Operating Systems and Languages Library

ISAM under MS-DOS

User Guide

olivetti

ADDENDUM

TO

ISAM UNDER MS-DOS USER GUIDE

Olivetti Personal Computer MS-DOS Rev 1.00 and following June 84

NOTICE

In order to use this implementation of ISAM you must have a system with 256K, or more, of memory.

PREFACE

This book describes the features of Indexed Sequential Access Method (ISAM) and how to use them in a GW-BASIC program on the Olivetti Personal Computer. The reader is assumed to have a working knowledge of GW-BASIC and to be familiar with the Olivetti Personal Computer

SUMMARY

Chapter 1 explains in general the concepts of keyed file management and is aimed at the reader who has little experience of ISAM techniques.

Chapter 2 explains the file structures of ISAM.

Chapter 3 describes how your GW-BASIC program communicates with ISAM.

Chapter 4 describes a tutorial program that teaches the concepts and functions of ISAM.

Chapter 5 describes in detail how ISAM can be used in a GW-BASIC Program.

Chapter 6 describes a file dump program that enables the user to examine the ISAM files.

RELATED PUBLICATIONS

M24 Installation and Operations Guide - Code 3986490 W

MS-DOS User Guide - Code 4001410 G

MS GW-BASIC Interpreter under MS-DOS User Guide - Code 4001420 H

DISTRIBUTION: General (G)

SECOND EDITION: JUNE 1984

RELEASE: Olivetti Personal Computer
MS-DOS Rev. 1.00 and Rev. 1.01

PUBLICATION ISSUED BY:

Ing. C. Olivetti & C., S.p.A.
Direzione Documentazione
77, Via Jervis - 10015 Ivrea (Italy)

TRADEMARKS NOTICE

- Concurrent CP/M-86 is a trademark of Digital Research
- Personal Basic is a trademark of Digital Research
- GSX-86 is a trademark of Digital Research
- CBASIC-86 is a trademark of Digital Research
- DR LOGO is a trademark of Digital Research
- DDT-86 is a trademark of Digital Research
- CP/M-86 is a trademark of Digital Research
- CB-86 is a trademark of Digital Research
- ASM-86 is a trademark of Digital Research
- The p-System is a trademark of Softech Microsystem, Inc.
- UCSD Pascal is a trademark of the Regents of the University of California
- PEACHPAK is a trademark of Peachtree Software International Ltd.
- UNIX is a trademark of Bell Laboratories
- PC-DOS is a trademark of International Business Machines Corp.
- Z80 is a registered trademark of Zilog Inc.
- Z8000 is a trademark of Zilog Inc.

CONTENTS

1. CONCEPTS OF KEYED FILE MANAGEMENT

ABOUT THIS CHAPTER

This chapter provides a general introduction to keyed file management. It describes the type 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.

INDEX

METHODS OF FILE HANDLING

A keyed file management system is a technique for organizing and processing files. It is used to access data records in logical sequential order or randomly on the basis of a meaningful data element such as a person's name, an account number, a part number, etc. This data element is known as the key. 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 organizations is suitable 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.

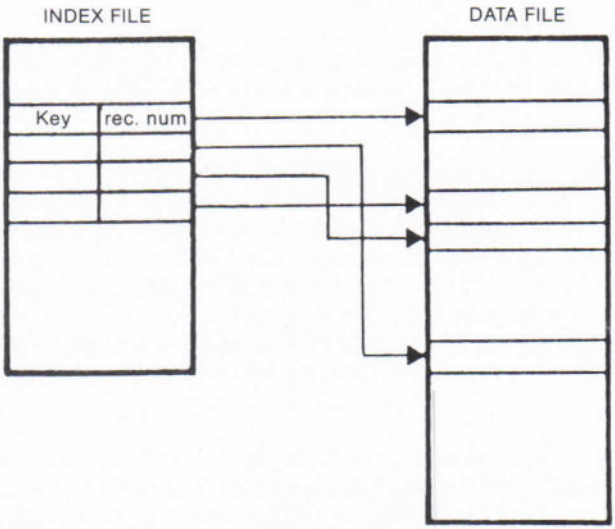


Fig. 1-1 Index and Data Files

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 indexes 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; for example, 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 is a random file containing the data records.

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.

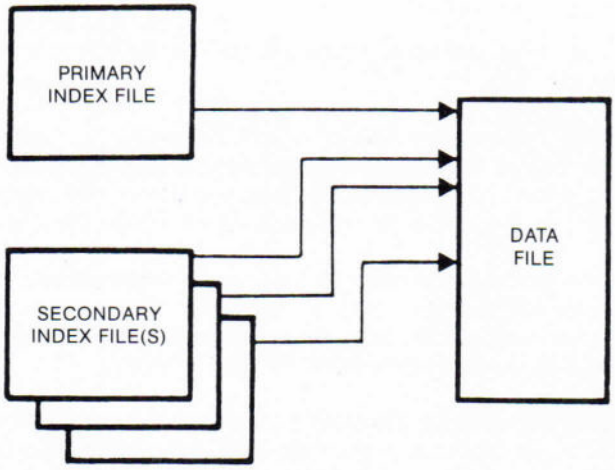


Fig. 1-2 Secondary Indexing

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 that when the specific record requested is not found, 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 organizing 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 reorganization.

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 indexes.

APPLICATIONS FOR KEYED FILE STRUCTURES

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

INDEX

ISAM FILES

ISAM uses two data structures: index files and data files. An ISAM structure will always have one data file and a corresponding primary index file. Secondary index files may also be present. Chapter 1 describes these concepts.

ISAM data files are stored in GW-BASIC format and are compatible with all GW-BASIC features and file facilities. Operations such as sorting and merging can therefore be done without modification. But note that any such action will destroy the relationship between the data file and the index file(s).

Up to 999,999 records may be accessed using ISAM keys.

THE DATA FILE

The data file is a GW-BASIC random file and can therefore be directly manipulated using GW-BASIC statements for random files. For instance, you would transfer a record from the data file into its buffer using a GW-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.

INDEX FILES

An index file is made up of records each containing a number of keys. Appended to each key is the corresponding record number. The keys are not stored in any physical sequence, but a series of forward and backward pointers maintain the keys logically in ASCII sequence. This logical ordering is established using a "tree" structure which also enables keys to be accessed randomly. The type of tree structure used by ISAM is known as a B-tree.

ISAM TREES

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 data 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.

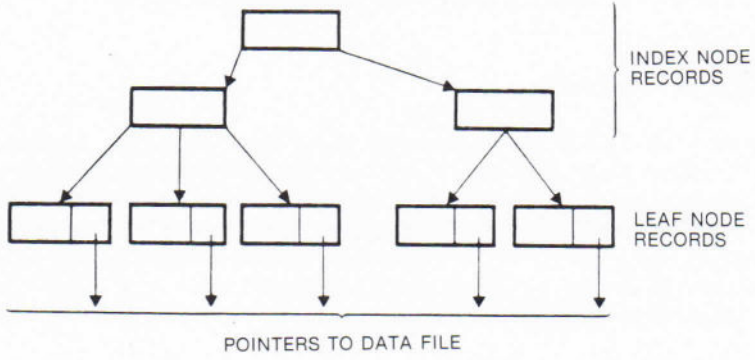


Fig. 2-1 ISAM Tree Structures

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 keys, to be accessed quickly in ASCII sequence.

At the top of the tree is the root node. This record contains pointers to the next (second) level of the index. For example, if the root node contains two keys - "000000" and "600000" - then it will contain two pointers, one to a second level index node that addresses that part of the index file containing keys "000000" to "600000", and one that

addresses keys higher than "600000". Each record in the second level of index also contains a number of keys, which in the case illustrated in Figure 2-1 point to the leaf node records. Normally, however, there will be more than two levels of index.

To access a certain key, ISAM compares the key value at each level in the index until the actual key and its corresponding data record number are found in a leaf node record. It is this technique that provides fast random access to the data file records.

As keys are added to or removed from the index the tree becomes unbalanced. For example, if a large number of keys beginning with "0" were inserted into the structure described above, additional levels of index in the left branches of the tree would be required. This would increase the number of comparisons and hence the access time. ISAM solves this problem with a set of algorithms that split the nodes and rebalance the tree. In our example ISAM might, for instance, balance the tree by accessing keys "000000" to "300000" via the left branch, and inserting another second level index node record to access keys "300001" to "600000". Keys 600001 onwards would still be accessed via the right branch.

Header information is held in a separate record called the header record. This contains general information about the file such as the key length, how many active key records the file contains, etc., and also contains pointers to the first and last keys in the file.

The ISAM File Dump program (ISAMD) can be used to examine the contents of an index file. This utility is described in Chapter 6, which by means of a sample file dump also provides further details about ISAM trees.

DUPLICATE AND UNIQUE KEYS

Duplicate keys require special treatment in that the key searching mechanism is slightly different. To identify a specific data record, the data record number must also be provided as well as the key. Moreover, this means that the data record number must be appended to the key value in the index nodes as well as the leaf nodes. This gives rise to two types of index file: one in which duplicate keys are permissible, and one which only allows unique keys.

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

When you open an index file you need to specify a DCB number. This corresponds to the BASIC file numbering scheme. All further access to the index is then done via the DCB number, not the file name.

Index file records are accessed via file buffers allocated from memory. Two buffers are automatically allocated so that the ISAM splitting algorithms can be used on the index file to keep the tree balanced. In addition, three buffers are allocated to each open index file to hold the header record, the root record and the last leaf node record accessed. To keep available memory to a maximum, you should therefore keep the number of open files to a minimum.

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 (LIFO) stack". That is, the last record to be deleted will be the first to be re-used.

3. COMMUNICATING WITH ISAM

ABOUT THIS CHAPTER

This chapter describes the mechanism by which your GW-BASIC program communicates with the ISAM subroutine.

INDEX

ISAM VARIABLES

All communication with ISAM takes place through two arrays: ISAM$ and ISAM.

The following tables describe these variables. The first such table describes the numeric variables to be passed to ISAM.

NUMERIC VARIABLE	EXPLANATION
ISAM(1)	DCB number. This is the file number known to GW-BASIC. It is assigned to a file by one of the file opening functions (Create File, Open/Create File or File Open); then used by all other functions that access the index file to indicate which of the currently open index files to use. Its value must be in the range 1 to n, where n is the maximum number of open files. For GW-BASIC this value is set by the /F parameter when you enter GW-BASIC and can have a value up to 15.
ISAM(2) ISAM(3) ISAM(4)	Unused.
ISAM(5)	Data record number. This variable must be specified to gain unique access to a record whose key is duplicated. Moreover, it is also required when building a secondary index to records that already exist; that is, when performing a Secondary Index Write Add (SA) function. ▶

NUMERIC VARIABLE	EXPLANATION
ISAM(6) ISAM(7) ISAM(8) ISAM(9)	Unused.

The following table describes the string variables to be 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 RB - Read Beginning RE - Read End 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 SB - Secondary Index Read Beginning ▶

STRING VARIABLE	EXPLANATION
	SE - Secondary Index Read End SA - Secondary Index Write Add SD - Secondary Index Key Delete OO - File Open OC - Create File OF - Open/Create File CL - Close File MS - ISAM Status.
ISAM$(2)	Key value. This is required by the following functions: RK/SK RG/SG WA/SA DR KD/SD The key can be any length up to 99 characters.
ISAM$(3)	Unused.
ISAM$(4)	Index name. This specifies the name of the index file. It is used by the Open File, Create File and Open/Create File functions. The file name must conform to the standard MS-DOS file naming conventions. (See the ''MS-DOS Disk Operating System User Guide''.)
ISAM$(5)	Unused.

STRING VARIABLE	EXPLANATION
ISAM$(6)	Duplicate Key flag. This flag determines whether or not duplicate keys are to be allowed in an index file. Its value must be set when you create a new index file using the Create File or Open/Create File function. Moreover, the flag must be set for any other function which uses it. The flag value can be either "D" for duplicate or null ("") for unique.
ISAM$(7) ISAM$(8) ISAM$(9)	Unused.

The following table describes the use of the numeric variables returned by ISAM to the calling program:

NUMERIC VARIABLE	EXPLANATION
ISAM(1) ISAM(2) ISAM(3)	Unused.

NUMERIC VARIABLE	EXPLANATION
ISAM(4) ISAM(5)	Unused.
ISAM(6)	Total records active. This value is returned by the ISAM Status function (MS) and reflects the total number of keys (and therefor ecords) that are active in the system.
ISAM(7)	Unreclaimed deleted records. This value is returned by the ISAM 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: 00 - normal return 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 the section entitled "Return Codes"
ISAM(9)	Data record number.

NUMERIC VARIABLE	EXPLANATION
	This is returned after a successful operation and can subsequently be used in GW-BASIC file I/O statements to retrieve or write data records.

The following table describes the string variables returned by ISAM to the calling program.

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 RB/SB RE/SE.
ISAM$(3) ISAM$(4) ISAM$(5)	Unused.

▶

STRING VARIABLE	EXPLANATION
ISAM$(6)	Duplicate Key flag. This variable is returned by the ISAM Status (MS) function to indicate the type of file. Its value is "D" if duplicate keys are allowed in the file, or null ("") if only unique keys are allowed.
ISAM$(7) ISAM$(8) ISAM$(9)	Unused.

RETURN CODES

The following table explains the meaning of the code returned by ISAM in ISAM(8).

CODE	EXPLANATION	FUNCTIONS
00	Normal return. This indicates that the requested function was performed successfully and that the returned record number is valid.	All

▶

CODE	EXPLANATION	FUNCTIONS
41	Syntax error. This indicates that some value passed to ISAM was invalid. For example • the specified file number was out of range • the function code was invalid. • the specified file is not open.	All
51	Record not found. This indicates that the desired record was not found 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 beginning has been made. • no keys in the index. In all cases the returned record number is invalid.	RK/SK KD/SD DR RG/SG RN/SN RP/SP RK/SK RB/SB RG/SG RE/SE RN/SN KD/SD RP/SP DR

CODE	EXPLANATION	FUNCTIONS
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, or if the file type is duplicate and the specified key and record number already exist.	WA/SA
71	Open/close error. This indicates that the specified function failed for one of the following reasons: • the requested index file number is already in use. • the requested file name does not exist.	 OO OC OF OO OC OF
81	Disk error. Indicates that a fatal disk error has occurred from which ISAM cannot recover. Probable causes are: • the directory is full • no more disk space is available.	ALL except MS CL

4. ISAM TUTORIAL (ISAMX)

ABOUT THIS CHAPTER

This chapter describes the ISAM tutorial program ISAMX.

INDEX

THE ISAM TUTORIAL PROGRAM

The ISAM Tutorial program (ISAMX) is designed to teach the concepts of ISAM. It is an interactive program that enables you to experiment with all the functions of ISAM without doing any programming.

Before using the program you must have booted MS-DOS and have on active drives GW-BASIC and the ISAM program files. Moreover, the disk to contain your trial files must also be in a drive.

SAMPLE SESSION

Let us create an ISAM file structure to contain the following stock information.

PART NUMBER	PART DESCRIPTION	STOCK LEVEL
00-12/10a	Saw 9 in.	11
00-12/11a	Saw 10 in.	25
00-12/11b	Saw 10 in.	15
00-12/11c	Saw 10 in.	8
00-12/12a	Bush saw 27 in.	3
01-18/02a	Spanner, ring - size 6	17
01-18/02b	Spanner, ring - size 8	25
01-18/02c	Spanner, ring - size 10	43
01-18/02d	Spanner, ring - size 12	18
01-18/03a	Spanner, open - size 6	4
01-18/03b	Spanner, open - size 8	7
01-18/04c	Spanner, open - size 10	15
01-18/04d	Spanner, open - size 12	0
01-04/00a	Wood glue	78
01-04/19a	Wood glue	139

Let us create a data file containing the stock level, and two index files; a primary index of part numbers, and a secondary index based on part description.

GETTING STARTED

Enter GW-BASIC as follows:

```
GWBASIC
```

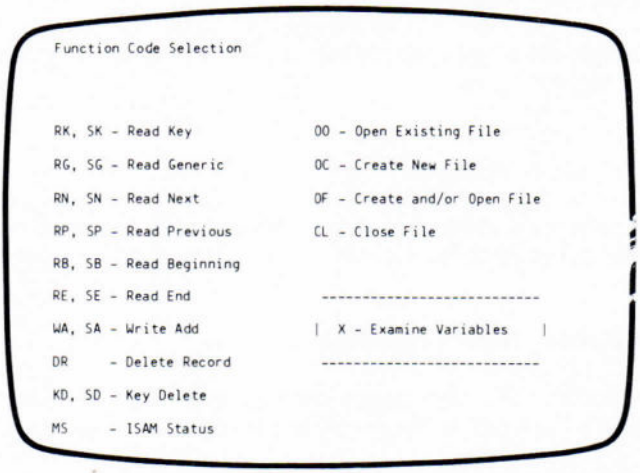
Then invoke the tutorial program by entering:

```
RUN"ISAMX
```

The program then asks you to enter information about the data file. Respond as indicated.

```
Enter Data File Name (Include Drive ID): a:stock.rec
Enter Data File Record Length: 16
```

The above sequence creates and opens a data file named stock.rec with a record length of 16 bytes. The following menu is then displayed:



Beneath the menu the following prompt will appear:

Enter Function Code or E to End (Enter ? for function menu);

If you enter **E** the tutorial program will terminate and return you to MS-DOS. If you enter **?** the Select Function menu is redisplayed. Otherwise you can enter any of the two-letter functions (in upper or lower case) listed on the menu followed by **CR** . The program responds with the full title of the function and prompts you for each of the variables that the function requires. The function is then executed.

Chapter 5 describes each of the functions in detail. For use with the tutorial program, however, the following summaries are adequate. Note that where two mnemonics are given (for example; RK, SK), the first applies to a primary index and the second to a secondary index.

RK, SK - Read Key

Accesses a data record whose key matches the specified key exactly.

RG, SG - Read Generic

Accesses a data record whose key is equal to or next greater than a specified key.

RN, SN - Read Next

Accesses a data record corresponding to the next sequential key in the index (that is, the key following the last key value entered).

RP, RN - Read Previous

Accesses the data record corresponding to the next lower key in the index file (that is, the key that precedes the last key value entered).

RB, SB - Read Beginning

Accesses the data record corresponding to the first key in the index.

RE, SE - Read End

Accesses the data record corresponding to the last key in the index.

WA, SA - Write Add

Inserts a new key into the index file. The WA function also writes the corresponding data record. The SA function sets up the pointer to the already existing data record.

DR - Delete Record

Deletes the data record that corresponds to the specified key.

KD, SD - Key Delete

Deletes a key from an index file but does not delete the corresponding data record.

MS - ISAM Status

Returns status information about a specific index file.

OO - Open Existing File

Opens an index file that already exists and assigns a DCB number to it.

OC - Create New File

Creates and opens a new index file, assigns a DCB number to it and defines the file to be either a unique or duplicate key type.

OF - Create and/or Open File

Creates and opens a new index file, or opens the file if it already exists. It also assigns a DCB number to the file and specifies the file to be either duplicate or unique key type.

CL - Close File

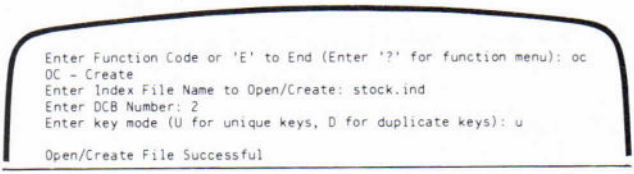
Closes an index file.

X - Examine Variables

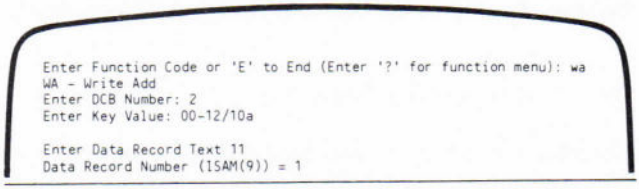
Displays the values of the ISAM and ISAM$ array variables that the tutorial program uses to communicate with ISAM. For details of ISAM variables refer to Chapter 3.

CREATING THE PRIMARY INDEX

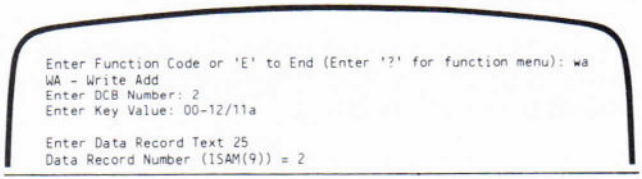
Now build your primary index by responding to the prompts as follows:



The above sequence creates and opens an index file named stock.ind that will contain unique keys only, and assigns DCB 2 to it.



The above sequence inserts key value 00-12/10a into the index file and enters stock level 11 in the first available data record, in this case record number 1. ISAM(9) is the variable in which the ISAM subroutine returns the data record number to ISAMX. Continue by entering the next key:



The above sequence inserts key value 00-12/11a into the index file and enters stock level 25 in the first available data record, in this case record number 2.

```
Enter Function Code or 'E' to End (Enter '?' for function menu): wa
WA - Write Add
Enter DCB Number: 2
Enter Key Value: 00-12/11b

Enter Data Record Text 15
Data Record Number (ISAM(9)) = 3
```

The above sequence inserts key value 00-12/11b into the index file and enters stock level 15 in the first available data record, in this case record number 3.

Enter the remaining keys listed on page 4-1 using successive Write Add functions. On completion the stock levels for each of the items will be contained in the data records numbered 1 to 15.

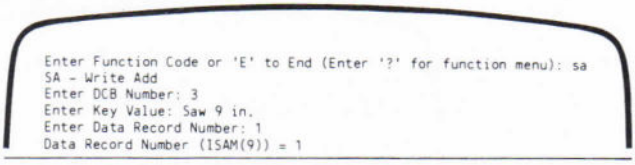
CREATING THE SECONDARY INDEX

To add the secondary index file to the file structure respond to the prompts as follows:

```
Enter Function Code or 'E' to End (Enter '?' for function menu): oc
OC - Create
Enter Index File Name to Open/Create: desc.ind
Enter DCB Number: 3
Enter key mode (U for unique keys, D for duplicate keys): d

Open/Create File Successful
```

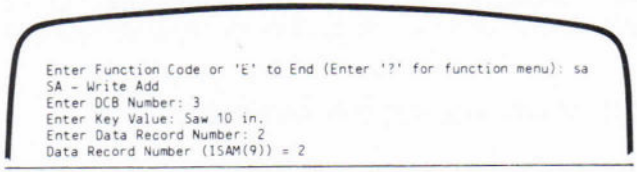
The above sequence creates and opens a secondary index file named desc.ind that will accept duplicate keys.

Create your secondary index using SA (Secondary Add) functions as follows:



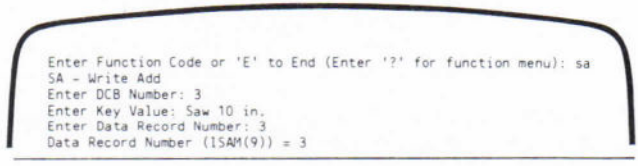
The above sequence inserts the key "Saw 9 in." into the secondary index as a key to record number 1. The last line confirms the operation.

Continue as follows:



The above sequence inserts the key "Saw 10 in." into the secondary index as a key to record number 2.

Continue as follows:



The above sequence inserts the key "Saw 10 in." into the secondary index as a key to record number 3. Note that "Saw 10 in." is now duplicated in the index.

Insert the remaining keys in the same manner. After doing so you will have completed the entry of your initial file structure.

ACCESSING AND UPDATING YOUR FILE STRUCTURE

This section describes how you can use the ISAMX program to retrieve data records and how to remove and add new keys to your structure. Respond to the prompts as follows:

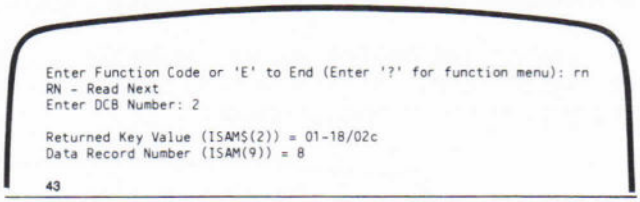
```
Enter Function Code or 'E' to End (Enter '?' for function menu): rk
RK - Read Key
Enter DCB Number: 2
Enter Key Value: 01-18/02c.

Returned Key Value (ISAM$(2)) = 01-18/02c
Data Record Number (ISAM(9)) = 8
43
```

The above sequence accesses the data record whose key is 01-18/02c. ISAM(9) is the variable in which the ISAM subroutine returns the data record number. "43" is the actual content of the data record, indicating that the stock level of item 0-18/02c is 43.

Supposing you wish to examine the stock level of all spanners. Assuming you know that the part numbers of all spanners begins with 01-18..., you can set the start of the browse using the Read Generic function, then examine each key in turn using the Read Next (RN) function as follows:

```
Enter Function Code or 'E' to End (Enter '?' for function menu): rg
RG - Read Generic
Enter DCB Number: 2
Enter Key Value: 01-18/00a

Returned Key Value (ISAM$(2)) = 01-18/02a
Data Record Number (ISAM(9)) = 6

17
```

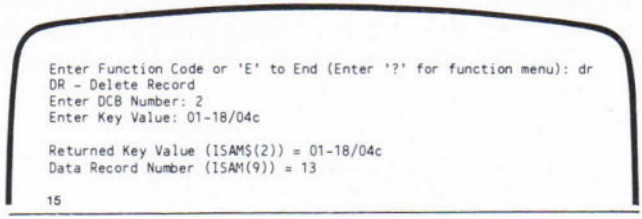
```
Enter Function Code or 'E' to End (Enter '?' for function menu): rn
RN - Read Next
Enter DCB Number: 2
Returned Key Value (ISAM$(2)) = 01-18/02b
Data Record Number (ISAM(9)) = 7

25
```



etc.

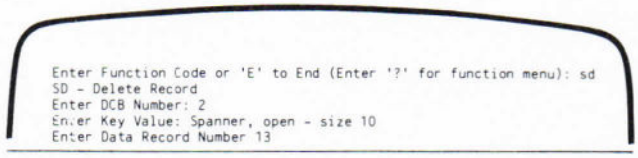
You can similarly set up a browse starting from the first key in the index (00-12/10a) using the Read Beginning (RB) function, followed by successive Read Next (RN) functions. Or start a browse from the end of the index using a Read End (RE) function followed by successive Read Previous (RP) functions.

To delete a record you need to use the Delete Record (DR) function as follows:



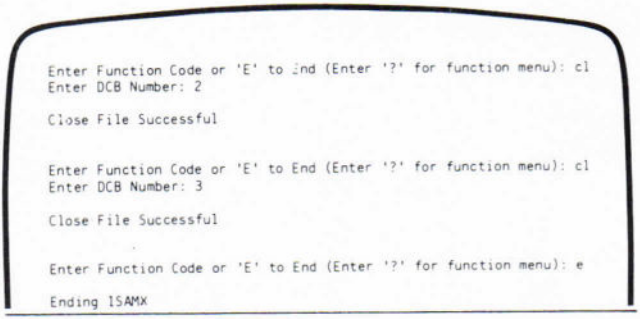
The above sequence deletes the key 01-18/04c from the index file and the corresponding record number from the data file. Data record number 13 is now placed on the deleted record stack for reclamation by a future Write Add (WA) function. Note, however, that the secondary index still contains a reference to this record.

Use the Key Delete (KD) or Secondary Key Delete (SD) function to remove the key as follows:



Note that the data record number is also required because desc.ind is a duplicate key type.

Finally, close both your index files using the Close File function, and close the data file by exiting ISAMX:



Now try some ideas of your own.

5. USING ISAM IN A GW-BASIC PROGRAM

ABOUT THIS CHAPTER

This chapter describes how to use the functions of ISAM within a GW-BASIC program.

INDEX

USING ISAM WITH GW-BASIC

ISAM is a subroutine to your GW-BASIC program. The subroutine itself is contained on your ISAM diskette in the file named isam.sav. The GW-BASIC interface to the subroutine is contained in isam.bas. It is invoked using the statement GOSUB 60000.

A GW-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:

- You must enter GW-BASIC with the following command

  GWBASIC

- ISAM source code must be included in the program; for example, having entered your GW-BASIC program you should then add the following two immediate statements:

  MERGE "ISAM.BAS" This statement appends the ISAM GW-BASIC interface.

  SAVE "B:MYPROG" This statement saves your GW-BASIC program with the ISAM subroutine appended to it on the diskette inserted in drive B in the file named MYPROG.

- no sequence number in the calling program must be greater than 60000.

- no variables in the BASIC program should start with "ISAM".

- two arrays are required, named

  ISAM$ and ISAM

These arrays must only be used for communicating with ISAM. It is not necessary to dimension them as only the elements with subscripts from 1 to 9 are used.

Any program that uses an ISAM file structure must perform the functions outlined in the following table:

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.

ISAM UTILITY FUNCTIONS

ISAM STATUS (MS)

The ISAM Status function returns information about an index file to the user program. This information comprises

- the number of keys in the index file (and hence the number of active data records)
- the number of unreclaimed deleted records in the associated data file
- a flag that indicates whether the index file is duplicate or unique type

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 in the following table:

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

▶

STEP	OPERATION
8	Close the data file using a GW-BASIC CLOSE statement
9	Close the index file using the Close File function

Example

DISPLAY	COMMENTS
100 OPEN"R",15,"DFILE",128 110 FIELD 15,128 AS A$ 120 ISAM(1) = 1 130 ISAM$(1) = "OO" 140 ISAM$(4) = "INDEX" 150 GOSUB 60000 160 IF ISAM(8)< > 0 THEN GOTO 900 • • •	Statements 100 and 110 open the data file named DFILE and partition the buffer. Statements 120 to 160 open the index file named INDEX and assign DCB 1 to it. The index file named INDEX has previously, been created using the 'OC' function.
200 ISAM(1) = 1 210 ISAM$(1) = "MS" 220 GOSUB 60000 230 IF ISAM(8)< > 0 THEN GOTO 1000	Statements 200 to 230 perform the ISAM Status function on the index file.

DISPLAY	COMMENTS
• • •	
500 CLOSE 15 510 ISAM(1) = 1 520 ISAM$(1) = "CL" 530 GOSUB 60000 540 IF ISAM(8)< > 0 THEN GOTO 1100	Statement 500 closes the data file. Statements 510 to 540 close the index file.

OPENING AND CLOSING ISAM FILES

CREATE FILE (OC)

The Create File function creates and opens an index file. You need to specify the file name 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 null ("") for unique before invoking the ISAM subroutine. The default value is null.

The following variables must be passed to ISAM:

- ISAM(1) - file DCB number
- ISAM$(1) - function code (OC)

- 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: 00, 41, 71 and 81.

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

STEP	OPERATION
1	Specify the DCB number
2	Set the function code to "OC"
3	Specify the file name
4	Set (or not) the Duplicate Key flag (D or null)
5	Call the ISAM subroutine
6	Check the return code

Example

DISPLAY	COMMENTS
100 ISAM(1) = 4 110 ISAM$(1) = "OC" 120 ISAM$(4) = "IFILE1" 130 ISAM$(6) = "D" 140 GOSUB 60000 150 IF ISAM(8) < > 0 THEN GOTO 600	Statements 100 to 150 create and open an index file named IFILE1, specify duplicate keys, and assign DCB 4 to the file.

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$(1) - function code (OO)
- ISAM$(4) - index file name

The following variable will be returned:

- ISAM(8) - return code

Possible return codes are: 00, 41 and 71.

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

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

Example

DISPLAY	COMMENTS
100 ISAM(1) = 2 110 ISAM$(1) = "OO" 120 ISAM$(4) = "IFILE" 130 GOSUB 60000 140 IF ISAM(8) < > 0 THEN GOTO 600	Statements 100 to 140 open the index file named IFILE - assuming it has already been created - and assign DCB 2 to it.

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. You need to specify the index file name and a DCB number. All subsequent operations on the file will only require the index file number.

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 null ("") for unique keys. The default value is null.

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 null)

The following variable will be returned:

- ISAM(8) - return code

Possible return codes are: 00, 41, 71 and 81.

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

STEP	OPERATION
1	Specify the DCB number
2	Set the function code to "OF"
3	Specify the file name
4	Set (or not) the Duplicate Key flag (D or null)
5	Call the ISAM subroutine
6	Check the return code

Example

DISPLAY	COMMENTS
100 ISAM(1) = 3 110 ISAM$(1) = "OF" 120 ISAM$(4) = "IFILE" 130 ISAM$(6) = "U" 140 GOSUB 60000 150 IF ISAM(8) < > 0 THEN GOTO 600	Statements 100 to 150 create and open an index file named IFILE1, which can only contain unique keys. Statement 100 assigns DCB 3 to the file.

CLOSE FILE (CL)

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

If the DCB number specified 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: 00, 41 and 71.

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

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

Example

DISPLAY	COMMENTS
200 ISAM(1) = 1 210 ISAM$(1) = "CL" 220 GOSUB 60000 230 IF ISAM(8) < > 0 THEN GOTO 600	Statements 200 to 230 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: 00, 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 in the following table:

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

STEP	OPERATION
9	Check the return code
10	Read the returned record using a GW-BASIC GET statement
11	Close the data file using a GW-BASIC CLOSE statement
12	Close the index file using the Close File function

Example

DISPLAY	COMMENTS
100 OPEN"R",15,"B:DATAFILE", 128 110 FIELD 15,128 AS A$ 120 ISAM(1) = 1 130 ISAM$(1) = "OO" 140 ISAM$(4) = "INDEXFILE" 150 GOSUB 60000 160 IF ISAM(8) < > 0 THEN GOTO 600 • • •	Statements 100 and 110 open the random data file on the diskette in drive B and partition the buffer. Statements 120 to 160 open the index file named INDEXFILE and assign DCB 1 to it.

DISPLAY	COMMENTS
200 ISAM(1) = 1 210 ISAM$(1) = "RK" 220 ISAM$(2) = "SMITH" 225 ISAM(5) = DATAREC% 230 GOSUB 60000 240 IF ISAM(8) < > 0 THEN GOTO 500 250 GET 15,ISAM(9) • • •	Statements 200 to 500 read the data record whose key is SMITH. Statement 225 assumes a duplicate key type file. It reads the data record from a variable list.
900 CLOSE 15 910 ISAM(1) = 1 920 ISAM$(1) = "CL" 930 GOSUB 60000 940 IF ISAM(8) < > 0 THEN GOTO 990	Statement 900 closes the data file. Statements 910 to 940 close the index file.

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 Generic function will always return the record number corresponding to the first occurrence 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 GW-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: 00, 41 and 51.

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

STEP	OPERATION
1	Open the data file using a GW-BASIC OPEN statement
2	Partition the data file buffer into fields using a GW-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)
7	Call the ISAM subroutine
8	Check the return code
9	Read the record using a GW-BASIC GET statement
10	Close the data file using a GW-BASIC CLOSE statement
11	Close the index file using the Close File function

Example

DISPLAY	COMMENTS
100 OPEN"R",14,"DFILE",128 110 FIELD 14,128 AS A$ 120 ISAM(1) = 4 130 ISAM$(1) = "OO" 140 ISAM$(4) = "IFILE1" 150 GOSUB 60000 160 IF ISAM(8) < >0 THEN GOTO 700 • • •	Statements 100 and 110 open the data file named DFILE and partition the buffer.
200 ISAM(1) = 4 210 ISAM$(1) = "SG" 220 ISAM$(2) = "1000" 225 ISAM(5) = DATAREC% 230 GOSUB 60000 240 IF ISAM(8) < >0 THEN GOTO 500 250 GET 14,ISAM(9) • • •	Statements 200 to 250 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 250 retrieves the record whose key is equal to or greater than 1000. The retrieved key value is returned in ISAM$(2).
900 CLOSE 14 910 ISAM(1) = 4 920 ISAM$(1) = "CL" 930 GOSUB 60000 940 IF ISAM(8) < >0 THEN GOTO 990	Statement 900 closes the data file. Statements 910 to 940 close the index file.

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 is established by the last Read Key, Read Generic, Read Next or Read Previous function or, if the first ISAM function performed on a file is a Read Next, the position defaults to the first key in the index. Do not use the Write Add, Key Delete or Delete Record functions to establish the starting position, as the results are unpredictable.

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 successive Read Next functions 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 GW-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: 00, 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 in the following table:

STEP	OPERATION
1	Open the data file using a GW-BASIC OPEN statement
2	Partition the data file buffer into fields using a GW-BASIC FIELD statement
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 GW-BASIC GET statement
9	Close the data file using a GW-BASIC CLOSE statement
10	Close the index file using the Close File function

Example

DISPLAY	COMMENTS
110 OPEN"R",13,"DFILE",128 110 FIELD 13,128 AS A$ 120 ISAM(1) = 2 130 ISAM$(1) = "OO" 140 ISAM$(4) = "INDEXFILE" 150 GOSUB 60000 160 IF ISAM(8)< > 0 THEN GOTO 600 • • •	Statements 100 and 110 open the data file named DFILE and partition the buffer. Statements 120 to 160 open the index file named INDEX-FILE and assign DCB 2 to it.
200 ISAM(1) = 2 210 ISAM$(1) = "RN" 220 GOSUB 60000 230 IF ISAM(8) < > 0 THEN GOTO 700 240 GET 13,ISAM(9) • • •	Statement 200 to 240 retieve the record whose key is next to greater than the previous key value. State-ment 240 reads the data into the file buffer. The key value is returned ISAM$(2).
300 CLOSE 13 310 ISAM(1) = 2 320 ISAM$(1) = "CL" 330 GOSUB 60000 340 IF ISAM(8)< > 0 THEN GOTO 800	Statement 300 close the data file. Statements 310 to 340 close the in-dex file.

READ PREVIOUS (RP, SP)

The Read Previous function reads the previous sequential key in the file and returns the key value and the corresponding data record number. RP and SP are functionally identical.

The Read Previous function allows records to be read in reverse sequential key order. The starting position is established by the last Read Key, Read Generic, Read Next or Read Previous function or, if the first ISAM function performed on a file is a Read Previous, the position defaults to the last key in the index. Do not use the Write Add, Key Delete or Delete Record functions to establish the starting position, as the results are unpredictable.

If the previous read operation used the first key in the index, the Read Previous function will issue a "not found" error - return code 51.

If duplicate keys exist in the index then successive Read Previous functions will return the record numbers associated with the key value in reverse record number sequence.

ISAM returns the record number in ISAM(9). The data in that record can therefore be retrieved using a GW-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 (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: 00, 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 in the following table:

STEP	OPERATION
1	Open the data file using a GW-BASIC OPEN statement
2	Partition the data file buffer into fields using a GW-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 GW-BASIC GET statement
9	Close the data file using a GW-BASIC CLOSE statement
10	Close the index file using the Close File function

Example

DISPLAY	COMMENTS
100 OPEN"R",12,"EMPFILE",128 110 FIELD 12,128 AS A$ 120 ISAM(1) = 1 130 ISAM$(1) = "OO" 140 ISAM$(4) = "EMPIND" 150 GOSUB 60000 160 IF ISAM(8) < >0 THEN GOTO 800 • • •	Statements 100 and 110 open the data file named EMPFILE and partition the buffer. Statements 120 to 160 open the index file named EMPIND and assign DCB 1 to it.
200 ISAM(1) = 1 210 ISAM$(1) = "SP" 220 GOSUB 60000 230 IF ISAM(8) < >0 THEN GOTO 900 240 GET 12,ISAM(9) • • •	Statements 200 to 240 perform the Read Previous function; that is; the data record retrieved has a key value that is next smaller than the last key value.
400 CLOSE 12 410 ISAM(1) = 1 420 ISAM$(1) = "CL" 430 GOSUB 60000 440 IF ISAM(8) < >0 THEN GOTO 950	Statement 400 closes the data file. Statements 410 to 440 close the index file.

READ BEGINNING (RB, SB)

The Read Beginning function returns the key value and record number associated with the first key in the index. RB and SB are functionally identical.

The Read Beginning function is particularly useful for starting or resetting a browse to the beginning of the index. Return code 51 is issued if there are no keys in the file.

The following variables must be passed to ISAM:

- ISAM(1) - index file number
- ISAM$(1) - function code (RB or SB)
- ISAM$(6) - Duplicate Key flag

The following variables are returned:

- ISAM(8) - return code
- ISAM(9) - data record number
- ISAM$(2) - key value.

Possible return codes are: 00, 41 and 51

To retrieve the data record corresponding to the first record in the index file you should therefore follow the sequence indicated in the following table:

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

STEP	OPERATION
3	Open the index file using the File Open function
4	Specify the DCB number
5	Specify the function code as "RB" or "SB"
8	Call the ISAM subroutine
9	Check the return code
10	Retrieve the data record using a GW-BASIC GET statement
11	Close the data file using a GW-BASIC CLOSE statement
12	Close the index file using the Close File function.

Example

DISPLAY	COMMENTS
100 OPEN"R",11, "CUSTOMERS",128 110 FIELD 11,128 AS A$ 120 ISAM(1) = 2 130 ISAM$(1) = "OO" 140 ISAM$(4) = "CUSTOMER INDEX" 150 GOSUB 60000 160 IF ISAM(8)< > 0 THEN GOTO 900 • • •	Statements 100 and 110 open the data file named CUSTOMERS and partition the buffer. Statements 120 to 160 open a duplicate key type index named CUSTOMERINDEX and assign DCB 2 to it.
300 ISAM(1) = 2 310 ISAM$(1) = "RB" 320 GOSUB 60000 330 IF ISAM(8)< > 0 THEN GOTO 900 340 GET 11,ISAM(9) • • •	Statements 300 to 340 retrieve the record whose key is first in the index file.
500 CLOSE 11 510 ISAM(1) = 2 520 ISAM$(1) = "CL" 530 GOSUB 60000 540 IF ISAM(8)< > 0 THEN GOTO 950	Statement 500 closes the data file. Statements 510 to 540 close the index file.

READ END (RE, SE)

The Read End function returns the key value and record number associated with the last key in the index. RE and SE are functionally identical.

The Read End function is particularly useful for starting or resetting a browse to the end of the file. Return code 51 is issued if there are no keys in the file.

The following variables must be passed to ISAM:

- ISAM(1) - index file number
- ISAM$(1) - function code (RE or SE)

The following variables are returned:

- ISAM(8) - return code
- ISAM(9) - data record number
- ISAM$(2) - key value.

Possible return codes are: 00, 41 and 51.

To retrieve the data record corresponding to the last record in the index file you should therefore follow the sequence indicated in the following table:

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

STEP	OPERATION
3	Open the index file using the File Open function
4	Specify the DCB number
5	.Specify the function code as "RE" or "SE"
7	Call the ISAM subroutine
8	Check the return code
9	Retrieve the record via a GW-BASIC GET statement
10	Close the data file using a GW-BASIC CLOSE statement
11	Close the index file using the Close File function.

Example

DISPLAY	COMMENTS
100 OPEN"R",11, "CUSTOMERS",128 110 FIELD 11,128 AS A$ 120 ISAM(1) = 2 130 ISAM$(1) = "OO" 140 ISAM$(4) = "CUSTOMER— INDEX" 150 GOSUB 60000 160 IF ISAM(8) < > 0 THEN GOTO 900 • • •	Statements 100 and 110 open the data file named CUSTOMERS and partition the buffer. Statements 120 to 160 open the index file named CUSTOMERINDEX and assign DCB 2 to it.
300 ISAM(1) = 2 310 ISAM$(1) = "RE" 320 GOSUB 60000 330 IF ISAM(8)< > 0 THEN GOTO 900 340 GET 11,ISAM(9) • • •	Statements 300 to 340 retrieve the record whose key is first in the index.
500 CLOSE 11 510 ISAM(1) = 2 520 ISAM$(1) = "CL" 530 GOSUB 60000 540 IF ISAM(8)< > 0 THEN GOTO 950	Statement 500 closes the data file. Statements 510 to 540 close the index file.

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 GW-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: 00, 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 GW-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 GW-BASIC OPEN statement
2	Partition the data file buffer into fields using a GW-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

STEP	OPERATION
9	Write the data record with a GW-BASIC PUT statement
10	Close the data file using a GW-BASIC CLOSE statement
11	Close the index file using the close file function

Example

DISPLAY	COMMENTS
100 OPEN"R",15,"DFILE-B",128 110 FIELD 15,128 AS A$ 120 ISAM(1) = 1 130 ISAM$(1) = "OO" 140 ISAM$(4) = "INDEX-B 150 GOSUB 60000 160 IF ISAM(8) < > 0 THEN GOTO 900 • • •	Statements 100 and 110 open the data file named DFILE-B and partition the buffer. Statements 120 to 160 open the index file named INDEX-B and assign DCB 1 to it.

DISPLAY	COMMENTS
200 ISAM(1) = 1 210 ISAM$(1) = "WA" 220 ISAM$(2) = "JONES" 230 GOSUB 60000 240 IF ISAM(8) < > 0 THEN GOTO 950 250 PUT 15,ISAM(9) • • •	Statements 200 to 250 write the data held in the data file buffer to the data record whose key is JONES.
300 CLOSE 15 310 ISAM(1) = 1 320 ISAM$(1, = "CL" 330 GOSUB 60000 340 IF ISAM(8) < > 0 THEN GOTO 1000	Statement 300 close the data file. Statements 310 to 340 close the index file.

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

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

STEP	OPERATION
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
10	Close the data file using a GW-BASIC CLOSE statement
11	Close the index file using the Close File function

Example

DISPLAY	COMMENTS
100 OPEN"R",15,"DFILE-B",128 110 FIELD 15,128 AS A$ 120 ISAM(1) = 2 130 ISAM$(1) = "OO" 140 ISAM$(4) = "INDEX-BS" 150 GOSUB 60000 160 IF ISAM(8) < > 0 THEN GOTO 700 • • •	Statements 100 and 110 open the data file named DFILE-B and partition the buffer. Statements 120 to 160 open the secondary index file named INDEX-BS and assign DCB 2 to it.
200 ISAM(1) = 2 210 ISAM(5) = RECNO% 220 ISAM$(1) = "SA" 230 ISAM$(2) = KEY$ 240 GOSUB 60000 250 IF ISAM(8) < > 0 THEN GOTO 750 • • •	Statements 200 to 250 assign a secondary key to a record, where both the key and the record are input from a variable list.
300 CLOSE 15 310 ISAM(1) = 2 320 ISAM$(1) = "CL" 330 GOSUB 60000 340 IF ISAM(8) < > 0 THEN GOTO 800	Statement 300 close the data file. Statements 310 to 340 close the index file.

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.

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 occurrence of the key will be deleted.

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: 00, 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 GW-BASIC OPEN statement
2	Partition the data file buffer into fields using a GW-BASIC FIELD statement
3	Open the index file using the File Open function
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
10	Close the data file using a GW-BASIC CLOSE statement
11	Close the index file using the Close File function

Example

DISPLAY	COMMENTS
300 OPEN"R",15,"DATA-1",128 310 FIELD 15,128 AS A$ 320 ISAM(1) = 1 330 ISAM$(1) = "OO" 340 ISAM$(4) = "INDEX-1" 350 GOSUB 60000 360 IF ISAM(8) < > 0 THEN GOTO 900 • • •	Statements 300 and 310 open the data file named DATA-1 and partition the buffer. Statements 320 to 360 open the index file named INDEX-1 and assign DCB 1 to it.
400 ISAM(1) = 1 410 ISAM$(1) = "KD" 420 ISAM$(2) = "1234" 425 ISAM(5) = DATAREC% 430 GOSUB 60000 440 IF ISAM(8) < > 0 THEN GOTO 950 • • •	Statements 400 to 440 delete from the index file the key whose value is 1234. Statement 425 is only necessary if INDEX-1 is a duplicate key type file.
500 CLOSE 15 510 ISAM(1) = 1 520 ISAM$(1) = "CL" 530 GOSUB 60000 540 IF ISAM(8) < > 0 THEN GOTO 1000	Statement 500 cloes the data file. Statements 510 to 540 close the index file.

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 reclamation 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 re-used.

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: 00, 41, 51 and 71.

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

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

Example

DISPLAY	COMMENTS
100 OPEN"R",12,"RECFILE",128 110 FIELD 12,128 AS A$ 120 ISAM(1) = 3 130 ISAM$(1) = "OO" 140 ISAM$(4) = "IFILE-B" 150 GOSUB 60000 160 IF ISAM(8)< > 0 THEN GOTO 600 • • •	Statements 100 and 110 open the data file named RECFILE and partition the buffer. Statements 120 to 160 open the index file named IFILE-B and assign DCB 3 to it.
200 ISAM(1) = 3 210 ISAM$(1) = "DR" 220 ISAM$(2) = "SMITH" 225 ISAM(5) = DATAREC% 230 GOSUB 60000 240 IF ISAM(8)< > 0 THEN GOTO 650 • • •	Statements 200 to 240 delete from the data file the record whose key is SMITH. Statement 225 reads from the data record number from a variable list - it is only necassary if the index file is a duplicate key type.
300 CLOSE 12 310 ISAM(1) = 3 320 ISAM$(1) = "CL" 330 GOSUB 60000 340 IF ISAM(8)< > 0 THEN GOTO 800	Statement 300 close the data file. Statements 310 to 340 close the index file.

6. ISAM FILE DUMP (ISAMD)

ABOUT THIS CHAPTER

This chaper describes thow you can examine your ISAM files using the ISAM file dump program ISAMD.

INDEX

THE ISAM FILE DUMP PROGRAM

The ISAM File Dump program (ISAMD) enables you to examine your ISAM files and thereby ensure your programs are working correctly. It can be used to examine data or index files on the screen, printer or both.

The program displays data files in record number sequence, one record beneath the other on the screen and/or printer.

Index files are displayed one record at a time on the screen with an option to select the next record you wish to examine.

Before using the program you must have initialized MS-DOS and have on active disks, GW-BASIC, the ISAM program files and your ISAM index and data files.

SAMPLE FILE DUMP SESSION

Consider an extended version of the file structure you created in Chapter 4.

Invoke the program by entering:

```
GWBASIC/S:256
RUN"ISAMD
```

The following series of prompts appear on the screen. Respond as indicated. Note that all responses must be followed by CR

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 (Include drive ID): a:desc.ind

Enter Data File name (Include drive ID): a:stock.rec

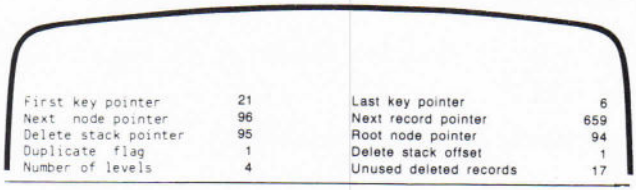
Enter data record length: 16

Do you want to display the Index File? (Y/N): y

The above sequence specifies a file structure made up of an index file named desc.ind and a data file named stock.rec. The data file has a record length of 16 bytes. The sequence also specifies the index file to be displayed on the screen only.

DISPLAYING THE HEADER RECORD

Following the above sequence the header record is always displayed as follows:



First key pointer	21	Last key pointer	6
Next node pointer	96	Next record pointer	659
Delete stack pointer	95	Root node pointer	94
Duplicate flag	1	Delete stack offset	1
Number of levels	4	Unused deleted records	17

The meaning of each of these entries is described below.

ENTRY	MEANING...
First key pointer	the number of the leaf node record that contains the lowest key value.
Last key pointer	the number of the leaf node record that contains the highest key value.

ENTRY	MEANING...
Next node pointer	the number of the first empty record in the index file. In this case the index uses the first 95 records, the 96th being the next available.
Next record pointer	the number of the first unused record in the data file; that is, the record immediately following the highest record used. Subsequent Write Add functions will first use up the records on the delete stack, after which the next record will be used.
Delete stack pointer	the number of the record in the index file that contains the delete stack.
Root node pointer	the index file record number that contains the root node.
Duplicate flag	a value that indicates whether the file is a duplicate key type or not. Its value will be: 1 - for a duplicate key type index 0 - for a unique key type index
Delete stack offset	the offset of the delete stack.
Number of levels	the number of levels in the index. The lowest level (level 0) always contains the leaf node records. The highest level is the root node.
Unused deleted records	the number of records on the delete stack. These records will be used by subsequent Write Add functions.

Following the display of the header record you are prompted:

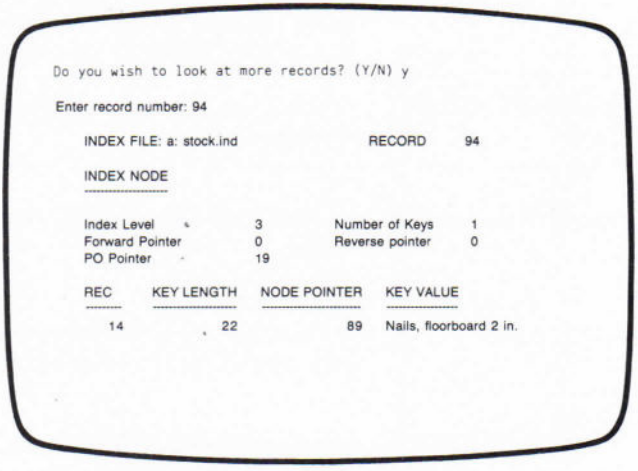
Do you wish to look at more records? (Y/N): y

An **n** answer displays a message that prompts you to look at the data file. A **y** answer issues the following prompt:

Enter record number:

DISPLAYING THE ROOT NODE

To display the root node record respond to the prompt with the root node record number as indicated in the display of the header record:



The entries in the above display have the following meaning:

ENTRY	MEANING...
Index level	the level in the tree above the leaf node records. The leaf node records all reside at level 0. In this case a value of three indicates that there are two levels of index between the leaf nodes and the current node.
Number of keys	the number of keys in the node. In this case the number is 1, indicating two level 2 index nodes; the lower one accessed via the PO pointer, and the higher one via the key.

ENTRY	MEANING...
Forward pointer	the index file record number at the same level as the current node containing the next higher node of keys; that is, the elder brother of the current node. As there can only be one root node, this value must be zero for the root node record.
Reverse pointer	the index file record number at the same level as current node containing the next lower node of keys; that is, the younger brother of the current node. As there can only be one root node, this value must be zero for the root node record.
PO pointer	for the root node this points to the record at the next lower level in the tree (level 2 in this case) that contains the lowest key values. That is, it points to the youngest child of the root node. At other levels in the index, however, the PO pointer has a more complex definition (see later).

The display also shows for each key the following information:

Key Length	the length of the key, in this case 22 characters.
Node Pointer	the number of the index file record that contains the index node addressed by this key. This record will be at the next lower level, in this case level 2.
Record Number	the data file record number containing the data record that corresponds to the key value.
Key Value	a key value used to determine which keys are accessed via this key. All keys greater than this value will be accessed via the corresponding node pointer. Had the node contained more than one key, all keys greater than this key value, and less than or equal to the next key value would be accessed via this key.

The following figure illustrates the structure.

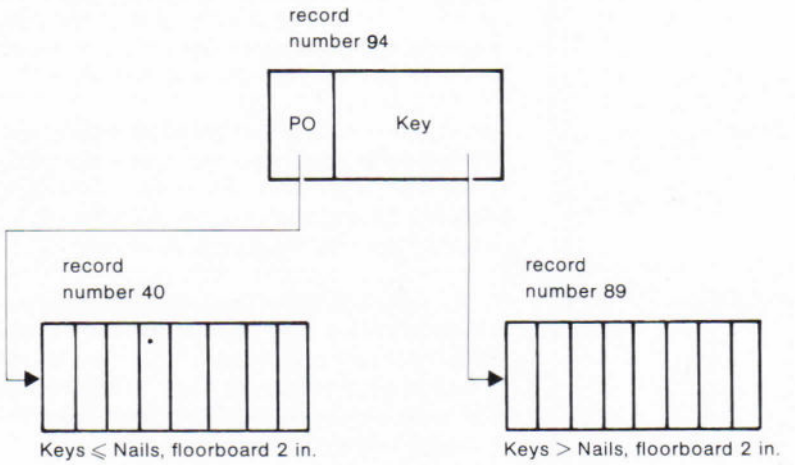


Fig. 6-1 A Root Node

DISPLAYING INDEX NODES

Following the display of the root node is a prompt that asks you if you wish to look at more records. Respond as follows and the corresponding index record will be displayed.

```
INDEX FILE:  a:stock.ind                              RECORD # 40

INDEX NODE
----------

Index level          2              Number of keys          7
Forward pointer      89             Reverse pointer         0
PO Pointer           19

REC #   KEY LENGTH   NODE POINTER    KEY VALUE
-----   ----------   ------------    ---------
  72           12              5    Fork, garden
 107           20             27    Hack-saw blade 5 in.
   .
   .
   .

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

Enter record number 5


INDEX FILE:  a:stock.ind                              RECORD #  5

INDEX NODE
----------

Index level          1              Number of keys          5
Forward pointer      27             Reverse pointer        19
PO Pointer           17

REC #   KEY LENGTH   NODE POINTER    KEY VALUE
-----   ----------   ------------    ---------
  53           12             22    Gas, camping
   .
   .
   .
```

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

The displays show two index nodes: record 40 at level 2; and its child - record 5 - at level 3. The following figure shows the structure:

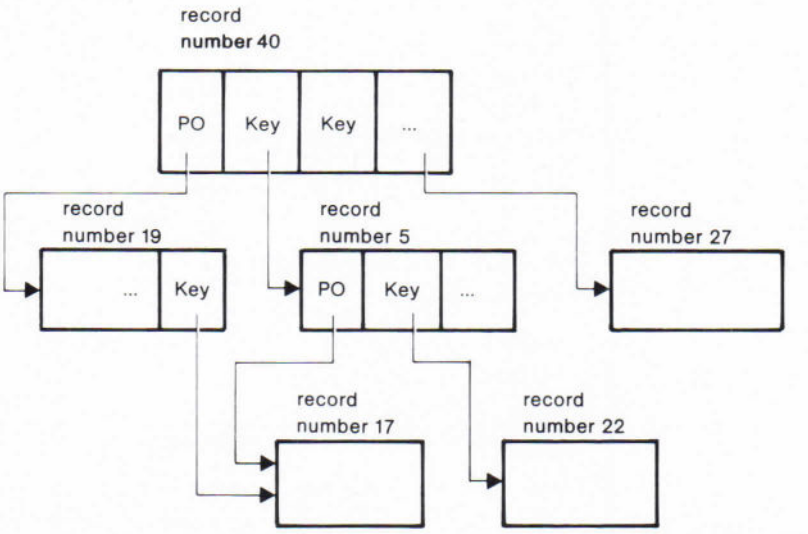


Fig. 6-2 Index Nodes

The P0 pointer for record 40 points to the record at the next level down that contains the lowest keys; that is, the youngest child of record 40. The P0 pointer for record 5, however, points to the eldest child of record 19. That is, where an index record has a younger brother (Reverse Pointer ⟨ ⟩ 0), the P0 pointer contains the record number of the younger brother's eldest child. If the index record does not have a younger brother (Reverse Pointer = 0),then the P0 pointer points to the record's own youngest child.

All other entries are as for the root node.

DISPLAYING A LEAF NODE

If you respond y to the last prompt the message that asks you to specify another record for display is again displayed. Respond as follows and the corresponding index record will be displayed. In this case the record contains a leaf node.

```
Enter record number: 22

    INDEX FILE: a:stock.ind                              RECORD # 22

    LEAF NODE
    ---------

    Index level          0          Number of keys          8
    Forward pointer      31         Reverse pointer         10

    REC #     KEY LENGTH   KEY VALUE
    -----     ----------   ---------

      53            18    Gas, Camping
       :
Do you wish to look at more records? (Y/N):
```

The display illustrates a leaf node. The Forward and Reverse Pointers link the record to its elder and younger brothers which provides the mechanism for accessing all keys sequentially.

DISPLAYING THE DELETE STACK

If you respond y to the last prompt, the message that asks you to specify another record for display is again displayed. Respond by entering the record number of the delete stack as indicated in the header record. If you have forgotten this value, redisplay the header record (record number 1).

```
Enter record number to display or 'E' to end: 95

    INDEX RECORD # 95
    NODE
    ----

    Index Level          = -1
    Forward Pointer      = 0
    Reverse Pointer      = 0

    Deleted Records
    ---------------
    4
    10
    132
    398
     .
     .
     .

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

The display simply lists the data records that have not yet been deleted.

DISPLAYING THE DATA FILE

Respond to the prompts as follows:

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

Do you want to display the Data File?(Y/N): y
```

The above sequence enables you to examine the data file. The entire data file is scrolled on the screen. Each data record is displayed as follows:

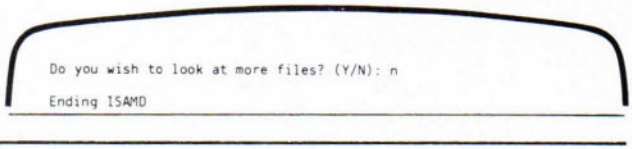
```
Data Record # 120
-----------------
"85"
```

TO EXIT THE PROGRAM

After terminating the data file display, you can exit the program as follows:

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

Ending ISAMD
```

The above sequence exits the program and returns control to GW-BASIC. Had you answered y to the prompt you would have been asked again for the index and data files you wish to examine.



A. ISAM VARIABLES

ABOUT THIS APPENDIX

This appendix provides a quick reference of the variables that are passed between your GW-BASIC program and the ISAM subroutine.

INDEX

VARIABLES PASSED TO ISAM

VARIABLE	DESCRIPTION
ISAM(1)	Index file number
ISAM(2)	Unused
ISAM(3)	Unused
ISAM(4)	Unused
ISAM(5)	Data record number
ISAM(6)	Unused
ISAM(7)	Unused
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 Key flag
ISAM$(7)	Unused
ISAM$(8)	Unused
ISAM$(9)	Unused.

VARIABLES RETURNED FROM ISAM

VARIABLE	DESCRIPTION
ISAM(1)	Unused
ISAM(2)	Unused
ISAM(3)	Unused
ISAM(4)	Unused
ISAM(5)	Unused
ISAM(6)	Total number of active records
ISAM(7)	Number of unreclaimed 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 Key flag
ISAM$(7)	Unused
ISAM$(8)	Unused
ISAM$(9)	Unused.

B. FUNCTION CODES

ABOUT THIS APPENDIX

This appendix provides a quick reference of all the ISAM function codes.

INDEX

FUNCTION CODES

FUNC-TION CODE	DESCRIPTION
OC	Create File
OO	Open File
OF	Open/Create File
CL	Close File.
RK	Read Key
RG	Read Generic
RN	Read Next
RP	Read Previous
RB	Read Beginning
RE	Read End.
SK	Secondary Index Read
SG	Secondary Index Read Generic
SN	Secondary Index Read Next
SP	Secondary Index Read Previous
SB	Secondary Index Read Beginning
SE	Secondary Index Read End.
WA	Write Add

FUNC-TION CODE	DESCRIPTION
SA	Secondary Index Write Add.
DR	Delete Record
KD	Key Delete
SD	Secondary Index Key Delete.
MS	ISAM Status.

C. RETURN CODES

ABOUT THIS APPENDIX

This appendix provides a quick reference of the return codes issued by the ISAM subroutine.

INDEX

RETURN CODES

RETURN CODE	MEANING
00	normal return
31	invalid function order
41	syntax error
51	record not found
61	duplicate key
71	open/close error
81	disk error

For more detailed information about the meaning of the return codes see Chapter 3.

NOTICE

Ing. C. Olivetti & C. S.p.A. reserves the right to make improvements in the product described in this manual at any time and without notice.

This material was prepared for the benefit of Olivetti customers. It is recommended that the package be test run before actual use.

Anything in the standard form of the Olivetti Sales contract to the contrary not withstanding, all software being licensed to Customer "as is". THERE ARE NO WARRANTIES EXPRESS OR IMPLIED INCLUDING WITHOUT LIMITATION THE IMPLIED WARRANTY OF FITNESS FOR PURPOSE AND OLIVETTI SHALL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL OR INCIDENTAL DAMAGES IN CONNECTION WITH SUCH SOFTWARE.

Code 4001480 P (1)
Printed in Italy

OLIVETTI
PERSONAL
COMPUTER

olivetti