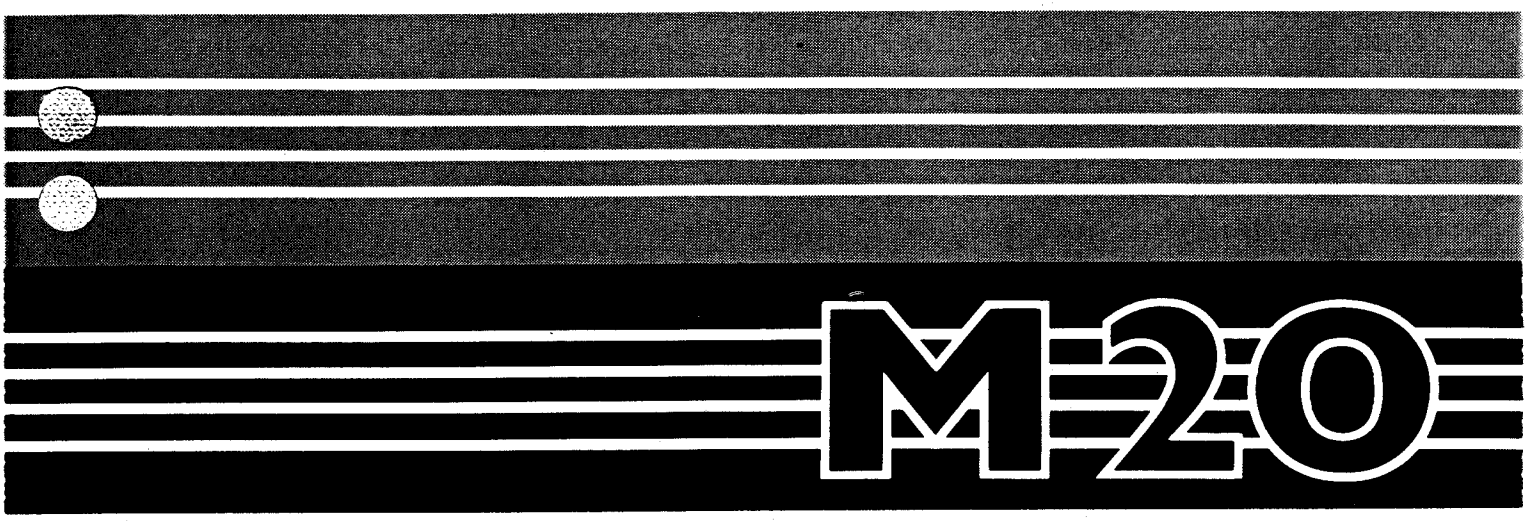
M20

olivetti L1



ISAM
(Index Sequential Access Method)

User Guide

olivetti L1

PREFACE

This book describes the features of ISAM (Index Sequential Access Method) and how to use them using BASIC programming language on the M20.

Chapter 1 explains in general the concepts of keyed file management and is aimed at the reader who has had little experience of ISAM techniques. It provides an introduction to the remainder of the manual which explains the file structures used by ISAM; how to implement ISAM and how to use the individual functions within a BASIC program.

Appendix A describes the tutorial program and contains examples which enable the user to become familiar with the techniques used by ISAM.

The reader is assumed to have a working knowledge of BASIC and to be familiar with the M20.

REFERENCES:

BASIC Language Reference Guide
Code 3982430 P

Professional Computer Operating System (PCOS) User Guide
Code 3987590 U

FIRST EDITION: July 1982

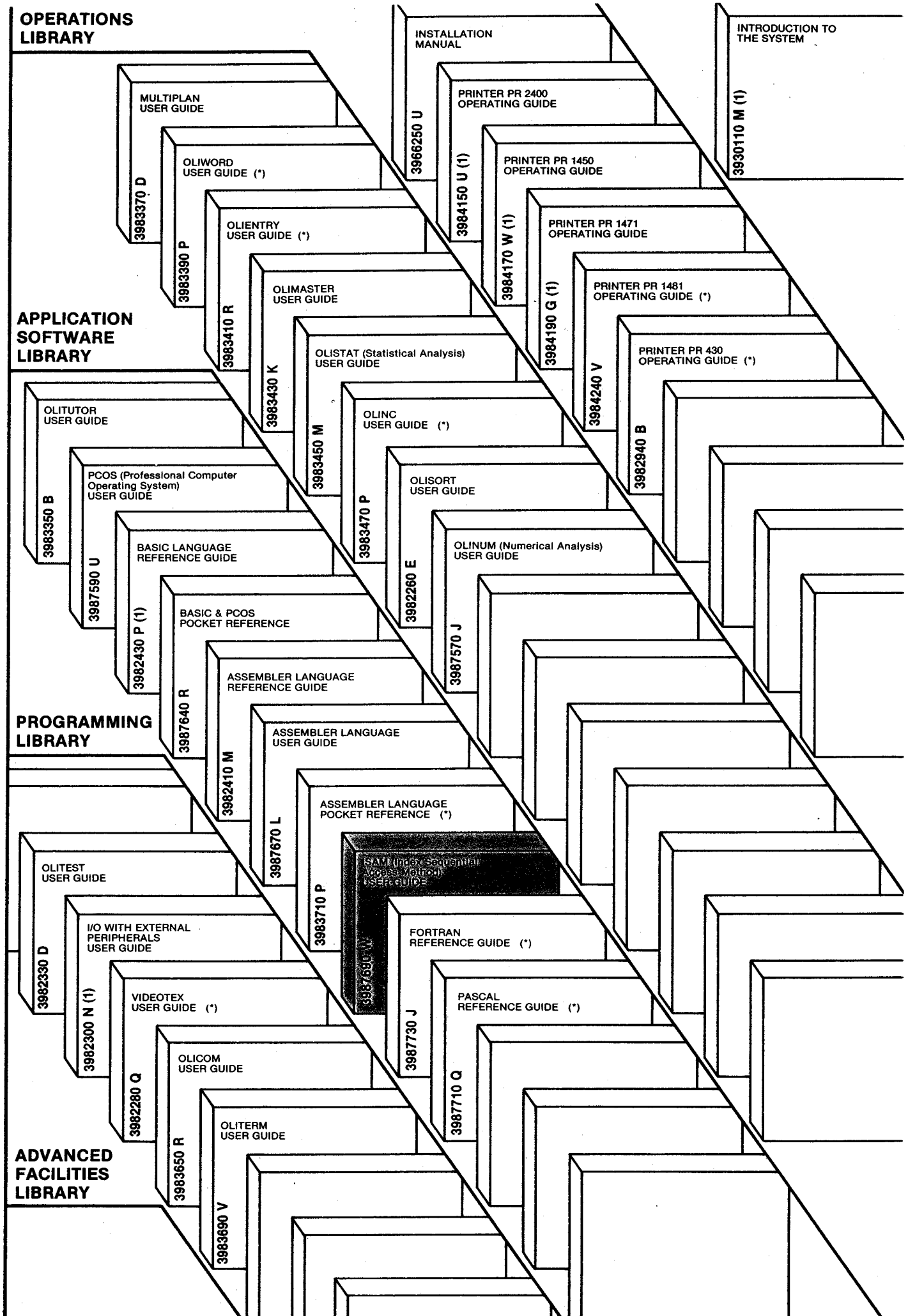
RELEASE: 1.1

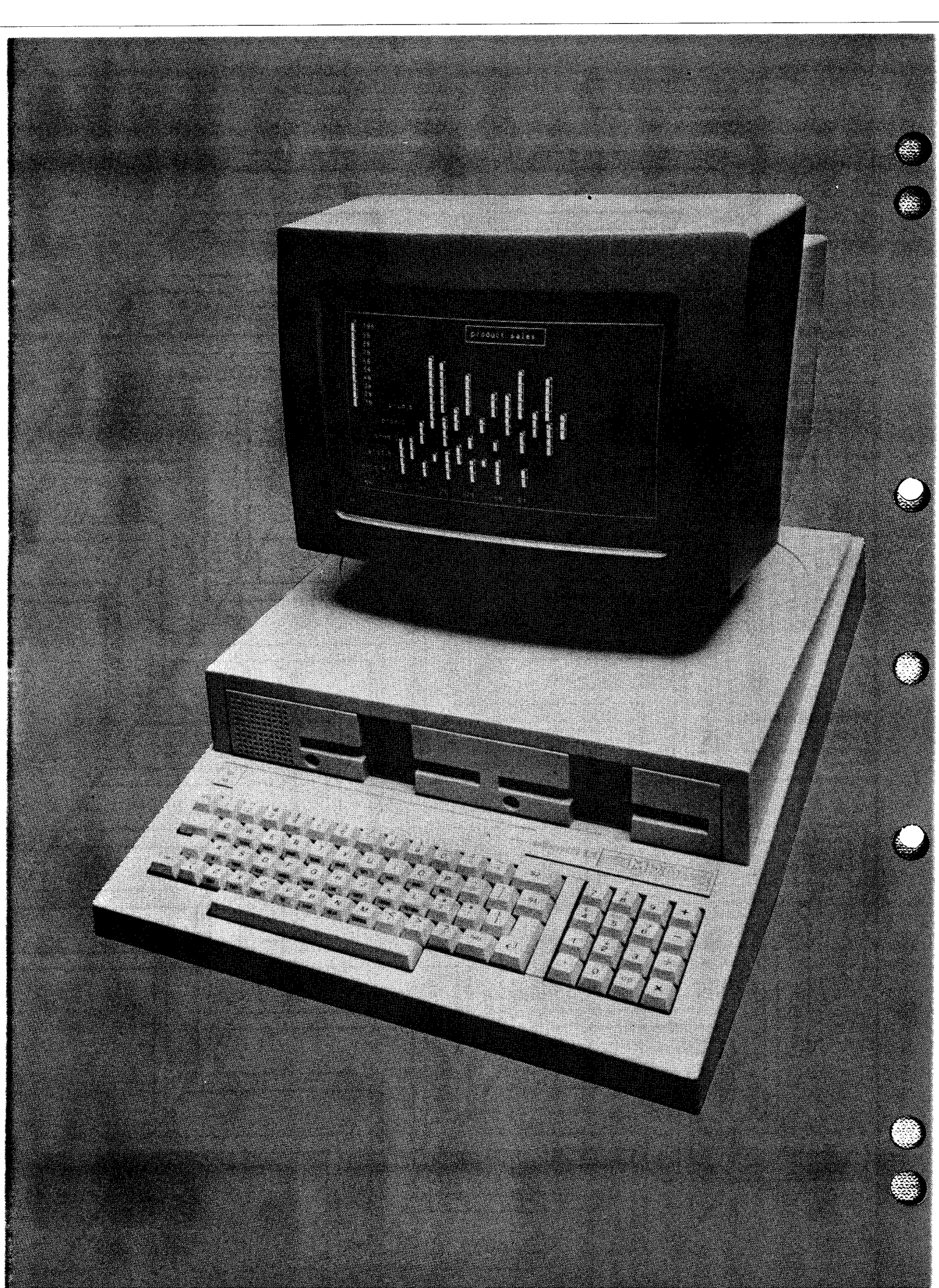
PUBLICATION ISSUED BY:

Ing. C. Olivetti & S.p.A.
Servizio Centrale Documentazione
77, Via Jervis-10015 IVREA (Italy)

The following are trademarks of Ing. C. Olivetti S.p.A.: OLICOM, OLITERM, OLIWORD, OLINUM, OLISTAT, OLIMASTER, OLITUTOR, OLITEST, OLIENTRY, OLISORT, GTL.

This manual also refers to Release 1.2.


OPERATIONS LIBRARY
INSTALLATION MANUAL
3966250 U
INTRODUCTION TO THE SYSTEM
3930110 M (1)
MULTIPLAN USER GUIDE
3983370 D
OLIWORD USER GUIDE (*)
3983390 P
OLIENTRY USER GUIDE (*)
3983410 R
OLIMASTER USER GUIDE
3983430 K
OLISTAT (Statistical Analysis) USER GUIDE
3983450 M
OLINC USER GUIDE (*)
3983470 P
OLISORT USER GUIDE
3982260 E
OLINUM (Numerical Analysis) USER GUIDE
3987570 J
PRINTER PR 2400 OPERATING GUIDE
3984150 U (1)
PRINTER PR 1450 OPERATING GUIDE
3984170 W (1)
PRINTER PR 1471 OPERATING GUIDE
3984190 G (1)
PRINTER PR 1481 OPERATING GUIDE (*)
3984240 V
PRINTER PR 430 OPERATING GUIDE (*)
3982940 B
APPLICATION SOFTWARE LIBRARY
OLITUTOR USER GUIDE
3983350 B
PCOS (Professional Computer Operating System) USER GUIDE
3987590 U
BASIC LANGUAGE REFERENCE GUIDE
3982430 P (1)
BASIC & PCOS POCKET REFERENCE
3987640 R
PROGRAMMING LIBRARY
ASSEMBLER LANGUAGE REFERENCE GUIDE
3982410 M
ASSEMBLER LANGUAGE USER GUIDE
3987670 L
ASSEMBLER LANGUAGE POCKET REFERENCE (*)
3983710 P
ISAM (Index Sequential Access Method) USER GUIDE
3987690 W
FORTRAN REFERENCE GUIDE (*)
3987730 J
PASCAL REFERENCE GUIDE (*)
3987710 Q
OLITEST USER GUIDE
3982330 D
I/O WITH EXTERNAL PERIPHERALS USER GUIDE
3982300 N (1)
VIDEOTEX USER GUIDE (*)
3982280 Q
OLICOM USER GUIDE
3983650 R
OLITERM USER GUIDE
3983690 V
ADVANCED FACILITIES LIBRARY
(*) T
ge made available with the next release.


product sales

ERRATA	CORRIGE
page 1-5 line 8 ... from the index and may also .... as deleted.	... from the index and makes the data record available for re-use when adding new records.
page 3-2 line 9 MERGE "ISAM.BAS"	MERGE "isam.bas"
page 3-2 line 10 SAVE 1:MYPROG	SAVE "1:MYPROG"
page 3-6 line 3 File DCB number.	Unused
page 3-8 line 11 and 12	**delete** RK/SK RG/SG
page 3-8 line 26 This code .... unique.	This code will be returned if the file type is unique, or if the file type is duplicate and the specified key and record number already exist.
page 3-9 line 3 00 0C 0F	00
page 3-13 line 3 The following ... -ISAM % (8) - return code	The following variable will be returned: -ISAM % (8) - return code

ERRATA	CORRIGE
page 3-17 line 19	
... on a duplicate key the ...	... on a duplicate index the ...
page 3-19 line 18	
... from a variable disk.	...from a variable list.
page 3-21 line 28	
... first greater and 1ØØØ.	... first greater than 1ØØØ.
page 3-22 line 9	
... key in the index, the Read ...	... key in the index, or if there are no keys in the index, the Read ...
page 3-27 line 20	
Statement 12Ø and 16Ø	Statement 12Ø to 16Ø
page 3-27 line 21	
ISAM$(4)="INDEX-B	ISAM$(4)="INDEX-B"
page 3-29 end of page	**add:**
	When performing a Key Delete function on a duplicate key type index, the record number must be specified to ensure that the intended key is deleted. If the record number is specified as zero, then the first occurrance of the key will be deleted.

ERRATA	CORRIGE
page 3-31 line before last	**add:**
	When performing a Delete Record function on a duplicate key type index, the record number must be specified to ensure that the key to the intended data record is deleted. If the record number is specified as zero, then the first matching key will be deleted.
page A-1 line 19	
RP,RN - Read Previous	RP,SP - Read Previous
page A-2 line 26	
the data file ...	the index file ...
page A-3 line 4	
data file IFILE1, ...	index file IFILE1,...
page A-3 line 5	
to the index record ...	to the data record ...
page B-1 line 18	
of record number, key number and record...	of record number and record ...
page B-2 line 20	
first key printer ...	**delete line**

ERRATA	CORRIGE
page B-3 line 18 data record number (.... ....), the key lenght ...	data record number, the key lenght ...
page C-2 line 3 File DCB number	Unused

CONTENTS

1. CONCEPTS OF KEYED FILE MANAGEMENT

METHODS OF FILE HANDLING 1-1

KEYED FILE STRUCTURES 1-1

SECONDARY INDEXING 1-2

CONCATENATED KEYS 1-3

KEYED RECORD RETRIEVAL 1-3

RECORD AND KEY DELETION 1-5

APPLICATIONS FOR KEYED FILE STRUCTURES 1-5

INQUIRIES 1-5

EDITS 1-6

UPDATES 1-6

ADDITIONS AND DELETIONS 1-6

2. ISAM FILE STRUCTURES

ISAM FILES 2-1

THE DATA FILE 2-1

THE INDEX FILE 2-1

B TREES 2-2

ISAM TREE STRUCTURES 2-4

DATA CONTROL BLOCKS 2-5

FILE BUFFERING AND NUMBER OF OPEN FILES 2-6

DELETED RECORDS 2-6

3. USING ISAM IN A BASIC PROGRAM

ISAM PROGRAM FILES 3-1

USING ISAM ON THE M20 3-1

USING ISAM WITH BASIC 3-1

COMMUNICATING WITH ISAM 3-3

RETURN CODES 3-7

ISAM UTILITY FUNCTIONS 3-9

METHOD STATUS (MS) 3-9

METHOD INITIALISE (MI) 3-11

OPENING AND CLOSING ISAM FILES 3-12

FILE OPEN (OO) 3-12

CREATE FILE (OC) 3-13

OPEN/CREATE FILE (OF) 3-15

CLOSE FILE (CL) 3-16

RETRIEVING DATA FROM AN ISAM FILE 3-17

READ KEY (RK, SK) 3-17

READ GENERIC (RG, SG) 3-19

READ NEXT (RN, SN) 3-22

READ PREVIOUS (RP, SP) 3-24

WRITING DATA TO AN ISAM FILE 3-25

WRITE ADD (WA, SA) 3-25

DELETE FUNCTIONS 3-29

KEY DELETE (KD, SD) 3-29

DELETE RECORD (DR) 3-31

APPENDICES

A. ISAM TUTORIAL PROGRAM A-1

B. ISAM FILE DUMP UTILITY B-1

C. ISAM VARIABLES C-1

VARIABLES PASSED TO ISAM C-1

VARIABLES RETURNED FROM ISAM C-2

D. FUNCTION CODES D-1

E. RETURN CODES E-1

1. CONCEPTS OF KEYED FILE MANAGEMENT

ABOUT THIS CHAPTER

This chapter provides a general introduction to keyed file management. It describes the types of file used, the methods by which information may be retrieved from the files, and gives a few examples of situations where keyed file management is particularly useful.

CONTENTS

METHODS OF FILE HANDLING 1-1

KEYED FILE STRUCTURES 1-1

SECONDARY INDEXING 1-2

CONCATENATED KEYS 1-3

KEYED RECORD RETRIEVAL 1-3

RECORD AND KEY DELETION 1-5

APPLICATIONS FOR KEYED FILE STRUCTURES 1-5

INQUIRIES 1-5

EDITS 1-6

UPDATES 1-6

ADDITIONS AND DELETIONS 1-6

CONCEPTS OF KEYED FILE MANAGEMENT

METHODS OF FILE HANDLING

A keyed file management system is a technique for organising and processing files. It is used to access data records in logical sequential order or randomly on the basis of a key or identifying data element of the individual data records. Indexed Sequential Access Method (ISAM) is one such implementation.

Before going into more detail about keyed file management it is worthwhile looking briefly at two simpler and better known access methods:

- Sequential access method (SAM) is simply the writing and reading of records one after the other, that is, in physical sequential order. Any time a new record is created it must be added to the end of the file. A record may be inserted into the file only by copying the entire file and writing the new record at the desired location in the new file. If a specific record in the file is desired the file must be searched in sequential order until that record is found. Record updates may or may not be allowed depending on the implementation.

- Random access method is more flexible. It is also called "direct access method" (DAM). DAM allows records to be written into or retrieved from any location within the file by knowing the actual physical location on the disk, or the record number (absolute position within the file), or the relative position of the desired record within the file. In this manner a specific record can be retrieved randomly by record number, physical disk location,or relative position from some unique data element in the record. In relative record addressing, the record number itself can serve as the key.

Each of these file organisations is ideal for particular uses. But for applications where there is a need to access records sequentially or randomly by their associated keys, a keyed file structure provides a much more useful approach.

KEYED FILE STRUCTURES

There are two major types of data structure that can be associated with a keyed file structure: the index file and the data file.

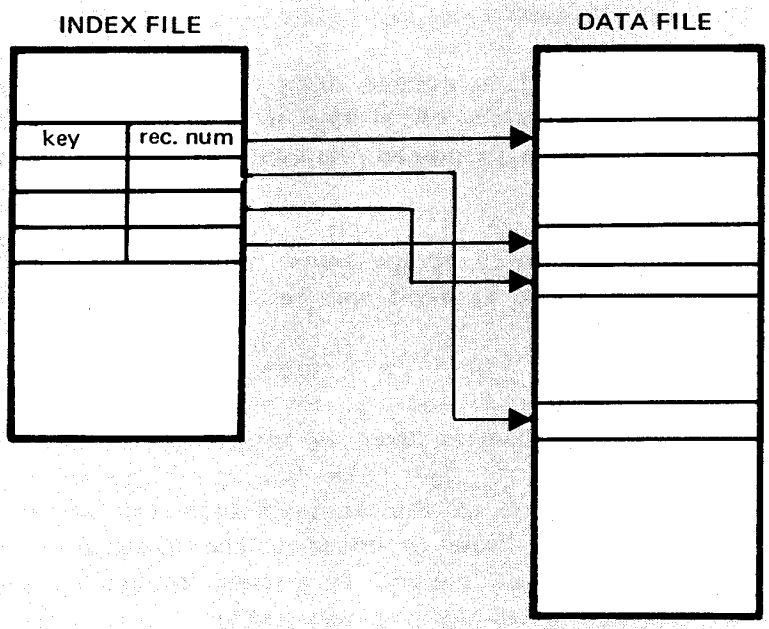


Figure 1-1 Index and Data Files

The index file contains index records that provide pointers to associated data records based on a unique and identifying data element in each record called the key. There may also be secondary or alternate indices which provide access to the data records via a secondary or alternate key. This provides multiple paths through the data.

It is also possible to have the same key repeated several times within an index file, e.g. if a person's name is to be used as the key, there might be several people with the same name. This does mean, however, that if unique access to data records is to be maintained, the data record number must be specified with the key value.

The data file contains the data records. The data file may be in key order or physical order depending on the implementation technique.

SECONDARY INDEXING

Secondary indexing allows data records to be accessed by different keys. For example, the records in a customer master file may need to be retrieved by customer number or by customer name. The primary index would be built using the customer number as the key and a secondary index would be built based on a customer name, thus providing two independent keyed paths through the data file.

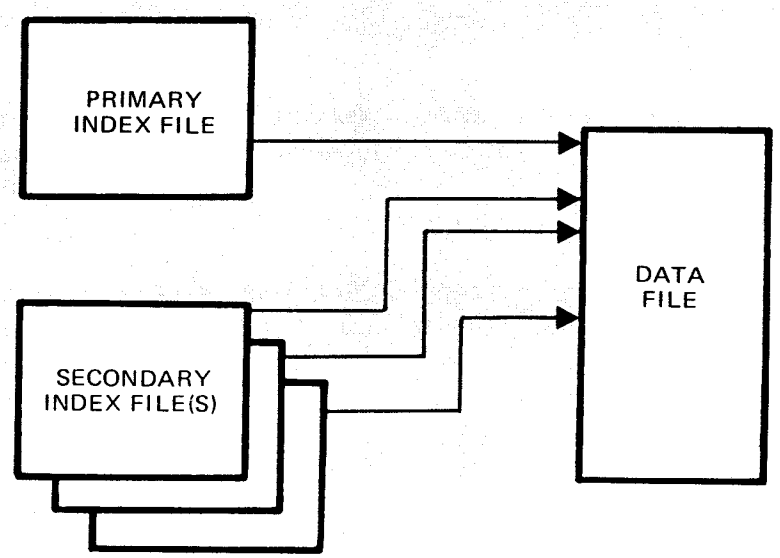


Figure 1-2 Secondary Indexing

CONCATENATED KEYS

It may be necessary to access data records using a combination of data elements as the key. Two or more data elements are combined (or concatenated) to form the actual key for the record.

For example, a company with several branch offices may want to keep the customer records for one branch separated from those of other branches so that reports may be easily produced by branch. It is also desirable to be able to retrieve information concerning any customer from any branch for inquiry and updating. If the key were composed of branch number and customer number, all customers for a given branch would be logically grouped and could be accessed independently of the other branches. Yet all customers would be available as a group for global inquiries and reports.

KEYED RECORD RETRIEVAL

There are five possible methods that can be used to retrieve records in a keyed file structure:

- Random retrieval by key is the technique whereby a key value is passed to the keyed access method which then returns a pointer to the associated data record.

  Upon receiving the key the access method searches the index file for the corresponding record pointer. The access method returns information indicating the result of the search. If the search was successful then a pointer to the record is also returned.

  Random retrieval by key is useful for inquiries and updates where a specific record is desired and the key of that record is known.

- Generic retrieval by key is similar to random retrieval by key except when the specific record requested is not found, in which case the pointer to the record of the next higher key is returned. Generic retrieval by partial key value can be used to group data records into logical sections.

  Generic retrieval provides a powerful tool for selecting and organising data records.

- Sequential retrieval by key provides the ability to read the data records in logical key sequence regardless of their physical sequence in the data file. This process can begin with the first logical record in the file, or it may begin from any point within the data file.

  Logical sequential retrieval can be used to retrieve data records in a particular sequence, that is, in key order. Or, in conjunction with generic retrieval, it can be used to accomplish retrieval of logical groupings of data records from the file structure.

- Sequential retrieval in physical order enables data records to be retrieved in physical order from the data file without reference to an index. This is convenient for data processing tasks where record retrieval sequence is unimportant, such as data analysis, as it reduces access time by eliminating the index search.

- Random retrieval by record number enables data records to be retrieved randomly by their record numbers. This allows the data file to be treated as a DAM file as well as a keyed structure.

RECORD AND KEY DELETION

A keyed file management system can provide the ability to remove data records and/or their associated keys from the file structure. This reduces the programming effort required to support deleted records, saves disk space and decreases the need for file reorganisation.

When a data record is no longer needed in the file structure, it can be removed by a record delete function. This removes the key associated with the data record from the index and may also flag the data record as deleted.

If access to a data record by its key is no longer required, but the record will be required for subsequent non-keyed access, a key delete function can be used. This removes the key from the index but leaves the data record unchanged. The responsibility for the removal of the data record is then left to the programmer. This is useful for applications where the data record will be required for later reporting, such as in archive generation. This function is also useful for removing the keys of deleted records from secondary indices.

APPLICATIONS FOR KEYED FILE STRUCTURES

Keyed file structures provide much more flexibility than other file organisations and lend themselves more readily to interactive and real time applications. The following examples illustrate some actual applications.

INQUIRIES

A primary application for keyed files is inquires. An inquiry may be serviced by retrieving information from a specific record by using the record key.

For example, in an inventory control system, management may want to know how much stock is on hand before committing themselves to an order. Using a keyed file structure with the item number as the key, a program can be written to request the item number from the operator, retrieve the corresponding record from the file structure, and display the stock status information on the screen.

Illustrating secondary indexing, a secondary key of item description could be assigned to the inventory file. Thus if the item number is not known, the description could be used to retrieve the desired item, or even a group of items with similar descriptions.

EDITS

A second useful application is data validation. A data element in one file may be the key to a keyed file structure. If this is the case, whenever that data element is entered by the operator it may be verified by performing a random retrieval by key against the file structure for which the data element serves as the key. If the key is not located, the operator can be immediately informed that the data is invalid and should be re-entered.

For example, in a payroll application the weekly time records will contain the employee number. The employee number can be entered and used as the key for a random retrieval against the employee master file. If a "no match" condition results, the operator can be informed that the employee number is invalid so that the correct value can be entered.

UPDATES

Possibly the most useful application for keyed file structures is real-time updating. A data record can be updated as soon as any transaction changes it, making current information available on a real-time basis. This would normally be used in conjunction with inquiries.

For example, in a banking environment, deposit and withdrawal transactions can be directly applied to the associated accounts by performing a random retrieval by key (account number), adding or subtracting the transaction to/from the account, and rewriting the record. Any inquiry or subsequent transaction would reflect the current balance, even if the previous transaction had been made only moments earlier.

ADDITIONS AND DELETIONS

Data records can be added to or deleted from keyed file structures in real-time. In this manner new data is immediately available for access and old data is removed making that space available for new data.

For example, in the inventory control system, a new item added to inventory immediately becomes available for ordering, and a discontinued item can be removed preventing orders being made against it.

2. ISAM FILE STRUCTURES

ABOUT THIS CHAPTER

This chapter describes the file structures used by ISAM and how they are handled.

CONTENTS

ISAM FILES	2-1
THE DATA FILE	2-1
THE INDEX FILE	2-1
B TREES	2-2
ISAM TREE STRUCTURES	2-4
DATA CONTROL BLOCKS	2-5
FILE BUFFERING AND NUMBER OF OPEN FILES	2-6
DELETED RECORDS	2-6

ISAM FILES

ISAM uses two data structures: index files and data files. Multiple index files will also be present if secondary indexing is being used. Chapter 1 describes these concepts.

ISAM data files are stored in BASIC format and are compatible with all BASIC features and file facilities. Utilities such as SORT and MERGE can therefore be used without modification. But note that any such action will destroy the relationship between the data file and the index file(s).

THE DATA FILE

The data file is a BASIC random file and can therefore be directly manipulated using BASIC statements for random files. For instance, you would transfer a record from the data file into its buffer using a BASIC GET statement; you would use a PUT statement to write a record from the file buffer to the data file; you would define the layout of the file buffer using FIELD statements. In fact ISAM performs none of these functions for you. It is left to the programmer.

THE INDEX FILE

The index file is made up of records each containing a number of keys, and appended to each key is the corresponding record number.

Duplicate keys require special treatment in that the key searching mechanism is slightly different. This gives rise to two types of index file: one in which duplicate keys are permissible, and one which only allows unique keys.

In either case the keys are not stored in any physical sequence, but a series of forward and backward pointers maintain the keys logically in alphanumeric order. This logical ordering is established using a "tree" structure. The type of tree structure used by ISAM is known as a B+ tree.

B TREES

Before discussing the B+ tree used by ISAM it is worthwhile considering a simpler, more general B tree in order to get a better understanding of how B trees work.

In one form of B tree each key occupies one node and has a forward and a backward pointer which point to keys in the next lower node. This is also known as a binary tree, e.g. if the first key to be entered into the index file is "7", then "9" and "4.5" are added, the structure shown in Figure 2-1 would result.

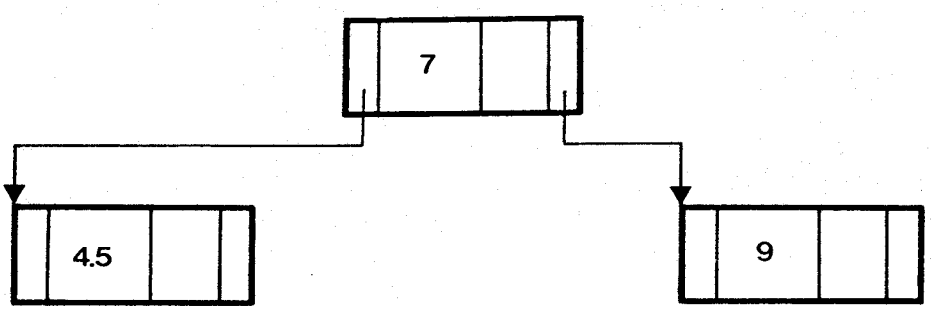


Figure 2-1 B Tree Structures I

When a record is added to the tree it is first compared to the key value at the top of the tree - this is known as the "root". If its value is greater than that of the root node it is then compared to the key value pointed to by the forward pointer. Conversely, if it is less than the key value of the root node it is compared to the key value pointed to by the backward pointer. Successive comparisons then take place until the correct position is found for the key at the lowest level of the tree. e.g. If 2, A3, A5, 4.5A and 9.9 are then added to the structure shown in Figure 2-1, then the structure shown in Figure 2-2 will result.

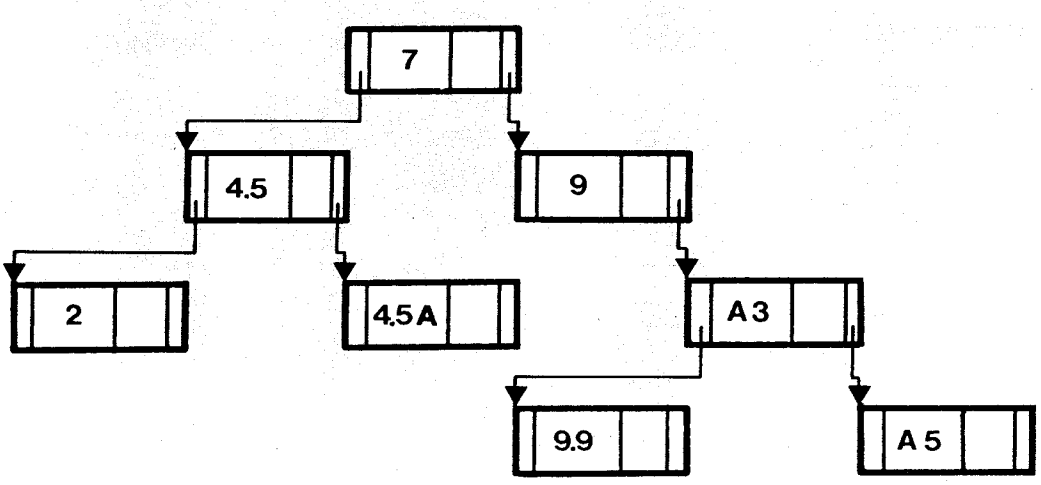


Figure 2-2 B Tree Structures II

As long as the tree remains balanced, i.e. there are a similar number of keys to the left of a node as to the right, search time is kept to a minimum. For instance, in the example shown in Figure 2-2 key A5 can be retrieved by examining nodes 7, 9, and A3 - i.e. three comparisons. But supposing the keys had been entered in sequential order, then you would get the structure shown in Figure 2-3.

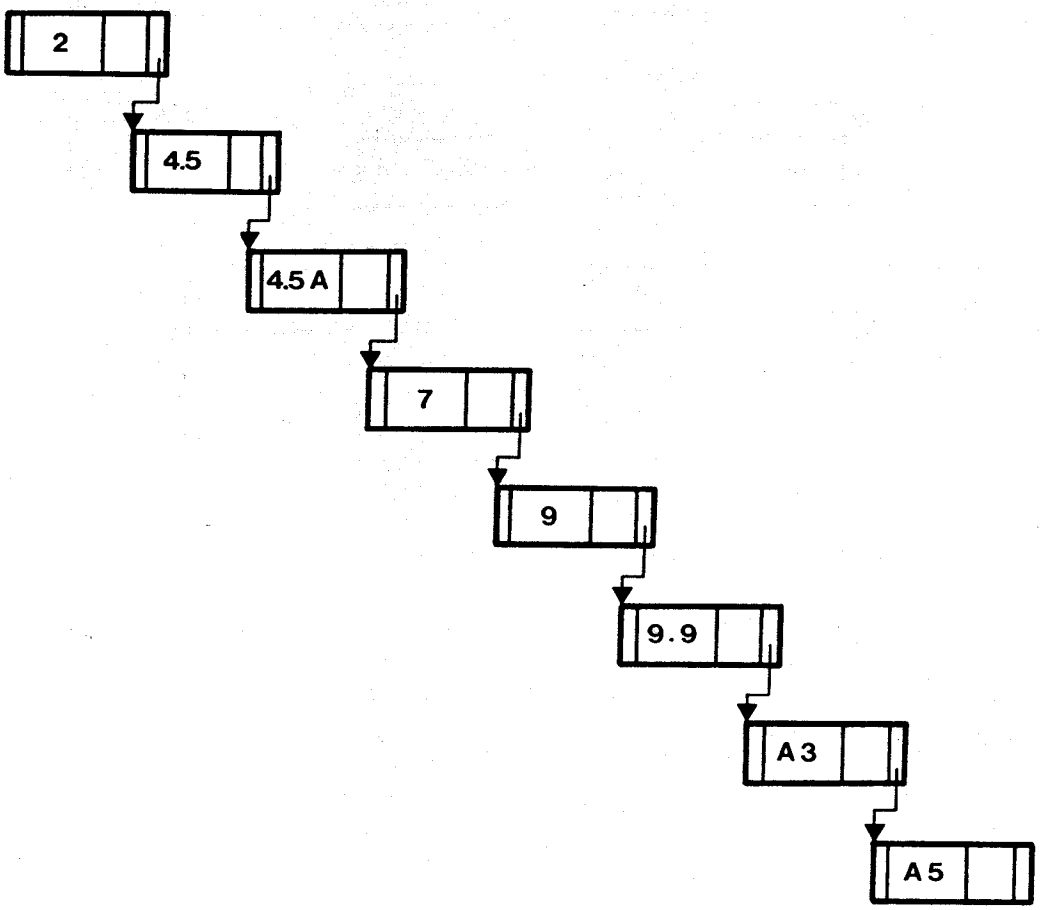


Figure 2-3 B Tree Structures III

In this case retrieving A5 would require seven comparisons. This sort of situation requires a "splitting" algorithm to balance the tree.

ISAM TREE STRUCTURES

The B+ trees used by ISAM are not so simple as the B tree described above. In fact ISAM trees use two types of record: index node records and leaf node records. Basically the leaf node records reside at the lowest level of the tree and contain all the keys and their corresponding record numbers. The index node records reside in all but the lowest level of the tree and provide the access path to the leaf node records.

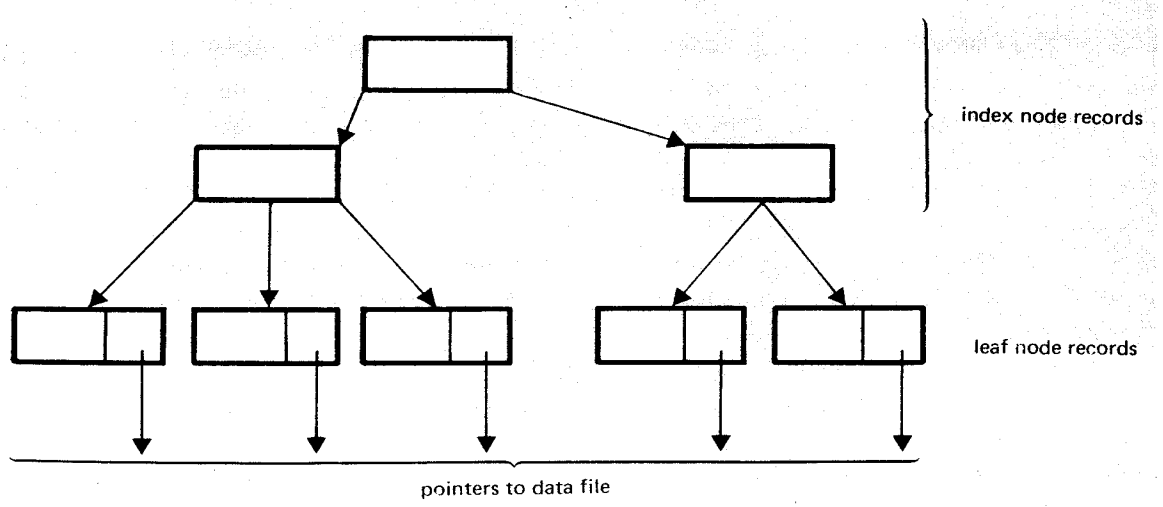


Figure 2-4 ISAM Data Structures

In addition, each leaf node record has a forward and a backward pointer which chain it to the next and previous leaf node records. This enables all leaf node records, and hence all leaf nodes, to be accessed quickly in ASCII sequence.

As the record size is fixed but the key length is variable, the number of keys per index file record is also variable.

DATA CONTROL BLOCKS

When an index file is opened it must have a data control block (DCB) assigned to it. This block will contain information pertaining to the current use of the file. It contains a pointer to the header control block (HCB), which is a record held within the index file itself and contains the header information.

This information is used for purposes such as assisting in the retrieval of the data record of the next sequential key; accessing the deleted record stack for the next available deleted record; and accessing the next available node for sequential writing.

The DCB also contains a flag that indicates whether the file is a duplicate or unique type.

FILE BUFFERING AND NUMBER OF OPEN FILES

Among the ISAM functions available is one which compels the user to define the maximum number of index files that can be open at one time. This number must be in the range 1 to 15. The associated file numbers then must not be used by the BASIC program, e.g. if ISAM is initialised for a maximum of four files, then the file numbers available to the BASIC program will be 5 to 15. If file numbers 1, 2, 3 or 4 are used by the BASIC program, unpredictable results will occur. One way of avoiding this conflict is to number your data files in descending order starting from the highest available file number.

Index file records are accessed via file buffers allocated from the system heap. A minimum of two buffers per file are automatically allocated so that the ISAM splittihg algorithms can be used on the index file to keep the B+ tree balanced. But the user is at liberty to specify additional buffers up to a limit imposed by the available memory, i.e. as the number of files increases, so does the usage of memory for control of these files. i.e. DCBs etc. are required and less memory becomes available for buffering. You are therefore advised to keep the number of open files to a minimum, thereby allowing as much memory as possible for the buffers.

While two additional buffers per index file are advisable ISAM will still function normally on less, but not so efficiently. ISAM is able to do so because it contains algorithms that allocate the available buffers to the most recently used files, thereby making the best possible use of the buffer space. i.e. when an index file requires an additional buffer and all buffers are currently in use, it is allocated the buffer which was the least recently used.

DELETED RECORDS

Record deletion is implemented by deleting the key from the index file and saving the associated record number so that the record can be re-used by a subsequent write function. Deleted records are saved on a last-in-first-out stack". i.e. the last record to be deleted will be the first to be re-used.

3. USING ISAM IN A BASIC PROGRAM

ABOUT THIS CHAPTER

This chapter describes how you communicate with the ISAM subroutine and how to use the individual functions within a BASIC program.

CONTENTS

ISAM PROGRAM FILES	3-1	READ KEY (RK, SK)	3-17
USING ISAM ON THE M20	3-1	READ GENERIC (RG, SG)	3-19
USING ISAM WITH BASIC	3-1	READ NEXT (RN, SN)	3-22
COMMUNICATING WITH ISAM	3-3	READ PREVIOUS (RP, SP)	3-24
RETURN CODES	3-7	WRITING DATA TO AN ISAM FILE	3-25
ISAM UTILITY FUNCTIONS	3-9	WRITE ADD (WA, SA)	3-25
METHOD STATUS (MS)	3-9	DELETE FUNCTIONS	3-29
METHOD INITIALISE (MI)	3-11	KEY DELETE (KD, SD)	3-29
OPENING AND CLOSING ISAM FILES	3-12	DELETE RECORD (DR)	3-31
FILE OPEN (OO)	3-12		
CREATE FILE (OC)	3-13		
OPEN/CREATE FILE (OF)	3-15		
CLOSE FILE (CL)	3-16		
RETRIEVING DATA FROM AN ISAM FILE	3-17		

USING ISAM IN A BASIC PROGRAM

ISAM PROGRAM FILES

ISAM is provided on a separate disk. On this disk are four files concerning ISAM:

- isamx.bas

- isamd.bas

- isam.bas

- isam.sav

isamx.bas is a tutorial program that teaches you the concepts of ISAM. It is described in Appendix A.

isamd.bas is a file dump utility that enables you to examine information about a particular file. ISAMD is described in Appendix B.

isam.bas is the file that contains the BASIC interface to the ISAM subroutine.

isam.sav contains the ISAM subroutine.

USING ISAM ON THE M20

Switch on the M20, insert the system disk and press **CR**.

After a few seconds the system header information will appear on the top of the screen followed by the ">" prompt.

Then use the SBASIC command to set the number of files which can be open concurrently.

You must now enter BASIC using the BA command.

You can now start to enter your BASIC program.

USING-ISAM WITH BASIC

ISAM is a subroutine to your BASIC programs. It is invoked using the statement GOSUB 6ØØØØ.

A BASIC program communicates with ISAM via variable values that indicate the file structure to be accessed, the function to be performed, and the key of the record to be manipulated.

When writing a program that uses ISAM, the following constraints must be applied:

- ISAM source code must be included in the program, e.g. having typed in your BASIC program you should then add the following two immediate statements:

  MERGE "ISAM.BAS" This statement appends the ISAM subroutine.
  SAVE 1:MYPROG This statement saves your BASIC program with the ISAM subroutine appended to it on the disk mounted in drive 1 in the file named MYPROG.

- no sequence number in the calling program must be between 6ØØØØ and 6Ø11Ø otherwise errors occur.

- two 1Ø element arrays are required, named

  ISAM$ and ISAM%

  These arrays must only be used for communicating with ISAM.

Any program that uses an ISAM file structure must perform the functions outlined below:

STEP	OPERATION
1	Open the index and data files
2	Assign the variable values to be passed to ISAM
3	Call the ISAM subroutine
4	Check the return code for successful completion
5	Read, write or delete data records as required
6	Close all files

COMMUNICATING WITH ISAM

All communication with ISAM takes place through the ISAM$ and ISAM% arrays.

The following tables describe these variables:

NUMERIC VARIABLE	EXPLANATION
ISAM%(1)	File DCB number. This is the number of the DCB used to monitor all file operations. It is used by all functions to indicate which of the currently open index files to use. Its value must be in the range 1 to 15.
ISAM%(2) ISAM%(3) ISAM%(4)	Unused. Unused. Unused.
ISAM%(5)	Data record number. This variable is used when building a secondary index to records that already exist. That is, it applies only to a Secondary Index Write Add (SA) function. This data record number must also be specified for all keyed access functions to a duplicate key.
ISAM%(6)	Additional disk buffers. This variable is used by the Method Initialise (MI) function to specify the total number of buffers to be allocated in addition to the two per file which are automatically allocated.
ISAM%(7)	Maximum number of open files. This variable is used by the Method Initialise (MI) function to specify the maximum number of index files that may be open at one time. Its value must be in the range 1 to 15..

Table 3-1 Numeric Variables Passed to ISAM (cont.)

NUMERIC VARIABLE	EXPLANATION
ISAM%(8)	Unused.
ISAM%(9)	Unused.

Table 3-1 Numeric Variables Passed to ISAM

STRING VARIABLE	EXPLANATION
ISAM$(1)	Function code. This must be specified for all functions as it specifies the function to be performed. Valid values are: RK - Read Key RG - Read Generic RN - Read Next RP - Read Previous WA - Write Add DR - Delete Record KD - Key Delete SK - Secondary Index Read Key SG - Secondary Index Read Generic SN - Secondary Index Read Next SP - Secondary Index Read Previous SA - Secondary Index Write Add SD - Secondary Index Key Delete OO - File Open OC - Create File OF - Open/Create File CL - Close File MI - Method Initialise MS - Method Status

Table 3-2 String Variables Passed to ISAM (cont.)

STRING VARIABLE	EXPLANATION
ISAM$(2)	Key value. This is required by the following functions: RK/SK RG/SG WA/SA DR KD/SD The key can be any length up to 11Ø ASCII characters.
ISAM$(3)	Unused.
ISAM$(4)	Index name. This specifies the name of the index file. It is used by the Open File, Create File and Open/Create File functions. The file name must conform to the standard PCOS file naming conventions
ISAM$(5)	Unused.
ISAM$(6)	Duplicate/Unique flag. This is used when creating new index files via the Create File or Open/Create File functions to specify whether or not duplicate keys are to be allowed in the file. The flag value can be either "D" for duplicate or "U" for unique. A null value is treated as a unique type.
ISAM$(7) ISAM$(8) ISAM$(9)	Unused. Unused. Unused.

Table 3-2 String Variables Passed to ISAM

The following tables describe the use of the ISAM% and ISAM$ arrays in returning variables from ISAM to the calling program:

NUMERIC VARIABLE	EXPLANATION
ISAM%(1)	File DCB number.
ISAM%(2) ISAM%(3) ISAM%(4) ISAM%(5)	Unused. Unused. Unused. Unused.
ISAM%(6)	Total records active. This value is returned by the Method Status function (MS) and reflects the total number of keys (and therefore records) that are active in the system.
ISAM%(7)	Unused deleted records. This value is returned by the Method Status (MS) function and reflects the number of deleted data records that have not yet been reclaimed by subsequent write functions.
ISAM%(8)	Return code. This can have the following values: ØØ - normal return 31 - invalid function order 41 - syntax error 51 - record not found 61 - duplicate key 71 - open/close error 81 - disk error A detailed description of each error code is given later in Table 3-5.
ISAM%(9)	Data record number. This is returned after a successful operation and can subsequently be used in BASIC file I/O statements to retrieve or write data records, as appropriate.

Table 3-3 Numeric Variables Returned from ISAM

STRING VARIABLE	EXPLANATION
ISAM$(1)	Unused.
ISAM$(2)	Key value. This value is returned by read functions other than the Read Key function: RG/SG RN/SN RP/SP
ISAM$(3) ISAM$(4) ISAM$(5)	Unused. Unused. Unused.
ISAM$(6)	Duplicate/Unique type. This variable is returned by the Method Status (MS) function to indicate the type of the file.
ISAM$(7) ISAM$(8) ISAM$(9)	Unused. Unused. Unused.

Table 3-4 String Variables Returned from ISAM

RETURN CODES

The following table describes the possible return codes:

CODE NO.	EXPLANATION	POSSIBLE FUNCTIONS
ØØ	Normal return This indicates that the requested function was performed succesfully and that the returned record number is valid.	All

Table 3-5 Return Codes (cont.)

CODE NO.	EXPLANATION	POSSIBLE FUNCTIONS
31	Invalid function order. This indicates that the sequence of function calls was incorrect. This occurs if, for instance, the first call was not a Method Initialise (MI) function.	All
41	Syntax error. This indicates that some value passed to ISAM was invalid.	All
51	Record not found. This indicates that the desired record was not located for one of the following reasons: - the requested key does not exist - the requested key is higher than the highest key in the file - an attempt to read past the logical end or begining has been made. In all three cases the returned record number is invalid	RK/SK RG/SG RK/SK RG/SG RN/SN RP/SP
61	Duplicate key. This indicates that the record to be added has the same key as a record already on the file. This code will only be returned if the file type is unique.	WA/SA
71	Open/close error. This indicates that the specified function failed for one of the following reasons: - the requested DCB number is already in use	 00 0C 0F

Table 3-5 Return Codes (cont.)

CODE NO.	EXPLANATION	POSSIBLE FUNCTIONS
	- the requested file name does not exist	00 0C 0F
	- the requested DCB has not been previously opened	ALL except 00 0C 0F
81	Disk error. Indicates that a fatal disk error has occurred from which ISAM cannot recover. Probable causes are:	
	- the directory is full	0C 0F
	- no more disk space is available.	WA/SA DR

Table 3-5 Return Codes

ISAM UTILITY FUNCTIONS

METHOD STATUS (MS)

The Method Status function returns information about a file to the user program. This information comprises the number of active data records and the number of unreclaimed deleted records.

The following variables must be passed to ISAM:

- ISAM%(1) - file DCB number

- ISAM$(1) - function code (MS)

The following variables will be returned:

- ISAM%(6) - number of active data records

- ISAM%(7) - number of unreclaimed deleted records

- ISAM%(8) - return code

- ISAM$(6) - duplicate/unique flag

To retrieve the above information using the ISAM Status function your program should follow the sequence indicated below:

STEP	OPERATION
1	Open the data file using a BASIC OPEN statement
2	Partition the buffer into fields using a BASIC FIELD statement
3	Open the index file using the File Open function
4	Specify the DCB number
5	Set the function code to "MS"
6	Call the ISAM subroutine
7	Check the return code
8	Close the data file using a BASIC CLOSE statement
9	Close the index file using the Close File function

Example

	BASIC PROGRAM	OPERATION
1ØØ	OPEN"R",15,"DFILE",128	Statements 1ØØ and 11Ø open the data file named DFILE and partition the buffer. Statements 12Ø to 16Ø open the index file named INDEX and assign DCB 1 to it.
11Ø	FIELD 15,128 AS A$	
12Ø	ISAM%(1)=1	
13Ø	ISAM$(1)="OO"	
14Ø	ISAM$(4)="INDEX"	
15Ø	GOSUB 6ØØØØ	
16Ø	IF ISAM%(8)<>Ø THEN GOTO 9ØØ	
	⋮	
2ØØ	ISAM%(1)=1	Statements 2ØØ to 23Ø perform the Method Status function on the index file.
21Ø	ISAM$(1)="MS"	
22Ø	GOSUB 6ØØØØ	
23Ø	IF ISAM%(8)<>Ø THEN GOTO 1ØØØ	

	BASIC PROGRAM	OPERATION
	. . .	
5ØØ	CLOSE 15	Statement 5ØØ closes the data file. Statements 51Ø to 54Ø close the index file.
51Ø	ISAM%(1)=1	
52Ø	ISAM$(1)="CL"	
53Ø	GOSUB 6ØØØØ	
54Ø	IF ISAM%(8)<>Ø THEN GOTO 11ØØ	

METHOD INITIALISE (MI)

The Method Initialise function must be the first call that your program makes to ISAM. Your program must specify the number of additional buffers - beyond the minimum of two per file which are atuomatically assigned - to be allocated to the program. Note that by specifying too many buffers you will exceed the available user memory.

Your program must also specify the maximum number of index files that can be kept open at one time. This value must be in the range 1 to 15 but should be dept to a minimum to allow as much memory space as possible for the buffers. Note, however, that the associated file numbers must not be used by the BASIC program, otherwise unpredictable results may occur.

The following variables must be passed to ISAM:

- ISAM%(6) - number of additonal buffers

- ISAM%(7) - number of open index files

- ISAM$(1) - function code (MI)

The following variable is returned:

- ISAM%(8) - return code

Possible return codes are: 31.

To initialise ISAM your program must, therefore, follow the sequence indicated below:

STEP	OPERATION
1	Specify the number of additonal buffers
2	Specify the maximum number of open files
3	Set the function code to "MI"
4	Call the ISAM subroutine
5	Check the return code

Example

	BASIC PROGRAM	OPERATION
2ØØ	ISAM%(6)=2	Statements 2ØØ to 24Ø set the maximum number of index files open to be two, and assign two additional buffers. File numbers 1 and 2 must not be used subsequently by the BASIC program.
21Ø	ISAM%(7)=2	
22Ø	ISAM$(1)="MI"	
23Ø	GOSUB 6ØØØØ	
24Ø	IF ISAM%(8)<>Ø THEN GOTO 9ØØ	

OPENING AND CLOSING ISAM FILES

FILE OPEN (OO)

The File Open function opens a previously created index file. You need to specify the index file name and a DCB number. All future operations on the file will then only require the DCB number. If the file does not exist, or if the specified DCB is already in use, then return code 71 is issued.

The following variables must be passed to ISAM:

- ISAM%(1) - file DCB number

- ISAM$(1) - function code (00)

- ISAM$(4) - index file name

The following variables will be returned:

- ISAM%(1) - file DCB number

- ISAM%(8) - return code

Possible return codes are: ØØ, 31, 41 and 71.

To open an index file using the File Open function your program must therefore follow the sequence indicated below:

STEP	OPERATION
1	Specify the DCB number
2	Set the function code to "00"
3	Specify the file name
4	Call the ISAM subroutine
5	Check the return code

Example

	BASIC PROGRAM	OPERATION
1ØØ 11Ø 12Ø 13Ø 14Ø	ISAM%(1)=2 ISAM$(1)="00" ISAM$(4)="IFILE" GOSUB 6ØØØØ IF ISAM%(8)<>Ø THEN GOTO 6ØØ	Statements 1ØØ to 14Ø open the index file named IFILE - assuming it has already been created - and assign DCB 2 to it.

CREATE FILE (OC)

The Create File function creates and opens an index file. You need to specify the filename and the DCB number.

If a file of the specified name already exists the file will not be created and return code 71 will be issued.

The file you are creating can be designated to contain only unique keys, or alternatively you can make it possible to have duplicate keys. This you do by setting the ISAM$(6) variable either to "D" for duplicate or "U" for unique before invoking the ISAM subroutine. The default value is "U".

The following variables must be passed to ISAM:

- ISAM%(1) - file DCB number

- ISAM$(1) - function code (0C)

- ISAM$(4) - index file name

- ISAM$(6) - duplicate/unique type (D or U)

The following variable will be returned:

- ISAM%(8) - return code

Possible return codes are: ØØ, 31, 41, 71 and 81.

To create an index file using the Create File function your program must therefore follow the sequence indicated below:

STEP	OPERATION
1	Specify the DCB number
2	Set the function code to "0C"
3	Specify the file name
4	Call the ISAM subroutine
5	Check the return code

Example

	BASIC PROGRAM	OPERATION
1ØØ 11Ø 12Ø 13Ø 14Ø 15Ø	ISAM%(1)=4 ISAM$(1)="OC" ISAM$(4)="IFILE1" ISAM$(6)="D" GOSUB 6ØØØØ IF ISAM%(8)<>Ø THEN GOTO 6ØØ	Statements 1ØØ to 15Ø create and open an index file named IFILE1, specify duplicate keys, and assign DCB 4 to the file.

OPEN/CREATE FILE (OF)

The Open/Create File function opens an index file if it already exists, or creates and opens it if it does not already exist.

If you are creating a file it can be designated to contain only unique keys, or alternatively you can make it possible for the file to contain duplicate keys. This you do by setting the ISAM$(6) variable to "D" for duplicate or "U" for unique keys. The default value is "U".

The following variables must be passed to ISAM:

- ISAM%(1) - file DCB number

- ISAM$(1) - function code (OF)

- ISAM$(4) - index file name

- ISAM$(6) - duplicate/unique type (D or U)

The following variable will be returned:

- ISAM%(8) - return code

Possible return codes are: ØØ, 31, 41, 71 and 81.

To open or create a file using the Open/Create File function your program must therefore follow the sequence indicated below:

STEP	OPERATION
1	Specify the DCB number
2	Set the function code to "OF"
3	Specify the file name
4	Call the ISAM subroutine
5	Check the return code

Example

	BASIC PPROGRAM	OPERATION
1ØØ 11Ø 12Ø 13Ø 14Ø 15Ø	ISAM%(1)=3 ISAM$(1)="OF" ISAM$(4)="IFILE" ISAM$(6)="U" GOSUB 6ØØØØ IF ISAM%(8)<>Ø THEN GOTO 6ØØ	Statements 1ØØ to 15Ø create and open an index file named IFILE which can contain unique keys only. Statement 1ØØ assigns DCB 3 to it.

CLOSE FILE (CL)

The Close File function closes an open index file that uses the DCB of the specified number.

If the DCB number specifed does not correspond to an open file, return code 71 is issued.

The following variables must be passed to ISAM:

- ISAM%(1) - file DCB number

- ISAM$(1) - function code (CL)

The following variable is returned:

- ISAM%(8) - return code

Possible return codes are: ØØ, 31, 41 and 71.

To close an index file using the Close File function your program must therefore follow the sequence indicated below:

STEP	OPERATION
1	Specify the DCB number
2	Set the function to "CL"
3	Call the ISAM subroutine
4	Check the return code

Example

	BASIC PROGRAM	OPERATION
2ØØ 21Ø 22Ø 23Ø	ISAM%(1)=1 ISAM$(1)="CL" GOSUB 6ØØØØ IF ISAM%(8)<>Ø THEN GOTO 6ØØ	Statements 2ØØ to 23Ø close the index file whose DCB number is 1.

RETRIEVING DATA FROM AN ISAM FILE

READ KEY (RK, SK)

The Read Key function searches for an exactly matching key in the index and returns the associated record number or a return code of 51 if the key does not exist. RK and SK are functionally identical.

When performing a Read Key function on a duplicate key the record number should also be passed so that the key value and record number can be used together to locate the matching key. If the record number is zero then the Read Key function will return the record number associated with the key value appearing first in the list of duplicates. Subsequent duplicates can then be retrieved using the Read Next function.

The following variables must be passed to ISAM:

- ISAM%(1) - file DCB number

- ISAM%(5) - data record number (duplicate keys only)

- ISAM$(1) - function code (RK or SK)

- ISAM$(2) - key value

The following variables will be returned:

- ISAM%(8) - return code

- ISAM%(9) - data record number

Possible return codes are: ØØ, 31, 41, 51 and 71.

To retrieve a record from the data file using the Read Key function your program should therefore follow the sequence indicated below:

STEP	OPERATION
1	Open the data file using a BASIC OPEN statement
2	Partition the data file buffer into fields using a BASIC FIELD statement
3	Open the existing index structure using the Open File function
4	Specify the DCB number
5	Set the function code to "RK" or "SK"
6	Specify the key value
7	Specify the data record number (duplicate key type files only)
8	Call the ISAM subroutine
9	Check the return code
1Ø	Read the returned record using a BASIC GET statement
11	Close the data file using a BASIC CLOSE statement
12	Close the index file using the Close File function

Example

	BASIC PROGRAM	OPERATION
1ØØ	OPEN"R",15,"1:DATAFILE",128	Statements 1ØØ and 11Ø open the random data file on the disk mounted in drive 1 and partition the buffer. Statements 12Ø to 16Ø open the index file called INDEXFILE and assign DCB 1 to it.
11Ø	FIELD 15,128 AS A$	
12Ø	ISAM%(1)=1	
13Ø	ISAM$(1)="OO"	
14Ø	ISAM$(4)="INDEXFILE"	
15Ø	GOSUB 6ØØØØ	
16Ø	IF ISAM%(8)<>Ø THEN GOTO 6ØØ	
	⋮	
2ØØ	ISAM%(1)=1	Statements 2ØØ to 25Ø read the data record whose key is SMITH. Statment 225 assumes a duplicate key file. It reads the data record number from a variable disk.
21Ø	ISAM$(1)="RK"	
22Ø	ISAM$(2)="SMITH"	
225	ISAM%(5)=DATAREC%	
23Ø	GOSUB 6ØØØØ	
24Ø	IF ISAM%(8)<>Ø THEN GOTO 5ØØ	
25Ø	GET 15,ISAM%(9)	
	⋮	
9ØØ	CLOSE 15	Statement 9ØØ closes the data file. Statements 91Ø to 94Ø close the index file.
91Ø	ISAM%(1)=1	
92Ø	ISAM$(1)="CL"	
93Ø	GOSUB 6ØØØØ	
94Ø	IF ISAM%(8)<>Ø THEN GOTO 99Ø	

READ GENERIC (RG, SG)

The Read Generic function accesses the data record whose key is the same or next greater than the requested key. This is useful when accessing the first record in a related group of records or for establishing a starting point for logical sequential retrieval. RG and SG are functionally identical.

If a group of records is to be processed it is your responsibility to check for the end of the group by checking for a change in the value of the group indicator.

If duplicate keys exist in the index, you should also specify the record number. The key value and record number are then used to locate the specific key. If you specify a record number of zero then the Read Gener-

ic function will always return the record number corresponding to the first occurrance of the key in the index, after which you can use the Read Next function to retrieve subsequent duplicates.

ISAM returns the record number in ISAM%(9) and the data in the record can therefore be retrieved using a BASIC GET statement. The key of the record found is returned in ISAM$(2).

The following variables must be passed to ISAM:

- ISAM%(1) - file DCB number

- ISAM%(5) - data record number (duplicate keys only)

- ISAM$(1) - function code (RG or SG)

- ISAM$(2) - generic key value

The following variables will be returned:

- ISAM%(8) - return code

- ISAM%(9) - data record number

- ISAM$(2) - value of the key found

Possible return codes are: ØØ, 31, 41 and 51.

To retrieve a record from the data file using the Read Generic function your program must therefore follow the sequence indicated below:

STEP	OPERATION
1	Open the data file using a BASIC OPEN statement
2	Partition the data file buffer into fields using a BASIC FIELD statement
3	Open the index file using the Open File function
4	Specify the DCB number
5	Set the function code to "RG" or "SG"
6	Specify the data record number (duplicate key type file only)

STEP	OPERATION
7	Call the ISAM subroutine
8	Check the return code
9	Read the record using a BASIC GET statement
1Ø	Close the data file using a BASIC CLOSE statement
11	Close the index file using the Close File function

Example

	BASIC PROGRAM	OPERATION
1ØØ	OPEN"R",14,"DFILE",128	Statements 1ØØ and 11Ø open the data file named DFILE and partition the buffer. Statements 12Ø to 16Ø open an index file called IFILE1 with DCB number 4.
11Ø	FIELD 14,128 AS $	
12Ø	ISAM%(1)=4	
13Ø	ISAM$(1)="OO"	
14Ø	ISAM$(4)="IFILE1"	
15Ø	GOSUB 6ØØØØ	
16Ø	IF ISAM%(8)<>Ø THEN GOTO 7ØØ	
	. . .	
2ØØ	ISAM%(1)=4	Statements 2ØØ to 25Ø perform a Read Generic function on the data file via the secondary index file named IFILE. Statement 225 is necessary only if IFILE is a duplicate key type. Statement 25Ø retrieves a record whose key is equal to or first greater and 1ØØØ. The retrieved key value is located in ISAM$(2).
21Ø	ISAM$(1)="SG"	
22Ø	ISAM$(2)="1ØØØ"	
225	ISAM%(5)=DATAREC%	
23Ø	GOSUB 6ØØØØ	
24Ø	IF ISAM%(8)<>Ø THEN GOTO 5ØØ	
25Ø	GET 14,ISAM%(9)	
	. . .	
9ØØ	CLOSE 14	Statement 9ØØ closes the data file. Statements 91Ø to 94Ø close the index file.
91Ø	ISAM%(1)=4	
92Ø	ISAM$(1)="CL"	
93Ø	GOSUB 6ØØØØ	
94Ø	IF ISAM%(8)<>Ø THEN GOTO 99Ø	

READ NEXT (RN, SN)

The Read Next function reads the next sequential key in the file and returns the key value and the corresponding data record number. RN and SN are functionally identical.

The Read Next function allows records to be read in sequential key order. The starting position can be established by a Read Key or Read Generic function or, if the first ISAM function performed on a file is a Read Next, the position defaults to the first record in the file. If the previous read operation used the last key in the index, the Read Next function will issue a "not found" error - return code 51.

If duplicate keys exist in the index then the Read Next function will return the record numbers associated with the key value in record number sequence.

ISAM returns the record number in ISAM%(9) and the data in that record can therefore be retrieved using a BASIC GET statement. The key of the record is returned in ISAM$(2).

The following variables must be passed to ISAM:

- ISAM%(1) - file DCB number

- ISAM$(1) - function code (RN or SN)

The following variables will be returned:

- ISAM%(8) - return code

- ISAM%(9) - data record number

- ISAM$(2) - key value

Possible return codes are: ØØ, 31, 41, 51 and 71.

To retrieve a record from the data file using the Read Next function your program should therefore follow the sequence indicated below:

STEP	OPERATION
1	Open the data file using a BASIC OPEN statement
2	Partition the data file buffer into fields using a BASIC FIELD statement

STEP	OPERATION
3	Open the index file using the File Open function
4	Specify the DCB number
5	Set the function code to "RN" or "SN"
6	Call the ISAM subroutine
7	Check the return code
8	Retrieve the record via a BASIC GET statement
9	Close the data file using a BASIC CLOSE statement
1Ø	Close the index file using the Close File function

Example

	BASIC PROGRAM	OPERATION
1ØØ	OPEN"R",13,"DFILE",128	Statements 1ØØ and 11Ø open the data file named DFILE and partition the buffer. Statements 12Ø to 16Ø open the index file named INDEXFILE and assign DCB 2 to it.
11Ø	FIELD 13,128 AS A$	
12Ø	ISAM%(1)=2	
13Ø	ISAM$(1)="OO"	
14Ø	ISAM$(4)="INDEXFILE"	
15Ø	GOSUB 6ØØØØ	
16Ø	IF ISAM%(8)<>Ø THEN GOTO 6ØØ	
	⋮	
2ØØ	ISAM%(1)=2	Statements 2ØØ to 24Ø retrieve the record whose key value is the next greater than the previous key value. Statement 24Ø reads the data record into the file buffer. The retrieved key value is returned in ISAM$(2).
21Ø	ISAM$(1)="RN"	
22Ø	GOSUB 6ØØØØ	
23Ø	IF ISAM%(8)<>Ø THEN GOTO 7ØØ	
24Ø	GET 13,ISAM%(9)	
	⋮	
3ØØ	CLOSE 13	Statement 3ØØ closes the data file. Statements 31Ø to 34Ø close the index file.
31Ø	ISAM%(1)=2	
32Ø	ISAM$(1)="CL"	
33Ø	GOSUB 6ØØØØ	
34Ø	IF ISAM%(8)<>Ø THEN GOTO 8ØØ	

READ PREVIOUS (RP, SP)

The Read Previous function is similar to the Read Next function exept that it returns the previous sequential key value instead of the next. RP and SP are functionally identical.

Return code 51 is issued if the previous read operation accessed the first key in the file.

The following variables must be passed to ISAM:

- ISAM%(1) - file DCB number

- ISAM$(1) - function code (RP or SP)

The following variables are returned:

- ISAM%(8) - return code

- ISAM%(9) - data record number

- ISAM$(2) - key value

Possible return codes are: ØØ, 31, 41, 51 and 71.

To retrieve a record from the data file using the Read Previous function your program should therefore follow the sequence indicated below:

STEP	OPERATION
1	Open the data file using a BASIC OPEN statement
2	Partition the data file buffer into fields using a BASIC FIELD statement
3	Open the index file using the File Open function
4	Specify the DCB number
5	Set the function code to "RP" or "SP"
6	Call the ISAM subroutine
7	Check the return code
8	Retrieve the record via a BASIC GET statement

STEP	OPERATION
9	Close the data file using a BASIC CLOSE statement
1Ø	Close the index file using the Close File function

Example

	BASIC PROGRAM	OPERATION
1ØØ	OPEN"R",12,"EMPFILE",128	Statements 1ØØ and 11Ø open
11Ø	FIELD 12,128 AS A$	the data file named EMPFILE
12Ø	ISAM%(1)=1	and partition the buffer.
13Ø	ISAM$(1)="00"	Statements 12Ø to 16Ø open
14Ø	ISAM$(4)="EMPIND"	the index file named EMPIND
15Ø	GOSUB 6ØØØØ	and assign DCB 1 to it.
16Ø	IF ISAM%(8)<>Ø THEN GOTO 8ØØ	
	⋮	
2ØØ	ISAM%(1)=1	Statements 2ØØ to 24Ø perform
21Ø	ISAM$(1)="SP"	the Read Previous function,
22Ø	GOSUB 6ØØØØ	i.e. the data record retriev-
23Ø	IF ISAM%(8)<>Ø THEN GOTO 9ØØ	ed has a key value which is
24Ø	GET 12,ISAM%(9)	the next smaller than the
	⋮	last key value.
4ØØ	CLOSE 12	Statement 4ØØ closes the data
41Ø	ISAM%(1)=1	file.
42Ø	ISAM$(1)="CL"	Statements 41Ø to 44Ø close
43Ø	GOSUB 6ØØØØ	the index file.
44Ø	IF ISAM%(8)<>Ø THEN GOTO 95Ø	

WRITING DATA TO AN ISAM FILE

WRITE ADD (WA, SA)

The Write Add function inserts a new key into the index file, assigns to that key the record number of the next available record and returns that record number in ISAM%(9). For a Write Add to a primary index (WA), your

program must then write the data record into the data file using a BASIC PUT statement using the record number returned in ISAM%(9). Failure to do so will result in an index structure that no longer corresponds to the data file and as a result the index file will need rebuilding.

This function can be used to build new ISAM file structures or add records to existing structures. Keys can be added in any order - they will be inserted automatically in the correct sequence.

When performing a Write Add function on a secondary index the function code SA must be used. The function in this case is to assign a secondary index key to an existing data record. Your program must therefore pass to ISAM both the new key value and the number of the record.

The following variables must be passed to ISAM:

- ISAM%(1) - file DCB number

- ISAM%(5) - data record number (SA only)

- ISAM$(1) - function code (WA or SA)

- ISAM$(2) - key value

The following variables are returned:

- ISAM%(8) - return code

- ISAM%(9) - data record number

Possible return codes are: ØØ, 31, 41, 61 and 71.

Remarks

It is imperative that all files are closed before terminating a program that uses the Write Add function. Failure to do so will cause permanent damage to your files. This is because BASIC does not write the end of file pointers until the files are closed.

To insert a new key into the primary index using the Write Add function, and to then write the new corresponding data record, your program must therefore follow the sequence indicated in the following table:

STEP	OPERATION
1	Open the data file using a BASIC OPEN statement
2	Partition the data file buffer into fields using a BASIC FIELD statement
3	Open the index file using the File Open, Create File or Open/Create File function, as appropriate
4	Specify the DCB number
5	Set the function code to "WA"
6	Specify the new key value
7	Call the ISAM subroutine
8	Check the return code
9	Write the data record with a BASIC PUT statement
1Ø	Close the data file using a BASIC CLOSE statement
11	Close the index file using the close file function

Example

	BASIC PROGRAM	OPERATION
1ØØ	OPEN"R",15,"DFILE-B",128	Statements 1ØØ and 11Ø open the data file named DFILE-B and partition the buffer. Statements 12Ø and 16Ø open the index file named INDEX-B and assign DCB 1 to it.
11Ø	FIELD 15,128 AS A$	
12Ø	ISAM%(1)=1	
13Ø	ISAM$(1)="OO"	
14Ø	ISAM$(4)="INDEX-B	
15Ø	GOSUB 6ØØØØ	
16Ø	IF ISAM%(8)<>Ø THEN GOTO 9ØØ	
	⋮	

	BASIC PROGRAM	OPERATION
2ØØ	ISAM%(1)=1	Statements 2ØØ to 25Ø write the data held in the data file buffer to the data record whose key is JONES.
21Ø	ISAM$(1)="WA"	
22Ø	ISAM$(2)="JONES"	
23Ø	GOSUB 6ØØØØ	
24Ø	IF ISAM%(8)<>Ø THEN GOTO 95Ø	
25Ø	PUT 15,ISAM%(9)	
	. . .	
3ØØ	CLOSE 15	Statement 3ØØ closes the data file. Statements 31Ø to 34Ø close the index file.
31Ø	ISAM%(1)=1	
32Ø	ISAM$(1)="CL"	
33Ø	GOSUB 6ØØØØ	
34Ø	IF ISAM%(8)<>Ø THEN GOTO 1ØØØ	

To insert a new key value into the secondary index file using the Write Add function your program should follow the sequence indicated below:

STEP	OPERATION
1	Open the data file using a BASIC OPEN statement
2	Partition the data file buffer into fields using a BASIC FIELD statement
3	Open the index file using the File Open, Create File, or Open/Create File function, as appropriate
4	Specify the DCB number
5	Specify the record number
6	Set the function code to "SA"
7	Specify the new key value
8	Call the ISAM subroutine
9	Check the error code
1Ø	Close the data file using a BASIC CLOSE statement
11	Close the index file using the Close File function

Example

	BASIC PROGRAM	OPERATION
1ØØ	OPEN"R",15,"DFILE-B",128	Statements 1ØØ and 11Ø open the data file named DFILE-B and partition the buffer. Statements 12Ø to 16Ø open the secondary index file named INDEX-BS and assign DCB 2 to it.
11Ø	FIELD 15,128 AS A$	
12Ø	ISAM%(1)=2	
13Ø	ISAM$(1)="OO"	
14Ø	ISAM$(4)="INDEX-BS"	
15Ø	GOSUB 6ØØØØ	
16Ø	IF ISAM%(8)<>Ø THEN GOTO 7ØØ	
	⋮	
2ØØ	ISAM%(1)=2	Statements 2ØØ to 25Ø assign a secondary key to a record, where both the key and the record are input from a variable list.
21Ø	ISAM%(5)=RECNO%	
22Ø	ISAM$(1)="SA"	
23Ø	ISAM$(2)=KEY$	
24Ø	GOSUB 6ØØØØ	
25Ø	IF ISAM%(8)<>Ø THEN GOTO 75Ø	
	⋮	
3ØØ	CLOSE 15	Statement 3ØØ closes the data file. Statements 31Ø to 34Ø close the index file.
31Ø	ISAM%(1)=2	
32Ø	ISAM$(1)="CL"	
33Ø	GOSUB 6ØØØØ	
34Ø	IF ISAM%(8)<>Ø THEN GOTO 8ØØ	

DELETE FUNCTIONS

KEY DELETE (KD, SD)

The Key Delete function enables you to delete a key from an index file but without deleting the associated record from the data file. KD and SD are functionally identical.

This is useful in a situation where a data record might be needed at a later time but access by key is no longer required.

This function physically removes a key from the index file thus making that space immediately available for subsequent additions.

The following variables must be passed to ISAM:

- ISAM%(1) - file DCB number

- ISAM%(5) - the data record number (duplicate key type file only)

- ISAM$(1) - function code (KD or SD)

- ISAM$(2) - key value for deletion

The following variables are returned:

- ISAM%(8) - return code

- ISAM%(9) - data record number

Possible return codes are: ØØ, 31, 41, 51 and 71.

To use this function to delete a key from an index file your program should follow the sequence indicated in the following table:

STEP	OPERATION
1	Open the data file using a BASIC OPEN statement
2	Partition the data file buffer into fields using a BASIC FIELD statement
3	Open the index file using the file open command
4	Specify the DCB number
5	Set the function code to "KD" or "SD"
6	Specify the key value to be deleted
7	Specify the data record number (duplicate key type file only)
8	Call the ISAM subroutine
9	Check the return code
1Ø	Close the data file using a BASIC CLOSE statement
11	Close the index file using the Close File function

Example

	BASIC PROGRAM	OPERATION
3ØØ	OPEN"R",15,"DATA-1",128	Statements 3ØØ and 31Ø open
31Ø	FIELD 15,128 AS A$	the data file named DATA-1
32Ø	ISAM%(1)=1	and partition the buffer.
33Ø	ISAM$(1)="OO"	Statements 32Ø to 36Ø open
34Ø	ISAM$(4)="INDEX-1"	the index file named INDEX-1
35Ø	GOSUB 6ØØØØ	and assign DCB 1 to it.
36Ø	IF ISAM%(8)<>Ø THEN GOTO 9ØØ	
	.	
	.	
	.	
4ØØ	ISAM%(1)=1	Statements 4ØØ to 44Ø delete
41Ø	ISAM$(1)="KD"	from the index file the key
42Ø	ISAM$(2)="1234"	whose value is 1234.
425	ISAM%(5)=DATAREC%	Statement 425 is only neces-
43Ø	GOSUB 6ØØØØ	sary if INDEX-1 is a dupli-
44Ø	IF ISAM%(8)<>Ø THEN GOTO 95Ø	cate key type file.
	.	
	.	
	.	
5ØØ	CLOSE 15	Statement 5ØØ closes the data
51Ø	ISAM%(1)=1	file. Statements 51Ø to 54Ø
52Ø	ISAM$(1)="CL"	close the index file.
53Ø	GOSUB 6ØØØØ	
54Ø	IF ISAM%(8)<>Ø THEN GOTO 1ØØØ	

DELETE RECORD (DR)

The Delete Record function deletes a record from a data file. It does this by deleting the associated key from the index file then placing the record number on top of the delete stack for later reclamaton by the Write Add function.

The Delete Record function should only be used for primary keys. If a secondary key exists to the deleted record, this must be removed using the Key Delete or Secondary key Delete function.

It is generally good practice to flag a deleted record by setting a delete code within the record. This indicates which records have been deleted but not yet reused.

The following variables must be passed to ISAM:

- ISAM%(1) - file DCB number

- ISAM%(5) - data record number (duplicate key type file only)

- ISAM$(1) - function code (DR)

- ISAM$(2) - key value for deletion

The following variables are returned:

- ISAM%(8) - return code

- ISAM%(9) - data record number

Possible return codes are: ØØ, 31, 41, 51 and 71.

To delete a data record using the Delete Record function your program should follow the sequence indicated below:

STEP	OPERATION
1	Open the data file using a BASIC OPEN statement
2	Partition the data file buffer into fields using a BASIC FIELD statement
3	Open the index file using the File Open command
4	Specify the DCB number
5	Set the function code to "DR"
6	Specify the key value
7	Specity the record number (duplicate key type file only)
8	Call the ISAM subroutine
9	Check the return code
1Ø	Close the data file using a BASIC CLOSE function
11	Close the index file using the Close File function

Example

	BASIC PROGRAM	OPERATION
1ØØ	OPEN"R",12,"RECFILE",128	Statements 1ØØ and 11Ø open the data file named RECFILE and partition the buffer. Statements 12Ø to 16Ø open the index file named IFILE-B and assign DCB 3 to it.
11Ø	FIELD 12,128 AS A$	
12Ø	ISAM%(1)=3	
13Ø	ISAM$(1)="OO"	
14Ø	ISAM$(4)="IFILE-B"	
15Ø	GOSUB 6ØØØØ	
16Ø	IF ISAM%(8)<>Ø THEN GOTO 6ØØ	
	⋮	
2ØØ	ISAM%(1)=3	Statements 2ØØ to 24Ø delete from the data file the record whose key is SMITH. Statement 225 reads the data record number from a variable list - it is only necessary if IFILE-B is a duplicate key type.
21Ø	ISAM$(1)="DR"	
22Ø	ISAM$(2)="SMITH"	
225	ISAM%(5)=DATAREC%	
23Ø	GOSUB 6ØØØØ	
24Ø	IF ISAM%(8)<>Ø THEN GOTO 65Ø	
	⋮	
3ØØ	CLOSE 12	Statement 3ØØ closes the data file. Statements 31Ø to 34Ø close the index file.
31Ø	ISAM%(1)=3	
32Ø	ISAM$(1)="CL"	
33Ø	GOSUB 6ØØØØ	
34Ø	IF ISAM%(8)<>Ø THEN GOTO 8ØØ	

APPENDICES

ABOUT THESE APPENDICES

This list of Appendices comprises an "ISAM Tutorial Program" in Appendix A; this program is meant as an instructive excercise for programmers using ISAM for the first time. The ISAM Dump Utility is described in Appendix B; this utility allows the user a closer examination of ISAM files. Appendices C, D and E are lists of ISAM variables, Function Codes and Return Codes respectively.

CONTENTS

A. ISAM TUTORIAL PROGRAM A-1

B. ISAM FILE DUMP UTILITY B-1

C. ISAM VARIABLES C-1

VARIABLES PASSED TO ISAM C-1

VARIABLES RETURNED FROM ISAM C-2

D. FUNCTION CODES D-1

E. RETURN CODES E-1

A. ISAM TUTORIAL PROGRAM

This program is designed to teach the concepts of ISAM.

Before using the program you must load your system disk containing the ISAM program files, enter the BASIC Interpreter using the BA command and have a ready-formated disk in one of the drives to contain your trial ISAM files.

Invoke the tutorial program by typing:

```
- run"isamx.bas"
```

The program then asks you to enter the name of the data file and the data record length:

```
Enter data file name: 1:dfile

Enter data record length: 128
```

The program then generates a menu which lists the selectable functions. This menu is shown in Figure A-1.

FUNCTION CODE SELECTIONS	
RK, SK - Read Key	OO - Open Existing File
RG, SG - Read Generic	OC - Create New File
RN, SN - Read Next	OF - Create and/or Open File
RP, RN - Read Previous	CL - Close File
WA, SA - Write Add	
DR - Delete Record	
KD, SD - Key Delete	X - Examine variables
MI - Method Initialise	
MS - Method Status	

Figure A-1 Function Code Selection Menu

You may now try out the functions listed in Figure A-1 but remember that the first function you use must be a Method Initialise (MI) function.

You may find it useful to get used to the program by trying out the example given below before experimenting with your own ideas.

Note that file number 1 is reserved for the data file. The first available DCB number is, therefore, 2.

DISPLAY	DESCRIPTION
Enter desired function code or 'end': mi MI - Method Initialise Enter maximum number of files open: 2 Enter number of additional buffers: 2 ISAM Initialised	You will subsequently be allowed a maximum of two index files open at one time. This must be the first function performed otherwise an error code 31 (invalid function order) will occur.
Enter desired function code or 'end': oc OC - Create New File Enter Index File Name: 1:ifile1 Enter DCB number: 2 Enter Duplicate/Unique type (D/U): u Create Function successful	This section creates and opens the index file IFILE1 on drive 1, assigns DCB 2 to it and specifies the file as being a unique type.
Enter desired function code or 'end': wa WA - Write Add Enter DCB number: 2 Enter key value: abc Enter data record text: 123 Error ISAM%(8)=71 Open/Close Error	This section attempts to insert key value 'abc' into the data file assigned DCB 3 and to write text '123' into the corresponding data record. It fails, however, because DCB 3 has not been assigned to an index file.

DISPLAY	DESCRIPTION
Enter desired function code or 'end': wa WA - Write Add Enter DCB number: 2 Enter key value: abc Enter data record text: 123 Data Record Number = 1	This section successfully inserts key value 'abc' into data file IFILE1, sets up a pointer to the index record and inserts text '123' in that record.
Enter desired function code or 'end': wa WA - Write Add Enter DCB number: 2 Enter key value: def Enter data record text: 567 Data Record Number = 2	This section inserts key 'def' into the index file and writes '567' into the corresponding data file record.
Enter desired function code or 'end': wa WA - Write Add Enter DCB number: 2 Enter key value: ØØ1 Enter data record text: ghi Data Record number = 3	This section performs another Write Add operation.

DISPLAY	DESCRIPTION
Enter desired function code or 'end': rk RK - Read Key Enter DCB number: 2 Enter key value: def Data Record Number = 2 Key = def 567	This section performs a Read Key function on key value 'def' and retrieves data 567.
Enter desired function code or 'end': rg RG - Read Generic Enter DCB number: 2 Enter key value: 1 Data Record Number = 1 Key = abc 123	This section performs a Read Generic function on key value 1. The first key greater than or equal to this is "abc" hence its corresponding data record is retrieved and has a value of "123".
Enter desired function code or 'end': rn RN - Read Next Enter DCB number: 2 Data Record Number = 2 Key = def 567	This section reads the next sequential key value, i.e. the next key greater than "abc" in this case "def".

DISPLAY	DESCRIPTION
Enter desired function code or 'end': rp RP - Read Previous Enter DCB number: 2 Data Record Number = 1 Key = abc 123	This section reads the data record corresponding to the previous key value, i.e. it reads key value "abc".
Enter desired function code or 'end': oc OC - Create New File Enter Index File Name: ifile2 Enter DCB number: 3 Enter Duplicate/Unique type (D/U): d Create function successful	This section opens and creates secondary index file IFILE2 and assigns DCB 3 to it. Furthermore, it is defined to be a duplicate type.
Enter desired function code or 'end': sa SA - Write Add Enter DCB number: 3 Enter key value: 125 Enter data record number: 1Ø	This section performs a Secondary Write function via index file IFILE2.
Enter desired function code or 'end': sa SA - Write Add Enter DCB number: 3 Enter key value: 125 Enter data record number: 1	This performs another Secondary Write function.

DISPLAY	DESCRIPTION
Enter desired function code or 'end': ms MS - Method Status Enter DCB number: 3 Total Records Active = 9 Unused deleted records = 1Ø Duplicate/Unique flag = D	This section performs a Method Status function on IFILE2 to determine how many records are active in the file, the number of unused deleted records and whether the file is a duplicate or unique type.
Enter desired function code or 'end': ms MS - Method Status Enter DCB number: 2 Total Records Active = 3 Unused Deleted Records = Ø Duplicate/Unique Flag = U	This section does the same for IFILE1.
Enter desired function code or 'end': cl Enter DCB number: 2 DCB number 2 Closed	This section closes IFILE1.
Enter desired function code or 'end': cl Enter DCB number: 3 DCB Number 3 Closed	This section closes IFILE2.
Enter desired function code or 'end': end ISAMX - complete Ok	

Now try some ideas of your own.

B. ISAM FILE DUMP UTILITY

The ISAM File Dump utility enables you to examine your ISAM files and thereby ensure your programs are working correctly.

Before using the program you must load your system disk containing the ISAM program files into memory, enter the BASIC Interpreter using the BA command and have in one of the drives the disk containing the files you are interested in.

Invoke the program by typing:

- run"isamd.bas"

The program gives you the option of console and/or line printer output then asks you to name your data and index files. It subsequently asks you if you wish to examine the index file. An affirmative reply will cause the program to display the header record (record Ø) of the index file and ask you if you wish to examine more records. If so, you will be asked for the record number.

Once you tell the program that you no longer wish to examine any more index records, it will then ask you if you would like to look at the data file. If so, each record will be displayed in terms of record number, key number and record contents.

Example

On executing the file dump utility a display such as that shown in Figure B-1 appears on the screen. In this case the console has been chosen to be the output medium. The index and data files selected are 1:IFILEA and 1:DFILEA, respectively, and the data record length defined as 128 bytes.

```
ISAM File Dump Utility
(c) Copyright 1982, OLIVETTI

How do you wish the output to be displayed?

1. At the console.

2. At the printer.

3. Both.

Enter desired selection: 1

Enter index file name: 1:ifilea

Enter data file name: 1:dfilea

Enter data record length: 128

Print index file? (Y/N): y
```

Figure B-1 Sample Display of File Dump

The response to the last prompt in Figure B-1 requests that the index file be displayed. The program responds by displaying record Ø of the index file - the header record which is illustrated in Figure B-2.

```
INDEX FILE 1:ifilea                                    RECORD Ø

HEADER CONTROL BLOCK

DCB use count              1          Length of file name      8
First key pointer          Ø          Last key pointer         Ø
Next node pointer          7          Next record pointer      22
Delete stack pointer       Ø          Root node pointer        1
Duplicate flag             Ø          Delete stack offset      Ø
Number of levels           2          Unused deleted records   Ø

Do you wish to look at more records? (Y/N): y

Enter record number: 1
```

Figure B-2 File Dump of Header Control Block (record Ø)

This display provides information about the index file. The responses given to the last two prompts then select record 1 for display - the root node record. This is shown in Figure B-3.

```
  INDEX FILE  1:ifilea
                                                    RECORD 1
  INDEX NODE

Index level                          2  Key count  4
PØ pointer                           Ø

REC              KEY LENGTH        NODE POINTER    KEY VALUE

4                    12                3          444567834523
7                    14                4          789432348765aa
19                   12                6          rem453976567
15                   16                5          u444567834523-aa

Do you wish to look at more records? (Y/N): y

Enter record number: 4
```

Figure B-3 Sample File Dump of Index Node Record

The display lists the keys contained in the record along with the corresponding data record number (displayed for duplicate files only), the key length, and the pointer to the next lower node. In this case node 4 is selected for display and is illustrated in Figure B-4. It is a leaf node containing four keys.

```
INDEX FILE  1:ifilea

LEAF NODE

Index level                          1  Number of keys  4
Forward pointer                      6  Reverse pointer  3

REC                      KEY LENGTH  KEY VALUE

7                            14          789432348765aa
8                            14          889432348765ab
9                            12          995676543765
2Ø                           2Ø          q567483456823branch5

Do you wish to look at more records? (Y/N): n

Print data file? (Y/N): y
```

Figure B-4 Sample File Dump of a Leaf Node Record

The responses given to the last two prompts given in Figure B-4 select the data file for display. This is illustrated in Figure B-5.

```
    DATA FILE 1:dfilea

    Data record number 1
    00000000000000000000000000000000000000000

    Data record number 2
    00000000000000011111111111111111111111111
       :
       :
```

Figure B-5 Sample file dump of the data file.

C. ISAM VARIABLES

VARIABLES PASSED TO ISAM

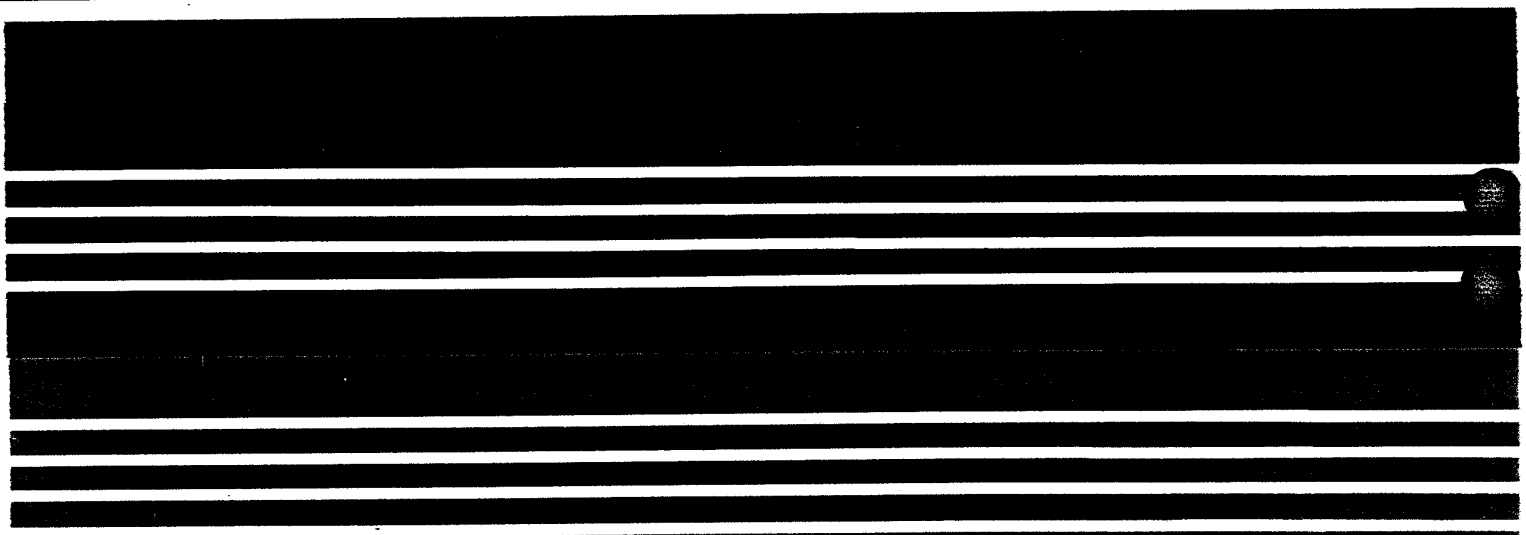
VARIABLE	DESCRIPTION
ISAM%(1)	File DCB number
ISAM%(2)	Unused
ISAM%(3)	Unused
ISAM%(4)	Unused
ISAM%(5)	Data record number
ISAM%(6)	Number of additional file buffers
ISAM%(7)	Maximum number of open index files
ISAM%(8)	Unused
ISAM%(9)	Unused
ISAM$(1)	Function code
ISAM$(2)	Key value
ISAM$(3)	Unused
ISAM$(4)	Index file name
ISAM$(5)	Unused
ISAM$(6)	Duplicate/Unique flag
ISAM$(7)	Unused
ISAM$(8)	Unused
ISAM$(9)	Unused

VARIABLES RETURNED FROM ISAM

VARIABLE	DESCRIPTION
ISAM%(1)	File DCB number
ISAM%(2)	Unused
ISAM%(3)	Unused
ISAM%(4)	Unused
ISAM%(5)	Unused
ISAM%(6)	Total number of records active
ISAM%(7)	Number of unused deleted records
ISAM%(8)	Return code
ISAM%(9)	Data record number
ISAM$(1)	Unused
ISAM$(2)	Key value
ISAM$(3)	Unused
ISAM$(4)	Unused
ISAM$(5)	Unused
ISAM$(6)	Duplicate/Unique flag
ISAM$(7)	Unused
ISAM$(8)	Unused
ISAM$(9)	Unused

D. FUNCTION CODES

FUNCTION CODE	DESCRIPTION
OO	Open File
OC	Create File
OF	Open/Create File
CL	Close File
RK	Read Key
RG	Read Generic
RN	Read Next
RP	Read Previous
SK	Secondary Index Read
SG	Secondary Index Read Generic
SN	Secondary Index Read Next
SP	Secondary Index Read Previous
WA	Write Add
SA	Secondary Index Write Add
DR	Delete Record
KD	Key Delete
SD	Secondary Index Key Delete
MI	Method Initialise
MS	Method Status



GTU Code: 898769/0 W (0)

Printed in Italy