0	0	0	0	0	0	0	0
16	3	0	0	0	0	0	0	83
17	3	0	19	0	0	1	0	307
18	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	8
20	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	75
22	1	0	1	0	0	2	0	42
23	14	1	0	0	0	0	0	493
24	0	0	0	0	0	0	0	28
25	2	0	0	0	0	0	0	44
26	0	0	0	0	0	0	0	28
27	0	0	0	0	0	0	0	0
TOTAL	111	12	177	13	26	153	0	13407

The second table is output in 3 sections.

```
F       TABL143,15,3,1[(HEAD 1)1(H3,H4)1(H6)1][(S1)1][(A4)1][(A4) ]
```


TEST TRAFFIC SURVEY

ORIGIN-DESTINATION TABLE AVERAGE DAILY FLOW

RURAL P.C.U'S

DESTINATION

	01	02	03	04	05	06	07	08	09	10
01	3	10	2	821	133	408	16	1894	1	169
02	1	10	2	37	4	5	1	74	1	4
03	0	0	0	0	0	0	2	1	0	0
04	672	12	0	46	10	43	13	1746	0	14
05	117	4	0	30	0	1	0	68	0	1
06	420	16	0	47	0	3	1	28	0	1
07	12	2	0	20	0	0	0	6	0	0
08	2084	89	3	1438	17	1	4	42	0	118
09	5	0	0	0	0	0	0	0	0	0
10	235	6	0	30	1	1	0	145	0	2
11	0	0	0	0	0	0	0	0	0	0
12	0	0	0	3	0	0	2	2	0	0
13	19	12	0	0	2	2	18	3	0	0
14	0	0	0	0	0	0	0	2	0	0
15	0	0	0	0	0	0	0	0	0	0
16	19	1	2	14	0	0	0	0	0	2
17	29	2	1	34	0	4	0	84	0	4
18	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	4	0	0
20	0	0	0	0	0	0	0	0	0	0
21	0	0	0	18	0	9	0	17	0	1
22	0	0	0	9	1	1	0	11	0	0
23	87	9	0	124	44	0	2	5	0	3
24	13	1	0	7	0	0	0	0	0	0
25	9	3	0	14	0	0	0	0	0	0
26	7	1	0	6	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0
TOTAL	3730	176	11	2700	212	478	59	4132	2	318

TEST TRAFFIC SURVEY

ORIGIN-DESTINATION TABLE AVERAGE DAILY FLOW

RURAL P.C.U'S

DESTINATION

	11	12	13	14	15	16	17	18	19	20
01	0	0	0	0	0	31	0	0	453	0
02	2	0	0	0	0	2	0	0	94	0
03	0	0	1	0	0	0	0	0	0	0
04	0	0	0	0	2	7	0	0	513	0
05	0	0	0	0	0	3	0	0	175	0
06	0	0	0	0	0	0	0	0	69	0
07	0	0	0	0	0	0	0	0	34	0
08	0	0	0	0	0	0	2	1	719	0
09	0	0	0	0	0	0	0	0	5	0
10	0	0	0	0	0	0	0	0	70	0
11	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	1	0
14	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	42	0
17	0	0	0	0	0	6	0	0	154	0
18	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	5	0
20	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	3	0	0	33	0
22	0	0	0	0	0	0	0	0	20	0
23	0	0	0	0	0	0	0	0	260	0
24	0	0	0	0	0	0	0	0	8	0
25	0	0	0	0	0	0	0	0	16	0
26	0	0	0	0	0	0	0	0	14	0
27	0	0	0	0	0	0	0	0	0	0
TOTAL	3	0	1	1	2	54	2	1	2686	0

TEST TRAFFIC SURVEY

ORIGIN-DESTINATION TABLE AVERAGE DAILY FLOW

RURAL P.C.U'S

DESTINATION

	21	22	23	24	25	26	27	TOTAL
01	22	8	53	5	11	150	0	4189
02	3	0	3	0	2	3	0	249
03	0	0	0	0	0	0	0	5
04	44	0	80	12	11	11	0	3235
05	3	0	32	0	2	2	0	439
06	3	1	1	0	0	0	0	592
07	1	0	0	0	0	1	0	76
08	33	1	0	0	0	1	0	4553
09	0	0	0	0	0	0	0	10
10	0	0	12	1	0	0	0	504
11	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	7
13	0	0	0	0	0	0	0	57
14	0	0	0	0	0	0	0	2
15	0	0	0	0	0	0	0	0
16	3	0	0	0	0	0	0	83
17	3	0	20	0	0	1	0	342
18	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	9
20	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	82
22	1	0	1	0	0	2	0	46
23	14	1	0	0	0	0	0	548
24	0	0	0	1	0	0	0	29
25	6	0	0	0	0	0	0	48
26	0	0	0	0	0	0	0	29
27	0	0	0	0	0	0	0	0
TOTAL	135	12	201	19	26	172	0	15136

END OF RUN

Appendix 3 #X3GX

TITLE

Traffic survey analysis tables conversion program.

HARDWARE REQUIREMENT

1728 words of core store.
1 paper tape reader or 1 card reader.
120 character-position line printer.
2 magnetic tape decks (excluding program library tape)

Note: If no hardware floating-point equipment is available then a suitable floating-point Executive is required.

USE OF PERIPHERALS

Peripheral	*Use*	*Allocation and Release*
Card Reader CR0 Paper Tape Reader TR0	Input of directives and parameters.	The reader is allocated at the start of the run and released when all parameters have been read in.
Magnetic tape deck MT0	Holds input tape containing tables from the Survey Analysis program.	The magnetic tape deck MT0 is retained throughout the run.
Magnetic tape deck MT1	Holds output tape containing the tables after conversion.	The magnetic tape deck MT1 is retained throughout the run.
Line Printer LP0	Listing of output and error messages.	The line printer is retained throughout the run.

The peripheral unit numbers given above are those to be used if the program is to be controlled by the GEORGE operating system.

DESCRIPTION

General

The program converts floating-point tables produced on magnetic tape by the 1900 Series Survey Analysis program (#XDSB) into integer form. The tables converted are generally origin-destination tables.

It is essential to use the conversion program #X3GX only when floating-point tables are to be input to the Capacity Restraint Traffic Assignment and Furness Prediction Mark 2 packages. Integer-form tables may be input directly as described in the manuals describing the use of these packages.

The following tables produced by the Survey Analysis program (#XDSB) will be in floating-point form:

1 Tables in which an entered record is a floating-point variable

2 Tables in which the contributions of records are weighted (see page 36)

3 Tables of squares

4 Tables read in by an RTAB statement (see page 29)

5 Tables formed by arithmetic operations.

The integer form tables are output on magnetic tape, and can be used as input data for the Traffic Assignment and Furness Prediction Mark 2 packages. Up to 10 tables may be converted in one run, and, if required, the resulting integer form tables may be multiplied by a scaling factor specified by the user.

The program is directed by the input of parameters (see Input Parameters, below).

Input parameters

Parameters may be input on punched cards or on paper tape. Each parameter must be punched left-justified in an 80 column card, or punched in 8 track paper tape and followed by a newline character. The parameters are input in the following order:

1 *File name of input magnetic tape*

2 *File name of output magnetic tape*

3 *Table parameters*

4 *End parameter*

FILE NAME OF INPUT MAGNETIC TAPE

Consists of the twelve alphanumeric characters by which the input magnetic tape is labelled.

FILE NAME TO BE WRITTEN ON OUTPUT MAGNETIC TAPE

Consists of the twelve alphanumeric characters specified by the user which will label the output magnetic tape.

TABLE PARAMETERS

One table parameter card is required for each table to be converted, and cards must be input in the same order that the corresponding tables will be found on the input magnetic tape. Table parameters are punched as a six-character alphanumeric table name referring to a particular table on the input tape, followed by a single digit n which specifies the scaling factor required for that table. All values in the converted table are then multiplied by 10^n.

END PARAMETER

If less than 10 tables are to be converted, the last table parameter card or paper tape line must be followed by an end parameter. This consists of four asterisks, and is punched on a new card or a new line.

RESTRICTIONS ON INPUT DATA

1 The data must not begin with a blank card or new line character. Blank data cards or multiple newlines should not occur anywhere within the data.

2 If there are more than 10 tables specified for conversion only the first 10 will be converted. The remainder will be ignored.

3 The scaling factor n must fulfil the following conditions:

$0 \leqslant n \leqslant 3$

4 If a table parameter refers to an input table already in integer form it is copied directly without scaling.

Example

A survey tables tape labelled SURVEYTABLE1 contains several tables. It is required to copy this tape to another tape, to be labelled SURVEYTABLE2, and at the same time convert tables named TEST01 and TEST06 from floating-point to integer form. Table TEST01 is to be reformed with its original values but in integer form. Table TEST06 is to be scaled by a factor of 100 (i.e. $n = 2$). The following parameters would be input in the order given:

SURVEYTABLE1

SURVEYTABLE2

TEST010

TEST062

****

Output

1 Magnetic tape output. The magnetic tape format of the output tape is given in the 1900 Series *Survey Analysis System Mark 1* manual. (Technical Publications 4066). The output tape format is similar to that of the input tape except that the tables will be in integer form.

2 Line printer output. The first six words of each table on the input tape are printed out together with an indication of whether the table was copied or converted. The use of these six words are described in the *Survey Analysis Mark 1* manual mentioned above.

Priority

The program as supplied has a priority of 90.

Operating instructions

Narrative		*Console message*
1	Load the program #X3GX.	
2	Load the input tape (without a write-permit ring) and a scratch tape (which becomes the output tape).	
3	Load the data on the card or paper tape reader.	
4	If the parameters are on card type:	ON #X3GX 1
5	To start the program type:	GO #X3GX 20
6	The program will then read in the parameters and open the appropriate magnetic tapes. The required input tape tables are converted and then written to the output tape. The program will normally end by typing:	DELETED:- 00

EXCEPTION CONDITIONS

Message	*Cause*	*Action*
HALTED: PE	Parameter error.	To continue run, ignoring this parameter, type: GO #X3GX
HALTED: ER	Table not found	Abandon run.
HALTED: TE	Tape transfer error.	Abandon run.

Appendix 4 #X3GV

TITLE

Traffic survey analysis tables conversion program

HARDWARE REQUIREMENTS

3136 words of core storage
1 paper tape reader or 1 card reader
120 character-position line printer
2 magnetic tape decks

Note: If there is no hardware floating-point equipment then a suitable floating-point Executive is required.

USE OF PERIPHERALS

Peripheral	*Use*	*Allocation & Release*
Card Reader CR7 Paper Tape Reader TR7	Input of directives and parameters.	The reader is allocated at the start of the run and released when all parameters and directives have been read.
Magnetic tape deck MT0	Holds input tape containing tables from Survey Analysis program.	The magnetic tape deck MT0 is retained throughout the run.
Magnetic tape deck MT1	Holds output tape containing tables, after conversion, in subfiles.	The magnetic tape deck MT1 is retained throughout the run.
Line Printer LP7	Listing of output and error messages.	The line printer is retained throughout the run.

The peripheral unit numbers given above are those to be used if the program is to be controlled by the GEORGE operating system.

GENERAL DESCRIPTION

The program converts tables produced by the 1900 Series Survey Analysis program (#XDSB) into a form acceptable to the 1900 Series Magnetic Tape File Handling for FORTRAN Data program (#X3GY) and to the 1900 Series Capacity Restraint Traffic Assignment Package.

The program copies any number of tables from any number of tapes produced by the Survey Analysis program and produces a variable number of output tapes. The output tapes contain the same tables, in subfiles acceptable to #X3GY and the Capacity Restraint Assignment Package. Values in converted tables may be multiplied by a user-specified scaling factor.

The program is controlled by *input directives*, each punched in a single card or a single line of paper tape. Each directive dictates the operation to be performed on the data immediately following it. In describing directives, reference is made only to card input. One card will be equivalent to a line of paper tape.

INPUT DIRECTIVES

Each directive and record must be punched left-justified in an 80 column card or punched in 8 track paper tape and followed by a newline character. The parameters are as follows:

Input

The INPUT directive states that the magnetic tape specified by the card following the INPUT directive is the tape holding the tables to be converted. The record following an INPUT directive must have the twelve character file name of the tape in the first twelve positions and the file generation number at or after position 13.

The program may have only one input tape open at a time; a second INPUT directive specifying another tape causes the current tape to be rewound before the new tape is opened.

Output

The OUTPUT directive opens a scratch magnetic tape as an output file for the program. It must be followed by a card containing the twelve character file name to be given to the tape in the first twelve positions, and the file generation number from position 13 onwards.

Only one output tape may be open at any one time. Successive OUTPUT directives requiring different tapes will cause an end of file marker to be written to the current tape, which will then be closed. The new tape is then opened.

Table

The TABLE directive specifies the table on the Survey Analysis Tape to be converted. The record following a TABLE directive should contain the name of the table to be converted in the first 6 character positions.

Create

The CREATE directive causes a subfile to be created on the output tape. The record following a CREATE directive should give the twelve character subfile name in the first twelve positions.

Finish

The FINISH directive causes the conversion of the required table into a form suitable for input to the Capacity Restraint Assignment Package and to program #X3GY, the converted table is then transferred to the required sub-file of the output tape.

As each record is written to the output tape a listing of that record is given on the line printer.

Stop

The STOP directive causes the program to write an end of file marker to the current output tape and to rewind and close both current input and output tapes.

The program then deletes itself.

Scale

The SCALE directive states that the scaling factor entered on the following card must be applied to tables following the scaling factor card; the factor may be either an integer or decimal number.

If more than one table is to be converted but not all are to be scaled, the scaling factor must be reset to unity by using the SCALE directive immediately after the last of a sequence of tables which were to be scaled.

Notes:

1 The tables to be converted must be given in the same order that they occur on the input tape.

2 The elements of a table under conversion are multiplied by $n/10^d$, where n is the specified scaling factor and d denotes the position of the decimal point. d is the mantissa of the floating-point value of the element output by the Survey Analysis package. At the time at which the program is entered, n is set to 1.

3 Any number of tables may be converted during one run by repeating the directives TABLE, CREATE and FINISH (and possibly INPUT, OUTPUT and SCALE where necessary) the required number of times.

4 Only the first four characters of any directive are significant.

5 An input tape and an output tape must be on-line to the program at all times. The first two directives for any sequence must specify the input and output tapes to be used. After each subfile has been written to the required output tape, the input and output tapes may be changed or maintained as necessary, ready for the next subfile to be processed.

Use of directives

Once the input and output tapes required to produce the subfile have been opened by INPUT and OUTPUT directives, the following cards must be input in the order given:

1 TABLE directive

2 table name

3 CREATE directive

4 subfile name

5 SCALE directive

6 scaling factor

7 FINISH directive

The SCALE directive and scaling factor are used only if the table is to be scaled.

The above procedure is repeated for each table to be copied from the given input tape and the given output tape. Other input and output tapes may be specified by further INPUT and OUTPUT directives, and the relevant tables converted by the same method. When all the required tables have been converted and placed in subfiles, the STOP directive is used to close the current input and output tapes, and end the run.

Example

A survey tables tape with file name SURVEY TAPE and file generation number 0 contains several tables including TEST01 and TEST03.

Another survey tables tape with file name SURV RESULTS and file generation number 1 contains several tables including TEST02 and TEST04.

Two output tapes are required. These are OD TABLES A with a file generation number 1, and OD TABLES B with a file generation number 1. OD TABLES A will contain TEST01 in a sub-file called OD TEST NO 1, and TEST02 scaled by a factor of 16 in a subfile called OD TEST NO 2. OD TABLES B will contain TEST03 in a sub-file called OD TEST NO3, and TEST04 in a sub-file called OD TEST NO 4.

The following sequence of input directives and parameters would be used:

INPUT

SURVEY TAPE 0

OUTPUT

OD TABLES A1

TABLE

TEST01

CREATE

OD TEST NO 1

FINISH

INPUT

SURV RESULTS 1

TABLE

TEST02

```
CREATE
OD TEST NO 2
SCALE
16
FINISH
INPUT
SURVEY TAPE 0
OUTPUT
OD TABLES B1
TABLE
TEST03
CREATE
OD TEST NO 3
SCALE
1
FINISH
INPUT
SURV RESULTS 1
TABLE
TEST04
CREATE
OD TEST NO 4
FINISH
STOP
```

MAGNETIC TAPE FORMATS

The format of the input tapes is given in the 1900 Series *Survey Analysis System Mark 1* manual (Technical Publication 4066). The format of the output tapes can be found in the *Magnetic Tape File Handling For Fortran Data* manual (Technical Publication 4203).

OUTPUT

Two types of output may be printed on the line printer. These are as follows:

1 Normal output

2 Error messages

NORMAL OUTPUT

All directives, their associated parameters and a listing of each record as it is written to the output magnetic tape are printed on the line printer.

ERROR MESSAGES

The following messages may be output on the line printer:

Message	*Cause*	*Action*
NO INPUT TAPE DIRECTIVE	The program has met a FINISH directive without having read an INPUT directive.	The program will delete itself.
NO OUTPUT TAPE DIRECTIVE	The program has met a FINISH directive without having read an OUTPUT directive.	The program will delete itself.
TABLE *xxxxxx* NOT ON INPUT TAPE	The table *xxxxxx* has been defined as a table to be converted but is not on the current input tape.	The program will carry on reading directives.
ERROR ON INPUT TAPE	A transfer error on the input tape has been encountered.	The program will delete itself.

Priority

The program as supplied has a priority of 50.

Operating instructions

	Narrative	*Console message*
1	Load the program #X3GV.	
2	Load the input magnetic tape and a scratch magnetic tape (which becomes the output tape).	
3	Load the data on the reader.	
4	If the parameters are on cards type:	ON #X3GV 1
5	To start the program type:	GO #X3GV 20
6	The program will read in the parameters, obeying them as they are read, with possible messages to the operator to load a new input or output magnetic tape. The program will normally end by typing:	DELETED:- 00

EXCEPTION CONDITIONS

Message	*Cause*
DELETED:- 01	Indicates an error has been found. The error message is printed on the line printer.

Index

Arithmetic expression 20,22
Array of text 33
Assignment program 1

Block transfer 20,22

Cell increment 35
Cell value format 46
Character code 65
Coding 7
Coding sheets 7 (data), 11 (control statements)
COLDIV statement 43
COLJN statement 42
COLMPY statement 43
COLMRG statement 42
Column total 36,37
Comment statement 11
COMMON statement 17,19
Common value 10
CONADD statement 42
CONDAT statement 27,28
CONDIV statement 42
CONMPY statement 42
Consistency expression 24,25
CONSUB statement 42
Control statement 4,11
Conversion programs, traffic survey analysis tables 93, 97

DA-REC statement 17,20
DACONV statement 17,23
Data record 2
Data tape 1
  creation 1,55
DELETE statement 42
Designation value 10,18,19

ERASE statement 53
Error condition 60
Error halt 59
Error message 60
Expansion factors 5,23,24,59,65

Floating point 37,46
FORMAT statement 4,17
Function statement 11
Furness prediction program 1

Key 3
  multiple key 3,32

Language 4
Limitations on record size 11
Line of text 33
LIST statement 13,15,(XTRACT group) 54

Magnetic tape input 1,11,28

Names 11

Offlining 1
OPEN statement 13
Operating instructions 55 to 58
OPERAT statements 5,39 to 44
Output 4
OUTPUT statement 5,45

Paper tape 10
Passenger car unit 4,5
Primary survey record 2
Progress report 35,39
Punched card 7,10

Questionnaire 1,7,8

R-DATA statement 27
Range check 2,19
READ statement 5,27
Record size, limitations 11
RECORD statement 17 to 20
Reduced record 10
REJECT statement 17,24,25
Relative sequence of 1900 Series characters 65
ROWDIV statement 43
ROWJN statement 42
ROWMPY statement 43
ROWMRG statement 42
Row total 36,37
RTAB statement 27,29,30

SET-UP statement 4,13
Standard headings 47
Sub-table 39
Survey form 1,7,8
Survey tape 4,14
System tape 4,14

Table 1,2
  manipulation 4
  section 47
  tape 2
  multi-level table 3
  result table 4
Table look-up expression 20,21,23
TABLES statement 5
TAB statement 35 to 38
Tabulation 2 to 4,36
T ADD statement 43
TAPES statement 13,14
T DIV statement 43
Terminating symbol 10,17

TEXT statement 27,33
T FIX statement 43
Title statement 11
T MPY statement 43
Traffic engineering programs, ICL 6
T SQRT statement 42
T SUB statement 43
Traffic survey analysis tables conversion programs 93, 97
T TSP statement 42

Validity check 2,24
Variables 2
recoding of 59
Vector 3,31
weight vector 43
VECTOR statement 27,30 to 33

WCSUM statement 43
Weighting factor 36
WRITE statement 45,46 to 50
WRSUM statement 43

XTRACT statement 5,53

AMENDMENT RECORD

Should any additions or amendments to this Manual be necessary, they will be issued in the form of Amendment Lists. When such changes have been incorporated, an appropriate entry should be made in the table on the left and the pink instruction sheet(s) should be filed behind this page.

List Number	*Item Numbers*
1	1 *to* 9
	to
	to
	to
	to
	to
	to
	to
	to
	to
	to
	to
	to
	to
	to
	to
	to
	to
	to
	to
	to
	to
	to
	to
	to
	to
	to
	to
	to
	to

ICL

1900 Series

Traffic Survey Analysis

First Edition August 1968

Amendment list 1
incorporating User Notices Traffic Survey Analysis 1 to 11

Each amendment list contains one or more numbered instructions to replace one or more existing pages or to add one or more new pages.

When a page is amended, significant technical changes on the re-issued page will be indicated by a vertical line in the margin against the changed passages. Any lines on the re-issued page which remain from a previous amendment, will be removed. New chapters or completely revised chapters will not be marked with amendment lines.

The date of issue appears at the foot of all new pages and re-issued pages in the form (month, year).

1	Contents	Remove and destroy pages iii to ix. Insert new pages iii to ix.
2	Chapter 1	Remove and destroy pages 3 to 5. Insert new pages 3 to 6.
3	Chapter 2	Remove and destroy pages 11/12. Insert new pages 11/12.
4	Chapters 4 and 5	Remove and destroy pages 23 to 32. Insert new pages 23 to 32.
5	Chapters 7 and 8	Remove and destroy pages 41 to 48. Insert new pages 41 to 48.
6	Chapter 10	Remove and destroy pages 59 to 64. Insert new pages 59 to 64.4.
7	Index	Remove and destroy pages 93/94. Insert new pages 93 to 104.
8		Destroy User notices 1 to 11.
9		Update the amendment record and file this list at the back of the manual.

4103(5.70)