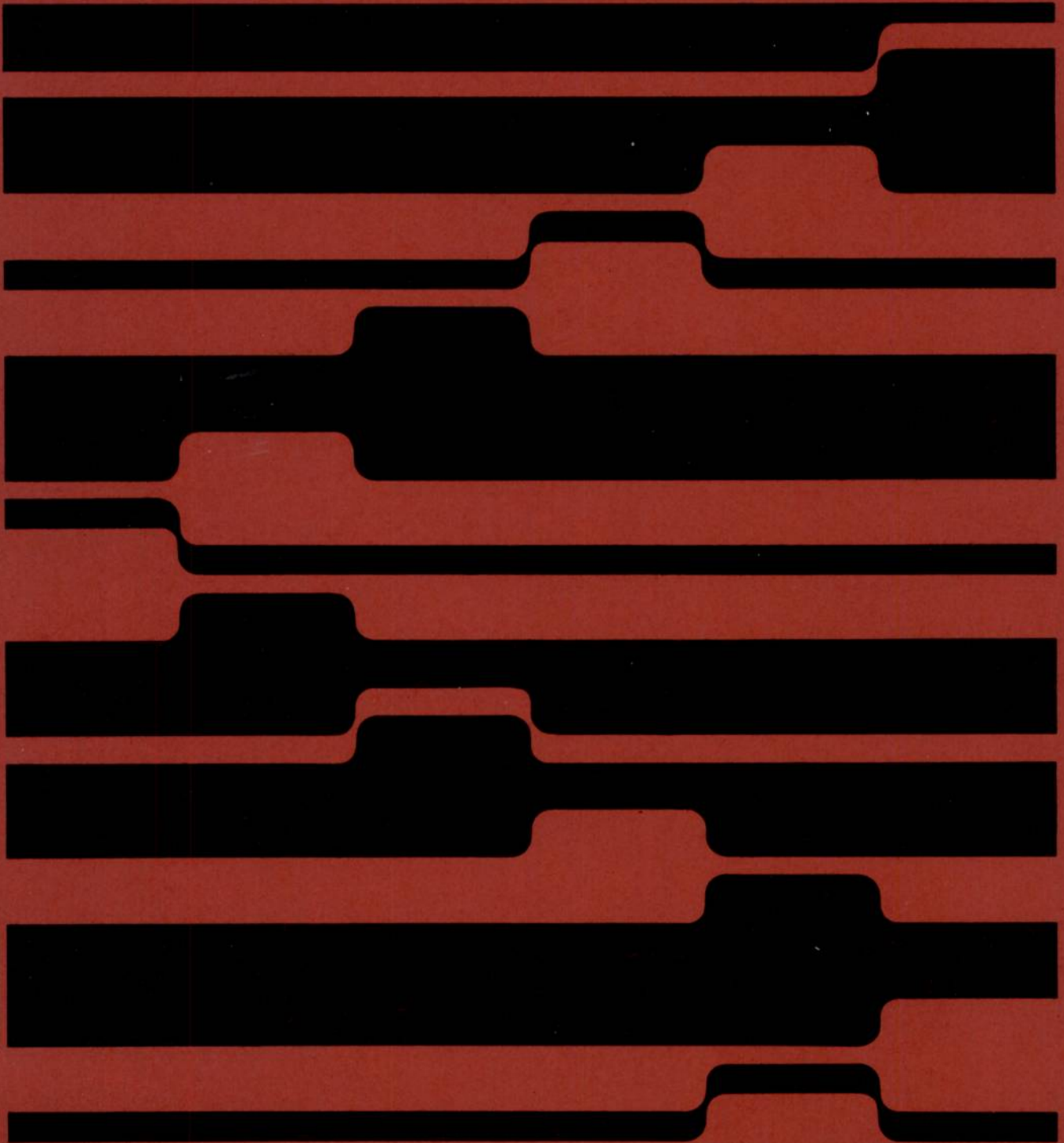


ICL

**Data
Management
Software
File Arrangement**

1900 Series



ICL

**Data
Management
Software
File Arrangement**

1900 Series



The policy of International Computers Limited is one of continuous development and improvement of its products and services, and the right is therefore reserved to alter the information contained in this document without notice. ICL makes every endeavour to ensure the accuracy of the contents of this document but does not accept liability for any error or omission. Any equipment or software performance figures and times stated herein are those which ICL expects to be achieved in normal circumstances. Wherever practicable, ICL is willing to verify upon request the accuracy of any specific matter contained in this document.

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

Technical Publication 4137

© International Computers Limited 1970

First Edition April 1970

Issued by Technical Publications Service
International Computers Limited
Head Office: ICL House, Putney, London SW15
Produced by ICL Printing Services
at Letchworth Hertfordshire

Preface

This manual describes the six Data Management Software programs which are concerned with file arrangement. The function of the first five programs is to arrange the contents of one or more input files in a form suitable to the user. The sixth program produces parameters suitable for input to standard 1900 Series Library sort programs.

The manual is one of a series describing ICL data Management Software, and should be read in conjunction with the manual *Data Management Software Introduction*, which explains the philosophy behind ICL's approach to data management.

The manual is intended to be read by those already using or considering the use of the Data Management Software programs described. These may be either programmers or people with no experience of programming: the software is designed to allow the user to control the programs entirely by parameters in a given format. The manual will also be of use to systems analysts and designers in so far as it explains what functions the programs are capable of performing.

There are six parts to the manual, each dealing with a single program as follows:

Part 1 describes COLLATE which writes selected data from two magnetic tape input files onto a single output file.

Part 2 describes COMPARE which compares records from two magnetic tape input files and outputs required details from such records to a single output file.

Part 3 describes SWAP which replaces data on one magnetic tape input file with data from a second magnetic tape input file, producing a single output file. An optional non-matching records file may also be output.

Part 4 describes COPY which is used for straightforward copying from one tape to another.

Part 5 describes CRAM which writes a variable number of magnetic tape input files to a single output file.

Part 6 describes SORT which produces parameters for input to the 1900 Series Library sort programs, #XSMC, #XSMD and #XSME.

Within each part there are chapters giving a general introduction to the program, a description of file input and output, and a detailed explanation of the parameter set needed for the program. There is an example of how the program could be used in an actual application and a chapter giving operating procedures and exception conditions.

It should be noted that a general description of all the Data Management Software programs will be found in the introductory manual. This manual also describes parameter standards and programming conventions which are not fully dealt with in the individual manuals. A typical example of the applications of the software is given.

The minimum configuration for each program is given in the introductory chapter to that program. No program requires more than 16K of store.

Contents

Preface

PART 1 COLLATE

Chapter 1 Introduction to COLLATE	1
PROCESSING SUMMARY	1
MODES	1
MINIMUM CONFIGURATION	3
Chapter 2 Input and output	5
INPUT	5
Input files	5
Parameters	5
OUTPUT	5
Output file	5
Line printer output	5
PARAMETER LIST	5
COUNTS STATEMENT	5
File errors	6
Console output	6
Chapter 3 The parameter set	9
READ TAPE LABEL PARAMETER (#READ)	9
WRITE TAPE LABEL PARAMETER (#WRITE)	9
DOUBLE FILE KEYS PARAMETER (#KEYS)	10
MODE PARAMETER (#MODE)	11
END OF PARAMETERS MARKER	11
Chapter 4 Example	13
Chapter 5 Operating instructions and exception conditions	17
OPERATING INSTRUCTIONS	17
EXCEPTION CONDITIONS	17

PART 2 COMPARE

Chapter 6 Introduction to COMPARE	19
USE OF COMPARE	19
PARAMETER PROCESSING	20
MAIN PROCESSING	20
Identification of output records	22

EXAMPLE 1	22
Set 1: Label JOIN	22
Set 2: Label LEFT	22
Set 3: Label DIFF	22
EXAMPLE 2	23
Set 3: Label DIFF	
Multiple Output Records	23
EXAMPLE 3	23
Set 1: Label MSRS	23
Set 1: Label PAYS	23
Set 1: Label FUND	23
Selection by Codes	24
EXAMPLE 4	24
Set 1: Label MAND	24
Set 1: Label MNOT	24
Set 1: Label REST	25
Ignoring input records	25
CONTENTS OF PARAMETER SETS	25
Common Transfers	26
LIMITATIONS TO COMPARE	26
MINIMUM CONFIGURATION	26
Chapter 7 Input and output	27
INPUT	27
Input files	27
Parameters	27
OUTPUT	27
Output file	27
Line printer output	27
Console output	28
Chapter 8 The parameter set	31
READ TAPE LABEL PARAMETER (#READ)	31
WRITE TAPE LABEL PARAMETER (#WRITE)	31
DOUBLE FILE KEYS PARAMETER (#KEYS)	32
COMMON TRANSFERS PARAMETER (#COMMON)	33
SET HEADER PARAMETER (#SET)	34
RECORD SELECTION PARAMETERS	34
#SKIP	34
#ONLY	35
CLEARING REQUIREMENTS PARAMETER (#CLEAR)	36
MOVE PARAMETER (#MOVE)	36
INSERT CONSTANTS PARAMETER (#FILL)	37
COMPARISON PARAMETER (#COMPARE)	38
END OF PARAMETERS MARKER	39
Chapter 9 Example	41
Notes on the example parameters	43

Chapter 10 Operating instructions and exception conditions	45
OPERATING INSTRUCTIONS	45
EXCEPTION CONDITIONS	45
PART 3 SWAP	
Chapter 11 Introduction to SWAP	47
INPUT FILE FORMAT	47
PARAMETER PROCESSING	47
MAIN PROCESSING	47
USE OF SWAP	48
MINIMUM CONFIGURATION	48
Chapter 12 Input and output	49
INPUT	49
Input files	49
Parameters	49
OUTPUT	49
Magnetic tape output	49
Line printer output	49
Console output	50
Chapter 13 The parameter set	53
READ TAPE LABEL PARAMETER (#READ)	53
WRITE TAPE LABEL PARAMETER (#WRITE)	53
DOUBLE FILE KEYS PARAMETER (#KEYS)	54
MODE PARAMETER (#MODE)	55
MOVE PARAMETER (#MOVE)	56
END OF PARAMETERS MARKER	56
Chapter 14 Example	59
Chapter 15 Operating instructions and exception conditions	61
OPERATING INSTRUCTIONS	61
EXCEPTION CONDITIONS	61
PART 4 COPY	
Chapter 16 Introduction to COPY	63
PROCESSING SUMMARY	63
MINIMUM CONFIGURATION	64
Chapter 17 Input and output	65
INPUT	65
Input file	65
Parameters	65
OUTPUT	65

Output file	65
Line printer output	65
Console output	65
Chapter 18 The parameter set	67
READ TAPE LABEL PARAMETER (#READ)	67
WRITE TAPE LABEL PARAMETER (#WRITE)	67
END OF PARAMETERS MARKER	68
Chapter 19 Operating instructions and exception conditions	69
OPERATING INSTRUCTIONS	69
EXCEPTION CONDITIONS	69
PART 5 CRAM	
Chapter 20 Introduction to CRAM	71
PROCESSING SUMMARY	71
MINIMUM CONFIGURATION	72
Chapter 21 Input and output	73
INPUT	73
Input files	73
Parameters	73
OUTPUT	73
Output file	73
Line printer output	73
Console output	73
Chapter 22 The parameter set	75
READ TAPE LABEL PARAMETER (#READ)	75
WRITE TAPE LABEL PARAMETER (#WRITE)	75
END OF PARAMETERS MARKER	76
Chapter 23 Operating instructions and exception conditions	77
OPERATING INSTRUCTIONS	77
EXCEPTION CONDITIONS	77
PART 6 SORT	
Chapter 24 Introduction to SORT	79
PARAMETER PROCESSING	79
MAIN PROCESSING	80
MINIMUM CONFIGURATION	80
Chapter 25 Input and output	81
INPUT	81
Parameters	81

OUTPUT	81
Card and paper tape output	81
Line printer output	81
Console output	81
Chapter 26 The parameter set	83
PROGRAM NAME PARAMETER (#XSMC, #XSMD or #XSME)	83
READ TAPE LABEL PARAMETER (#READ)	83
WRITE TAPE LABEL PARAMETER (#WRITE)	84
KEYS PARAMETER (#KEYS)	84
END OF PARAMETERS MARKER	85
Chapter 27 Example	87
Chapter 28 Operating instructions and exception conditions	89
OPERATING INSTRUCTIONS	89
EXCEPTION CONDITIONS	89

Illustrations

Figure 1	Outline flow diagram for COLLATE	Facing page 1
Figure 2	The effect of different COLLATE modes, shown by using the same input files	2
Figure 3	COLLATE parameter map	8
Figure 4	COLLATE example flowchart	14
Figure 5	COLLATE example printouts	15
Figure 6	Outline flow diagram for COMPARE	Facing page 19
Figure 7	Logical flow in COMPARE	21
Figure 8	COMPARE parameter map	30
Figure 9	COMPARE example output record formats	40 and 42
Figure 10	COMPARE example printouts	44
Figure 11	Outline flow diagram for SWAP	Facing page 47
Figure 12	SWAP parameter map	52
Figure 13	SWAP example outline flowchart	58
Figure 14	SWAP example input and output record formats	58
Figure 15	SWAP example printout	59
Figure 16	Outline flow diagram for COPY	63
Figure 17	Example of printout from COPY	65
Figure 18	Outline flow diagram for CRAM	71
Figure 19	Example of printout from CRAM	73
Figure 20	Outline flow diagram for SORT	Facing page 79
Figure 21	SORT parameter map	82
Figure 22	SORT example printout: parameters for SORTED ADATA file	86
Figure 23	SORT example printout: parameters for SORTED BDATA file	86
Figure 24	SORT example printout: card input with paper tape output.	87

PART 1 COLLATE

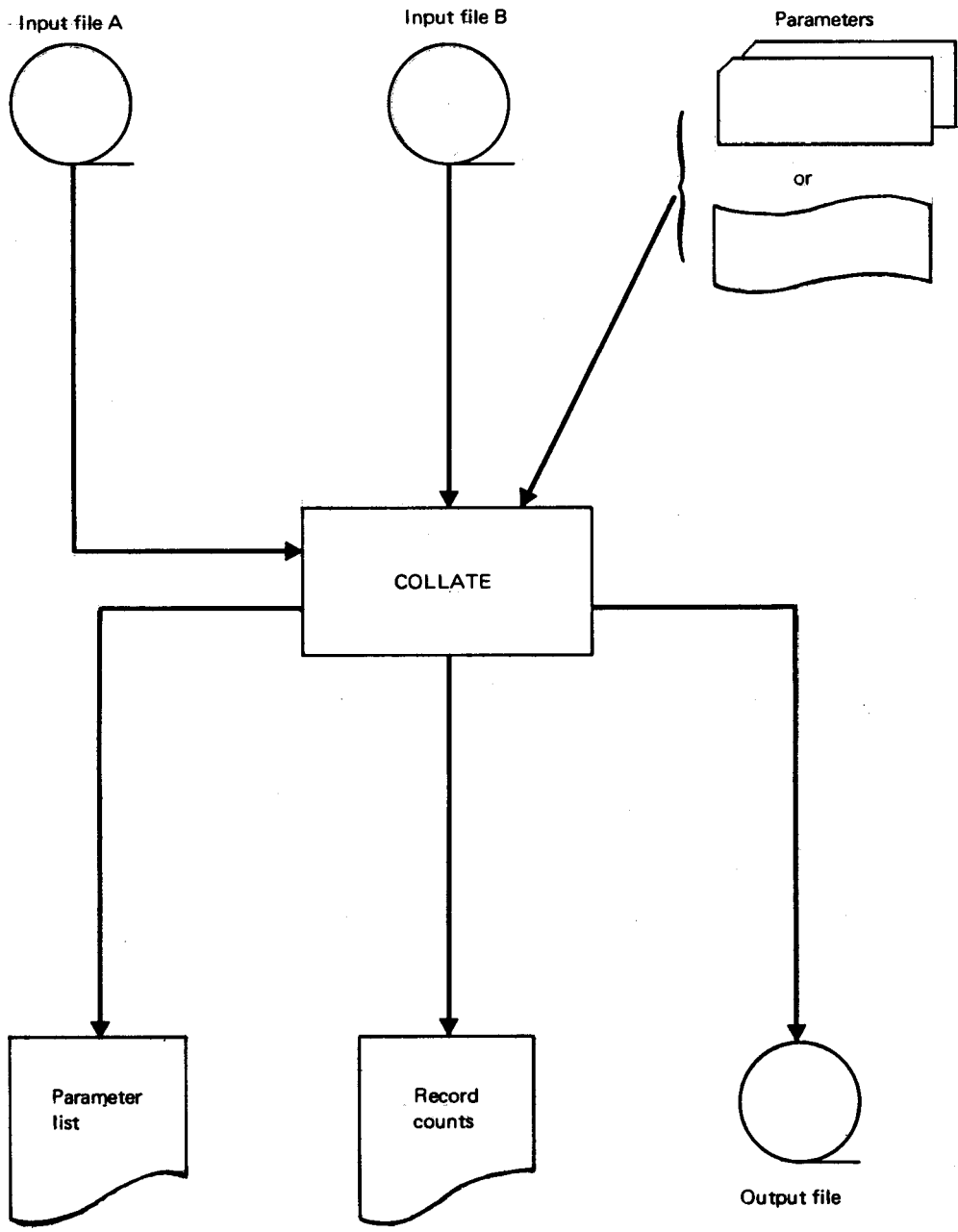


Figure 1 Outline flow diagram for COLLATE

Chapter 1 Introduction to COLLATE

This program merges selected records from two magnetic tape files, the selection being according to rules specified by the user. No alteration is made to the content or format of the records on the files. Output consists of a list of parameters with indications of any errors and, if the parameters are accepted, a single magnetic tape file containing the required records together with a line printer listing of the counts of records read and written. This is illustrated in Figure 1.

PROCESSING SUMMARY

At the start of the run, the program reads six parameters similar in form to those used in other programs of the Data Management Software suite. The first three of these define the two input files, and the label which is to be given to the output file. The next parameter defines the keys of the records which are to be related. Up to six keys may be specified, and they may be at any points within the records. The program then reads a parameter which defines the *mode* in which it is to operate. This allows the user to say whether he wants the output file to contain records from one file only or from both, whether they should be matched or unmatched records, and so on.

The parameter set is terminated by the usual end of parameters marker. When this has been read the program checks that the parameters are correct and complete before continuing. A list of all parameters is printed, so that a record will be available of the parameters used for each run.

During the run records from the two input files are read, and their keys are compared. The action taken depends on the mode specified; this defines which records are to be written to the output tape. The various modes are described in detail in the next section.

When all records from the two input files have been read and processed, the program closes the tapes, prints counts of records read and written and the run is terminated.

MODES

COLLATE considers the two input files to be a *main file*, input file A, and a *subsidiary file*, input file B, and interprets the #MODE parameter accordingly.

The various modes, whose effects are illustrated in Figure 2, are as follows:

The first two modes output only records selected from the main file.

MODE 1 This mode outputs all main records whose key is not present in the subsidiary file, i.e. unmatched main records. As an example, if the main file details items to be invoiced whilst the subsidiary file has customers' names and addresses, the output file would contain details of items to be invoiced for which the customer's name and address is not held.

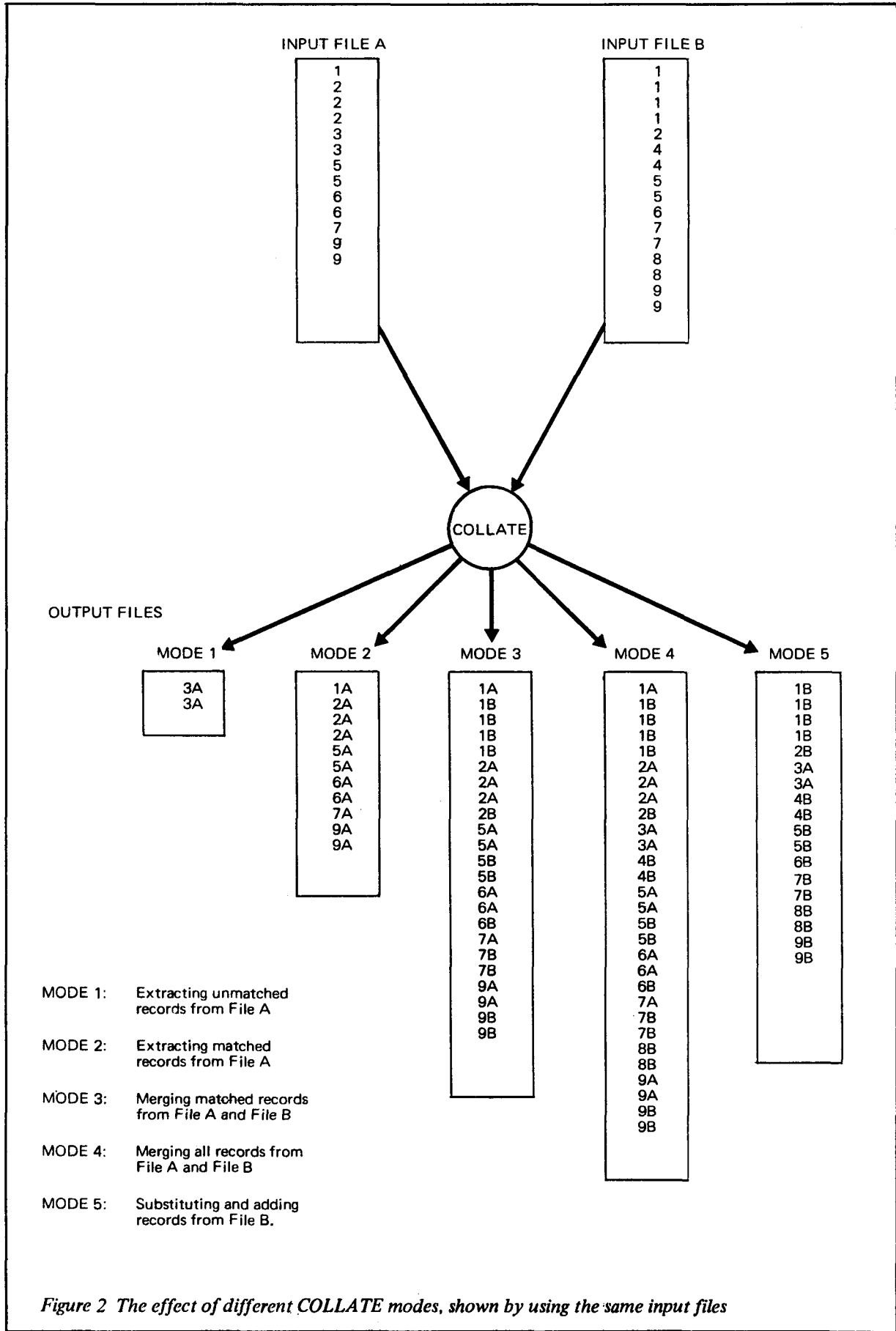
MODE 2 This mode outputs only matched main records. Using the example mentioned above, in this mode the output file would hold items to be invoiced for which the customer's name and address is not known.

The remaining three modes output records from both input files according to whether their keys match or not.

MODE 3 This mode outputs matched records from both files. The main file's record precedes the subsidiary file's record where matching occurs. Using the same example again, in this mode the output file would be composed of the item to be invoiced, followed by the name and address. If the system required the name and address record to appear first on the file, i.e. before the item to be invoiced to which it refers, the role of the two files would have to be reversed.

MODE 4 This mode merges all input records to the output file. Main records again precede subsidiary records where matching occurs.

MODE 5 This mode updates the main file from the subsidiary file. Any unmatched records from either file are output i.e. subsidiary records are insertions. When records match, only the subsidiary one is output i.e. the main record is amended by replacement. This does not cover the deletion of records but this can be done by using SOLO (see the manual *Data Management Software : Validation and Editing*),



or by using COLLATE a second time with the subsidiary file holding deletion records only and with the program operating in mode 1.

MINIMUM CONFIGURATION

The minimum configuration for the COLLATE program comprises:

1 1900 Series central processor with 8K core store

1 card reader or paper tape reader

1 line printer

3 magnetic tape decks.

Chapter 2 Input and output

INPUT

Input to COLLATE consists of the two magnetic tape files to be collated, and parameters.

Input files

The two input files must observe the conventions of 1900 Series Magnetic Tape Housekeeping. They may be single or multi-reel files. User tape sentinels will be ignored, and will not be copied to the output file. A file may contain one or more records of the same keys. Key fields must appear in all records and must always be in the same place in the records: subject to this limitation, records may be of varying length.

Parameters

The parameters are described in Chapter 3.

OUTPUT

Output from COLLATE consists of a single output file, and line printer output.

Output file

One file is output containing appropriate records, as determined by the mode being used. It will conform to the usual MTH standards.

Line printer output

PARAMETER LIST

The program will print out each parameter in full exactly as it appears in the card or on paper tape. Errors are indicated as follows:

ERROR is printed out against the listing of any parameter found to be incorrect.

SEQ is printed out if an error is discovered in the parameter sequence.

If an error message has appeared against any parameter in the listing the message at the end of the listing is:

PARAMETERS INCORRECT

In addition, the printer will output:

PARAMETERS INCOMPLETE

if the parameter set is incomplete.

COUNTS STATEMENT

At the end of the run the program prints out counts as follows:

Records read from File A

Records read from File B

File A records written to the output file.

File B records written to the output file.

If the mode in use does not require both input files to be read to end of file, the record counts will show the number of records read up to the point at which processing was completed; to emphasize that this may be different from the total number of records on the file, the count may be accompanied by the additional text; 'FILE A [or B] WAS CLOSED BEFORE END OF FILE.'

File errors

If there is a sequence error on the File A tape the program will be suspended and the printer will output:

SEQUENCE ERROR FILE A

This message will also be typed out on the console; if it occurs the run should be abandoned.

A similar message will be printed and typed if a sequence error is detected in file B.

Console output

See under *Exception Conditions*, Chapter 5.

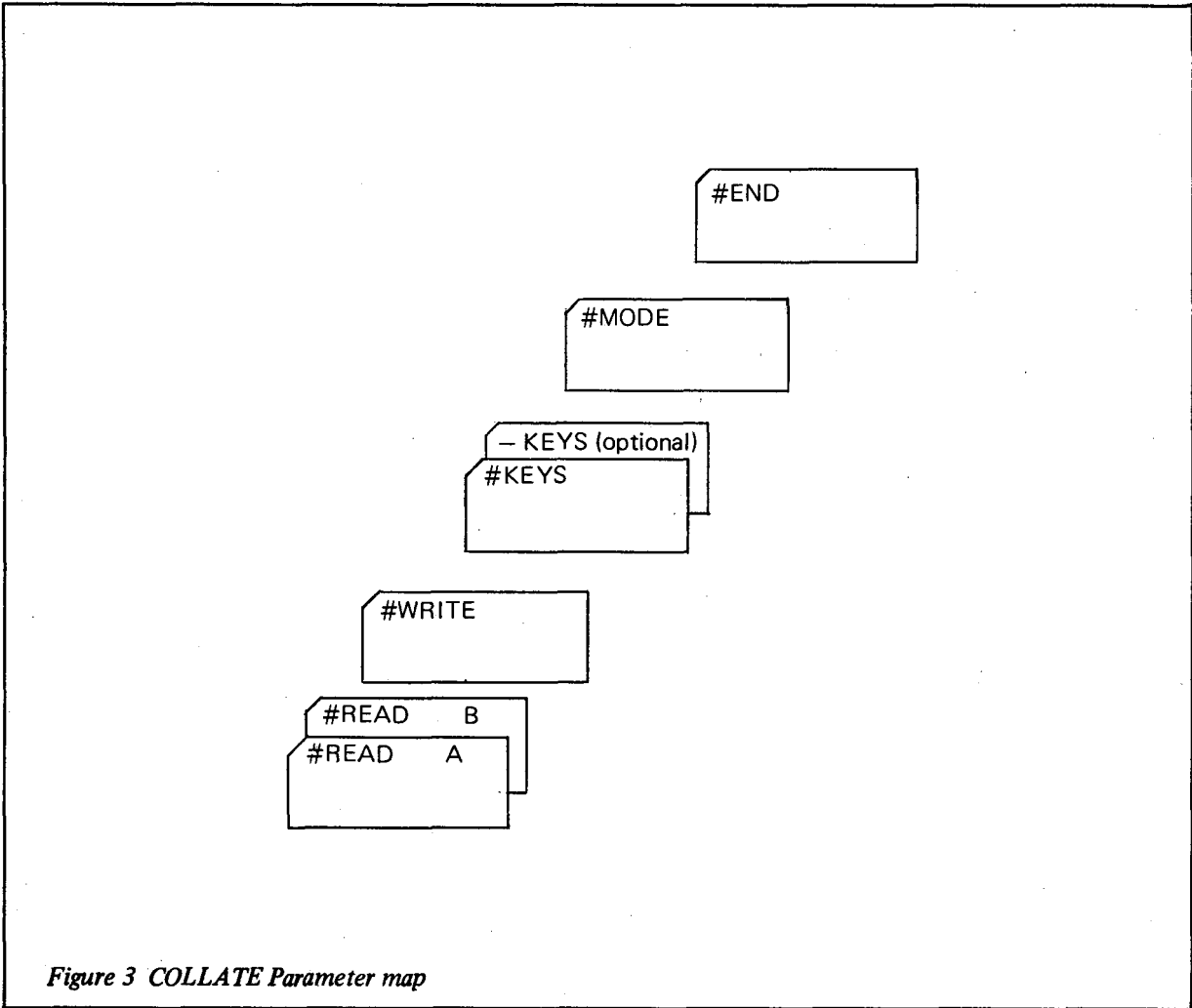


Figure 3 COLLATE Parameter map

Field	Columns	Contents
A	1 to 6	The directive #WRITE
B	8 to 19	12-character file name .
C	21 to 24	Reel sequence number: 4 digits .
D	26 to 29	File generation number: 4 digits.
E	31 to 34	Retention period: 4 digits .
F	36	Designation: 1 alphabetic character .
G	38 to 49	Old file name: 12 characters. The existing file name on the tape to be used for output.
H	51 to 54	Old reel sequence number: 4 digits.
I	56 to 59	Old file generation number: 4 digits.

Example

1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1	2	3	4	5	6	7	8	9	40						
#	W	R	I	T	E	I	N	V	O	I	C	E	I	N	G	T	P	1

DOUBLE FILE KEYS PARAMETER (#KEYS)

This parameter is in standard form for double file input. It allows the user to specify the keys which are to be used to match records from the two input files. Up to six keys may be specified; the first four of these are defined in the #KEYS parameter, the key with highest significance being defined first.

Field	Columns	Contents
A	1 to 5	The directive #KEYS.
B	7 to 11	The start address of a key in File A expressed in words and characters relative to the start of the record, in the form <i>NNN.N</i>
C	13 to 17	The start address of a key in File B expressed in words and characters relative to the start of the record, in the form <i>NNN.N</i>
D	19 and 20	Field size: a 2 digit integer giving the length of the key in characters for character keys or words for binary keys.
E	21	A single character indicating the type of field: H = characters ascending % = binary ascending D = characters descending * = binary descending
F,G,H,I	23 to 37	Similar to fields B to E, defining the second key.
J,K,L,M	39 to 53	Similar to fields B to E, defining the third key.
N,O,P,Q	55 to 69	Similar to fields B to E, defining the fourth key.

Example

1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1	2	3	4	5	6	7	8	9	40																														
#	K	E	Y	S	1	1	2	.	3	1	3	H	5	3	1	2

If more than four keys need to be specified, a continuation card can be punched; this takes the following form:

Field	Columns	Contents
A	1 to 5	The directive-KEYS
B	7 to 11	The start address of a key in File A expressed in words and characters relative to the start of the record, in the form <i>NNN.N</i>
C	13 to 17	The start address of a key in File B expressed in words and characters relative to the start of the record, in the form <i>NNN.N</i>
D	19 and 20	Field size: 2 digit integer giving the length of the key in characters for character keys or words for binary keys.
E	21	A single character indicating the type of field: H = characters ascending % = binary ascending D = characters descending * = binary descending
F,G,H,I	23 to 37	Similar to fields B to E, defining the sixth key.

Example

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			
#	K	E	Y	S					1	.	0				4	.	3			1	3	H					4	.	1					4	.	0				3	D
-	K	E	Y	S					8	.	0				2	.	0				2	2																			

1	2	3	4	5	6	7	8	9	50	1	2	3	4	5	6	7	8	9	60	1	2	3	4	5	6	7	8	9	70	1	2	3	4	5	6	7	8	9	80		
5	.	0							1	.	0				1	2				6	.	0						8	.	0						2	*				

MODE PARAMETER (#MODE)

This parameter specifies the mode of operation of the COLLATE program; the effect of the various modes is defined on page 2.

Field	Columns	Contents
A	1 to 5	The directive #MODE
B	7	The number of the mode to be used either 1, 2, 3, 4 or 5.

Example

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	
#	M	O	D	E					1																															
#	M	O	D	E					4																															

END OF PARAMETERS MARKER

This parameter is in standard form. It marks the end of the parameter set.

Field	Columns	Contents
A	1 to 4	The directive #END

Example

1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1	2	3	4	5	6	7	8	9	40		
#	E	N	D																																						

Chapter 4 Example

In this example a company maintains a file on magnetic tape which contains details of all its customers. Each record is identified by a customer number, and contains, among other details, the address to which any invoices are to be sent.

The company receives orders for its products, which it processes in the usual way, resulting in the despatch of goods to its customers. The system which processes orders and creates despatch documents outputs a record on magnetic tape for each item that has been despatched, and which should therefore now be invoiced. This record contains details of the goods and their value, to provide the information that has to be shown on the invoice; the record also contains the number of the customer to whom the goods have been sent. When they have been created, these records are sorted to customer number order.

The periodic invoicing procedure starts by matching these two files. The purpose of the matching procedure is two-fold:

- 1 There should be no records of goods with customer numbers that do not occur in the Customer file.
- 2 To speed up the actual invoicing run, customer records are to be entered to the run only if the customers have received goods during the current cycle of the system.

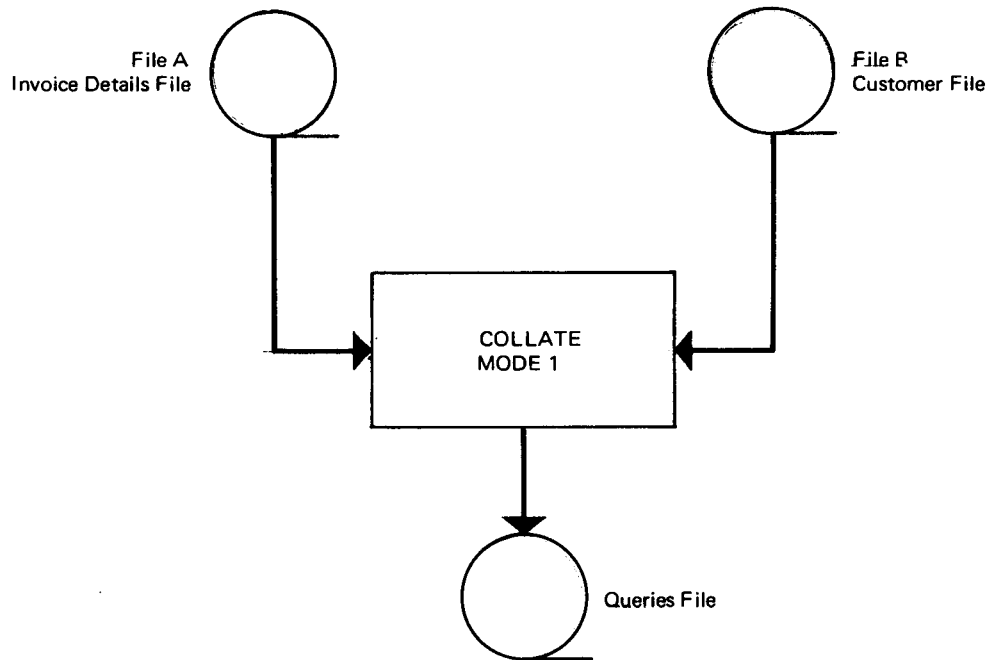
These requirements can be met by two runs of COLLATE. In the first, the invoice details file is used as File A, and the Customer file as File B. MODE 1 is used, to extract any File A records that are not matched by records on File B.

These records will then be available on a Queries file for investigation. They may refer to new customers whose records have not yet reached the Customer file: or the customer number may have been quoted wrongly by the salesman.

While the queries are being investigated, it is necessary to continue with invoicing all the valid records. COLLATE is therefore used again to produce an invoicing file. In this run the Customer file is used as File A, and the invoice details file as File B. On the invoicing file the customer record will then precede any corresponding invoice detail records. MODE 3 is used for this run, to create an output file which contains records from both files, but only those records which are matched by records in the opposite file. Thus on the one hand customers are excluded if they have not bought any goods during the current cycle, and on the other hand unmatched invoice details, which have already been isolated as queries, are also excluded. The result is a compact and accurate invoicing file.

The following pages contain a flowchart of these procedures, and details of the parameters required. The files are matched on customer number, which is taken to be a six-character field, starting at Word 1.0 of the customer file record and at Word 16.2 of the invoice details record.

RUN 1 – EXTRACTING QUERIES



RUN 2 – CREATING INVOICING FILE

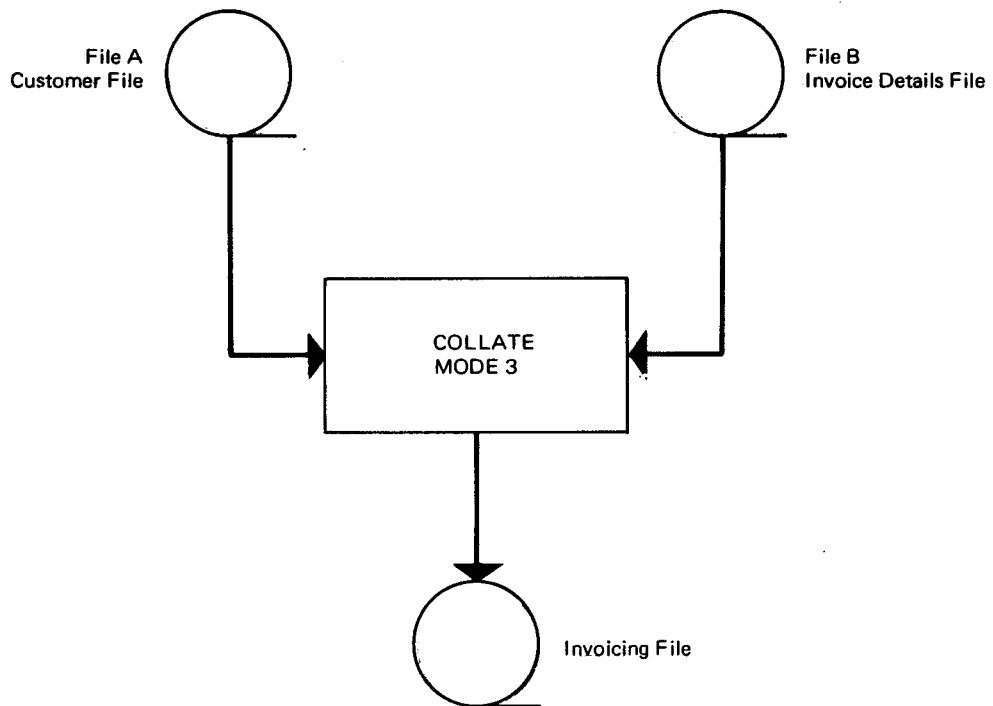


Figure 4 COLLATE example flowchart.

```
04/11/68  COLLATE PARAMETERS

#READ2 CUSTOMERFILE 0000 0010 A
#READ2 INVCEDETAILS 0000 0025 B
#WRITE INVOICESFILE 0000 0025 0030 A INVOICESFILE 0000 C022
#KEYS 001.0 016.2 06H
#MODE 3
#END
```

```
04/11/68  COLLATE RECORD COUNTS

COUNT OF RECORDS READ FROM FILE A           716
COUNT OF RECORDS READ FROM FILE B           3919
COUNT OF RECORDS FROM FILE A WRITTEN TO OUTPUT FILE    320
COUNT OF RECORDS FROM FILE B WRITTEN TO OUTPUT FILE    3902
FILE A WAS CLOSED BEFORE END OF FILE
```

```
04/11/68  COLLATE PARAMETERS

#READ2 INVCEDETAILS 0000 0025 A
#READ2 CUSTOMERFILE 0000 0010 B
#WRITE INVCEQUERIES 0000 0025 0030 A INVCEQUERIES 0000 0022
#KEYS 016.2 001.0 06H
#MODE 1
#END
```

```
04/11/68  COLLATE RECORD COUNTS

COUNT OF RECORDS READ FROM FILE A           3919
COUNT OF RECORDS READ FROM FILE B           716
COUNT OF RECORDS FROM FILE A WRITTEN TO OUTPUT FILE    17
COUNT OF RECORDS FROM FILE B WRITTEN TO OUTPUT FILE     0
FILE B WAS CLOSED BEFORE END OF FILE
```

Figure 5 COLLATE example printouts

Chapter 5 Operating instructions and exception conditions

OPERATING INSTRUCTIONS

<i>Action</i>	<i>Message</i>
1 Load input File A without write permit ring.	
2 Load input File B without write permit ring.	
3 Load output tape with write permit ring	
4 Load the program COLLATE by inputting:	FI #X684 #TAPE
5 Load parameters in appropriate reader.	
6 To read in the parameter either	
(a) if the parameters are on paper tape input:	GO #X684 20
or	
(b) if the parameters are on cards input:	GO #X684 21
7 The program reads the parameters, opens the tapes and processes the records. At end of run the program outputs the message:	HALTED:- END OF RUN
8 When it is required to terminate a partly-completed run, input:	GO #X684 26
the program closes all tapes, allots printer, prints record counts, releases printer and outputs the message:	HALTED:- RUN ABANDONED
following any tape error, the program closes all tapes, allots printer, prints details of tape error type and position, prints record counts, releases printer and closes with the relevant exception condition message, see below.	

EXCEPTION CONDITIONS

<i>Message</i>	<i>Reason</i>	<i>Action</i>
HALTED:- PARAMETERS INCORRECT	The program has detected errors of format or content in one more of the parameters.	Correct the parameters, and re-start from step 5 of Operating Instructions.
HALTED:- PARAMETERS INCOMPLETE	At least one parameter has been omitted.	Complete the parameter set, and re-start from step 5 of Operating Instructions.
HALTED:- CR	No card reader is available.	Make a card reader available and GO #X684.
HALTED:- TR	No paper tape reader is available.	Make a paper tape reader available and GO #X684.
HALTED:- LP	No line printer is available.	Make a line printer available and GO #X684.

<i>Message</i>	<i>Reason</i>	<i>Action</i>
HALTED:- INPUT FILE A EXCEPTION CONDITION 01	A long block has been found on the A input file.	Abandon the run.
HALTED:- INPUT FILE A EXCEPTION CONDITION 04	A parity failure has occurred on the A input file.	Abandon the run.
HALTED:- INPUT FILE B EXCEPTION CONDITION 01	A long block has been found on the B input file.	Abandon the run.
HALTED:- INPUT FILE B EXCEPTION CONDITION 04	A parity failure has occurred on the B input file.	Abandon the run.
HALTED:- OUTPUT FILE EXCEPTION CONDITION 04	A parity failure has occurred on the output file.	Abandon the run.
HALTED:- SEQUENCE ERROR FILE A	A record on the A input file has been found with keys out of sequence. The record counts will be printed, as at end of run, to allow the position of the faulty record to be determined.	Abandon the run.
HALTED:- SEQUENCE ERROR FILE B	A record on the B input file has been found with keys out of sequence. The record counts will be printed, as at the end of run, to allow the position of the faulty record to be determined.	Abandon the run.

PART 2 COMPARE

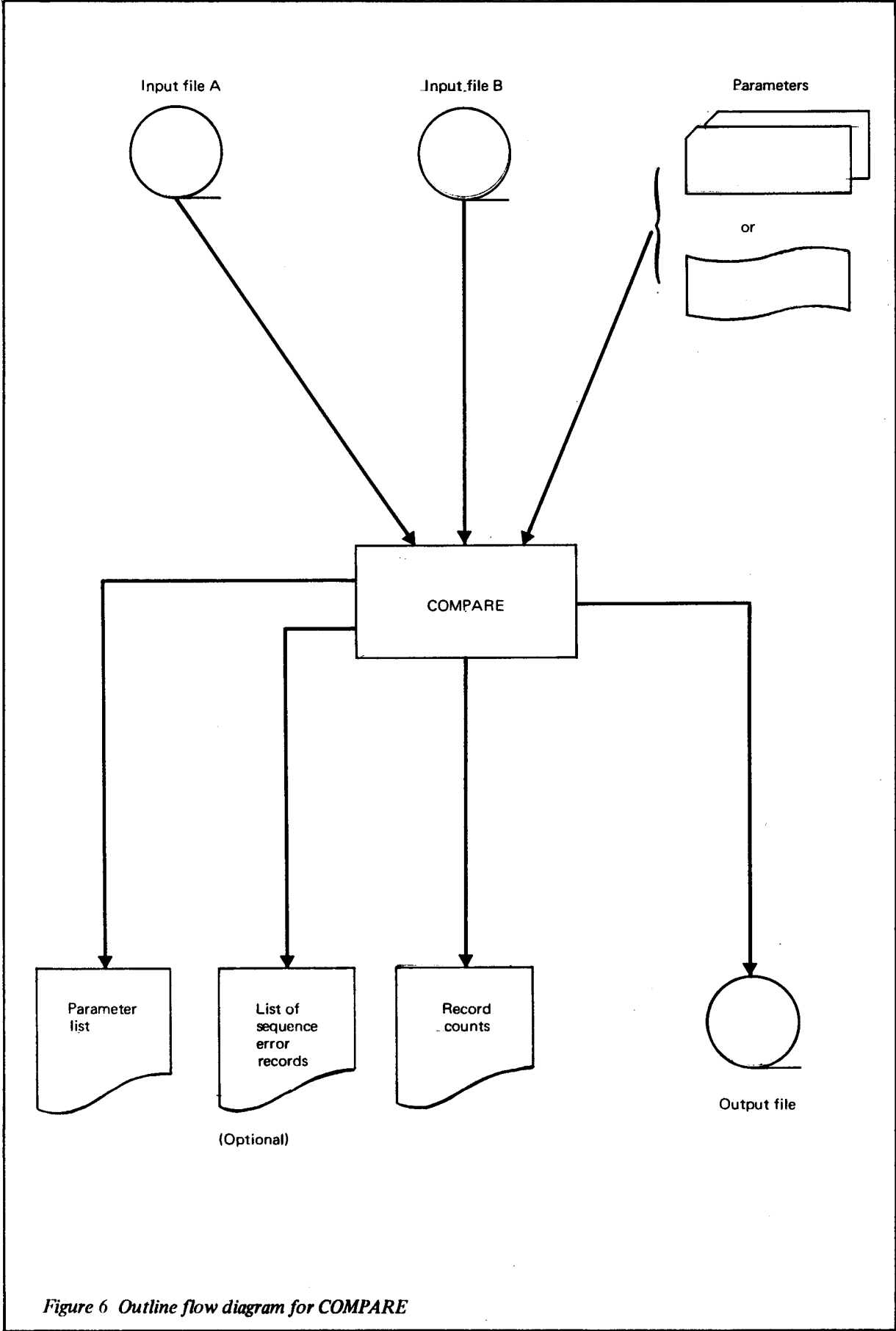


Figure 6 Outline flow diagram for COMPARE

Chapter 6 Introduction to COMPARE

COMPARE is designed to make easier the task of comparing the information held on two input magnetic tape files referred to as *File A* and *File B*. The comparison process begins with a comparison of the record keys in order to identify records on one file which have no corresponding records on the other file, secondly, a further comparison is made between the fields of matching records. In this way it is possible to isolate discrepancies between files at either record or field level.

Output from COMPARE consists of a single output file, which is created in accordance with parameters input by the user. The parameters (which are described in outline in this chapter and in detail in Chapter 8) enable the user to create one or more output records from any of the following:

- 1 An unmatched record from File A
- 2 An unmatched record from File B
- 3 A pair of records which are matched in keys, but which differ in the contents of other selected field or fields.
- 4 A pair of records which are matched both on keys and on the selected field or fields.

An output record may comprise the whole of the input record or records from which it is created, or only the contents of certain fields within such record or records. Certain other fields, not contained in the original records, may be output in accordance with the user's parameters. The final word of each output record is the *Selection word*, used by COMPARE to identify the output record type (see *Identification of output records*, below).

COMPARE provides the following facilities:

- 1 identification of output records according to the comparisons made
- 2 creation of multiple output records
- 3 selective comparison procedures

Each of these features is described in detail below.

USE OF COMPARE

COMPARE can be used for isolating discrepancies between two files which should be identical. Another use is to compare the contents of two versions of a master data file before and after an updating run. COMPARE is able to detect deletions (records on the earlier version which do not appear on the later), additions (records which appear on the later version but not on the earlier) and amendments (records which appear on both versions but disagree in detail). All relevant details of these records can be output by COMPARE.

The output file created by a COMPARE run may be used as a print tape to be input to the Data Management Software program REPRINT, see *Data Management Software: Reporting*. The records on the file must, however, be in the standard format for REPRINT as follows:

<i>Words</i>	<i>Contents</i>
0	Word count
1	Paper feed control character in position 1.3
2 to 41	The edited print line of up to 160 print positions
42	Selection word

The COMPARE user should insert the paper feed control character into the records by means of #FILL parameters. The various control characters are listed below:

<i>Type of Paper Movement</i>	<i>Control Character</i>	
	<i>With Printing</i>	<i>Without Printing</i>
	<i>Decimal</i>	<i>Decimal</i>
No movement	32	
Single line spacing	33	1
Double line spacing	34	2
Throw to head of form (track 1)	41	9
Throw to track 2	42	10
Throw to track 3	43	11
Throw to track 4	44	12
Throw to track 5	45	13
Throw to track 6	46	14
Throw to track 7	47	15

This table gives the decimal values which the user must enter in a #FILL parameter in order to insert the binary equivalents into the paper feed control character.

The tracks referred to in the table are tracks on a loop of paper tape fitted to the line printer which controls the position of the printing on the page.

PARAMETER PROCESSING

At the start of the run, a series of parameters is read. The format and logical validity of each parameter is checked immediately after reading.

The parameters allow the user to specify the following:

- 1 the files which are to be compared
- 2 the keys which are to be used to associate each pair of matching records
- 3 the comparisons which are then to be made
- 4 the parts of both records which are to be output

A list of parameters is printed, to assist in the correction of any errors and for subsequent reference. When all the parameters have been read, the program makes a further check: if the parameter set is incomplete, or if there have been any errors in the parameters, the program halts; otherwise it continues to the main processing stage. An outline flow diagram is given in Figure 6 facing page 19.

MAIN PROCESSING

The main processing stage of the system compares the two input files in accordance with the rules that have been specified, and prepares an output tape, containing all the details required for subsequent processing.

The input files are read in parallel, and the contents of the key fields are compared. Records with unmatched keys are identified as either File A or File B records and are processed accordingly.

Further user-specified comparisons are carried out on selected fields in the records found to have matching keys. This results in a further subdivision into those records which differ on any one or more of the specified fields and those which agree on all the specified fields.

This process is illustrated diagrammatically in Figure 7, page . The numbers 1 to 4 on the diagram mark the four conditions under which the user is able to specify that output records should be created.

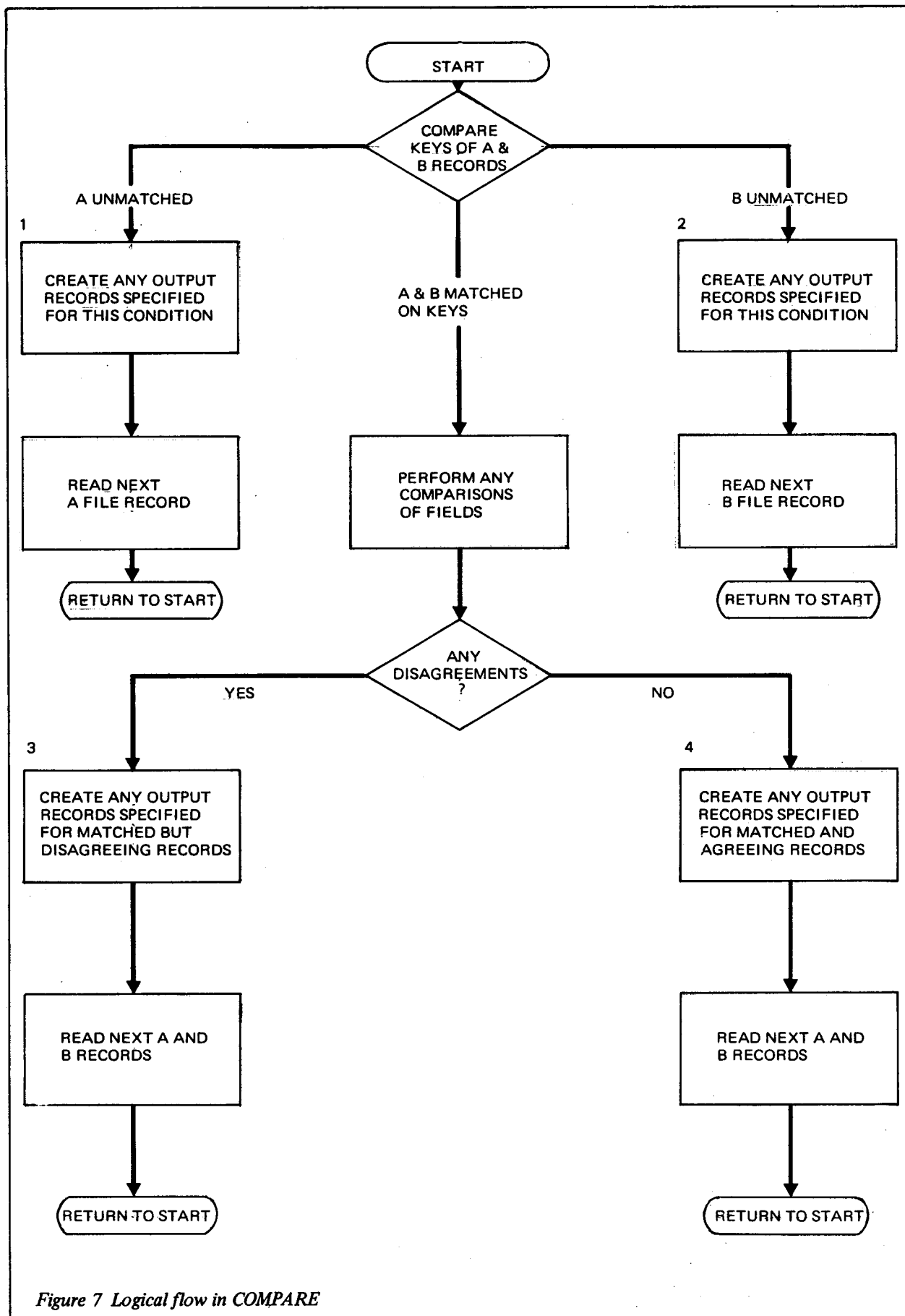


Figure 7 Logical flow in COMPARE

Each condition calls in the coding generated by one of the four sets of parameters specified by the user to produce the required output records.

Identification of output records

The final word of each output record, the *selection* word, is used to distinguish the various different types of output records and to identify the conditions under which those records were created.

Within each set of parameters there may be several groups of parameters. Each group must have a label and the additional comparisons that may be carried out on matching records must also have labels.

Each label is uniquely associated on input with one of the 24 bits in the selection word of each record. This means that all output records will have at least one bit set in the final word. Those records output under the third condition, with matching keys but differences in selected fields, may have several bits set since each comparison which fails will set a bit in the word.

Different labels may not set the same bit but the same label may be used for several different groups and/or comparisons. This allows records of a similar type to be grouped in a particular class. This is illustrated in *EXAMPLE 2* below.

The user may also include definitions of any identification words he considers to be necessary to facilitate the subsequent use of the file.

EXAMPLE 1

In this example a personnel file that has been updated is being compared with the same file before updating; the aim is to obtain a file containing details of all the changes that have taken place. For subsequent processing, the user wants to be able to distinguish changes in salary, changes in tax code, and changes in deductions from salary. The user also wishes to identify the records of staff who have joined the file, and of staff who have left the file. The parameters to cope with such a requirement could be arranged as shown below.

Set 1: Label JOIN

If File A is the most recent version of the personnel file, and File B is the same file before the most recent updating, any unmatched File A record must therefore represent a member of staff who has joined the file. The group of parameters in this set will cause an output record to be formed every time that an unmatched file A record is found.

Set 2: Label LEFT

The group of parameters in this set will cause an output record to be created every time that an unmatched record is found on the File B input. Such a record must have been removed in the updating of the file, and therefore represents a member of staff who has left.

Set 3: Label DIFF

Comparison of salary. Label SALY

Comparison of tax code. Label TAXC

Comparison of deductions. Label DEDS

The groups of parameters in this set will cause matched input records to be compared on the salary, tax code and deductions fields. An output record will be created if there is any difference in any of these fields.

This arrangement of parameters will cause the program to associate labels and bits in the selection word of each record as follows:

<i>Bit</i>	<i>Label</i>
0	JOIN
1	LEFT
2	DIFF
3	SALY
4	TAXC
5	DEDS

Thus the output record representing a new employee will have Bit 0 set; a record representing a leaver will have Bit 1 set. Records representing changes will all have Bit 2 set with at least one of Bits 3 to 5 also set; a record that has thoroughly changed would have all of Bits 2, 3, 4 and 5 set.

EXAMPLE 2

In the first example a single comparison is being made to establish whether there has been any change in the man's deductions from salary. In fact, more than one comparison would be necessary to establish this. So the parameters for Set 3 might be expressed as:

Set 3: Label DIFF

- Comparison of salary. Label SALY
- Comparison of tax code. Label TAXC
- Comparison of pension contribution. Label DEDS
- Comparison of national insurance. Label DEDS
- Comparison of social club subscription, Label DEDS

The same label can be used in more than one comparison. This means that, where it is helpful, different changes in the records can be indicated as belonging to the same class of change, and the associated bit will record any difference in that class. In this case any difference in the man's deduction will cause Bits 2 and 5 to be set, Bit 2 because it is a difference in matched records, and Bit 5 because it is a difference in deduction fields.

Multiple Output Records

The examples of parameters given have assumed that only one output record would be created under each of the conditions that the program was told to recognise. The user can create more than one output record under each condition, the different records again being distinguished by the use of different labels.

Thus, if the requirements of the personnel system which has been used in the previous examples are extended, three records may be required for each member who joined the staff; these records are to be created by COMPARE, and they will then be passed on to the computer systems which require them. The records are to be used to

- 1 Provide the employee's manager with a new staff record sheet;
- 2 Pay the employee through the salary system;
- 3 Record the employee for pension fund and social club requirements.

These requirements should be expressed by parameters as follows:

EXAMPLE 3

Set 1: Label MSRS

The group of parameters in this set will cause a record to be output with all the details necessary to create a new Manager's Staff Record Sheet.

Set 1: Label PAYS

The group of parameters in this set will create a record for use in the salary payment system.

Set 1: Label FUND

The group of parameters in this set will create a record for use in the pension fund, social club, and associated systems.

Each of these groups of parameters is identified as being a set 1. This refers back to the original diagram of the logic-flow of the program (Figure 7 on page 21) where it was shown that all unmatched File A records cause the program to follow the line of processing marked '1'.

The use of these labels will cause the program to allocate bits in the selection word as follows:

<i>Bit</i>	<i>Label</i>
0	MSRS
1	PAYS

<i>Bit</i>	<i>Label</i>
2	FUND

Multiple output records can be created on all four branches of the program. The program is able to cater for a maximum of 24 output record specifications in each run; these may be shared among the four branches in any proportion that suits the user's requirements.

Selection by Codes

A further facility is the ability for the user to specify that a certain output record is to be created only if certain conditions are fulfilled in the information held in the input record or records.

This ability to select by codes is simple and if more complex selections have to be performed, the input files should be pre-processed using the program GAMP (see the manual, *Data Management Software: Updating*) so that the different logical requirements can be met by a program with more extensive selection facilities.

COMPARE is able to select records in two ways. It can skip records in which a specified value occurs; or it can select only those records in which a specified value occurs.

Separate *skipping* and *only* requirements can be specified in each set of parameters.

If a set of parameters contains more than one *skipping* requirement, the input record will be ignored if any one of the requirements is met. This corresponds to an 'OR' operator in terms of GAMP logic. Thus a set could contain instructions to skip if the input record represented a man who was on weekly-paid staff, *or* who did not subscribe to the social club.

If a set of parameters contains more than one *only* requirement, the input record will be ignored unless it meets all the *only* requirements. This corresponds to an 'AND' operator in terms of GAMP logic. A set could therefore contain instructions that an output record was to be formed only if the input record represented a man who was on monthly salary *and* who contributed to the pension fund.

The same set of parameters may contain both *skip* and *only* requirements. These are specified separately, but the overall logic may be expressed as:

'An output record will be formed if the input record(s) meets *all the only* requirements *but none of the skip* requirements'.

This selection of input records allows the user to control the output of records in a more complex manner. In the examples which have been given so far, if an unmatched record was found on File A, the employee was considered to be a new employee and all set 1 parameters would cause output records to be created. These output records would be the same for all new employees. Selection could be used to give a different result, as in the following example.

EXAMPLE 4

Using the same overall framework of a personnel file being compared before and after updating, and with unmatched File A records still representing staff who have joined the file, separate output records are to be created for:

- 1 Staff on monthly salary and contributing to the pension fund
- 2 Staff on monthly salary but not contributing to the pension fund
- 3 Others

(Monthly salary is indicated by the value 7 in a field for type of payroll; pension fund contribution is decided by zero or non-zero in the contribution field.)

Set 1: Label MAND

Only if type of payroll is 7.

Skip if contribution is zero.

Output records will be created for the first category of new employees.

Set 1: Label MNOT

Only if type of payroll is 7.

Only if contribution is zero.

Output records will be created for the second category of new employees.

Set 1: Label REST

Skip if type of payroll is 7.

Output records will be created for the third category of new employees.

Ignoring input records

Example 4 also illustrates in more detail what is meant by the program 'ignoring' an input record. There are three groups of parameters all belonging to set 1. The selection criteria are such that an unmatched File A record can satisfy the requirement of only one of the three groups. An output record will therefore be created from the group which is satisfied, but the input record will be ignored by the other two groups, and therefore no such output records will be created. A record will be totally ignored only if it fails to satisfy the criteria for any group in the appropriate set.

Input records can also be ignored if the user does not supply parameters for the sets that are to be ignored. A case might arise where no output was required from matched pairs of input records, but output was required for unmatched File A records and for unmatched File B records. To achieve this, the user should supply parameters under set 1 and under set 2, but no parameters under either set 3 or 4.

The opposite case could also arise, that is, where output was required only from matched pairs of records. This can be achieved if the user supplies no parameters under set 1 or set 2. At least one set 3 or set 4 would have to be supplied. If the user wished to distinguish between agreeing and disagreeing matched records, he could supply parameters under set 3 and set 4.

CONTENTS OF PARAMETER SETS

Several of the facilities which the user will require have been discussed in general terms already. This section explains the formal rules by means of which the user's field comparison requirements can be expressed in parameters, and also explains the editing facilities which can be used to create the output records.

The set header parameter (**#SET**) introduces the parameters which define the criteria for forming a particular output record. These can be summarised as follows:

- 1 The **#SET** parameter contains a label which causes the same bit of the section word to be set on all output records formed from this set of parameters.
- 2 The **#SET** parameter defines whether the parameters refer to conditions 1, 2, 3 or 4 of the program; i.e. whether the output record is to be formed from unmatched File A records, unmatched File B records, matched File A and B records which disagree in the details of specified fields, or matched File A and B records which agree in the detail of the specified fields.
- 3 Within the set, **#SKIP** and **#ONLY** parameters determine whether the set is to be obeyed for all relevant input conditions or only under specific sub-sets of those conditions.

The remainder of the parameters that form a set are concerned with the actual formation of the output record. They are described in detail in Chapter 8; this section merely outlines some possibilities.

A **#CLEAR** parameter may be given, instructing the program to clear the area in which the output record will be formed to all zeroes or all blanks. Only one **#CLEAR** parameter may be given for each group. This parameter allows the user to ensure that all areas not filled by actual transfers of data will be consistently zero or consistently blank.

#MOVE parameters allow the user to specify fields that are to be transferred from the input records to the output record. During transfer no change is made in the content of the field. No conversion facilities are included.

#FILL parameters allow the user to insert constant information in all output records created under the control of the current set. This can be used if, for example, output records cannot be adequately distinguished by the configuration of the bits set in the selection word. If different sets could create the same pattern of bits, the output records could be distinguished by different 'filled' characters.

#COMPARE parameters allow the user to define the fields of matched input records which are to be compared. These parameters may only occur in set 3. The fields are compared by the program for any disagreement; the program detects only difference or identity; it does not distinguish between greater and lesser, positive and negative, zero and non-zero by indicating which field was the greater, etc. The user may specify what action is to be taken with the compared fields, depending on the level of disagreement found. The three possibilities are:

- 1 Do not output the field.
- 2 Output the field if it disagrees with the corresponding field in the other record.
- 3 Output the field if there is any disagreement in any of the comparisons specified in the current set.

These three possibilities can be specified for each of the two fields in each comparison.

In all transfers, whether under #MOVE, #FILL or #COMPARE parameters, the user has full control over the amount of information transferred and over its destination address in the output record.

Common Transfers

Since many applications which require different output records will require each of these records to contain certain standard fields in standard positions, the program is able to carry out such insertion of common fields in output records.

The most obvious case where such a facility is required is for dealing with key information; for example, the personnel number which identifies a record as referring to a particular employee. This would clearly have to appear in every output record, or the following systems would be unable to deal with them correctly.

There may be many different patterns of output record and the user can therefore specify that certain transfers are *common*. These are presented to the program under the control of a special set header parameter #COMMON which identifies them as common transfers.

A move transfer or a fill transfer can be specified as common. Other types of parameters must always be under the control of the normal set header parameter.

If the user has specified common transfers from the File A input record and from the File B input record, all such transfers will be obeyed when a matched pair of input records is found. Where an unmatched File A record is found, the program will ignore any common transfers which refer to File B and vice versa.

LIMITATIONS TO COMPARE

The program is able to cope with the following:

Maximum block size	:	512 words
Maximum record size	:	512 words; on output this means 1 for word-count 510 for data 1 for selection word
Maximum number of groups of parameters	:	24, which may be shared in any proportion among the four sets of parameters in the program.
Maximum number of comparisons	:	256, which may be shared in any proportion among set 3 groups of parameters.
Maximum number of moves	:	256, which may be shared in any proportion between Common and sets 1 to 4.
Maximum number of fills	:	48, which may be shared in any proportion between Common and sets 1 to 4.
Maximum number of fill characters	:	256, which may be shared in any proportion among the 48 Fills.
Maximum number of skips	:	48, which may be shared in any proportion among sets 1 to 4.
Maximum number of onlys	:	48, which may be shared in any proportion among sets 1 to 4.
Maximum number of clears	:	1 per set.

MINIMUM CONFIGURATION

The minimum configuration for the COMPARE program is:

- 1 1900 Series central processor with 16K core store
- 1 card reader or paper tape reader
- 1 line printer
- 3 tape decks.

Chapter 7 Input and output

INPUT

Input to COMPARE consists of two input files, and input parameters.

Input files

The program requires two input files on magnetic tape. These may be single-reel or multi-reel files. They must observe all MTH standards. The details of the tape labels are specified by standard #READ parameters.

The files must be sorted to the same order, so that the program is able to match records from the two files. The user specifies the key fields on which the files are sorted, and on which the records are to be matched. The keys may be in character or binary form; they may be in ascending or descending sequence.

Parameters

The function of the parameters was explained under *Contents of Parameter Sets*. The format of the parameters is given in detail in Chapter 8.

OUTPUT

Output from COMPARE consists of a single output file, line printer output, and console output.

Output file

The main output of the program is a file on magnetic tape of the output records formed in accordance with the user's parameters. This file observes all MTH standards and may be single-reel or multi-reel. The details for the label of any first output reel are provided by the #WRITE parameter. If the program has to open continuation output reels the same file name and file generation number are used, but the reel sequence number is increased by one for each output reel opened.

The sequence of records on the output file will be determined overall by the sequence to which the input files were sorted. Thus output file records will follow the key order of the input files. But where several output records have been created from a single input record or matched pair of input records, the sequence in which these are written will be determined by the sequence in which the parameter sets were presented.

Line printer output

At the beginning of the run the program produces a printed list of the parameters. Each parameter is printed out exactly as it was read, with no editing or conversion of the information. Error codes may appear alongside the parameter depending on whether the program was able to validate the parameter fully, partly, or not at all.

When all the parameters have been printed, the program prints a verdict line, showing whether there has been any error or incompleteness in the parameters. If there have been any errors, a list is also printed of the interpretation of the error codes.

The program also prints a table of the labels which it has accepted, showing which bit in the selection word of each record is associated with each label.

All the above items are printed at the start of the run. As soon as this has been completed, the program releases the printer, so that it is available for use by other programs.

During the run the program does not use a line printer unless it detects that an input record is out of sequence, in which case it allots a printer and prints out details of the record which is out of sequence. The program halts on the occurrence of the first sequence error.

At the end of the run, the program prints a statement of the numbers of records which it has processed, both on input and output. It states how many of these were matched or unmatched; it also shows how many output records occur for each bit in the selection word that is set.

Console output

See under *Exception conditions*, Chapter 10.

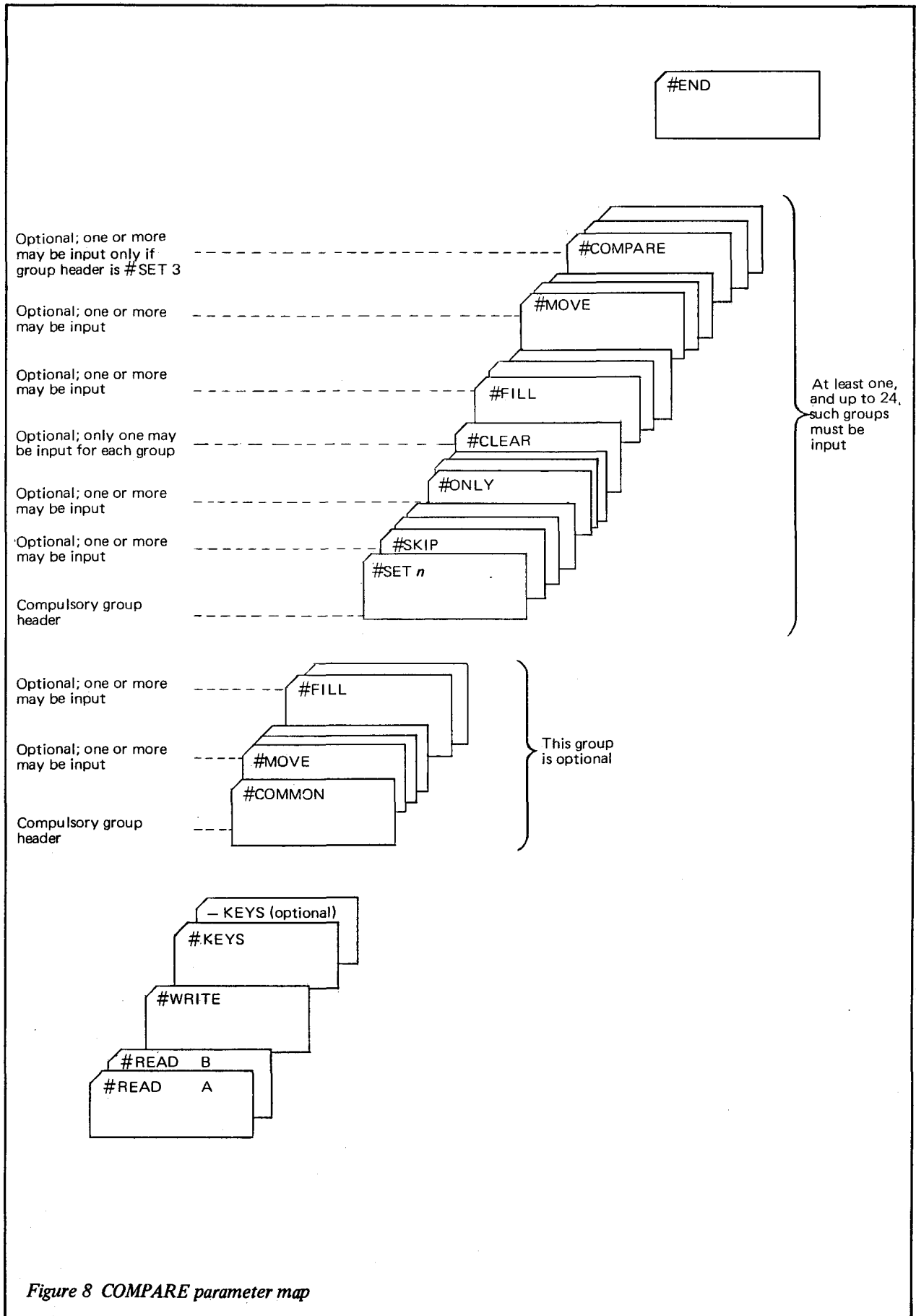


Figure 8 COMPARE parameter map

Chapter 8 The parameter set

The parameters may be supplied to the program in the sequence shown in Figure 8. The following points should be noted:

- 1 The parameters must finish with the standard #END, end of parameter marker.
- 2 Each set of parameters defining the requirements for a particular output record must be preceded by the appropriate set header.
- 3 The order in which sets are presented to the program determines the order in which bits in the selection word of each record are allocated to particular labels. Since the user's further processing will be dependent upon patterns of bits set by COMPARE he must present his sets of parameters in the sequence which corresponds with the requirement of such further processing.

READ TAPE LABEL PARAMETER (#READ)

This parameter is in standard form. Its function is to define the labels on the input tape files, and the modes in which they are to be opened. Two #READ parameters must be provided for every run, one for File A and the other for File B.

Field	Columns	Contents
A	1 to 5	The directive #READ
	6	Opening mode: 1 character as follows: 1 = no checks for presence or absence of write permit ring, 2 = checks for absence of write permit ring. 3 = check for presence of write permit ring. ∇ (space) in this position is interpreted as 2.
B	8 to 19	12 character file name
C	21 to 24	Reel sequence number: 4 digits.
D	26 to 29	File generation number: 4 digits.
E	31	Designation: 1 alphabetic character, A or B.

Note: These are equivalent to the standard modes of opening tapes, that is #100, #200 and #300. Details of additional checks by Executive on the 1904 and above will be found in the ICL manual *Magnetic Tape*.

Example

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			
#	R	E	A	D			P	E	R	S	O	N	N	E	L	D	T	L																								
#	R	E	A	D			P	E	R	S	O	N	N	E	L	D	T	L																								

WRITE TAPE LABEL PARAMETER (#WRITE)

This parameter is in standard form. Its function is to define the label to be written to the output file. If fields G, H and I are punched, the program will search for a tape with the specified label and open it for output; if these fields are blank, the program will open any available scratch tape.

If more than four keys need to be specified, a continuation card can be punched; this takes the following form:

Field	Columns	Contents
A	1 to 5	The directive -KEYS.
B	7 to 11	The start address of a key in File A expressed in words and characters relative to the start of the record, in the form <i>NNN.N</i> .
C	13 to 17	The start address of a key in File B expressed in words and characters relative to the start of the record, in the form <i>NNN.N</i> .
D	19 and 20	Field size: a 2 digit integer giving the length of the key in characters for character keys or words for binary keys.
E	21	A single character indicating the type of field: H = characters ascending % = binary ascending D = characters descending * = binary descending
F,G,H,I	23 to 37	Similar to fields B to E, defining the sixth key.

Example

1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1	2	3	4	5	6	7	8	9
#	K	E	Y	S					1	.	0				1	.	0			1	%				2	.	0			2	.	0			1	6	H	
-	K	E	Y	S					8	.	0				8	.	0			1	%																	

1	2	3	4	5	6	7	8	9	50	1	2	3	4	5	6	7	8	9	60	1	2	3	4	5	6	7	8	9	70	1	2	3	4	5	6	7	8	9	80
6	.	0							6	.	0								1	*																			
																				7	.	0																	

COMMON TRANSFERS PARAMETER (#COMMON)

As explained on page 26, transfers which are to be performed on all records are more economically specified by means of #MOVE and #FILL parameters grouped in a special set. This set is introduced by the #COMMON parameter, which is given below and indicates that the parameters which follow are to have a common application; #COMMON is therefore a special kind of set header.

When transfers specified as COMMON are not relevant, the program will bypass them; for example if a COMMON MOVE parameter transfers information from a File B input record to output, this MOVE will be bypassed if the program is forming an output record from an unmatched File A input record; and vice versa.

The use of the #COMMON parameter is optional.

Field	Columns	Contents
A	1 to 7	The directive #COMMON

Example

1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1	2	3	4	5	6	7	8	9	40
#	C	O	M	M	O	N																																	

SET HEADER PARAMETER (#SET)

Each set of parameters must be preceded by a #SET parameter which defines the nature of the records on which the parameters that follow it are to operate. (These parameters may be any or all of #CLEAR, #SKIP, #ONLY, #COMPARE, #MOVE, #FILL.)

According to the digit in Field B of the set header the parameters will operate on

- 1 non-matching File A records
- 2 non-matching File B records
- 3 matching File A and File B records which disagree at selected field level
- 4 File A and B records which match on both keys and the selected fields.

Field	Columns	Contents
A	1 to 4	The directive #SET.
B	5	A single digit, either 1, 2, 3 or 4. This defines the records on which the set is to operate. 1 = non-matching File A records. 2 = non-matching File B records. 3 = matching File A and B records which disagree at the selected field level. 4 = File A and B records which match on both keys and the selected fields.
C	7 to 10	Set label: 4 characters. This is described in more detail on page 22.
D	12 to 14	Output record length: a 3 digit integer, minimum value 2 and maximum value 512. The record length includes the count-word and selection word. If this field is blank, the program assumes a value of 43.

Example

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		
#	S	E	T	1			J	O	I	N		3	7																												
#	S	E	T	3			I	N	C	R																															

RECORD SELECTION PARAMETERS

#SKIP

This parameter can be used to select the conditions under which a particular output record specified in a set which contains this parameter will not be formed. The corresponding input record must contain the designation entered in the relevant fields of the #SKIP.

There may be more than one #SKIP parameter in a group. If this is the case, the output record will not be formed if any one of the skipping requirements is met.

The group in which #SKIP parameters are present must always be preceded by a #SET parameter. Different skipping requirements may be specified in different groups.

Field	Columns	Contents
A	1 to 5	The directive #SKIP.
B	7	File identity defines the input file which will determine whether the set is to be skipped or not.

Field	Columns	Contents
		A single character, which must be either A or B.
C	9 to 13	Designation starting position: the starting position of the field which contains the designation that is to control skipping; it is expressed in words and characters relative to the start of the input record, in the form <i>NNN.N</i> .
D	15 and 16	Designation length: this determines the length of the control designation; if the designation is a character field the length is expressed in characters, if it is a binary field the length is expressed in words. The maximum size of field is 08 characters or 02 words. (Two digits are allowed for this field in order to conform with the standard for position and length fields in all parameters; the first digit must always be zero.)
E	17	Designation type: a single character, which defines the type of the designation as it is held in the input record. The permitted values are: H = characters % = binary integer.
F	18 to 25	Designation value: if type is H, this field may contain up to 8 characters, left justified. If type is % the field contains the decimal equivalent of the required number; this is left justified.

Example

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	
#	S	K	I	P		A				6	.	0				1	%	7	3	4	1	3																		
#	S	K	I	P		B				7	.	0				4	H	A	C	C	T																			

#ONLY

This parameter can also be used to select the conditions under which an output record is to be formed from the transfers specified in a particular set. It works in the opposite sense to #SKIP, in that an output record will be formed only if the input record contains the specified designation.

There may be more than one #ONLY parameter in a group; in this case the output record will be formed only if all the only requirements are met.

The group in which #ONLY parameters are present must always be preceded by a #SET parameter. Different selection requirements may be specified in different groups.

Field	Columns	Contents
A	1 to 5	The directive #ONLY
B	7	File identify: either A or B; as in #SKIP
C	9 to 13	Designation starting position as in #SKIP
D	15 to 16	Designation length as in #SKIP
E	17	Designation type as in #SKIP
F	18 to 25	Designation value as in #SKIP

Example

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	
#	O	N	L	Y		B				1	3	.	0			2	%	4	3	6	7	4	8																	
#	O	N	L	Y		A				1	7	.	2			2	H	C	R																					

CLEARING REQUIREMENTS PARAMETER (#CLEAR)

This parameter can be used to clear the area in which each output record is formed, before any of the transfers which create the record are made. In this way, the user can be sure that any areas of the output record which are not filled by specific transfers will contain constant information, and not any unwanted data, e.g. data from a previous record formed in the same area.

This parameter is optional. By omitting this parameter the user could save a small amount of processing time on each record, if each output record is completely filled by transfers specified under other parameters.

The program clears to zero the area in which all output records are formed before the start of the main processing. Therefore if all output records have the same format so that there is no risk of a gap in one record corresponding to information in another record, the user can omit the clearing parameter even when there are gaps in the specification of each output record; this is because these gaps will be filled by the zeros originally placed there by the program.

Field	Columns	Contents
A	1 to 6	The directive #CLEAR
B	8	Clearing requirement: a single character, either blank or zero. This character will be used to fill the output record area before the record is formed. No other characters can be used for clearing.

Example

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	
#	C	L	E	A	R																																			
#	C	L	E	A	R	0																																		

Note: The group in which a #CLEAR parameter is present must be preceded by a #SET. If there is more than one #SET, clearing requirements must be specified separately in each group. Each group can contain only one #CLEAR parameter.

MOVE PARAMETER (#MOVE)

This parameter defines a transfer of data that is to take place if an output record is being formed. The parameter is in standard form.

The group in which this parameter appears may be preceded either by #COMMON or by #SET. #MOVE must not appear without one of these headers.

There may be more than one #MOVE parameter in a group, and there may be more than one after #COMMON.

Up to three moves may be specified in a single parameter; when reckoning up the number of moves which the program is able to accept, a parameter with three moves is counted as three, and not as one.

Field	Columns	Contents
A	1 to 5	The directive #MOVE.
B	7	File identity: defines the input file from which the transfer is to be made. Either A or B.
C	9 to 13	Source starting position: the starting position of the field in the input record from which the transfer is to be made. This is expressed in words and characters relative to the start of the input record, in the form <i>NNN.N</i> .
D	15 to 19	Destination starting position; the starting position of the field in the output record to which it is to be transferred. This is expressed in words and characters relative to the start of the output record, in the form <i>NNN.N</i> .
E	21 to 22	Field length: the length of the field which is to be transferred. This can be expressed in words or characters, without regard to the form of the actual content of the field;

Field	Columns	Contents
		it should be in characters if either the source address or the destination address or both are incomplete words; it may be in words if the field in both input and output records fits exactly within word boundaries.
F	23	Field type: H if the field length is expressed in characters; blank if the field length is expressed in words.
G	25 to 41	Specification of a second move, as columns 7 to 23 above.
H	43 to 59	Specification of a third move, as columns 7 to 23 above.

Example

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	
#	M	O	V	E		B					2	.	0						2	.	0		4		A				7	.	0					6	.	2

40	1	2	3	4	5	6	7	8	9	50	1	2	3	4	5	6	7	8	9	60	1	2	3	4	5	6	7	8	9	70	1	2	3	4	5	6	7	8
1	H		A							7	.	1							7	.	0		3	H														

Note: In translating the parameter, the program confirms that the terms used for defining field position length and type are consistent and will reject as invalid a move which tries to transfer as words items which do not exactly fit between word boundaries. Wherever possible, word transfers should be specified in the interest of faster processing.

INSERT CONSTANTS PARAMETER (#FILL)

This parameter defines information which is to be inserted in the output record, but which cannot be derived from the input records. Whereas varying information can be derived from the input records, this parameter provides a means of inserting constant information in output records. The parameter is in standard form.

Where more than one type of output record is being produced, this parameter provides a means of inserting codes to identify the several types of output record.

The group in which this parameter appears must be preceded by either #COMMON or #SET; it must not appear without one of these headers.

There may be more than one insertion under #COMMON. There may be more than one insertion in any group, and different insertions can be used in different groups.

Field	Columns	Contents
A	1 to 5	The directive #FILL.
B	7 to 11	Destination starting position: the starting position of the field in the output record in which the insertion is to be made. This is expressed in words and characters relative to the start of the output record, in the form <i>NNN.N</i> .
C	13 to 14	Insertion length: the length of the field in the output record in which the insertion is to be made. Columns 13 and 14 give the number of characters to be inserted if the Column 15 contains H, and the number of words if Column 15 contains %. Maximum is 60 if Column 15 contains H, and 02 if column 15 contains %.
D	15	Insertion type - a single character, defining the type of insertion to be made: H = characters % = binary integer.
E	16 to 75	Insertion value: the actual information to be inserted. If insertion type is H, this is in the form of a string of characters, left-justified.

Field *Columns* *Contents*

If insertion type is % the decimal equivalent of the required binary value inserted: this is also left-justified. It will be converted to a single or double length binary integer before the insertion is made (the length is defined in Columns 13 and 14, 01 for one word, and 02 for two words).

Example.

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					
#	F	I	L	L					1	1	.	3			1	0	H	D	E	P	A	R	T	M	E	N	T																	
#	F	I	L	L					1	0	.	0			1	%	1	7	3	4	0																							

COMPARISON PARAMETER (#COMPARE)

This parameter defines the comparisons which are to be made between fields from matching File A and B input records.

The group in which this parameter appears must be preceded by #SET, and the only set-type which is valid is #SET3. If the parameter occurs without the appropriate header it will be rejected.

More than one comparison requirement parameter may be specified in the same group and different comparisons may be specified in different groups.

<i>Field</i>	<i>Columns</i>	<i>Contents</i>
A	1 to 8	The directive #COMPARE
B	10 to 13	Comparison label: a four character label, which defines which bit of the selection word is to be set if there is any disagreement as a result of the comparison. This is discussed in more detail on page 22.
C	15 to 19	File A address: the starting address of the field in the File A input record which is to be compared. This is expressed in words and characters, relative to the start of the A file input record, in the form <i>NNN.N</i> .
D	21 to 25	File B address: the starting address of the field in the File B input record which is to be compared. This is expressed in words and characters, relative to the start of the File B input record, in the form <i>NNN.N</i> .
E	27 to 28	Field length: the length of the fields to be compared. This is expressed in characters; since both fields must be the same length, only one such field is present.
F	29	Field type: since the length is expressed in characters, this field must contain H regardless of the actual content of the fields to be compared. It is included in order to preserve the standards for defining field length and type.
G	31	Action code for File A: defines the action to be taken with the File A field depending on the result of the comparison. The possible values are: 0 Do not transfer File A field to the output record. 1 Output the File A field if there is a disagreement in this comparison. 2 Output the File A field if there is any disagreement in any of the comparisons in the current set.
H	33	Action code for File B field: as above. The transfer of the File B field to output can be independently controlled.
I	35 to 39	Destination address for the File A field: the starting address of the field in the output record to which the File A field is to be transferred. This is expressed in words and characters relative to the start of the output record, in the form <i>NNN.N</i> . The field may be left blank if the A field action code is 0; if the action code is 1 or 2, an address must be given.

<i>Field</i>	<i>Columns</i>	<i>Contents</i>
J	41 to 45	Destination address for File B: this corresponds to the information for the File A field.

Example

1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1	2	3	4	5	6	7	8	9	40																																								
#	C	O	M	P	A	R	E			D	I	F								1	3									2										1	7	.	3							6	H									0										1									
1	2	3	4	5	6	7	8	9	50	1	2	3	4	5	6	7	8	9	60	1	2	3	4	5	6	7	8	9	70	1	2	3	4	5	6	7	8	9	80																																								
2	1	.	1																																																																												

END OF PARAMETERS MARKER

This parameter is in standard form. It indicates the end of the entire parameters set.

<i>Field</i>	<i>Columns</i>	<i>Contents</i>
A	1 to 4	The directive #END

Example

1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1	2	3	4	5	6	7	8	9	40
#	E	N	D																																				

1 All staff joining the file

0	1 – 5	6	–	21	22
23	KEYS	CURRENT SALARY DETAILS			

2 Staff joining the file and joining the pension fund

0	1 – 5	6	–	21	22	–	31	32
33	KEYS	CURRENT SALARY DETAILS			PENSION FUND DETAILS			

3 Staff with changes of salary details

0	1 – 5	6	–	21	22	–	36	37
38	KEYS	CURRENT SALARY DETAILS			PREVIOUS SALARY DETAILS			

Figure 9 COMPARE example – output record formats

Continued on page 42

Chapter 9 Example

This example brings together some of the points mentioned in the earlier examples of the use of parameters (see pages 22 to 25).

A company's personnel file is updated, and the two versions of the file (before and after) are compared in order to distinguish the changes that have been made.

The records on the file contain the following information:

Words 1 to 5	Basic identifying information: personnel number, date of birth, etc. These items of information form the keys of the record.
Words 6 to 9	Location details: building code, department code.
Words 10 to 19	Name and initials.
Words 20 to 49	Home address.
Words 50 to 59	Telephone numbers: office internal, office external, home.
Words 60 to 75	Current salary, current year's PAYE details.
Words 76 to 85	Pension fund contribution details.
Words 86 to 94	Miscellaneous deductions: social club, sports club, etc.
Words 95 to 99	Job code, experience, qualifications.

The following conditions are to be covered by distinct output records of the given format:

1 Staff joining the file.

For all staff joining the file, records are to be created containing key fields (personnel number, date of birth, etc.) and also complete salary details. These records are associated with the label JON1.

2 Staff joining the file, pensionable.

For staff joining the pension fund, records are to be created containing key fields, and also containing current salary details and current pension fund details. These records are associated with the label JON2. Joining the pension fund is indicated by a code '7' occurring in the first character of word 76 of the input record.

3 Staff with changes of salary details.

If there has been any change in the salary details fields, a record is to be created containing key fields, revised salary details, and previous salary details. These records are associated with the label CHA1.

4 Staff with changes of telephone number.

Records are to be created with the key fields and the revised telephone numbers. All three telephone numbers are to be output, even though not all of them may have changed. These records are associated with the label CHA2, and with labels CHA3, CHA4 and CHA5 according to which telephone numbers have changed.

5 Staff with changes of miscellaneous deductions.

Records are to be created with the key fields and the revised miscellaneous deductions. The records are also to contain the text: 'DEDUCTIONS ARE NOW:'. These records are associated with the label CHA6.

6 Staff leaving the file, pensionable.

Records are to be created containing key fields, previous salary details, and previous pension fund details. These records are associated with the label DEL1.

7 Staff leaving the file, non-pensionable.

Records are to be created containing key fields and previous salary details; the fields used for pension fund

4 Staff with changes of telephone number

0	1 - 5	6	-	15	16
17	KEYS	NEW TELEPHONE NUMBERS			

5 Staff with changes of miscellaneous deductions

0	1 - 5	6	-	10	11	-	19	20
21	KEYS	"DEDUCTIONS ARE NOW:"			NEW MISCELLANEOUS DEDUCTIONS			

6 Staff leaving the file, ex-members of the pension fund

0	1 - 5	6	-	21	22	-	31	32
33	KEYS	PREVIOUS SALARY DETAILS			PENSION FUND DETAILS			

7 Staff leaving the file, not ex-members of the pension fund

0	1 - 5	6	-	21	22	-	31	32
33	KEYS	PREVIOUS SALARY DETAILS			ALL ZERO			

Figure 9 continued: COMPARE example -- output record formats

details in the previous type of record are to contain zero's. The records are associated with the label DEL2.

Notes on the example parameters

Two versions of the same file are being compared. The #READ parameters therefore contain the same filename and reel sequence number, but differ on file generation. File A has a higher file generation number; it is therefore the most recent version of the file.

The output tape is given a file generation number to match that of File A input file, and a retention period of 30 days.

The keys are treated as a single string of 20 characters. For the purpose of this example a very simple key-structure is adequate.

Under #COMMON a pair of moves are specified. These will serve to move the key fields from the input record to the output record. This saves specifying a move for the key fields in each of the following sets. The same information is moved from each input file, and the same destination address is given in the output record. If the record is only present in File A, the File B move will be ignored and vice versa. If the record is present in both input files, both moves will be made; one of these in fact will be redundant, since the same information will be moved twice to the same location. Note that the two moves are specified slightly differently; one as 20 characters, the other as 5 words. The latter form is preferable, as it will be performed slightly faster.

#SKIP and #ONLY are used to distinguish pension fund contributors from non-contributors. In the different groups of set 3 parameters all changes of salary are indicated under label CHA1; any change of telephone number will be indicated by the label CHA2, but in addition CHA3, CHA4, and CHA5 are used to indicate precisely which number has changed. Finally, all changes of miscellaneous deductions are indicated by the label CHA6.

In this example the different output records have been given different lengths. The selection word will therefore be in a different place for each type of record. Accordingly it will be rather more complicated to distinguish output records of a particular type in any subsequent analysis jobs, than it would be if all the records were of the same length. The alternative would be to give all records the same length and to include a #CLEAR parameter in each set to ensure that unused fields were properly emptied.

```
11/10/68    COMPARE - LIST OF PARAMETERS

#READ2 PERSONELFILE 0000 0016 A
#READ2 PERSONELFILE 0000 0015 B
#WRITE CHANGES FILE 0000 0016 0030 A
#KEYS 001.0 001.0 20H
#COMMON
#MOVE A 001.0 001.0 20H B 001.0 001.0 05
#SET1 JON1 023
#MOVE A 060.0 006.0 16
#SET1 JON2 033
#ONLY A 076.0 01H7
#MOVE A 060.0 006.0 16 A 076.0 022.0 10
#SET3 CHA1 038
#COMPARE CHA1 060.0 060.0 64H 2 2 006.0 022.0
#SET3 CHA2 017
#COMPARE CHA3 050.0 050.0 12H 2 0 006.0
#COMPARE CHA4 053.0 053.0 12H 2 0 009.0
#COMPARE CHA5 056.0 056.0 16H 2 0 012.0
#SET3 CHA6 021
#FILL 006.0 20HDEDUCTIONS ARE NOW
#COMPARE CHA6 086.0 086.0 36H 2 0 011.0
#SET2 DEL1 033
#ONLY B 076.0 01H7
#MOVE B 060.0 006.0 16 B 076.0 022.0 10
#SET2 DEL2 033
#SKIP B 076.0 01H7
#CLEAR 0
#MOVE B 060.0 006.0 16
#END
```

Figure 10 COMPARE example printouts

(continued overleaf)

11/10/68 COMPARE - LIST OF SELECTION WORD BIT INDEX

RECORD LABELS AND SELECTION WORD BITS ARE ASSOCIATED AS FOLLOWS

BIT	LABEL
0	JON1
1	JON2
2	CHA1
3	CHA2
4	CHA3
5	CHA4
6	CHA5
7	CHA6
8	DEL1
9	DEL2

04/11/68 COMPARE - RECORD COUNTS AT END OF RUN

FILE A IS : PERSONELFILE/0000/0016 TSN: 10236

FILE B IS : PERSONELFILE/0000/0015 TSN: 10721

OUTPUT FILE IS: CHANGES FILE/0000/0016 TSN: 10558

FILE A RECORDS READ : MATCHED- 25126 UNMATCHED- 319 TOTAL- 25445

FILE B RECORDS READ : MATCHED- 25126 UNMATCHED- 278 TOTAL- 25404

OUTPUT RECORDS: FROM UNMATCHED A- 517 FROM UNMATCHED B- 278 FROM MATCHED A B- 1329 TOTAL- 2124

SELECTION WORD BITS ARE SET IN THE FOLLOWING NUMBERS OF RECORDS:

BIT	LABEL	COUNT
0	JON1	319
1	JON2	198
2	CHA1	601
3	CHA2	378
4	CHA3	214
5	CHA4	208
6	CHA5	17
7	CHA6	350
8	DEL1	134
9	DEL2	144

NORMAL END OF RUN

Figure 10 COMPARE example printouts (continued)

Chapter 10 Operating instructions and exception conditions

OPERATING INSTRUCTIONS

Action	Console Message
1 Load input tape A without write permit ring,	
2 Load input tape B without write permit ring.	
3 Load output tape with write permit ring.	
4 Load parameters in the appropriate reader.	
5 Load the program COMPARE by inputting:	FI #X685 #TAPE
6 To read in the parameters either	
(a) if the parameters are on paper tape input:	GO #X685 20
or	
(b) if the parameters are on cards input:	GO #X685 21
7 The program reads the parameters, prints a list of parameters and an index of bits in the selection word. If there have been parameter errors, the program prints a list of error code interpretations, releases the reader and the printer and ends the run with the appropriate exception condition message, see below.	
8 If all parameters are correct, the program continues automatically: opens all tape files, reads input tapes, writes output tape, closes all tapes, allots printer, prints record counts, releases printer, ends the run with the message:	HALTED:- END OF RUN
9 When it is required to terminate a partly-completed run, input:	GO #X685 26
the program closes all tapes, allots printer, prints record counts, releases printer and outputs the message:	HALTED:- RUN ABANDONED
following any tape error, the program closes all tapes, allots printer, prints details of tape error type and position, prints record counts, releases printer and closes with the relevant exception condition message, see below.	

EXCEPTION CONDITIONS

<i>Message</i>	<i>Reason</i>	<i>Action</i>
HALTED:- PARAMETERS INCORRECT	There are errors in the parameters but the program would be able to operate on the valid parameters.	Examine the errors on the printout and if the errors are acceptable GO #X685, otherwise abandon the run.
HALTED:-PARAMETERS WRONG - NO RESTART	There are errors in the parameters and the program is unable to continue.	Abandon the run.

<i>Message</i>	<i>Reason</i>	<i>Action</i>
HALTED:–PARAMETERS INCOMPLETE	One or more of mandatory parameters has been omitted.	Abandon the run.
HALTED:–PARAMETER STORAGE FULL	Too many parameters have been provided.	Abandon the run.
HALTED:– SEQUENCE ERROR	Sequence error detected in input files.	Abandon the run.
HALTED:–INPUT FILE A EXCEPTION CONDITION 01	A long block has been found on the the A input file.	Abandon the run.
HALTED:–INPUT FILE A EXCEPTION CONDITION 04	A parity failure has occurred on the A input file.	Abandon the run.
HALTED:– INPUT FILE B EXCEPTION CONDITION 01	A long block has been found on the B input file.	Abandon the run.
HALTED:–INPUT FILE B EXCEPTION CONDITION 04	A parity failure has occurred on the B input file.	Abandon the run.
HALTED:–OUTPUT FILE EXCEPTION CONDITION 04	A parity failure has occurred on the output file.	Abandon the run.

PART 3 SWAP

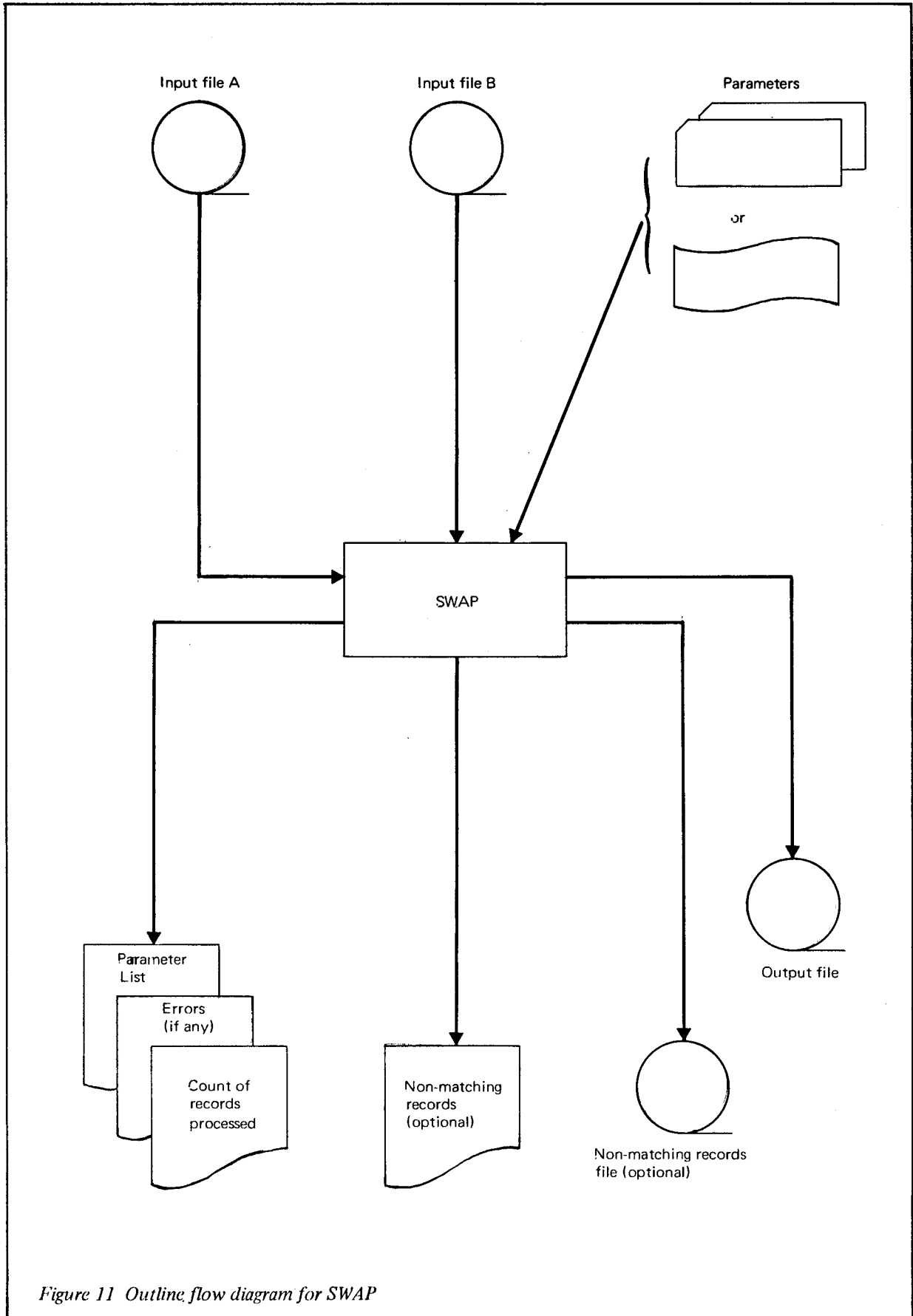


Figure 11 Outline flow diagram for SWAP

Chapter 11 Introduction to SWAP

SWAP is designed to alter a data file record by cross-reference to another file. The alteration consists of replacing a field on the data file (henceforth referred to as File A) by a field extracted from the other file (File B).

SWAP is most usefully employed where File B is an index file, i.e. a file containing elements of data common to several data files. One run of SWAP will allow up to twenty data fields to be taken from the index file and transferred to the data file. This concept is further elaborated on page 8 below and in Chapter 14 which consists of a comprehensive example of SWAP.

INPUT FILE FORMAT

Both input files must be sorted to common keys, but apart from this there is no implied relationship between them. The only restriction is that there must be *no more than* one File B record to each File A record, i.e. on File B there should not be more than one record with the same key.

However:

- 1 There may be more than one File A record to each File B record.
- 2 There need be no File A record for a File B record.
- 3 There need be no File B record for a File A record.

PARAMETER PROCESSING

At the start of the run parameters are input as follows:

#READ parameters for input Files A and B.

A #WRITE parameter for the output file.

A #KEYS parameter giving the address of the keys on the two input files.

A #MODE parameter specifying the mode of operation of the SWAP program. The effect of the various modes is described in Chapter 12 page 49.

#MOVE parameters specifying the address of fields in File B which are to be transferred and the address in File A to which transfers are to be made. The program makes use of these parameters to form a table to control the modification of File A.

A full description of all SWAP parameters will be found in Chapter 13. The program checks all parameters for validity and prints them out.

MAIN PROCESSING

The program reads each record from File A in key sequence and tests for a match with records on File B.

If there is a match, the program refers to the tables set up by the #MOVE parameters and uses these as a basis for arranging the transfer of fields from File B to File A. The altered File A records are then output to tape.

If there is no corresponding File B record for a File A record, the user has the option of outputting the unmatched record in various ways:

- 1 To the output tape, just as for matched records
- 2 To the line printer but not to the output tape
- 3 To the line printer and the output tape

4 To a special output tape, created for this purpose and labelled NON-MATCHING

During the run the program maintains a check on the sequence of input records and displays an error message on finding a record out of sequence (see page 62).

At the end of the run the program prints out a count of the number of records read from File A.

Figure 11 shows in flowchart form how the SWAP system operates.

USE OF SWAP

In a possible SWAP application, File A might consist of a file of data records referring to stock parts. The same stock part may be identified by alternative part numbers on different records. For example, three numbering systems may be in use and the same part may be numbered as AAB, as 001, or as AB2, on the different records. As the part number is used as the record key, records such as these would appear in other parts of the file.

For this data file to be used for amendment purposes, it is necessary to standardise these part numbers; this can be done using SWAP.

File A records are first sorted by part number without regard for the fact that various numbering systems are in use.

The second input file (File B) is an index file containing one record for each possible part number use by File A. These records are sorted to the same sequence as those on File A. However each File B record also contains in another field a Master Part Number which is designed to replace all alternative numbers referring to the same stock part.

Figure 3 on page 58 shows the format of the records on the two files.

Parameters to the SWAP program tell the program to transfer the Master Part Number from File B to the key field of output File A, whenever a match occurs.

If, for example it is required to replace AAB, 001 and AB2 (which all refer to the same part) by Master Part Number XXX, the following procedures will take place.

When SWAP reads in a File A record with the key 001 it will match it with the File B record with the key 001. The program will then transfer the Master Part Number XXX (which it finds on the File B record 001) to the key field of the output File A record, overwriting the old part number 001.

The same process will be applied to File A records with keys AAB and AB2, which will all be output with a key XXX.

At the end of the run all part numbers on output File A records will have been standardised, although the file will need to be sorted into sequence a second time because of the changes made to the key field.

MINIMUM CONFIGURATION

The minimum configuration for the SWAP program is:

1 1900 Series central processor with 16K core store.

1 card reader or paper tape reader.

1 line printer.

3 tape decks.

Chapter 12 Input and output

INPUT

Input to SWAP consists of two input files, and input parameters.

Input files

As described in Chapter 11 the two input files, File A and File B are both sorted to the same sequence.

Parameters

The parameters are described in detail in Chapter 13.

OUTPUT

Output from SWAP consists of an output file, an optional non-matching records file, line printer output and console output.

Magnetic tape output

All File A records which have been matched with File B records and processed by SWAP are output to a magnetic tape file as specified by the #WRITE parameter.

If no #MODE parameter is input, or if mode 0 is requested, unmatched File A records will also be output to this tape.

If mode 1 is requested, unmatched File A records will be output to the line printer, but not to magnetic tape.

If mode 2 is requested the unmatched records will be output to this tape, as well as to the line printer (see below).

If mode 3 is requested the program will open a second output tape exclusively for non-matching records. This tape will be labelled NON-MATCHING, will have a reel sequence number of 0, a file generation number of 0, and a retention period of 7 days.

Line printer output

- 1 The first printed output is the heading line, consisting of the page number, program title and date.
- 2 There follows a list of all parameters. If any errors have been found, the program will halt when it has read all the parameters and an appropriate message will be printed and also displayed by the console typewriter (see below).
- 3 If mode 1 or 2 is requested non-matching records are output as a continuous string of characters.
- 4 At the end of the run the printer outputs a count of File A records read and a count of all matching records (i.e. in effect those records on File A which have been amended by the swapping process).

PARAMETER ERRORS

These are indicated on the parameter listing as follows:

ERROR is printed out against the listing of any parameter found to be incorrect.

If an **ERROR** message has appeared against any parameter in the listing the message at the end of the listing is:

PARAMETERS INCORRECT

In addition, the printer will output:

PARAMETERS INCOMPLETE

if less than the minimum parameter set has been input. The minimum set is:

two #READ parameters
one #WRITE parameter
one #KEYS parameter
one #MOVE parameter
one #END parameter

Console output

Messages may be output to the console typewriter at run time. For details of these, see Chapter 15, 'Operating Instructions'.



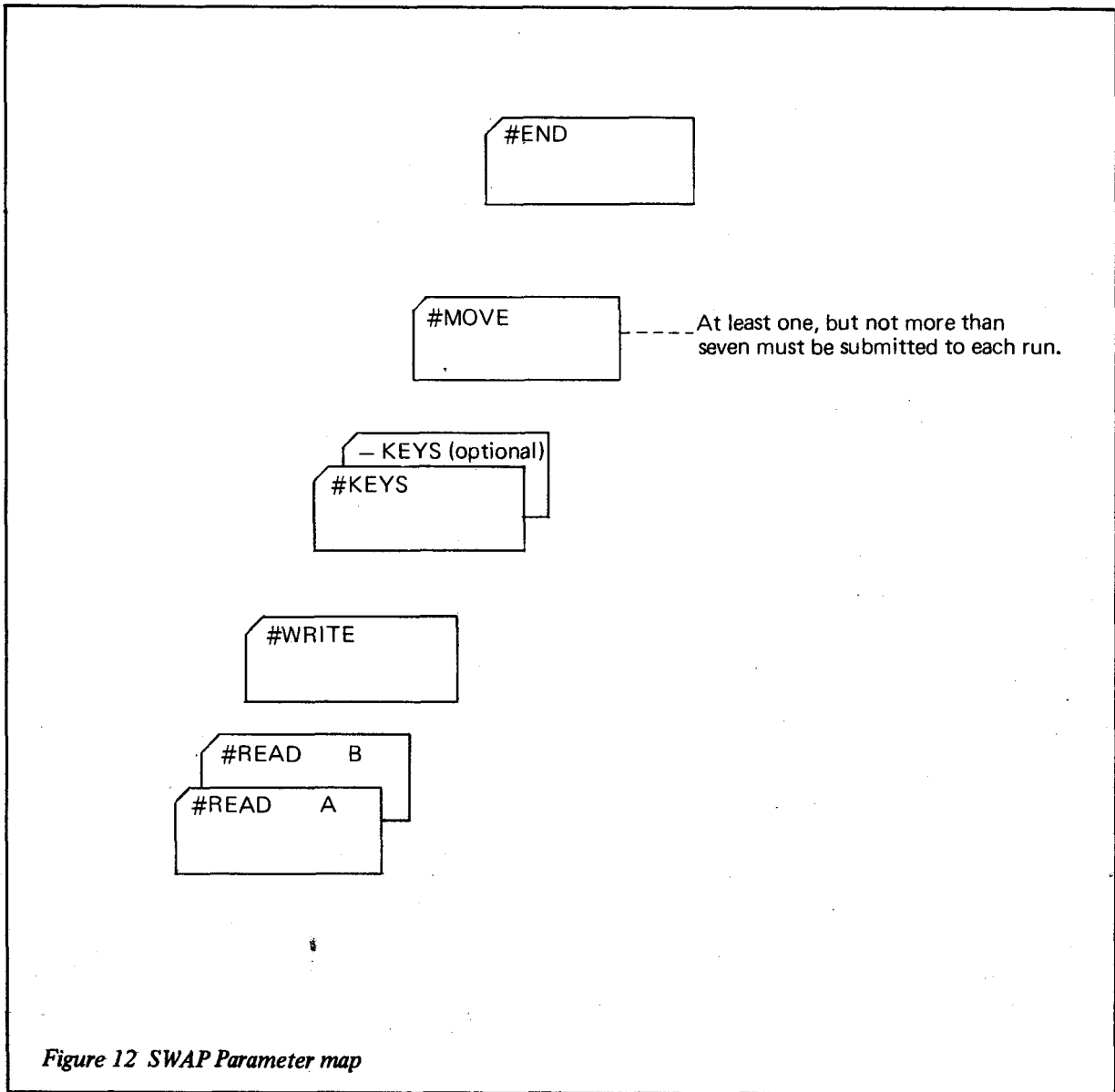


Figure 12 SWAP Parameter map

Chapter 13 The parameter set

SWAP uses the standard #READ, #WRITE and #END parameters and also the double-file form of the standard #KEYS parameter. The form of these (as used by SWAP) is given below, together with a description of the standard #MODE parameter which specifies the mode of operation and the standard #MOVE parameter which is used to control the modification of File A.

Parameters may be presented in any order, terminated by #END.

READ TAPE LABEL PARAMETER (#READ)

This parameter is in standard form. Its function is to define the labels on the input tape files, and the modes in which they are to be opened.

Field	Columns	Contents
A	1 to 5	The directive #READ
	6	Opening mode: 1 character as follows: 1 = no checks for presence or absence of write permit ring. 2 = check for absence of write permit ring. 3 = check for presence of write permit ring. ∇ (space) in this position is interpreted as 2.
		Note: These are equivalent to the standard modes of opening tape, that is #100, #200 and #300. Details of additional checks by Executive on the 1904 and above will be found in the ICL manual <i>Magnetic Tape</i> .
B	8 to 19	File name: 12 characters
C	21 to 24	Reel sequence number: 4 digits.
D	26 to 29	File generation number: 4 digits.
E	31	Designation: A or B.

Example

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			
#	R	E	A	D						D	E	T	A	I	L	F	I	L	E																							
#	R	E	A	D																																						

WRITE TAPE LABEL PARAMETER (#WRITE)

This parameter is in standard form. Its function is to define the labels to be written to output files. If fields G, H, and I are punched the program will search for a tape with the specified label and open it for output; if these fields are blank, the program will open any available scratch tape.

Field	Columns	Contents
A	1 to 6	The directive #WRITE

Field	Columns	Contents
B	7 to 11	12 character file name
C	21 to 24	Reel sequence number: 4 digits.
D	26 to 29	File generation number: 4 digits
E	31 to 34	Retention period: 4 digits
F	36	Designation: 1 alphabetic character
G	38 to 49	Old file name: 12 characters. The existing file name to be used for output.
H	51 to 54	Old reel sequence number: 4 digits.
I	56 to 59	Old file generation number: 4 digits.

Example

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			
#	W	R	I	T	E		D	E	T	A	I	L	F	I	L	E					0							3	8						2	8		
8	9	40	1	2	3	4	5	6	7	8	9	50	1	2	3	4	5	6	7	8	9	60	1	2	3	4	5	6	7	8	9	70	1	2	3			
P	R	O	G	R	A	M		T	A	P	E					0						0																

DOUBLE FILE KEYS PARAMETER (#KEYS)

This parameter is in standard form for double file input. It allows the user to specify the keys which are to be used to match records from the two input files. Up to six keys may be specified; the first four of these are defined in the #KEYS parameter, the key with highest significance being defined first.

Field	Columns	Contents
A	1 to 5	The directive #KEYS.
B	7 to 11	The start address of a key in File A expressed in words and characters relative to the start of the record, in the form <i>NNN.N</i> .
C	13 to 17	The start address of a key in File B expressed in words and characters relative to the start of the record, in the form <i>NNN.N</i> .
D	19 and 20	Field size: a 2 digit integer giving the length of the key in characters for character keys or words for binary keys.
E	21	A single character indicating the type of field: H = characters ascending % = binary ascending D = characters descending * = binary descending
F,G,H,I	23 to 37	Similar to fields B to E, defining the second key.
J,K,L,M	39 to 53	Similar to fields B to E, defining the third key.
N,O,P,Q	55 to 69	Similar to fields B to E, defining the fourth key.

Example

1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1	2	3	4	5	6	7	8	9	40		
#	K	E	Y	S	-	-	-	-	1	.	0	-	-	-	1	.	0	-	2	4	H	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Note that the key addresses for File A and File B are given in pairs, File A preceding. The most significant key on each file is always specified first.

If more than four keys need to be specified, a continuation card can be punched; this takes the following form:

Field	Columns	Contents
A	1 to 5	The directive -KEYS.
B	7 to 11	The start address of a key in File A expressed in words and characters relative to the start of the record, in the form <i>NNN.N</i> .
C	13 to 17	The start address of a key in File B expressed in words and characters relative to the start of the record, in the form <i>NNN.N</i> .
D	19 and 20	Field size: a 2 digit integer giving the length of the key in characters for character keys or words for binary keys.
E	21	A single character indicating the type of field: H = characters ascending % = binary ascending D = characters descending * = binary descending
F,G,H,I	23 to 37	Similar to fields B to E, defining the sixth key.

Example

1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1	2	3	4	5	6	7	8	9		
#	K	E	Y	S	-	-	-	-	1	.	0	-	-	-	1	.	0	-	1	2	H	-	-	-	4	.	0	-	-	7	.	2	-	-	6	H	-	-	-	-
-	K	E	Y	S	-	-	-	-	8	.	0	-	-	-	5	.	0	-	8	H	-	-	-	9	.	0	-	-	7	.	0	-	-	2	H	-	-	-	-	-

1	2	3	4	5	6	7	8	9	50	1	2	3	4	5	6	7	8	9	60	1	2	3	4	5	6	7	8	9	70	1	2	3	4	5	6	7	8	9	80	
5	.	2	-	-	-	-	-	9	.	0	-	-	-	2	H	-	-	-	6	.	0	-	-	-	4	.	0	-	-	2	%	-	-	-	-	-	-	-	-	-

MODE PARAMETER (#MODE)

This parameter specifies the mode of operation of the SWAP program; the effect of the various modes is defined on page 49. The absence of a mode parameter is taken to mean mode 0.

Field	Columns	Contents
A	1 to 5	The directive #MODE
B	7	The number of the mode to be used, either 0, 1, 2 or 3.

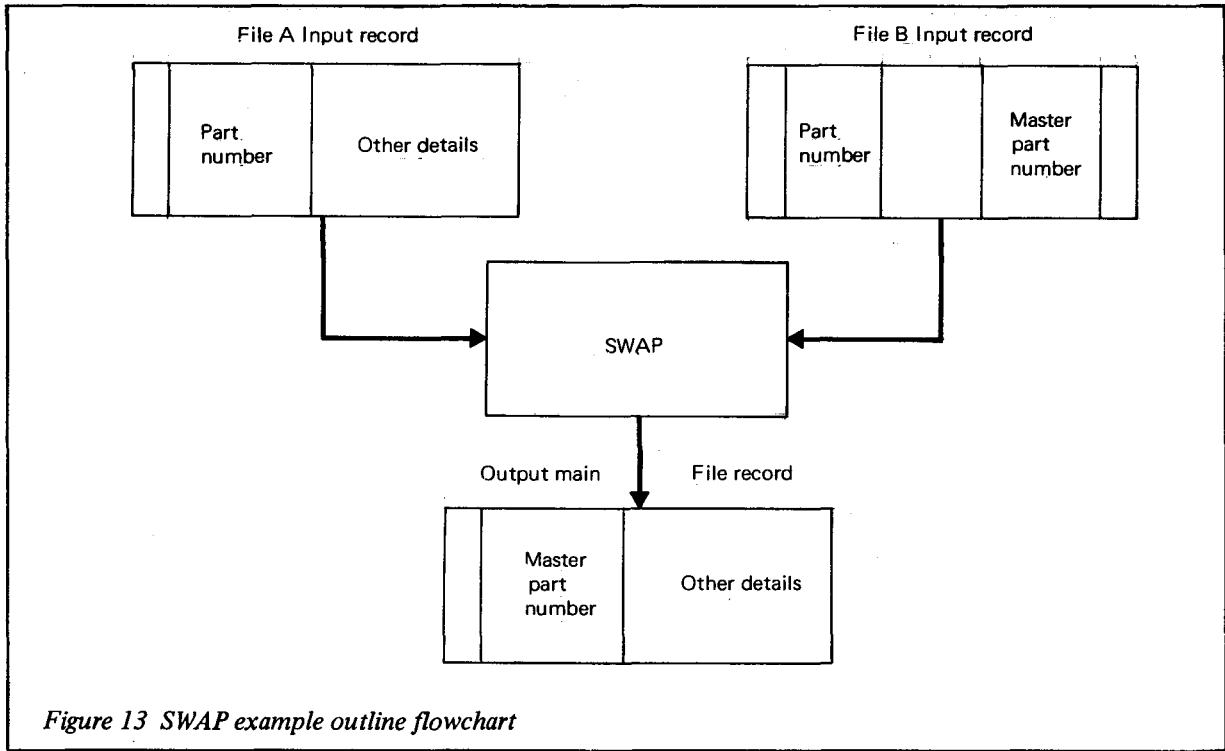


Figure 13 SWAP example outline flowchart

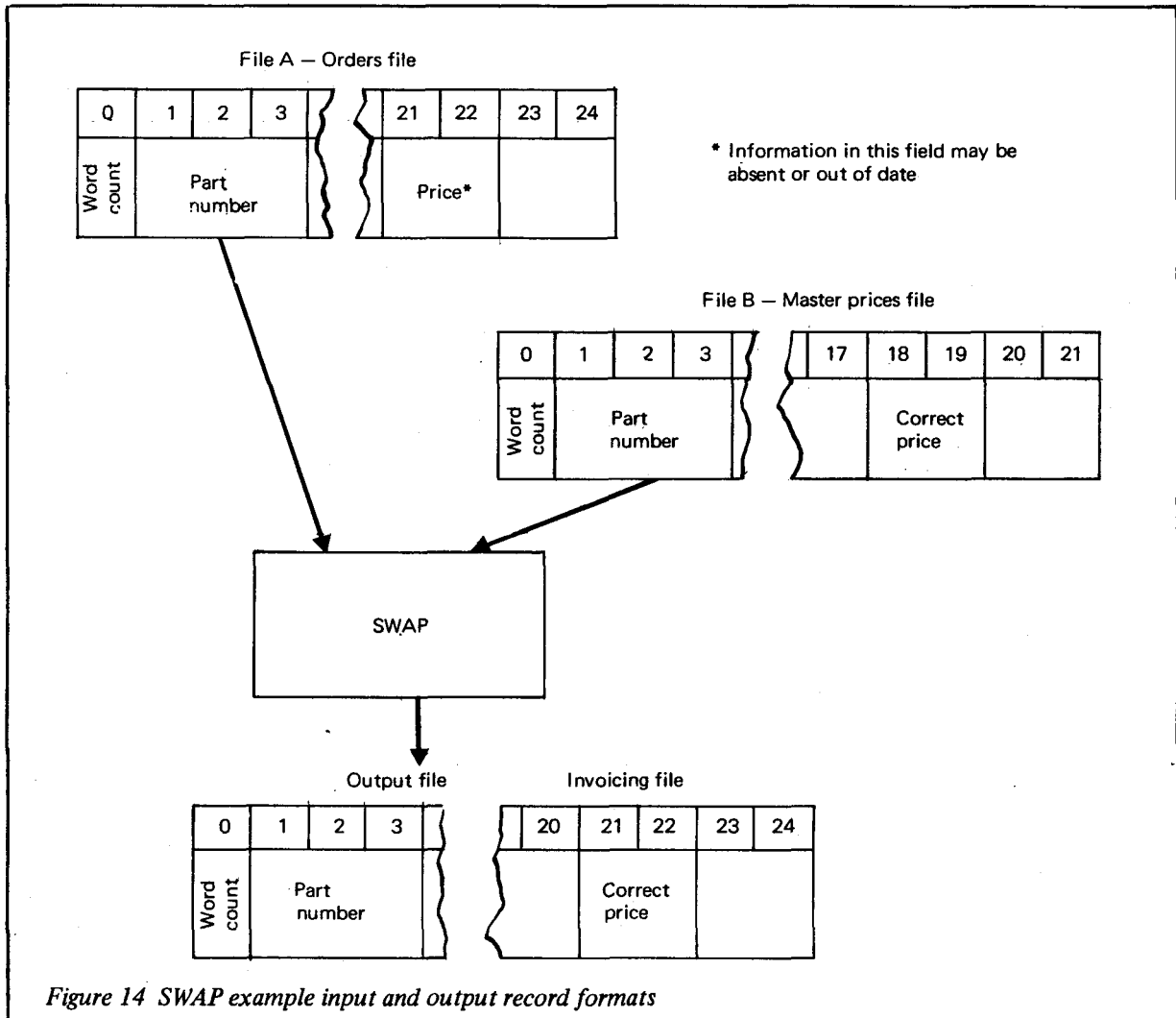


Figure 14 SWAP example input and output record formats

Chapter 14 Example

In Chapter 11 an example was given of the way in which SWAP can be used to transform file keys. In this example SWAP is used for updating by indexing that is, replacing old information on a file by information taken from a master reference file.

The master reference file in this case contains records relating to all items manufactured by a certain company; one field on each record contains the current price of each item. This file is File B.

File A contains records of all outstanding orders for items manufactured. There is also a price field on records in this file, and it is required to overwrite this field with the price field from the master file. This will ensure that all prices are brought up to date. Note that there will not necessarily be a File A record for every File B record (since there may be some items for which there are no outstanding orders); on the other hand there may well be several File A records corresponding to one File B record (since for a particular item there may be more than one outstanding order).

Both files contain a field giving the part number of the item; the files are sorted, using this field as the key, before being input to SWAP.

The diagram (Figure 14 on page 58) shows one record from each file; only the fields relevant to this description are shown.

Note:

- 1 On File A the part number field occupies Words 1 to 3 of each record and the price field occupies Words 21 and 22.
- 2 On File B the part number field also occupies Words 1 to 3 of each record and the price field occupies Words 18 and 19.

Figure 15 shows a listing of the parameters input for this run. File A has the File Name STOCK_vORDERS and File B has the File Name STOCK_vMASTER; the output file has the name STOCK_vORDERS and a generation number of 1 to distinguish it from the input file of which it is a revised version.

```
04/11/68 SWAP PARAMETER LIST

#READ2 STOCK ORDERS 0000 0000 A
#READ2 STOCK MASTER 0000 0000 B
#WRITE STOCK ORDERS 0000 0001 0060 A
#KEYS 001.0 001.0 12H
#MOVE B 018.0 021.0 08H.
#END

04/11/68 SWAP RECORD COUNTS

FILE A RECORDS READ 686
FILE B RECORDS READ 257
MATCHING RECORDS 645
```

Figure 15 SWAP example printout

Chapter 15 Operating instructions and exception conditions

OPERATING INSTRUCTIONS

<i>Narrative</i>	<i>Console Message</i>
1 Load the File A input tape without write permit ring.	
2 Load the File B input tape without write permit ring.	
3 Load output tape with a write permit ring.	
4 Load parameters in the appropriate reader.	
5 Load the program SWAP by inputting:	FI #X686 #TAPE
6 To read in the parameters either (a) if the parameters are on paper tape input:	GO #X686 20
or	
(b) if the parameters are on cards input:	GO #X686 21
7 When the parameters have been read, the reader and line printer will be released. When the input tapes have been read, and the output tape written, the program will allot a line printer to print out record counts, close all tapes, and release the printer. The run ends with the message:	HALTED:- END OF RUN
8 If the record counts at the end of the run are required to be output to the console instead of to the line printer, the following message should be input during the run:	ON #X686 0
9 When it is required to terminate a partly-completed run, input:	GO #X686 26
the program closes all tapes, allots printer, points record counts, releases printer and outputs the message:	HALTED:- RUN ABANDONED
following any tape error, the program closes all tapes, allots printer, prints details of tape error type and position, prints record counts, releases printer and closes with the relevant exception condition message, see below.	

EXCEPTION CONDITIONS

<i>Message</i>	<i>Reason</i>	<i>Action</i>
HALTED:- CR	No card reader is available.	Make a card reader available and GO #X686
HALTED:- TR	No paper tape reader is available.	Make a paper tape reader available and GO #X686
HALTED:- LP	No line printer is available.	Make a line printer available and GO #X686
HALTED:- PARAMETERS INCORRECT	Incorrect parameters.	Restart with correct parameters or abandon run.
HALTED:- PARAMETERS INCOMPLETE	Incomplete parameters.	Restart with complete set or abandon the run.

<i>Message</i>	<i>Reason</i>	<i>Action</i>
HALTED:- SEQUENCE ERROR FILE A	A record on the A input file has been found with keys out of sequence. The record counts will be printed, as at end of the run, to allow the position of the faulty record to be determined.	Abandon the run.
HALTED:- SEQUENCE ERROR FILE B	A record on the B input file has been found with keys out of sequence. The record counts will be printed, as at end of run, to allow the position of the faulty record to be determined.	Abandon the run.
HALTED:- INPUT FILE A EXCEPTION CONDITION 01	A long block has been found on the A input file.	Abandon the run.
HALTED:- INPUT FILE A EXCEPTION CONDITION 04	A parity failure has occurred on the A input file.	Abandon the run.
HALTED:- INPUT FILE B EXCEPTION CONDITION 01	A long block has been found on the B input file.	Abandon the run.
HALTED:- INPUT FILE B EXCEPTION CONDITION 04	A parity failure has occurred on the B input file.	Abandon the run.
HALTED:- OUTPUT FILE EXCEPTION CONDITION 04	A parity failure has occurred on the output file.	Abandon the run.
HALTED:- SHORT RECORD IN FILE A	A record has been read which is too short for one of the specified transfers.	Abandon the run.
HALTED:- SHORT RECORD IN FILE B	As above, for File B.	As above.

PART 4 COPY

Chapter 16 Introduction to COPY

COPY is a program designed to copy the contents of a file held on magnetic tape on to another tape, without altering the contents of the original except for the file name. The input and the output file may have continuation reels.

PROCESSING SUMMARY

The input and output tapes are loaded. The program COPY is loaded, followed by the three standard parameters, #READ, #WRITE and #END. The input tape is read, and written to the output tape. The program ends by releasing both tapes and displaying a count of records read and records written. An outline flow diagram is given in Figure 16.

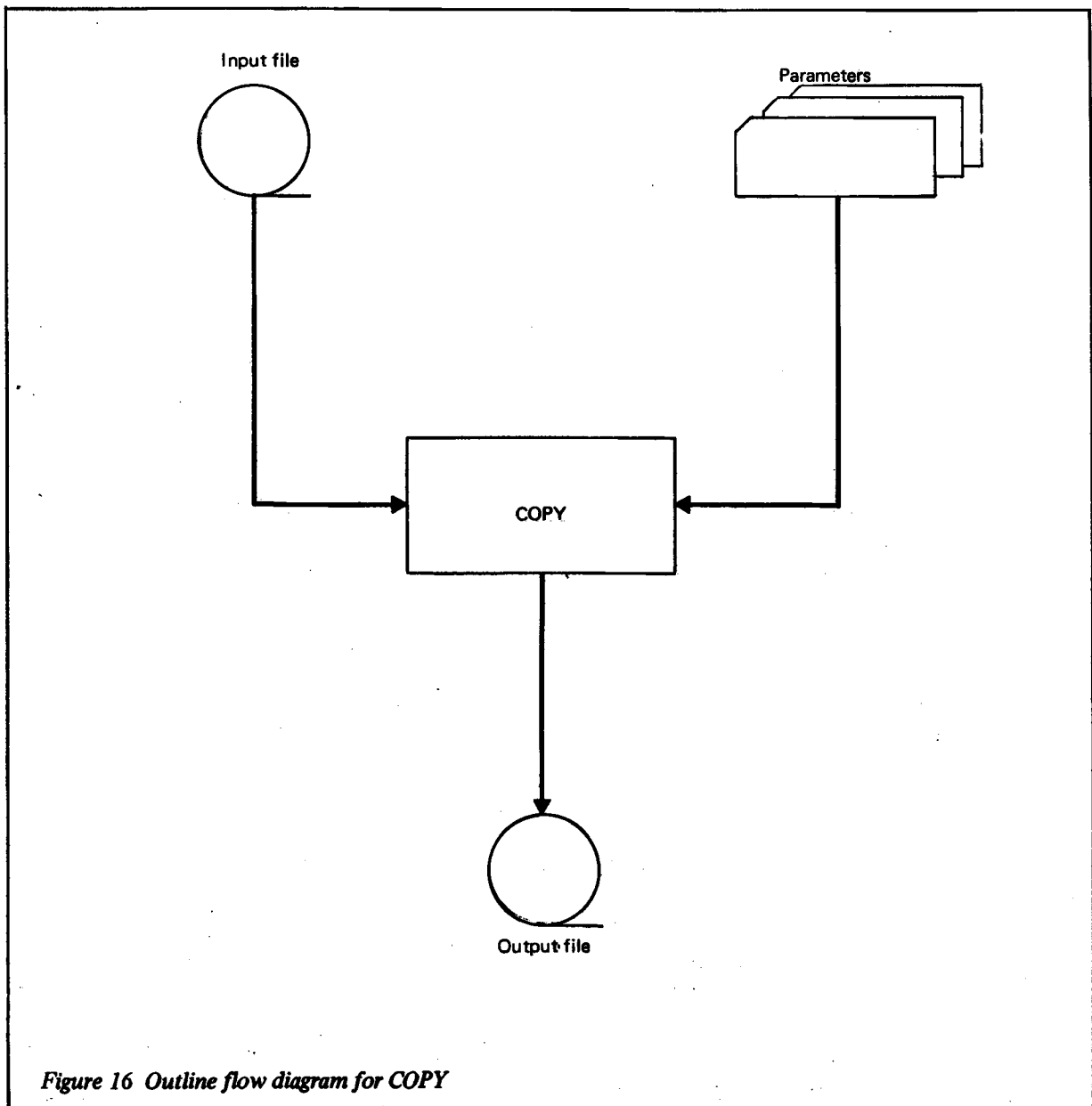


Figure 16 Outline flow diagram for COPY

Other magnetic tape copying programs already exist in the 1900 Series Library; COPY does not provide any extra facilities. However, the user who is familiar with other programs in this series of Data Management Software, and who is used to defining tape labels by means of #READ and #WRITE parameters, may wish to take advantage of a copying program which possesses this facility. This program, by permitting the use of a #WRITE parameter, allows the user to create a file which is a copy of the data on an existing file, but which has a different file identification.

MINIMUM CONFIGURATION

The minimum configuration for COPY is:

1 1900 Series central processor with 8K core store.

1 card reader or paper tape reader.

2 magnetic tape decks.

Chapter 17 Input and output

INPUT

Input to COPY consists of one input magnetic tape file and three parameters.

Input file

This tape should be similar to the other tapes used in Data Management Software programs, i.e. it should conform to MTH standards. Continuation reels may be used.

Parameters

Three parameters must be input: one #READ, one #WRITE and one #END, each in the standard format.

OUTPUT

The output from COPY consists of one magnetic tape file, line printer output and console output.

Output file

The data from the input tape is copied to the output tape without change.

Line printer output

The line printer output consists of a listing of the parameters together with counts of records read and written. The counts should be identical.

Console output

The operator can arrange for the counts of records to be output to the console instead of to the line printer by setting a switch, see Chapter 19.

Other messages output to the console are described in detail in Chapter 19.

```
DATE OF RUN:- 10/09/69          COPY-PARAMETER LIST
#READ1 SCHOOL FILE 0000 0000
#WRITE TEST FILECRA 0000 0000 4095
#END

      10/09/69  COPY RECORD COUNTS
COUNT OF OUTPUT RECORDS WRITTEN  203
COUNT OF INPUT RECORDS READ      203
```

Figure 17 Example of printout from COPY

Chapter 18 The parameter set

Three parameters must be input to COPY, one #READ, one #WRITE and one #END.

READ TAPE LABEL PARAMETER (#READ)

This parameter is in standard form. Its function is to define the label on the input tape file, and the mode in which it is to be opened.

Field	Columns	Contents
A	1 to 5	The directive #READ
	6	Opening mode: 1 character as follows: 1 = no checks for presence or absence of write permit ring. 2 = check for absence of write permit ring 3 = check for presence of write permit ring. ∇ (space) in this position is interpreted as 2. Note: These are equivalent to the standard modes of opening tapes, that is #100, #200 and #300. Details of additional checks by Executive on the 1904 and above will be found in the ICL manual <i>Magnetic Tape</i> .
B	8 to 19	12 character file name.
C	21 to 24	Reel sequence number: 4 digits.
D	26 to 29	File generation number: 4 digits.
E	31	Designation: not used in this program.

Example

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			
#	R	E	A	D																																						

WRITE TAPE LABEL PARAMETER (#WRITE)

This parameter is in standard form. Its function is to define the label to be written to the output file. If fields G, H and I are punched, the program will search for a tape with the specified label and open it for output, relabelling it as specified in fields B, C, D, E. If fields G, H and I are blank, the program will open any available scratch tape.

Field	Columns	Contents
A	1 to 6	The directive #WRITE
B	8 to 19	12 character file name.
C	21 to 24	Reel sequence number: 4 digits.
D	26 to 29	File generation number: 4 digits.
E	31 to 34	Retention period: 4 digits.

Field	Columns	Contents
F	36	Designation: not used in this program.
G	38 to 49	Old file name: 12 characters. The existing name on the tape to be used for output.
H	51 to 54	Old reel sequence number: 4 digits.
I	56 to 59	Old file generation number: 4 digits..

Example

1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1	2	3	4	5	6				
#	W	R	I	T	E	_	C	O	P	Y	O	F	F	I	L	E	_	_	_	_	_	_	_	_	_	0	_	_	_	3	3	_	_	_	4	0	_	_	_
8	9	40	1	2	3	4	5	6	7	8	9	50	1	2	3	4	5	6	7	8	9	60	1	2	3	4	5	6	7	8	9	70	1	2	3				
F	I	L	E	T	O	B	E	U	S	E	D	_	_	_	0	_	_	_	0	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_				

END OF PARAMETERS MARKER

This parameter is in standard form. It marks the end of the parameter set.

Field	Columns	Contents
A	1 to 4	The directive #END

Example

1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1	2	3	4	5	6	7	8	9	40
#	E	N	D	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	

Chapter 19 Operating instructions and exception conditions

OPERATING INSTRUCTIONS

Narrative

Console Message

- | | | |
|---|--|---|
| 1 | Load the input tape, without a write permit ring. | |
| 2 | Load the output tape, with a write permit ring. | |
| 3 | Load the program COPY by inputting: | FI #X687 #TAPE |
| 4 | Load the parameters in the appropriate reader. | |
| 5 | To read in the parameters either | |
| | (a) if the parameters are on paper tape input: | GO #X687 20 |
| | or | |
| | (b) if the parameters are on cards input: | GO #X687 21 |
| 6 | When the parameters have been read, the reader and line printer will be released. When the input tape has been read, and the output tape written the program will allot a line printer to print out record counts, close all tapes and release the printer. The run ends with the message: | HALTED:- END OF RUN |
| 7 | If the record counts at the end of the run are required to be output to the console instead of the line printer, the following message should be input during the run: | ON #X687 0 |
| 8 | When it is required to terminate a partly completed run, input:
the program closes all tapes, allots printer, prints record counts, releases printer and outputs the message:
following any tape error the program closes all tapes, allots printer, prints details of tape error type and position, prints record counts, releases printer and closes with the relevant exception condition message, see below. | GO #X687 26

HALTED:- RUN ABANDONED |

EXCEPTION CONDITIONS

The following messages may appear on the console typewriter.

<i>Message</i>	<i>Reason</i>	<i>Action</i>
HALTED:- PARAMETERS INCORRECT	Incorrect parameters.	Restart with correct parameters or abandon the run.
DISPLAY:- CONTINUATION REEL OPENED, OUTPUT SO FAR <i>nnnn</i>	Continuation reel opened.	None.
HALTED:- INPUT FILE EXCEPTION CONDITION 01	A long block has been found on the input file.	Abandon the run.
HALTED:- INPUT FILE EXCEPTION CONDITION 04	A parity failure has occurred on the input file.	Abandon the run.
HALTED:- OUTPUT FILE EXCEPTION CONDITION 04	A parity failure has occurred on the output file.	Abandon the run.

Message

HALTED:- LINE PRINTER
NOT AVAILABLE FOR
TOTALS

Reason

Record counts are ready to be
output.

Action

Either

a) make line printer available and GO
#X687

or

b) if record counts are to be output to
console instead,
ON #X687 0
GO #X687

PART 5 CRAM

Chapter 20 Introduction to CRAM

CRAM is a program designed to write to a single output file a variable number of magnetic tape input files, up to seven in all. Any or all of the input files or the output file may have continuation reels. The program does not reorganise the records on the output file into key sequence.

PROCESSING SUMMARY

The program reads all the parameters, i.e. a single #WRITE card for the output file, a #READ card for each input file and a #END, end of parameters marker. It then reads the input tapes one by one, and writes the output tape. The program ends with a display of the counts of records read and written. An outline flow diagram is given in Figure 18.

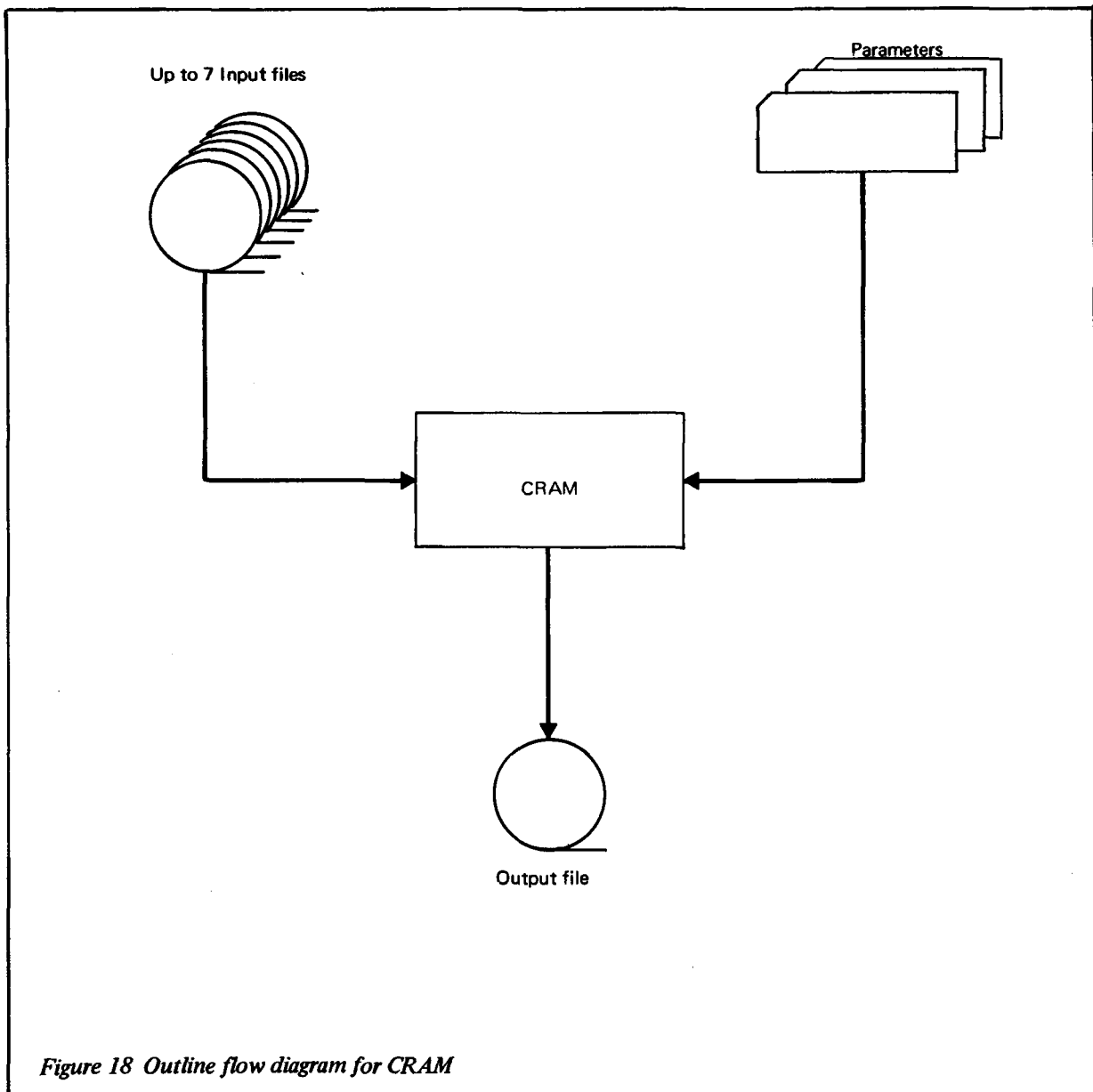


Figure 18 Outline flow diagram for CRAM

Other magnetic tape merging programs already exist in the 1900 Series Library; CRAM does not provide any extra facilities. However, the user who is familiar with other programs in this series of Data Management Software, and who is used to defining tape labels by means of #READ and #WRITE parameters, may wish to take advantage of a merging program which possesses this facility.

MINIMUM CONFIGURATION

The minimum configuration for CRAM is:

1 1900 Series central processor with 8K core store.

1 card reader or paper tape reader and at least

2 magnetic tape decks

Chapter 21 Input and output

INPUT

Input to CRAM consists of up to seven input magnetic tape files and three types of parameters.

Input files

There are no restrictions on the input tapes, except that they should conform to magnetic tape housekeeping standards. It should be noted that if CRAM is unable to distinguish between any of the input files (i.e. they have the same file name, and file generation number) it will choose first the tape on the deck with the lower number, and write that tape before the other identical tape or tapes.

Parameters

These consist of one #READ parameter for each input file, a #WRITE parameter for the output file and a #END, end of parameters marker.

OUTPUT

Output from CRAM consists of one magnetic tape file, line printer output and console output.

Output file

This tape has the header label specified in the #WRITE parameter, and consists of all the input files in sequence, that is, all the records from the first file followed by all the records from the second file, and so on up to seven input files.

Note: the sequence is determined by the sequence in which the #READ parameters were read.

Line printer output

If the parameters are found to be invalid the program will print out one of the following messages:

INCORRECT PARAMETERS
TOO MANY READ PARAMETERS

A count of records read from each input file and records written to the output file are printed at the end of the run.

Console output

The operator can arrange for the counts of records to be output to the console instead of to the line printer by setting a switch, see Chapter 23. Other messages output to the console are described in detail in Chapter 23.

```
DATE OF RUN:- 15/09/69          CRAM-PARAMETER LIST
#READ1 SCHOOL FILE 0000 0000
#READ1 SCHOOL FILE 0000 0000
#READ1 SCHOOL FILE 0000 0000
#WRITE CRAM FILECRA 0000 0000 4095
#END
15/09/69  CRAM RECORD COUNTS
COUNT OF OUTPUT RECORDS WRITTEN 609
COUNT OF INPUT RECORDS FILE 1 203
COUNT OF INPUT RECORDS FILE 2 203
COUNT OF INPUT RECORDS FILE 3 203
```

Figure 19 Example of printout from CRAM



Chapter 22 The parameter set

Three types of parameters must be input to CRAM, one #READ for each input file, one #WRITE and one #END, end of parameters marker.

READ TAPE LABEL PARAMETER (#READ)

This parameter is in standard form. Its function is to define the labels on the input tape files and the modes in which they are to be opened.

Field	Columns	Contents
A	1 to 5	The directive #READ
	6	Opening mode: 1 character as follows: 1 = no checks for presence or absence of write permit ring. 2 = check for absence of write permit ring. 3 = check for presence of write permit ring. ∇ (space) in this position is interpreted as 2. Note: These are equivalent to the standard modes of opening tapes, that is #100, #200 and #300. Details of additional checks by Executive on the 1904 and above will be found in the ICL manual <i>Magnetic Tape</i> .
B	8 to 19	12 character file name.
C	21 to 24	Reel sequence number: 4 digits.
D	26 to 29	File generation number: 4 digits.
E	31	Designation: not used in this program.

Example

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			
#	R	E	A	D			F	I	R	S	T	F	I	L	E								0						0													
#	R	E	A	D			S	E	C	O	N	D	F	I	L	E							0						1													
#	R	E	A	D			T	H	I	R	D	F	I	L	E								0						2													

WRITE TAPE LABEL PARAMETER (#WRITE)

This parameter is in standard form. Its function is to define the labels to be written to the output file. If fields G, H and I are punched, the program will search for a tape with the specified label and open it for output: if these fields are blank the program will open any available scratch tape.

Field	Columns	Contents
A	1 to 6	The directive #WRITE
B	8 to 19	12 character file name.

Chapter 23 Operating instructions and exception conditions

OPERATING INSTRUCTIONS

Narrative

Console Message

- | | | |
|---|---|------------------------|
| 1 | Load input tapes, without write permit rings. | |
| 2 | Load output tape, with write permit ring. | |
| 3 | To load the program CRAM input: | FI #X688 #TAPE |
| 4 | Load the parameters in the appropriate reader. | |
| 5 | To read in the parameters either | |
| | (a) if the parameters are on paper tape input: | GO #X688 20 |
| | or | |
| | (b) if the parameters are on cards input: | GO #X688 21 |
| 6 | When the parameters have been read, the reader and line printer will be released. When the input tape has been read, and the output tape written, the program will allot a line printer to print out record counts, close all tapes and release the printer. The run ends with the message: | HALTED:- END OF RUN |
| 7 | If the record counts at the end of the run are required to be output to the console instead of to the line printer, the following message should be input during the run: | ON #X688 0 |
| 8 | When it is required to terminate a partly completed run, input: | GO #X688 26 |
| | the program closes all tapes, allots printer, prints record counts, releases printer and outputs the message: | HALTED:- RUN ABANDONED |
| | following any tape error the program closes all tapes, allots printer, prints details of tape error type and position, prints record counts, releases printer and closes with the relevant exception condition message, see below. | |

EXCEPTION CONDITIONS

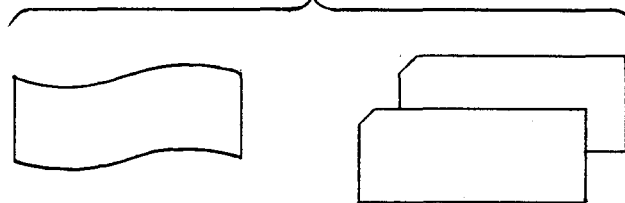
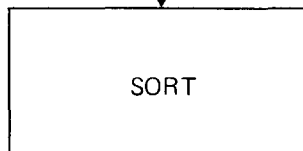
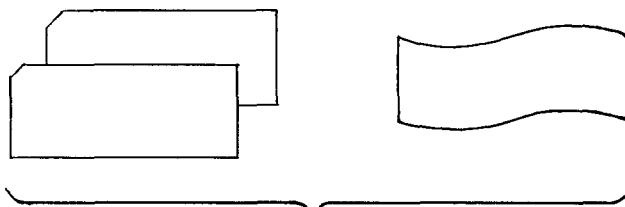
The following messages may appear on the console typewriter:

<i>Message</i>	<i>Reason</i>	<i>Action</i>
HALTED:- PARAMETERS INCORRECT	Incorrect parameters.	Restart with correct parameters or abandon the run.
HALTED:- TOO MANY READ PARAMETERS	Too many #READ parameters have been input.	As above.
DISPLAY:- CONTINUATION REEL OPENED, OUTPUT SO FAR <i>nnnn</i>	Continuation reel opened.	None.
HALTED:- INPUT FILE EXCEPTION CONDITION 01	A long block has been found on the input file.	Abandon the run.

<i>Message</i>	<i>Reason</i>	<i>Action</i>
HALTED:-INPUT FILE EXCEPTION CONDITION 04	A parity failure has occurred on an input file.	Abandon the run.
HALTED:- OUTPUT FILE EXCEPTION CONDITION 04	A parity failure has occurred on the output file.	Abandon the run.
HALTED:- LINE PRINTER NOT AVAILABLE FOR TOTALS	Record counts are ready to be output	Either a) make line printer available and GO #X688 or b) if record counts are to be output to console instead, ON #X688 0 GO #X688

PART 6 SORT

SORT PARAMETERS ON EITHER CARDS OR PAPER TAPE



PARAMETERS ON EITHER CARDS OR PAPER TAPE FOR
INPUT TO #XSMC, #XSMD or #XSME

Figure 20 Outline flow diagram for SORT

Chapter 24 Introduction to SORT

This program is designed for the user who is familiar with the parameters required by other Data Management Software programs. It enables such a user to make use of the standard 1900 Series Library sort programs, #XSMC, #XSMD and #XSME. The parameters required by these three programs are different in form from those parameters used by Data Management Software programs.

The user indicates to SORT which sort program he wishes to use and provides parameters in the standard form for Data Management Software. The program translates and edits the details from these parameters and produces on either cards or paper tape the parameters required for input to #XSMC, #XSMD or #XSME.

The operation of the sort programs with these parameters will be greatly simplified and this results in the following limitations:

- 1 The maximum block size of the input tape to be sorted is considered to be 512 words
- 2 The maximum record size for the input tape to be sorted is considered to be 512 words
- 3 The maximum block size for the tape output by the chosen sort program is considered to be 512 words
- 4 The chosen sort program will use four work-tapes
- 5 Neither #XSMC nor #XSMD will check the generation number of the input tape
- 6 Neither #XSMC nor #XSMD will set the output tape generation number from that specified in the #WRITE parameter
- 7 Neither #XSMC nor #XSMD will set the output tape retention period from that specified in the #WRITE parameter
- 8 The maximum number of keys is 6, although #XSME can normally cope with a maximum of 10 keys
- 9 The input tape should not be fitted with a write permit ring
- 10 Floating point keys may not be defined
- 11 A consolidated version of #XSME may not be renamed nor may the program to be loaded after #XSME
- 12 An account number may not be specified with #XSME
- 13 Existing labels on tapes to be used for output may not be defined

If the user wishes to use any of the sort programs without these limitations he must prepare parameters for input directly to such a program, as described in the ICL manual *Magnetic Tape Sorting*.

PARAMETER PROCESSING

At the start of the run the program reads and checks a complete SORT parameter set. One or more parameter sets consisting of four parameters, may be input.

The first parameter, the program name parameter, gives to SORT the name of the sort program for which parameters are to be created. The standard #READ and #WRITE parameters identify the input and output tapes that are to be used by the chosen sort program not by SORT, itself. The #KEYS parameter defines the keys on which the sort program eventually used is to order the input file. Only one of each parameter should be provided in each set.

Each set of SORT parameters is printed on input. If the set is acceptable, it is processed and a list of the parameters generated is printed beneath the listing of the SORT parameters set.

If the set is not acceptable it is printed but not processed. Any further sets of parameters are then read, unless an end of parameters marker has already been input.

MAIN PROCESSING

The details of the input and output files for the chosen sort program are transferred to the appropriate output fields without change.

The program calculates the number of keys defined and sets the counts in the appropriate output field. Each key definition is edited into the required form for output and the result is moved to the appropriate output field.

When creating output intended for #XSME, the program generates a key sequence number and inserts this in the appropriate place in each output key definition. The program also calculates the minimum record length when creating output intended for #XSME, by finding the highest numbered word that is affected by the input key definitions.

When the parameters created by each SORT parameter set are complete, the program lists them on the same page as the appropriate SORT parameter set.

The newly created parameters are then punched on either cards or paper tape. When output is to paper tape, each set of parameters is followed by a suitable stretch of run-out.

MINIMUM CONFIGURATION

The minimum configuration for the SORT program comprises:

- 1 1900 Series central processor with 4K core store
- 1 card reader or paper tape reader
- 1 card punch or paper tape punch
- 1 line printer

Chapter 25 Input and output

INPUT

Input to SORT consists of parameters on either cards or paper tape.

Parameters

The parameters are described in Chapter 24.

OUTPUT

Output from SORT consists of cards or paper tape containing the parameters required for input to #XSMC, #XSMD, or #XSME, line printer output and console output.

Card and paper tape output

The parameters give details of any checks to be made on the generation number of the input tape to be used by the chosen sort program, together with a number of key definitions.

Line printer output

The input parameters to SORT are printed out in a list. Where more than one set of input parameters is provided, each set is printed on a separate page. A list of the parameters created for input to one of the sort programs is printed on the same page as the parameter set which created those parameters.

Console output

See under *Exception Conditions*, Chapter 27.

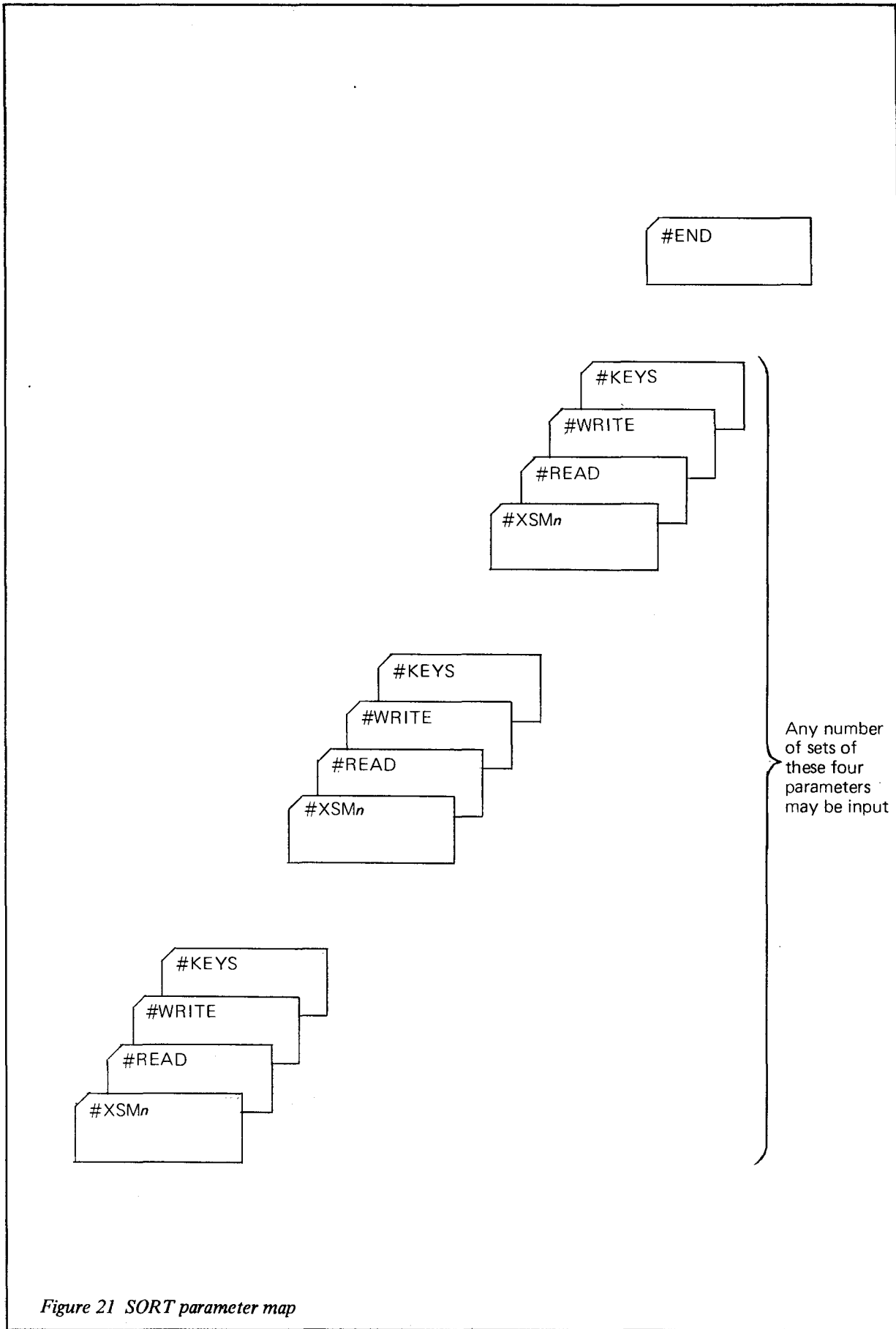


Figure 21 SORT parameter map

Chapter 26 The parameter set

At least five parameters must be input to the SORT program. They are (in the order in which they must be input):

- One program name parameter
- One #READ parameter
- One #WRITE parameter
- One #KEYS parameter (single-file form)
- One #END parameter

The first four constitute a set all of which must be present. Any number of such sets can be input and the last or only set must be terminated by an end of parameters marker.

PROGRAM NAME PARAMETER (#XSMC, #XSMD OR #XSME)

The function of this parameter is to give to SORT the name of the sort program for which parameters are to be created.

Field	Columns	Contents
A	1 to 5	The name of the program for which parameters are to be created. This may be either #XSMC, #XSMD or #XSME.
B	7 to 11	The name of the search program to be used in loading the chosen sort program. This field is optional; if it is left blank the program assumes that #TAPE will be used.

Example

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	
#	X	S	M	E		#	T	A	P	E																														
#	X	S	M	D		#	U	T	I	L																														

READ TAPE LABEL PARAMETER (#READ)

This parameter is in standard form. Its function is to define the label on the input tape file that is to be read and sorted by the chosen sort program, and the modes in which it is to be opened. This parameter does not define an input tape for SORT.

Field	Columns	Contents
A	1 to 5 6	The directive #READ Opening mode: 1 character as follows: 1 = no checks for presence or absence of write permit ring. 2 = check for absence of write permit ring. 3 = check for presence of write permit ring.

Field	Columns	Contents
A	1 to 5	The directive #KEYS
B	7 to 11	The start address of key, expressed in words and characters relative to the start of the tape record, in the form <i>NNN.N</i> .
C	13 to 14	Field size: a 2 digit integer giving the length of the key in characters for character keys and words for binary keys.
D	15	Key type: a single character indicating the type of field: H = characters ascending % = binary ascending D = characters descending * = binary descending
E	17 to 25	Similar to fields B, C and D defining the second key.
F	27 to 35	Similar to fields B, C and D defining the third key.
G	37 to 45	Similar to fields B, C and D defining the fourth key.
H	47 to 55	Similar to fields B, C and D defining the fifth key.
I	57 to 65	Similar to fields B, C and D defining the sixth key.
J	67	Designation: not required for this program.

Example

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		
#	K	E	Y	S					1	.	0		1	4	H				4	.	2			2	D				5	.	0			1	%						

END OF PARAMETERS MARKER

This parameter is in standard form. Its function is to indicate that the last parameter set has been read.

Field	Columns	Contents
A	1 to 4	The directive #END.

Example

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				
#	E	N	D																																								

```

06/08/69      #SORT - LIST OF PARAMETERS

1. INPUT PARAMETERS
#XSMC
#READ RANDOM DATA 0025 0169
#WRITE SORTED ADATA 0072 0444 0987
#KEYS 004.3 04H 006.1 09D 025.0 01% 020.0 02% 145.0 01* 124.0 02*

2. OUTPUT PARAMETERS
SORT      1RANDOM DATA0000250      SORTED ADATA0000725124
SORT      2RANDOM DATA000025600190040102500910025000200200001114500021124000

1. INPUT PARAMETERS
#XSME
#READ RANDOM DATA 0025 0169
#WRITE SORTED ADATA 0072 0444 0987
#KEYS 004.3 04H 006.1 09D 025.0 01% 020.0 02% 145.0 01* 124.0 02*

2. OUTPUT PARAMETERS
INPUT FILE (RANDOM DATA(0169/0025));
OUTPUT FILE (SORTED ADATA(0444/0072,0987));
FORMAT 146,512,512,512;
CONTROL 4;
KEY 6(0,0A,004.3,04;1,0D,006.1,09;2,1A,025;3,2A,020;4,1D,145;5,2D,124);
****
#END

```

Figure 22 SORT example printout: parameters for SORTED ADATA file

```

1. INPUT PARAMETERS
#XSME
#READ RANDOM DATA 0025 0169
#WRITE SORTED BDATA 0072 0444 0987
#KEYS 004.3 04H 006.1 09H 025.2 17H 020.0 25H 145.3 01H 124.3 66H

2. OUTPUT PARAMETERS
INPUT FILE (RANDOM DATA(0169/0025));
OUTPUT FILE (SORTED BDATA(0444/0072,0987));
FORMAT 146,512,512,512;
CONTROL 4;
KEY 6(0,0A,004.3,04;1,0A,006.1,09;2,0A,025.2,17;3,0A,020.0,25);
KEY 6(4,0A,145.3,01;5,0A,124.3,66);
****
#END

```

Figure 23 SORT example printout: parameters for SORTED BDATA file

Chapter 27 Example

In this example a file named *RANDOM DATA* is to be sorted. If the file is sorted on the set of keys specified by the parameters in Figure 22, a file named *SORTED ADATA* will be produced. Parameters to produce this file are provided for both #XSMC and #XSME.

The parameters for #XSME in Figure 23 however, will cause the *RANDOM DATA* file to be sorted on a different set of keys and a file named *SORTED BDATA* will be produced.

In figures 22 and 23 the input parameters have been input on cards and the output is also on cards. In Figure 24, the same input parameters on cards are illustrated together with output on paper tape.

```
06/08/69      #SORT - LIST OF PARAMETERS

1. INPUT PARAMETERS

#XSMC

#READ RANDOM DATA 0025 0169

#WRITE SORTED ADATA 0072 0444 0987

#KEYS 004.3 04H 006.1 09D 025.0 01% 020.0 02% 145.0 01* 124.0 02*

2. OUTPUT PARAMETERS
SORT†
†
RANDOM DATA†
0025†
0†
†
4†
SORTED ADATA†
0072†
512†
6†
0†
0†
019†
004†
0†
1†
025†
009†
1†
0†
025†
†
2†
0†
020†
†
1†
1†
145†
†
2†
1†
124†
†
```

Figure 24 SORT example printout: card input with paper tape output (continued overleaf)

```

1 INPUT PARAMETERS
#XSME
#READ RANDOM DATA 0025 0169
#WRITE SORTED ADATA 0072 0444 0987
#KEYS 004.3 04H 006.1 09D 025.0 01% 020.0 02% 145.0 01* 124.0 02*

2. OUTPUT PARAMETERS
INPUT FILE (RANDOM DATA(0169/0025));†
OUTPUT FILE (SORTED ADATA(0444/0072 0987));†
FORMAT 146,512,512,512;†
CONTROL 4;†
KEY 6(0,0A,004.3,04;1,0D,006.1,09;2,1A,025;3,2A,020;4,1D,145;5,2D,124);†
****†

1. INPUT PARAMETERS
#XSME
#READ RANDOM DATA 0025 0169
#WRITE SORTED BDATA 0072 0444 0987
#KEYS 004.3 04H 006.1 09H 025.2 17H 020.0 25H 145.3 01H 124.3 66H

2. OUTPUT PARAMETERS
INPUT FILE (RANDOM DATA(0169/0025));†
OUTPUT FILE (SORTED BDATA(0444/0072,0987));†
FORMAT 146,512,512,512;†
CONTROL 4;†
KEY 6(0,0A,004.3,04;1,0A,006.1,09;2,0A,025.2,17;3,0A,020.0,25);†
KEY 6(4,0A,145.3,01;5,0A,124.3,66);†
****†
#END

```

Figure 24 SORT example printout: card input with paper tape output (continued)

Chapter 28 Operating instructions and exception conditions

OPERATING INSTRUCTIONS

<i>Action</i>	<i>Message</i>
1 Load the parameters in the appropriate reader.	
2 Load the program SORT by inputting:	FI #X689 #TAPE
3 To read in the parameters either	
(a) if the parameters are on paper tape and the output will be on paper tape, input:	GO #X689 20
or	
(b) if the parameters are on cards and the output will be on cards, input:	GO #X689 21
or	
(c) if the parameters are on paper tape and output will be on cards, input:	GO #X689 22
or	
(d) if the parameters are on cards and output will be on paper tape, input:	GO #X689 23
4 The program reads and processes the parameters. At the end of the run the program halts with one of two messages: either	
(a) if only one parameter set has been input	DELETED:- FI #XSMx nnnn
or	
(b) if several parameter sets have been input	HALTED:- END OF RUN

EXCEPTION CONDITIONS

<i>Message</i>	<i>Reason</i>	<i>Action</i>
HALTED:- CR	No card reader is available.	Make a card reader available and GO #X689.
HALTED:- TR	No paper tape reader is available.	Make a paper tape reader available and GO #X689.
HALTED:- CP	No card punch is available.	Make a card punch available and GO #X689.
HALTED:- TP	No paper tape punch is available.	Make a paper tape punch available and GO #X689.
HALTED:- LP	No line printer is available.	Make a line printer available and GO #X689.

