# INFOS™

# UTILITIES

# User's Manual

093-000182-01

Ordering No. 093-000182

Printed in the United States of America
Rev. 01, August 1976

Original Release - July 1975
First Revision - August 1976

This document has been extensively revised from revision 00; therefore, change indicators have not been used.

# PREFACE

This manual describes the utility programs available for INFOS System users, each utility being described in its own chapter.

The descriptions provided here rely, in part, on your knowledge of INFOS and RDOS. You should be familiar with the contents of the manuals for these systems. INFOS information appears in the INFOS Planning Manual (093-000115) and in the INFOS Programmer's Manual (093-000114). RDOS is described in the RDOS Reference Manual (093-000129) and in the RDOS CLI Manual (093-000146).

# CONTENTS

## CHAPTER 4 - (Continued)



## CHAPTER 5 - FILE COPY UTILITY

## CHAPTER 6 - PERMANENT FILE SPECIFICATION RETRIEVAL UTILITY



## CHAPTER 7 - FILE RENAME UTILITY



## CHAPTER 8 - FILE DELETE UTILITY



## CHAPTER 9 - LABELED MAGNETIC TAPE INITIALIZATION UTILITY

# CHAPTER 1
# INTRODUCTION

## GENERAL DESCRIPTION

At all stages of a file's existence, you have access to a collection of utility programs designed to reduce the amount of programming you need to do for INFOS System file management. The degree of difficulty associated with file creation and maintenance ranges from relatively simple to very complex. Quite often you'll have to do a good deal of planning and design work to generate a file which provides the desired characteristics, while simultaneously taking advantage of and efficiently using system resources. Manipulating and maintaining files also can become a complex task, depending of the file's structure and what you, as a programmer, want to do with it.

You invoke the utility programs through the operating system's Command Line Interpreter (CLI). In general, there are two types of utility programs; those that augment the operating system's file manipulation facilities, and those that assist you in designing, creating, and maintaining INFOS System files. The utility programs currently available are:

### File Create (ICREATE)

You can use this utility program to create the required file and volume definition for any type of INFOS System disk file. After you make the desired specifications, this utility creates an empty file on disk, and makes a permanent record of its characteristics.

### Index File Parameter Calculation (INDEXCALC)

You can use this program to determine an optimum set of index file parameters for a particular application. INDEXCALC will calculate the potential structure of a planned index and determine the effects of any change in the parameters you select. It outputs the information calculated to a line printer, if you desire, so you can study it and plan the file off-line.

### ISAM/DBAM File Inquiry (INQUIRE)

This utility lets you examine, modify, add, and delete data base records, and create, modify, and manipulate index files.

### File Copy (ICOPY)

With ICOPY, you can transfer all or part of an ISAM or DBAM file to a SAM disk file, transfer a SAM disk file to an ISAM or DBAM file, or transfer all or part of a SAM source file to a SAM destination file. Both the source and destination files can be labeled or unlabeled magnetic tape files, or they can be disk files. You can specify a preorder traversal of the index structure when copying a DBAM file.

### Permanent File Specification Retrieval (IFILE)

You can retrieve and examine the system-maintained Permanent File Specification (PFS) for any INFOS System disk file with IFILE.

### File Rename (IRENAME)

You can change the name of any INFOS System disk file through this utility. If you change the name of a multi-volume file, the utility renames each volume and makes the appropriate changes in the system-generated and-maintained Volume Definition and Index Definition files. You can also change the names of one or more volumes of a multi-volume file without renaming the file itself.

### File Delete (IDELETE)

You can delete any INFOS System disk file, and one or more indexes from a multiple-index DBAM file with IDELETE.

### Labeled Magnetic Tape Initialization (LBINIT)

With this utility, you can fully or partially initialize a labeled magnetic tape file.

## INVOKING A UTILITY

You invoke each utility through a call to the Command Line Interpreter (CLI). You can append various local and global switches to each call. The call for each utility is described later, with the detailed description of the utility.

You respond to any utility prompt by entering one of the characters included in the prompt, a value of your own, or a carriage return. The carriage return is the default response; if you key a carriage return, the utility assigns a system-determined value, where appropriate, or it allows (or disallows) a system feature according to the established system convention. In either case, whenever you default a response the utility displays the system-generated response on the line immediately below the prompt.

In all examples throughout this manual user entries are underlined so you can readily distinguish them from the prompt. The carriage return, whether used as the command line terminator or as the default response, is shown by the symbol ).

## LOG FILES

With most utilities you have the option of keeping a log file, which is a copy of your interaction with a utility for a particular invocation. You can produce the log file in either of two ways: where permitted, you can include $LPT/A on the command line, and immediately output the results of your interaction to the line printer. Or, you can include a non-device name with the switch /A on the command line (for example, LOG/A) and print that file later.

If you use a non-device file name with /A, you can use the resulting disk file as input to the utility. For example, if you create a file named ANALYSIS, you might also save on disk a log file named DIALOGUE:

```
ICREATE ANALYSIS DIALOGUE/A
```

You can later input DIALOGUE by entering:

```
ICREATE ANALYSIS DIALOGUE/B
```

Note that all the file names and local switches that you used in the original command line must be present when you input the log file to a utility, and the local switch /B (not /A) must immediately follow the dialogue file name.

Keeping a log file on disk has many advantages. For example, you might create a file that you use temporarily, and then delete. When you require that file structure again, simply invoke ICREATE with the name of the file containing the dialogue for it. You can also use this feature when you're trying to find an optimum file design. For example, you can create a file and keep its design in a dialogue file. If the file proves unsuitable, delete it, edit the dialogue file with SUPEREDIT, and try again.

We recommend that you normally keep a dialogue file, as there is no way to "back up" a utility to change a response. You can proceed through an entire interactive session, then edit the dialogue file later. This can save you a lot of grief when you're creating a relatively complex ISAM or DBAM file. Also, if you keep a dialogue file, you can terminate the interactive session at any time without losing your previous work. Simply input the dialogue file, and the utility accepts the dialogue up to the point where you ended the previous session.

END OF CHAPTER

# CHAPTER 2

# FILE CREATE UTILITY

## INTRODUCTION

You define and create INFOS System disk files interactively at the system console with ICREATE. You can define SAM disk files and RAM, ISAM, and DBAM files. Each can be single- or multi-volume, and you can create multi-indexed DBAM files. ICREATE accepts file specifications which you enter at the console, and builds a Permanent File Specification (PFS). It then opens and closes the file, leaving a fully-specified empty file on disk that you can subsequently open without respecifying all its characteristics.

ICREATE is easy to use. For example, you can enter all the information needed to create a file on a single command line. Or, through the interactive dialogue, you can enter values for only those options you require, and default the rest to system values. You can save the file creation dialogue on disk and edit it whenever necessary. Later, you can enter the dialogue file in a 'Batch-Like' mode as input to ICREATE.

## INVOKING ICREATE

The format of the CLI call that invokes ICREATE has a significant effect on the resulting dialogue and on the file ultimately created. You can call the utility simply by entering:

```
ICREATE
```

In this case, you'll see the entire dialogue displayed line-by-line, and the file created depends on your entries.

Alternatively, you can invoke the utility with the file's name and access method on the command line. ICREATE checks to see that the file name you enter is not already in use. The local switch (/S, /R, /I, or /D) following the file name indicates the file's access method. Thus,

```
ICREATE filename/S
ICREATE filename/R
ICREATE filename/I
ICREATE filename/D
```

tells the utility to use the SAM, RAM, ISAM, or DBAM dialogue respectively, and to build the file accordingly.

To create a file that uses all system-assigned values, include the global switch /S on the command line. For example,

```
ICREATE/S ANALYSIS/R
```

creates a single volume RAM file named ANALYSIS. The file will have 512 byte records, a 512 byte block, and one buffer. The volume will be allocated randomly, be of maximum size (up to 65,535 blocks), and use a pad character of zero. Note that you need not include the file's name and access method on the command line when you use the global switch /S, in which case the utility prompts for them.

Include a file name with the local switch /A on the command line when you want a hard copy of the dialogue. For example,

```
ICREATE $LPT/A
```

outputs the dialogue to the line printer, line-by-line. Or, for example, use:

```
ICREATE LOG/A
```

and print LOG later. You can also include the file's name and access method on the command line when you generate a hard copy file.

## ICREATE DIALOGUE SUMMARY

After you invoke ICREATE, it proceeds through an interactive dialogue. The prompts used in the dialogue depend on the access method you specify. Certain prompts are common to all access methods, and some apply only to a particular method. Some prompts accept a default response (the carriage return) and some don't. If you default an entry, ICREATE displays the system-assigned value. If it finds a default response unacceptable, it repeats the prompt or returns to the CLI.

Table 2-1 summarizes ICREATE's prompts and describes your responses to them.

Table 2-1. ICREATE Dialogue

| ICREATE PROMPT | YOUR RESPONSE | COMMENTS |
|---|---|---|
| ACCESS METHOD: | ) | This prompt is displayed only if you do not include a local switch indicating the file's access method.<br><br>The following prompt is displayed:<br>S = SAM, R = RAM, I = ISAM, D = DBAM |
| | S | SAM |
| | R | RAM |
| | I | ISAM |
| | D | DBAM |
| ALLOW SUBINDICES? (Y OR N) | Y | Allows the subsequent creation of subindexes under this index. |
| | N or ) | Prohibits the subsequent creation of subindexes under this index. |
| BLOCK SIZE: | number | Enter the block size, in bytes, for SAM, RAM, and data base files. |
| | ) | ICREATE assigns 512-byte blocks for SAM, RAM, and data base files. |
| BLOCK (PAGE) SIZE: | | You may first want to use the INDEXCALC utility to determine an appropriate page size. |
| | number | Enter the page size, in bytes, for index files. |
| | ) | ICREATE assigns 512-byte pages used for index files. |
| COMPRESSED KEYS? (Y OR N) | Y | The system will compress index keys. |
| | N or ) | The system will not compress index keys. |
| CONTIGUOUS STORAGE? (Y OR N) | Y | The volume currently being defined will be allocated contiguously. See 'SIZE(DISK BLOCKS):'. |
| | N or ) | The volume currently being defined will be allocated randomly. See 'SPECIFY MAXIMUM SIZE? (Y OR N):'. |

(Continued on the next page)

Table 2-1. ICREATE Dialogue (Continued)

| ICREATE PROMPT | YOUR RESPONSE | COMMENTS |
|---|---|---|
| ENABLE SPACE MANAGEMENT? (Y OR N) | Y | The system builds an Available Space Index at the beginning of each logical volume. You can calculate the number of disk blocks required for each such index built from:<br><br>1+ (Blocks per Volume + 255)/256<br><br>You must account for the space required for each such index when you specify volume size. |
| | N or ) | Space management is not used. |
| FILE: | | This prompt is displayed only if you did not include a file name on the command line. |
| | name | ICREATE checks to see whether the file name is in use; if it is, ICREATE repeats the prompt. File names must correspond to the operating system's naming conventions. NOTE: The file has the name given to its first volume; see 'VOLUME NAME:'. |
| | ) | ICREATE returns to the CLI. |
| INITIALIZE TO ZEROS? (Y OR N) | | This prompt is displayed only if you specified contiguous allocation. |
| | Y | The system zeros out the disk blocks allocated on the current volume. |
| | N or ) | The space is not zeroed out on the current volume. |
| MAXIMUM KEY LENGTH: | number | Enter the length, in bytes, of the longest key in the index. |
| | ) | ICREATE will calculate the largest possible key size for the minimum node size. |
| MAXIMUM NUMBER OF LEVELS: | | This prompt is displayed only if you allowed subindexing. |
| | number | Enter the number of sub-index levels the file will eventually have, plus 1, since the index you are currently creating is subindex level zero. |

(Continued on the next page)

Table 2-1. ICREATE Dialogue (Continued)

| ICREATE PROMPT | YOUR RESPONSE | COMMENTS |
|---|---|---|
| MAXIMUM RECORD LENGTH: | number<br><br>) | This prompt is displayed only if you specified variable or undefined length records.<br><br>Enter the length, in bytes, of the longest record in the file.<br><br>For variable length records, the maximum record size will be 8 bytes less than block size, and for undefined length records record size will be equal to block size. |
| MINIMUM NODE SIZE: | <br><br><br><br><br><br><br><br><br><br><br><br>number<br><br>) | You can use the INDEXCALC utility to determine an appropriate node size. Otherwise, for ISAM use:<br><br>36 + k * (10 + max key length)<br><br>For DBAM files with subindexing use:<br><br>36 + k * (14 + max key length + partial record length)<br><br>and for DBAM files without sub-indexing use:<br><br>36 + k * (10 + max key length + partial record length)<br><br>where k = the number of index entries per node.<br><br>Enter the node size in bytes.<br><br>506 bytes if you also defaulted page size. Otherwise, node size will be 6 bytes less than page size. |
| NUMBER OF BUFFERS: | number<br><br>) | Enter the desired number of buffers.<br><br>ICREATE assigns 1 buffer for SAM, RAM, and data base files; 2 for index files. |
| OPTIMIZE RECORD DISTRIBUTION? (Y OR N)? | <br><br>Y<br><br><br><br>N | You can use the INDEXCALC utility to help you determine appropriate index file page sizes.<br><br>The system places index nodes and data base records on different-speed devices, according to merit factors you subsequently specify.<br><br>Optimized record distribution is not used. |

(Continued on the next page)

Table 2-1. ICREATE Dialogue (Continued)

| ICREATE PROMPT | YOUR RESPONSE | COMMENTS |
|---|---|---|
| PAD CHARACTER (OCTAL CODE 0-377): | number<br><br>) | Enter any three-octal-digit value to represent any character in a set.<br><br>The pad character is the null character (000). |
| PARTIAL RECORD LENGTH: | number<br><br>0 or ) | Enter the length, in bytes, of the index file partial records.<br><br>The index file does not have partial records. |
| RECORD FORMAT: | F<br>V<br>U<br>D<br><br>) | Fixed length records.<br>Variable length records.<br>Undefined length records.<br>Data Sensitive records. The system assumes the delimiter is either null, carriage return, or form feed.<br><br>ICREATE displays the following prompt.<br><br>F=FIXED, V=VARIABLE, U=UNDEFINED, D=DATA SENSITIVE |
| RECORD LENGTH: | number<br><br>) | This is displayed only if you specified fixed length records.<br><br>Enter the exact record length in bytes.<br><br>The prompt is repeated. |
| ROOT NODE MERIT FACTOR: | number | Enter the number corresponding to (or in the range of) the merit factor of the volume on which the root node will be placed. (See VOLUME MERIT FACTOR:.)<br><br>You cannot default this question. |
| SIZE (DISK BLOCKS): | number | This prompt is displayed only if you specified contiguous allocation or if you responded 'Y' to 'SPECIFY MAXIMUM SIZE?'<br><br>Enter the number of 512-byte disk blocks for the volume currently being defined. If you specified space management, include the blocks required for the Available Space Index.<br><br>You cannot default this question. |

(Continued on the next page)

Table 2-1. ICREATE Dialogue (Concluded)

| ICREATE PROMPT | YOUR RESPONSE | COMMENTS |
|---|---|---|
| SPECIFY MAXIMUM SIZE? (Y OR N) | | This prompt is displayed only if you specified random allocation. |
| | Y | This response allows you to set an upper limit on the size of a randomly-allocated volume. |
| | N or ) | The volume you are currently defining is unlimited in size; that is, it can have 65,535 disk blocks. |
| VOLUME MERIT FACTOR: | | You must define volumes in order of decreasing I/O speed. The highest numeric merit factor applies to the volume on the fastest device. |
| | number | Enter the merit factor of the volume being defined. |
| | | NOTE: If, for example, there are three volumes with merit factors of 9, 6, and 3, then the range of merit factors for the first (fastest) volume is 9 and above; the range for the second volume is 6, 7, and 8; and the range for the slowest (third) volume is 0, 1, 2, 3, 4, and 5. |
| VOLUME n NAME: | | This prompt initiates a repeated series of questions. Proceed through the series for each volume you're defining and then respond with a carriage return to this prompt. |
| | | If you want optimized record distribution you must define the volumes in order of decreasing speed. |
| | name | If this is the first volume being defined, the name you enter becomes the file name. You must give a name to each volume you define. |
| | ) | If this is the first volume being defined, the system uses the name you entered on the command line, or your response to the 'FILE:' prompt. |
| | | If this is not the first volume being defined, ICREATE terminates volume definition and goes on to its next task. |

## EXAMPLE 1 - SAM FILE CREATION

This example shows your interaction with ICREATE to define a SAM disk file having 80-byte records blocked at 12 records per block. ICREATE produces a single-volume, randomly-allocated file with a maximum size of 2334 disk blocks which can hold 14,000 records. One buffer is allocated.

```
ICREATE $LPT/A)
FILE: SAMD)
ACCESS METHOD (S=SAM,R=RAM,I=ISAM,D=DBAM): S)
RECORD FORMAT: F)
BLOCK SIZE: 960)
NUMBER OF BUFFERS: )
                   1
RECORD LENGTH: 80)
VOLUME 1 NAME: )
               SAMD
        CONTIGUOUS STORAGE? (Y OR N) )
                                     N
        SPECIFY MAXIMUM SIZE? (Y OR N) Y)
                SIZE (DISK BLOCKS): 2334)
        PAD CHARACTER (OCTAL CODE 0-377): )
                                          0
VOLUME 2 NAME: )
```

## EXAMPLE 2 - RAM FILE CREATION

Here is an interaction with ICREATE to define a single volume RAM file that can hold 500 records. The records are unblocked (one per block) and are stored in contiguously allocated space that has been zeroed out. The asterisk character (252) pads the unused space in each block. Two buffers are allocated.

```
ICREATE ANALYSIS/R $LPT/A)
BLOCK SIZE: )
            512
NUMBER OF BUFFERS: 2)
RECORD LENGTH: 500)
VOLUME 1 NAME: )
               ANALYSIS
        CONTIGUOUS STORAGE? (Y OR N) Y)
        SIZE (DISK BLOCKS): 500)
        INITIALIZE TO ZEROS? (Y OR N) Y)
        PAD CHARACTER (OCTAL CODE 0-377): 252)
VOLUME 2 NAME: )
```

## EXAMPLE 3 - ISAM FILE CREATION

This example shows user interaction with ICREATE to define the ISAM file, ISAM1. The index file uses default page and node sizes, 512 bytes and 506 bytes respectively. Twelve bytes is specified as key length and three buffers are specified. The entire index is stored on a single volume, randomly allocated, with its size limited to 100 disk blocks.

A maximum record length of 80 bytes and a block size of 512 bytes are specified for the data base file. The data base is stored on a single volume of unlimited size, randomly allocated. One buffer is specified.

```
ICREATE ISAM1 $LPT/A)
ACCESS METHOD (S=SAM,R=RAM,I=ISAM,D=DBAM): I)
 ****** INDEX FILE ******
BLOCK (PAGE) SIZE: )
                   512
NUMBER OF BUFFERS: 3)
MINIMUM NODE SIZE: )
                   506
MAXIMUM KEY LENGTH: 12)
ENABLE SPACE MANAGEMENT? (Y OR N) )
                                  N
VOLUME 1 NAME: )
               ISAM1
        CONTIGUOUS STORAGE? (Y OR N) )
                                     N
        SPECIFY MAXIMUM SIZE? (Y OR N) Y)
                SIZE (DISK BLOCKS): 100)
        PAD CHARACTER (OCTAL CODE 0-377): )
                                          0
VOLUME 2 NAME: )

 ****** DATA BASE FILE ******
NAME: ISAM2)
BLOCK SIZE: )
            512
NUMBER OF BUFFERS: )
                   1
MAXIMUM RECORD LENGTH: 80)
ENABLE SPACE MANAGEMENT? (Y OR N) )
                                  N
VOLUME 1 NAME: )
               ISAM2
        CONTIGUOUS STORAGE? (Y OR N) )
                                     N
        SPECIFY MAXIMUM SIZE? (Y OR N) )
                                       N
        PAD CHARACTER (OCTAL CODE 0-377): )
                                          0
VOLUME 2 NAME: )
```

## EXAMPLE 4 - DBAM FILE CREATION

User interaction with ICREATE to define a DBAM file is shown in this example. Index entries in the level zero node have a key length of 10 bytes and a partial record length of 20 bytes. Subindexes cannot be defined under this index. Page and node sizes are 1024 bytes and 1018 bytes, respectively. The index resides on two volumes, with the root node on the fastest. This volume uses contiguous storage of 98 disk blocks and is assigned a merit factor of 1. The second volume contains the level zero node in randomly allocated, unlimited space. Its merit factor is 0.

The data base resides on a single volume of unlimited size. Both block size and maximum record length were defaulted, giving a maximum record length 8 bytes less than block size.

```
ICREATE ACCT.V1/D $LPT/A)
 ****** INDEX FILE ******
BLOCK (PAGE) SIZE: 1024)
NUMBER OF BUFFERS: )
                   2
ALLOW SUBINDEXES? (Y OR N) )
                           N
PARTIAL RECORD LENGTH: 20)
MINIMUM NODE SIZE: 1018)
MAXIMUM KEY LENGTH: 10)
COMPRESSED KEYS? (Y OR N) )
                          N
ENABLE SPACE MANAGEMENT? (Y OR N) )
                                  N
OPTIMIZE RECORD DISTRIBUTION? (Y OR N) Y)
ROOT NODE MERIT FACTOR: 1)
VOLUME 1 NAME: ACCT.V1)
        CONTIGUOUS STORAGE? (Y OR N) Y)
        SIZE (DISK BLOCKS): 98)
        INITIALIZE TO ZEROS? (Y OR N) )
                                      N
        PAD CHARACTER (OCTAL CODE 0-377): )
                                          0
        VOLUME MERIT FACTOR: 1)
VOLUME 2 NAME: ACCT.V2)
        CONTIGUOUS STORAGE? (Y OR N) )
                                     N
        SPECIFY MAXIMUM SIZE? (Y OR N) )
                                       N
        PAD CHARACTER (OCTAL CODE 0-377): )
                                          0
        VOLUME MERIT FACTOR: 0)
VOLUME 3 NAME: )

 ****** DATA BASE FILE ******
NAME: ACCTDB)
BLOCK SIZE: )
            512
NUMBER OF BUFFERS: )
                   1
MAXIMUM RECORD LENGTH: )
                      504
ENABLE SPACE MANAGEMENT? (Y OR N) )
                                  N
OPTIMIZE RECORD DISTRIBUTION?  (Y OR N) )
                                        N
VOLUME 1 NAME: )
               ACCTDB
        CONTIGUOUS STORAGE? (Y OR N) )
                                     N
        SPECIFY MAXIMUM SIZE? (Y OR N) )
                                       N
        PAD CHARACTER (OCTAL CODE 0-377): )
                                          0
VOLUME 2 NAME: )
```

## EXAMPLE 5 - MULTI-INDEX DBAM FILE CREATION

This example shows user interaction with ICREATE to define a second index for the data base file created in Example 4.

Note the interaction defining the data base file. Since it already exists under the file name ACCTDB, entering that name links the index (CUSTNAME) to it. Now you can access ACCTDB through ACCT.V1 or CUSTNAME, or both indexes.

```
ICREATE CUSTNAME/D $LPT/A)
 ****** INDEX FILE ******
BLOCK (PAGE) SIZE: 1024)
NUMBER OF BUFFERS: 3)
ALLOW SUBINDEXES? (Y OR N) )
                           N
PARTIAL RECORD LENGTH: )
                       0
MINIMUM NODE SIZE: 1018)
MAXIMUM KEY LENGTH: 50)
COMPRESSED KEYS?  (Y OR N) Y)
ENABLE SPACE MANAGEMENT? (Y OR N) )
                                  N
OPTIMIZE RECORD DISTRIBUTION? (Y OR N) )
                                       N
VOLUME 1 NAME: CUSTNAME)
        CONTIGUOUS STORAGE? (Y OR N) )
                                     N
        SPECIFY MAXIMUM SIZE? (Y OR N) Y)
                SIZE (DISK BLOCKS): 100)
        PAD CHARACTER (OCTAL CODE 0-377): )
                                          0
VOLUME 2 NAME: )

 ****** DATA BASE FILE ******
NAME: ACCTDB)
(PRE-EXISTING DATA BASE FILE)
TEMPORARY INDEX? (Y OR N) )
                          N
VOLUME 1 NAME: )
               ACCTDB
```

## EXAMPLE 6 - USING A DIALOGUE FILE TO CREATE A DBAM FILE

The only user input in this example is the command line. ICREATE creates the file using the specifications shown in the dialogue.

```
ICREATE ACCTS/D DLOG/B $LPT/A)
****** INDEX FILE ******
BLOCK (PAGE) SIZE:
                  512
NUMBER OF BUFFERS:
                   2
ALLOW SUBINDICES?  (Y OR N) Y
MAXIMUM NUMBER OF LEVELS:  3
PARTIAL RECORD LENGTH: 24
MINIMUM NODE SIZE
                  506
MAXIMUM KEY LENGTH: 5
COMPRESSED KEYS? (Y OR N)
                         N
ENABLE SPACE MANAGEMENT? (Y OR N)
                                  N
OPTIMIZE RECORD DISTRIBUTION? (Y OR N)
                                       N
VOLUME 1 NAME:
             ACCTS
        CONTIGUOUS STORAGE? (Y OR N) Y
        SIZE (DISK BLOCKS): 100
        INITIALIZE TO ZEROS? (Y OR N) Y
        PAD CHARACTER (OCTAL CODE 0-377):
                                         0
VOLUME 2 NAME:

 ****** DATA BASE FILE ******
NAME: ACCTSDB1
BLOCK SIZE:
            512
NUMBER OF BUFFERS:
                  1
MAXIMUM RECORD LENGTH: 80
ENABLE SPACE MANAGEMENT? (Y OR N)
                                 N
OPTIMIZE RECORD DISTRIBUTION? (Y OR N) Y
VOLUME 1 NAME:
              ACCTSDB1
        CONTIGUOUS STORAGE? (Y OR N) Y
        SIZE (DISK BLOCKS): 100
        INITIALIZE TO ZEROS? (Y OR N) Y
        PAD CHARACTER (OCTAL CODE 0-377):
                                         0
        VOLUME MERIT FACTOR: 1
VOLUME 2 NAME: ACCTSDB2
        CONTIGUOUS STORAGE? (Y OR N) Y
        SIZE (DISK BLOCKS): 100
        INITIALIZE TO ZEROS? (Y OR N) Y
        PAD CHARACTER (OCTAL CODE 0-377):
                                         0
        VOLUME MERIT FACTOR: 0
VOLUME 3 NAME:
```

END OF CHAPTER

# CHAPTER 3
# INDEX FILE PARAMETER CALCULATION UTILITY

## INTRODUCTION

INDEXCALC helps you find the right combination of main index or sub-index parameters for a particular application. You can calculate the potential structure of an index, using a given set of specifications, and then determine the effects on its structure of changing one or more of them. INDEXCALC is easy to use. It simply asks you a series of questions about the index and then asks which parameter you want to solve for. After you reply, it asks for additional specifications, which you can change later if you desire.

You can use INDEXCALC to find out how many tree levels the index should have to give optimum performance, the maximum number of index entries it will be able to contain, and what its optimum page and node sizes are. To do this, you input design characteristics such as key size, partial record length, and whether or not you intend to allow subindexing. After computing a set of parameters, you can re-specify various elements to compute another set.

INDEXCALC treats all input and display values as integers, either truncated or rounded up. When you input specifications for key length and partial record length, INDEXCALC rounds them up, if necessary, to be a multiple of two bytes.

## INVOKING INDEXCALC

Your CLI call to invoke INDEXCALC is:

```
INDEXCALC
```

## INDEXCALC DIALOGUE SUMMARY

The INDEXCALC interactive dialogue is depicted in Table 3-1. The questions you are ultimately asked depend wholly on your responses to previously-asked questions.

Table 3-1. INDEXCALC Dialogue

| INDEXCALC PROMPT | YOUR RESPONSE | COMMENTS |
|---|---|---|
| HARD COPY OF ALL RESULTS? (Y OR N) | Y | You'll get the prompt, SPECIFY HARD COPY HEADER. |
| | N | A header is not requested of you now; but later after you specify what is being solved for, you'll get the prompt, HARD COPY OF THESE RESULTS? |
| SPECIFY HARD COPY HEADER: | header | Enter an alphanumeric string that identifies the current calculation results. |
| SPECIFY KEY LENGTH: | number | Enter a decimal number in the range 1-255 that gives the length, in bytes, of the longest key in the sub index. |
| SPECIFY PARTIAL RECORD LENGTH: | number | If partial records are not used enter 0. Otherwise, enter a decimal number between 1 and 255 that gives the length in bytes. |
| SUB-INDEXES ALLOWED? (Y OR N) | Y | Index entry size is adjusted to accommodate a pointer to another sub-index. |
| | N | Index entries will not have room for a pointer to another sub-index. |
| SOLVING FOR NUMBER OF LEVELS (L), PAGE SIZE (P), OR NUMBER OF INDEX ENTRIES (E): | L | When solving for the number of index tree levels, you are asked to specify page size and the number of index entries. |
| | P | When solving for page size you are asked to specify the number of index entries and the number of index tree levels. |
| | E | When solving for the number of index entries, you are asked to specify page size and the number of index tree levels. |
| SPECIFY PAGE SIZE (BLOCKS): | number | Enter a decimal integer between 1 and 12 for the number of 512-byte blocks for a page. |

(Continued on the next page)

Table 3-1. INDEXCALC Dialogue (Continued)

| INDEXCALC PROMPT | YOUR RESPONSE | COMMENTS |
|---|---|---|
| SPECIFY NUMBER OF INDEX ENTRIES: | number | Enter the number of index entries the index will contain. |
| SPECIFY NUMBER OF LEVELS: | number | Enter a decimal number between 1 and 256 representing the desired number of index tree levels. |
| ITERATE (I), CHOOSE ANOTHER PARAMETER (C), RESTART (R), OR STOP (S): | I | When solving for the number of index tree levels, you are asked to respecify the number of index entries. The page size you previously specified is reused.<br><br>When solving for page size, you are asked to respecify both the number of index entries and the number of index tree levels.<br><br>When solving for the number of index entries, you are asked to respecify the number of index tree levels. The page size you previously specified is reused. |
| | C | INDEXCALC returns to the 'SOLVING FOR' prompt so you can solve for a different parameter. |
| | R | INDEXCALC returns to the beginning of the dialogue so you can enter a different set of specifications. |
| | S | INDEXCALC returns to the CLI. |

```
INDEXCALC = VERSION 1.8                    DATE: 7/ 8/76   14:17
TEST ONE


SOLVING FOR PAGE SIZE

KEY LENGTH =  10        PART. REC. LENGTH =  24  SUB-INDEXES ALLOWED = Y
PAGE SIZE =  1          NODE SIZE =  506         MIN. INITIAL NODE SIZE =     0
NUMBER OF LEVELS =  5
DESIRED ENTRIES =    100000.                     MAXIMUM ENTRIES =    3179520.

ROOT NODE B.F. =  23    INTERM. NODE B.F. =  24  LEVEL 0 NODE B.F. =  10

 LEVEL   I       NODES      BLOCKS   I   MAX NODES  MAX BLOCKS
     0   I     10000.       10000.   I     317952.     317952.   I
     1   I       416.         416.   I      13248.      13248.   I
     2   I        17.          17.   I        552.        552.   I
     3   I         0.           0.   I         23.         23.   I
     4   I         0.           0.   I          1.          1.   I
TOTALS   I     10433.       10433.   I     331776.     331776.   I
 BYTES   I                5341696.   I               169869312.   I


SOLVING FOR NUMBER OF LEVELS

KEY LENGTH =  10        PART. REC. LENGTH =  24  SUB-INDEXES ALLOWED = Y
PAGE SIZE =  1          NODE SIZE =  506         MIN. INITIAL NODE SIZE =     0
NUMBER OF LEVELS =  4
DESIRED ENTRIES =    100000.                     MAXIMUM ENTRIES =     132480.

ROOT NODE B.F. =  23    INTERM. NODE B.F. =  24  LEVEL 0 NODE B.F. =  10

 LEVEL   I       NODES      BLOCKS   I   MAX NODES  MAX BLOCKS
     0   I     10000.       10000.   I      13248.      13248.   I
     1   I       416.         416.   I        552.        552.   I
     2   I        17.          17.   I         23.         23.   I
     3   I         1.           1.   I          1.          1.   I
TOTALS   I     10434.       10434.   I      13824.      13824.   I
 BYTES   I                5342208.   I                7077888.   I


SOLVING FOR NUMBER OF ENTRIES

KEY LENGTH =  10        PART. REC. LENGTH =  24  SUB-INDEXES ALLOWED = Y
PAGE SIZE =  1          NODE SIZE =  506         MIN. INITIAL NODE SIZE =     0
NUMBER OF LEVELS =  4
DESIRED ENTRIES =    132480.                     MAXIMUM ENTRIES =     132480.

ROOT NODE B.F. =  23    INTERM. NODE B.F. =  24  LEVEL 0 NODE B.F. =  10

 LEVEL   I       NODES      BLOCKS   I   MAX NODES  MAX BLOCKS
     0   I     13248.       13248.   I      13248.      13248.   I
     1   I       552.         552.   I        552.        552.   I
     2   I        23.          23.   I         23.         23.   I
     3   I         1.           1.   I          1.          1.   I
TOTALS   I     13824.       13824.   I      13824.      13824.   I
 BYTES   I                7077888.   I                7077888.   I
```

END OF CHAPTER

# CHAPTER 4

# ISAM/DBAM FILE INQUIRE UTILITY

## INTRODUCTION

The INQUIRE Utility makes it easy for you to examine and make modifications to an ISAM or DBAM file. For example, you can position to any index entry and retrieve the key and its occurrence number, determine the current octal-value link to its data base record and the key's index level, and/or get the length of the key and data base record. You can position through relative motion, keyed access, or a combination of both techniques.

Generally, when you position to an entry, the data base record is displayed if one is present. When a data base record for the key does not exist, and when the key positioned to is a duplicate, you are so informed. You can write a new entry and rewrite an existing entry and/or partial record, and define duplicate keys. You can also use the writing functions with the suppress data base and inverting options.

You can logically delete entries either locally or globally, or delete them physically. You can define and link subindexes as well as delete them. Many other features and capabilities are available, all of which are described in detail in this chapter.

Even though writing functions are provided, the utility is not designed for loading large amounts of data into an ISAM or DBAM file. You can perform this function more easily (and accurately) with the ICOPY Utility, or with an application program.

## INVOKING INQUIRE

Your CLI call to invoke INQUIRE is

```
INQUIRE [index file name][listfile/A]
```

When you give the call without an index file name, you'll see dialogue like this:

```
INQUIRE $LPT/A)
INDEX: ACCTS)
OVERRIDE # OF BUFFERS? (Y OR N) Y)
NUMBER OF BUFFERS: 4)
ENTER C<CR> TO GET LIST OF COMMANDS

COMMANDS: C)


                              COMMANDS

A          ACQUIRE FEEDBACK
C          COMMANDS                      W          WRITE
K          KEYED                         X          REWRITE
R          RELATIVE                      I          INVERTED (W OR X)
B          SUPPRESS DATA BASE            D          DELETE RECORD
P          INCLUDE PARTIAL RECORD        Q          REINSTATE RECORD
O          CHANGE OUTPUT FILE            S          DEFINE SUBINDEX
N          NO POSITION CHANGE            L          LINK SUBINDEX
F          SET FORMATS                   U          UNLINK SUBINDEX
V          VERIFY KEY                    #          SET REPEAT COUNT
^          MOVE UP                       H          RETRIEVE HIGH KEY
->         MOVE FORWARD                  E          EXTRACT STATUS
<-         MOVE BACK                     G          GET INDEX DEF INFO
DOWN       MOVE DOWN & FORWARD           DEFAULT    READ FORWARD
ESCAPE     NEW INDEX                     RUBOUT     DELETE LINE


COMMANDS:
```

When your call includes an index file name, the dialogue will look like this:

```
INQUIRE ACCTS $LPT/A)
OVERRIDE # OF BUFFERS? (Y OR N) )
                                N
ENTER C<CR> TO GET LIST OF COMMANDS

COMMANDS:
```

When you include the global switch /S and an index file name you'll get the following simplified dialogue:

```
INQUIRE/S ACCTS $LPT/A)
ENTER C<CR> TO GET LIST OF COMMANDS

COMMANDS:
```

If you are not using a Data General 6012 video display terminal, you must include a global switch /I with the INQUIRE command.

## INQUIRE COMMANDS

INQUIRE commands that invoke ISAM/DBAM processing or utility functions are:

| | | |
|---|---|---|
| W | - | WRITE |
| X | - | REWRITE |
| D | - | DELETE |
| Q | - | REINSTATE |
| S | - | DEFINE SUBINDEX |
| L | - | LINK SUBINDEX |
| U | - | DELETE SUBINDEX |
| <ESC> | - | CLOSE |
| H | - | RETRIEVE HIGH KEY |
| V | - | RETRIEVE KEY |
| E | - | RETRIEVE STATUS |
| G | - | RETRIEVE SUBINDEX DEFINITION |

The INQUIRE command modifiers represent the following options:

| | | |
|---|---|---|
| K | - | Keyed Access |
| R | - | Relative Motion |
| ↑ | } | Motion Control |
| → | | |
| ← | | |
| ↓ | | |
| B | - | Suppress Data Base |
| I | - | Inverting |
| N | - | Don't Set Current Position |
| P | - | Include Partial Record |
| A | - | Data Record Feedback |

These INQUIRE commands have no corresponding ISAM/DBAM feature:

| | | |
|---|---|---|
| C | - | Display list of INQUIRE commands. |
| O | - | Change output file. |
| F | - | Set key and/or record formats. |
| # | - | Set repeat count. |
| <RUBOUT> | - | Delete line. |

An INQUIRE command string consists of one or more single-character INQUIRE commands terminated by a carriage return. You can enter a command string whenever INQUIRE displays the prompt

```
COMMANDS:
```

After you key in a single-character command, INQUIRE types a space to separate each command from the next in the command string. You should not attempt to space between the individual commands; if you do so, you'll get an error.

An empty command string is acceptable; it invokes a read with forward relative motion. Thus, when you enter:

```
COMMANDS: )
```

INQUIRE displays the record for the next index entry (relative to the current position prior to entering the command).

INQUIRE assumes the desired function is reading. Consequently, a command to specifically invoke the read function is not provided, and you'll execute a read operation unless you specify otherwise in the command string. INQUIRE also assumes the desired relative motion is

static motion. So, a motion control command is not provided to expressly specify static motion. Entering

```
COMMANDS: R)
```

specifies relative motion but no direction. This command string, therefore, is interpreted as a read relative with static motion request.

When you enter a command string containing the modifier K (for keyed access) the current position is set to the absolute beginning of the index structure before the function you request is executed. When you enter a command string containing both the K and R modifiers (for keyed access and relative motion) the relative motion is performed first. Motion is relative within the current index level. The current position is first set to the index level specified by the up or down motion control specifier, if you include one in the command string. Note that the initial current position is set logically just above the level zero index after opening the file. It is your responsibility to keep track of the current position. Most of the commands that invoke a function change the current position on executing the requested function. INQUIRE has a command to specify that current position not be changed when executing the requested function. Thus:

```
COMMANDS: K W )
```

requests a write operation. The current position first moves to the absolute beginning of the index structure.

```
COMMANDS: K W R )
```

requests a write operation at the current index level.

```
COMMANDS: K W ↑)
```

requests a write operation. The current position is first set at the index entry that owns the subindex which held the current position when you made the request.

```
COMMANDS: K W ↓)
```

requests a write operation. The current position is first set at the beginning of the subindex defined under the current index entry.

Whenever the command string includes the K modifier, INQUIRE asks you to specify a key, which you then enter on the line immediately following the command string. For write and delete commands you must specify an actual key. If the actual key is a duplicate, you indicate this by terminating the actual key with the # character. For other keyed accesses (read, rewrite, etc.) you can, if you want, specify a search key rather than an actual key. A search key can be a duplicate, approximate, or generic key. When the search key is a duplicate, you terminate the actual key with # followed by an occurrence number. When the search key is an approximate key you terminate it with the + character, and when it's a generic key you terminate it with the - character.

# ISAM/DBAM FUNCTIONS

## WRITE

### *Inquire Command*

W Invokes the write function.

### *Required Modifier*

K Specifies keyed access. See the description of the # command if the key you're writing is a duplicate. See the description of the F command for setting key and/or data record formats.

### *Optional Modifiers*

R Specifies relative motion. If you want to write the entry at the current index level, you must include this modifier command in the command string.

If you want to write the entry at an index level different than that of current position, this modifier command is not required, but you must include either ↑ or ↓ in the command string.

↑ Specifies upward relative motion. The relative motion is performed first, then the keyed access starts at the new index level.

↓ Specifies down and forward relative motion. The relative motion is performed first, then the keyed access starts at the new index level.

B Specifies suppress data base. This writes the index entry, but not a data base record.

I Specifies inverting. This writes an index entry pointing to an existing data base record that is also pointed to through some other index entry; it does not write the specified data base record.

N Specifies no position change. This writes an entry at the appropriate index level, but on completing the command the current position is unchanged.

P Specifies partial record. INQUIRE prompts for the partial record, which it writes along with the rest of the index entry at the appropriate index level.

### *Examples*

```
COMMANDS: W K )
KEY: 012 )
RECORD
FIELD #1: NAILS, 10D $0.65/LB )
```

This writes an index entry at index level zero, and the data base record to the data base file. The current position is at key 012 when the operation is completed.

```
COMMANDS: W K R )
KEY: 012 )
RECORD
FIELD #1: NAILS, 10D $0.65/LB )
```

This writes an entry at the current index level, and the data base record to the data base file. The current position is at key 012 when the operation is completed.

```
COMMANDS: W K ↓ P )
KEY: BOLTS, STEEL )
PARTIAL RECORD: NOT A STOCK ITEM. )
RECORD
FIELD #1: 1/4 IN - $0.35 EA. )
FIELD #2: 1/8 IN - $0.32 EA. )
FIELD #3: 1/16 IN - $0.30 EA. )
```

This writes an entry at the subindex defined under BOLTS. It positions to the subindex owned by the current entry, then, in that subindex, locates the key BOLTS which owns a subindex in which the index entry STEEL and its partial record are written. It also writes the three-field data base record to the data base file. The current position is at key STEEL when the operation is completed.

```
COMMANDS: W K I )
KEY: 007 )
```

This is an inverted write. It writes the index entry in the index, but links to the most-recently-accessed data base record in the data base file. The current position is at key 007 when the operation is completed.

```
COMMANDS: W K ↑ B P )
KEY: 007# )
PARTIAL RECORD: JOB CLASS. SPY )
```

This writes an entry containing the specified partial record at the index level of the entry that owns the current subindex entry. The current position is at key 007 (possibly a duplicate) when the operation is completed.

```
COMMANDS: W K ↑ N)
KEY: 200, 210, 220)
RECORD
FIELD #1:  NAILS IN STOCK)
```

In this example, the relative motion is performed first; the move is to the index entry that owns the current subindex. Then the keyed access is applied. The first key (200) must be in the sub-index containing the entry that owns the previous current subindex. INQUIRE writes an index entry with key 220 in the appropriate subindex, and the record to the data base file. Because you included N in the command string the current position remains unchanged after this operation. Current position, therefore, is the same both before and after you call for the write.

## REWRITE

### *Inquire Command*

X Rewrites the data base and/or partial record associated with the index entry at the current position. When you put only the X command in the command string, the current position is not changed by executing a rewrite function.

### *Required Modifiers*

None

### *Optional Modifiers*

K When you include K in the command string, INQUIRE asks you to specify a key. You can specify either a single- or multilevel key. The key is accessed before the rewriting; the current position is at the entry accessed when the rewrite is completed.

R When R is the only modifier you provided in the command string, you have specified static relative motion. Thus, this accesses the data base record for the current entry.

When both R and K appear in the command string, you have specified that you want to rewrite a data base record whose entry is in the current index level. INQUIRE asks you to specify that key, which is then accessed. The current position is at the entry accessed when the operation is completed.

→ Entering → specifies forward relative motion. Thus, the data base record for the next entry (relative to current position) is accessed. The current position is at the entry accessed when the operation is completed.

← Entering ← specifies backward relative motion. So, the data base record for the entry immediately preceding the current index entry is accessed. The current position is on the entry accessed when the rewrite is completed.

↑ Entering ↑ specifies upward relative motion. The index entry that owns the current subindex is accessed, then the keyed access (if you specified one) is done. The current position is at the entry accessed when the rewrite is completed.

(Continued on the next page)

**REWRITE (Continued)**

↓ Entering ↓ specifies down and forward relative motion. The first entry in the subindex owned by the current entry is accessed, then the keyed access (if you specified one) is done. The current position is on the entry accessed when the rewrite is completed.

I Entering I specifies an inverted rewrite, which is a linking function. INQUIRE links the data base record of the previously-accessed entry to the entry you specify. This entry cannot have a data base record linked to it. The current position is on the entry you specified when the operation is completed.

P Entering P specifies that the desired index entry has a partial record you want to rewrite.

B Entering B specifies that the data base record not be rewritten. Consequently, you should only use B when you also include P in the command string.

N The current position is not changed, regardless of positioning modifiers.

*Examples*

```
COMMANDS: X)
RECORD
FIELD #1: TOM SAWYER)
```

This rewrites the data base record for the current index entry. The current position does not change.

```
COMMANDS: X K)
KEY: BOLTS, STEEL)
RECORD
FIELD #1: 1/4 IN - $0.42 EA)
FIELD #2: 1/8 IN - $0.40 EA)
FIELD #3: 1/16 IN - $0.38 EA)
```

This rewrites the record for the multilevel key BOLTS, STEEL. The current position is now on STEEL.

```
COMMANDS: X K P B)
KEY: BOLTS, STEEL)
PARTIAL RECORD: 1/16 IN NOT A STOCK ITEM.)
```

This rewrites the partial record for the key BOLTS, STEEL. The current position is now on STEEL.

In this sequence of commands you do the following:

1)
```
COMMANDS: K)
KEY: 007)
ERROR: FUNCTION
OCTAL
400 NO DATA BASE RECORD FOR
    INDEX ENTRY
```

A keyed read, specifying key 007. INQUIRE tells you that there is no data base record for that entry. (The entry was previously written with the suppress data base flag.)

2)
```
COMMANDS: K)
KEY: BOND, JAMES)
RECORD
FIELD #1: JOB CLASS - SPY
```

A keyed read to BOND, JAMES.

3)
```
COMMANDS: X K I)
KEY: 007)
```

A rewrite inverted, specifying key 007, which links key 007 to the data base record linked to key JAMES.

4)
```
COMMANDS: N)
FIELD #1: JOB CLASS - SPY
```

A read with no position change to verify that the entry 007 is linked to the data base record JOB CLASS - SPY. The current position is now at key 007.

Note that to do a rewrite inverted, you must first access an entry linked to a data base record. The entry you specify for the rewrite inverted must not be linked to a data base record. At the completion of the rewrite both the original entry (BOND, JAMES) and the entry specified as key (007) are linked to the same data base record (JOB CLASS - SPY). The current position is on the entry specified as key.

## DELETE

### *Inquire Command*

D Logically deletes the current entry and/or data base record, when you include only D in the command string.

A logical delete does not change the current position. That is, the entry accessed prior to the delete operation is the same when the operation is completed.

### *Required Modifier*

K YOU MUST INCLUDE K IN THE COMMAND STRING TO DO A PHYSICAL DELETE.

The entry you specify as the key is accessed and deleted.

The current position is on the next entry (relative to the entry accessed) when the operation is completed. If you delete the last entry in a subindex, the current position is moved up to the entry that owns the subindex. If you delete the last entry in index level zero, the current position is moved to logically just above the level zero index.

### *Optional Modifiers*

K You can include K in the command string even if you don't want to do a physical delete. If the command string also contains ↑ or ↓, the relative motion is performed first, then the keyed access is done. You can specify either a single- or multilevel key.

If you also include R in the command string, you are specifying that the entry to be accessed is in the current index level.

R When R is the only modifier you include in the command string, you have specified static relative motion. Thus, the current entry is accessed.

↑ Specifies upward relative motion. The relative motion is done first, then the keyed access (if specified).

→ Specifies forward relative motion. The next entry (relative to current position) is accessed.

← Specifies backward relative motion. The immediately preceding entry (relative to current position) is accessed.

↓ Specifies down and forward relative motion. The relative motion is performed first, then keyed access is done (if specified).

N The current position is not changed, regardless of positioning modifiers.

### *Examples*

```
COMMANDS: D)
LOGICAL DELETION? (Y OR N): Y)
LOCAL? (Y OR N): N)
GLOBAL? (Y OR N): Y)
```

This operation marks as logically deleted the data base record linked to the current entry. The current position is not changed by this execution.

```
COMMANDS: D K)
KEY: 007)
LOGICAL DELETION? (Y OR N): N)
```

Physically deletes the index entry 007 (in index level 0) and its data base record. The current position is on the next entry (relative to 007) when the operation is completed.

```
COMMANDS: D ↑K)
KEY: BOND)
LOGICAL DELETION? (Y OR N): Y)
LOCAL? (Y OR N): Y)
GLOBAL? (Y OR N): N)
```

Marks the index entry BOND as logically deleted. BOND is in the subindex whose entry owns the current subindex when you give the command. The current position is on BOND when the operation is completed.

```
COMMANDS: D ↑K  N)
KEY: BOND)
LOGICAL DELETION? (Y OR N): Y)
LOCAL? (Y OR N): Y)
GLOBAL? (Y OR N): N)
```

This command is the same as in the previous example, except that the current position remains unchanged.

## REINSTATE

### *Inquire Command*

Q Removes the flag(s) marking the current index entry and/or data base record as logically deleted.

A reinstate operation does not change the current position when you enter Q as the only command in the command string. That is, the current position is the same when the operation is completed as it was prior to your invoking the function.

### *Required Modifiers*

None

### *Optional Modifiers*

K Specifies keyed access. If you also specify ↑ or ↓, the relative motion is done first. You can specify a single- or multilevel key. The current position is on the entry accessed when the operation is completed.

R When R is the only modifier in the command string, you are specifying static relative motion. Thus, the current entry is accessed and the current position remains unchanged when the operation is completed.

If you also include K in the command string, you are specifying keyed access starting in the current subindex. The current position is on the entry accessed when the operation is completed.

↑ Specifies upward relative motion. The relative motion is done first, then keyed access (if specified).

→ Specifies forward relative motion; this accesses the next entry (relative to current position). The current position is on the entry accessed when the operation is completed.

← Specifies backward relative motion; this accesses the immediately-preceding entry (relative to the current position). The current position is on the entry accessed when the operation is completed.

↓ Specifies down and forward relative motion. The relative motion is done first, then the keyed access (if specified). The current position is on the entry accessed when the operation is completed.

N The current position is not changed, regardless of positioning modifiers.

### *Examples*

```
COMMANDS: Q)
LOCAL? (Y OR N): Y)
GLOBAL? (Y OR N): N)
```

This operation removes the "logically deleted" flag from the current index entry. The current position does not change with this operation.

```
COMMANDS: Q K R)
KEY: 007)
LOCAL? (Y OR N): Y)
GLOBAL? (Y OR N): N)
```

This operation accesses key 007 in the current subindex, and reinstates it. The current position is on key 007 when the operation is completed.

```
COMMANDS: Q ↓ K)
KEY: BOND, JAMES)
LOCAL? (Y OR N): N)
GLOBAL? (Y OR N): Y)
```

This operation accesses key JAMES in the subindex owned by the entry with key BOND in the subindex owned by the current entry, and removes the flag marking the data base record as logically deleted. The current position is on JAMES when the operation is completed.

## DEFINE SUBINDEX

### *Inquire Command*

S Creates a subindex under the entry at the current position.

Defining a subindex does not change the current position when you make S the only command in the command string. That is, the current position is the same when the operation is completed as it was prior to your invoking the function.

### *Required Modifiers*

None

### *Optional Modifiers*

K Specifies keyed access. If you also specify ↑ or ↓, the relative motion is done first. You can specify a single- or multilevel key. The current position is on the entry accessed when the operation is completed.

R When you put R as the only modifier in the command string, you are specifying static relative motion. Thus this accesses the current entry, and the current position remains unchanged when the operation is completed.

If you also include K in the command string, you are specifying keyed access starting in the current subindex. The current position is on the entry accessed when the operation is completed.

↑ Specifies upward relative motion. The relative motion is done first, then the keyed access (if specified).

→ Specifies forward relative motion. This accesses the next entry (relative to the current position). When the operation is completed, the current position is on the entry accessed.

← Specifies backward relative motion. This accesses the immediately-preceding entry (relative to the current position). When the operation is completed the current position is on the entry accessed.

↓ Specifies down and forward relative motion. This first does the relative motion, then the keyed access (if specified). When the operation is completed the current position is on the entry accessed.

N Specifies no position change. The current position is not changed, regardless of position modifiers; it is the same after you invoke the operation as it was before.

### *Examples*

```
COMMANDS: S K)
KEY: 11,21)
MINIMUM NODE SIZE: 506)
MAXIMUM KEY LENGTH: 10)
PARTIAL RECORD LENGTH: 24)
COMPRESSED KEYS? (Y OR N): N)
SUBINDICES ALLOWED? (Y OR N): Y)
```

Access the specified key (11,21). Define a subindex under that entry according to the specifications.

The current position is on key 21 when the operation is completed.

```
COMMANDS: S R K)
KEY: JAMES)
USE PREVIOUS INFO? (Y OR N): Y)
```

Access the entry whose key is JAMES in the current subindex, and define a subindex under it according to previously-supplied subindex definition information.

The current position is at key JAMES when the operation is completed.

```
1) COMMANDS: K)
   KEY: 007)
   FIELD #1: JOB CLASS - SPY

2) COMMANDS: S)
   USE PREVIOUS INFO? (Y OR N): N)
   MINIMUM NODE SIZE: 506)
   MAXIMUM KEY LENGTH: 5)
   PARTIAL RECORD LENGTH: )
                          0
   COMPRESSED KEYS? (Y OR N): )
                              N
   SUBINDICES ALLOWED? (Y OR N): )
                                 N
```

In this sequence you: 1) positioned to entry 007, whose data record is displayed; and 2) defined a subindex under that entry.

The current position is at key 007 when the operation is completed.

```
COMMANDS: S K R N)
KEY: BOND)
USE PREVIOUS INFO? (Y OR N): Y)
```

This sequence defines a subindex under BOND in the current subindex. Executing this command string does not change the current position.

## LINK SUBINDEX

### *Inquire Command*

L Reproduces the current entry's link to its subindex for the entry whose key you are asked to specify. You must be positioned at an entry with a subindex defined under it (source key), and you must specify a destination key that does not have a subindex under it.

When L is the only command in the command string, the current position is on the source key when the operation is completed.

### *Required Modifiers*

None

### *Optional Modifiers*

K Specifies keyed access to the source key. If you also specify ↑ or ↓ the relative motion is done first. You can specify a single- or multi-level source key. The current position is on the entry you specify as the source key when the operation is completed.

R When you provide only the R modifier in the command string you have specified static relative motion. Thus, this uses the current entry as the source key. The current position is on the entry you specify as the source key when the operation is completed.

When you also include K in the command string, you are specifying keyed access starting at the current subindex. You can specify a single- or multilevel key as the source key. The current position is on the entry you specify as the source key when the operation is completed.

↑ Specifies upward relative motion. The relative motion is done first, then the keyed access (if specified).

→ Specifies forward relative motion. This accesses the next entry (relative to the current position).

← Specifies backward relative motion. This accesses immediately-preceding entry (relative to the current position).

↓ Specifies down and forward relative motion. This first does the relative motion, then the keyed access (if specified).

N Specifies no position change. The current position is not changed, regardless of position modifiers.

### *Examples*

```
COMMANDS: L)
DESTINATION KEY: 01,11,21)
RELATIVE KEY? (Y OR N): N)
```

You are positioned at an entry with a subindex defined under it. You specify a three level key (01,11,21) as the destination entry, which does not have a subindex under it. In this example the source key is not in the same subindex as the destination key. Hence, you specified that the destination key is not a relative key.

The current position is at the source key when the operation is completed.

```
COMMANDS: L ↑ )
DESTINATION KEY: BOND)
RELATIVE KEY? (Y OR N): Y)
```

You specified the source key as the entry that owns the current subindex. The destination key (BOND) is in the same subindex as the source key. You therefore specified that it is a relative key.

The current position is on the source key when the operation is completed.

```
COMMANDS: L R K)
KEY: 007)
DESTINATION KEY: BOND, JAMES )
RELATIVE KEY? (Y OR N): N)
```

You positioned to key 007 in the current subindex. Key 007 becomes the source key. The destination key (BOND, JAMES) is in a different subindex so it is not a relative key.

The current position is on 007 key when the operation is completed.

## DELETE SUBINDEX

### *Inquire Command*

U When U is the only command in the command string, you execute one of two possible actions.

1) If the subindex linked to the entry at the current position is not linked to by any other entry in the index structure, (and no subindexes are defined under any of its entries) then U physically deletes all entries in the subindex, and the subindex structure itself.

2) If the subindex linked to the entry at the current position has other links to it, then U deletes only the link between the current entry and the subindex.

When the operation is completed the current position is at the entry that was previously linked to the subindex.

### *Required Modifiers*

None

### *Optional Modifiers*

K Specifies keyed access. If you also specify ↑ or ↓ the relative motion is done first. You can specify a single- or multilevel key. The current position is on the entry accessed when the operation completes.

R When you provide only the R modifier in the command string, you specify static relative motion. Thus, this accesses the current entry, and the current position does not change when the operation completes.

If you also include K in the command string you are specifying keyed access starting in the current subindex. The current position is on the entry accessed when the operation completes.

↑ Specifies upward relative motion. The relative motion is done first, then keyed access (if specified). The current position is on the entry accessed when the operation completes.

→ Specifies forward relative motion. This accesses the next entry (relative to the current position). The current position is on the entry accessed when the operation completes.

← Specifies backward relative motion. This accesses the immediately-preceding entry (relative to the current position). When the operation is completed, the current position is on the entry accessed.

↓ Specifies down and forward relative motion. This first does the relative motion, then the keyed access (if specified). When the operation is completed, the current position is on the entry accessed.

N Specifies no position change. The current position is not changed, regardless of position modifiers; it is the same after you invoke the operation as it was before.

### *Examples*

```
COMMANDS: U)
```

Deletes the link between the current entry and its subindex. If there are no other links to or from the subindex, this physically deletes the subindex and its entries. The current position is not affected by this operation.

```
COMMANDS: U K R)
KEY: 007)
```

This accesses key 007 in the current subindex, and deletes the link between it and its subindex. (The subindex may or may not be deleted, depending on its link status.)

The current position is on key 007 when the operation completes.

```
COMMANDS: U↑)
```

This accesses the entry that owns the current subindex, and deletes the link between that entry and the subindex. (The subindex may or may not be deleted, depending on its link status.)

The current position is on the entry accessed when the operation completes.

```
COMMANDS: U K R N)
KEY: 007)
```

This unlinks the subindex, but doesn't change the current position.

## CLOSE

### *Inquire Command*

ESCAPE — Enter this command by depressing the ESC key on the console.

When you enter ESC, INQUIRE closes the ISAM/DBAM file you opened when you invoked the INQUIRE utility.

### *Required Modifiers*

None

### *Optional Modifiers*

None

### *Examples*

```
COMMANDS: ESCAPE)
INDEX: )
R
```

This closes the current file. Because you responded with a carriage return to the prompt "INDEX:", you return to the CLI.

```
COMMANDS: ESCAPE)
INDEX: ACCTS)
OVERRIDE # OF BUFFERS? (Y OR N): )
                                 N
ENTER C<CR> TO GET LIST OF COMMANDS
```

This closes the current file and opens a new file, ACCTS.

## RETRIEVE HIGH KEY

### *Inquire Command*

H — When you key in H as the only command in the command string, it retrieves the highest key in the current subindex. The current position is not changed by this operation.

### *Required Modifiers*

None

### *Optional Modifiers*

K — When you use K in the command string, you are asked to specify a key. You can specify either a single- or multilevel key; you thus access that key and retrieve the highest key in its subindex. The current position is on the key accessed when the operation completes.

↓ — When you use ↓ in the command string, the current position moves to the first entry in the subindex owned by the current entry. The highest key in that lower subindex is retrieved.

↑ — When you use ↑ in the command string, the current position moves to the entry that owns the current subindex. The highest key in that higher subindex is retrieved.

N — The current position is not changed, regardless of motion modifiers.

### *Examples*

```
COMMANDS: H)
HIGH KEY: 007
```

This retrieves the highest key in the current subindex. The current position does not move with this operation.

```
COMMANDS: H↓)
HIGH KEY: JAMES
```

This retrieves the highest key in the subindex owned by the current entry. The current position is on the first key in the owned subindex when the operation completes.

```
COMMANDS: H↑ )
HIGH KEY: 007
```

This retrieves the highest key in the subindex of the owner entry. The current position is on the owner entry when the operation completes.

## RETRIEVE KEY

### *Inquire Command*

V When V is the only command in the command string, it retrieves the key (and its occurrence number) for the current entry. The current position is not changed by this operation.

### *Required Modifiers*

None

### *Optional Modifiers*

K Specifies keyed access. If you also specify ↑ or ↓, the relative motion is done first. When the operation is completed, the current position is on the entry accessed.

R When you provide only the R modifier in the command string, you have specified static relative motion. Thus, this retrieves the status for the current entry. The current position does not change.

When you also include K in the command string, you are specifying keyed access starting in the current subindex. You can specify either a single- or multilevel key. When the operation is completed, the current position is on the entry accessed.

↑ Specifies upward relative motion. The relative motion is done first, then the keyed access (if specified).

→ Specifies forward relative motion. This accesses the next entry (relative to the current position). When the operation is completed, the current position is on the entry accessed.

← Specifies backward relative motion. This accesses the immediately-preceding entry (relative to the current position). When the operation is completed, the current position is on the entry accessed.

↓ Specifies down and forward relative motion. You first access the first entry in the subindex owned by the current entry, then do the keyed access (if specified). The current position is on the entry accessed when the operation completes.

N Specifies no position change. The current position is not changed, regardless of position modifiers; it is the same after you invoke the operation as it was before.

### *Examples*

```
COMMANDS: V)
RETRIEVED KEY: 007
OCCURRENCE: 0
```

Retrieves the key for the current position entry.

```
COMMANDS: V K R)
KEY: BOND)
RETRIEVED KEY: BOND
OCCURRENCE: 0
```

Retrieves key BOND in the current subindex. The current position is on BOND when the operation completes.

```
COMMANDS: V K R N)
KEY: BOND)
RETRIEVED KEY: BOND
OCCURRENCE: 0
```

Retrieves the key BOND in the current subindex, but the current position remains changed.

```
COMMANDS: V ←)
RETRIEVED KEY: 006
OCCURRENCE: 0
```

Moves backward to immediately preceding entry (relative to current position), and retrieves key 006. The current position is on key 006 when the operation completes.

## RETRIEVE STATUS

### *Inquire Command*

E When E is the only command in the command string, it retrieves the status of the entry at the current position. The current position does not change.

### *Required Modifiers*

None

### *Optional Modifiers*

K Specifies keyed access. If you also specify ↑ or ↓, the relative motion is done first. When the operation is completed the current position is on the entry accessed.

R When you provide only the R modifier in the command string, you have specified static relative motion. Thus, this retrieves the status for the current entry. The current position does not change.

When you also include K in the command string, you are specifying keyed access starting in the current subindex. You can specify either a single- or multilevel key. When the operation is completed, the current position is on the entry accessed.

↑ Specifies upward relative motion. The relative motion is done first, then the keyed access (if specified).

→ Specifies forward relative motion. This accesses the next entry (relative to the current position). When the operation is completed, the current position is on the entry accessed.

← Specifies backward relative motion. This accesses the immediately-preceding entry (relative to the current position). When the operation is completed, the current position is on the entry accessed.

↓ Specifies down and forward relative motion. This first does the relative motion, then the keyed access (if specified). When the operation is completed, the current position is on the entry accessed.

N Specifies no position change. The current position is not changed, regardless of position modifiers; it is the same after you invoke the operation as it was before.

### *Examples*

```
COMMANDS: E)
RECORD LENGTH: 5
KEY LENGTH: 2
```

Retrieves the status of the current entry. The current position does not change.

```
COMMANDS: E↑)
RECORD LENGTH: 80
KEY LENGTH: 6
```

Retrieves the status for the entry that owns the current subindex. The current position is on the entry accessed when the operation completes.

```
COMMANDS: E K R N)
KEY: 007)
RECORD LENGTH: 60
KEY LENGTH: 5
```

Retrieves the status for key 007 in the current subindex. The current position does not change.

## RETRIEVE SUBINDEX DEFINITION

### *Inquire Command*

G When G is the only command in the command string, it retrieves the definition for the current subindex. The current position does not change.

### *Required Modifiers*

None

### *Optional Modifiers*

↑ Specifies upward relative motion. This accesses the entry that owns the current subindex. The current position is on the entry accessed when the operation completes.

↓ Specifies down and forward relative motion. This accesses the first entry in the subindex owned by the current entry. The current position is on the entry accessed when the operation completes.

N Specifies no position change. Either the definition for the current subindex or the definition for the index indicated by the ↑ or ↓ modifier is retrieved. The current position does not change.

### *Examples*

```
COMMANDS: G)
MINIMUM NODE SIZE: 506
MAXIMUM KEY LENGTH: 10
PARTIAL RECORD LENGTH: 24
KEYS ARE NOT COMPRESSED
SUBINDICES ARE ALLOWED
```

Retrieves the current subindex's definition. The current position does not change.

```
COMMANDS: G↓)
MINIMUM NODE SIZE: 506
MAXIMUM KEY LENGTH: 5
PARTIAL RECORD LENGTH: 0
KEYS ARE NOT COMPRESSED
SUBINDICES ARE NOT ALLOWED
```

Retrieves the definition for the subindex owned by the current entry. The current position is on the first entry in the owned subindex when the operation completes.

# COMMAND MODIFIERS

## Keyed Access

### *Command Modifier*

K This modifier tells INQUIRE to use keyed access in positioning to the desired entry. If R, ↑, or ↓ also appear in the command string, it does the relative motion first. Otherwise, keyed access begins at index level zero.

Whenever you include K in the command string, INQUIRE asks you to specify a key. You can enter either a single- or multilevel key, but the first in a multilevel key must be relative to the current index level, or be in index level zero.

Multilevel keys must be separated with commas. For example, 007 JAMES BOND is a single level key and 007, JAMES, BOND is a three level key.

Whenever K modifies a write operation, the last or only key you enter on the key definition line must be an actual key; that is, it must be a key you want to become part of the subindex. When you want to enter a duplicate key into a subindex, follow the key with the # symbol. For example, to write a duplicate of key 007 in the level zero subindex, do the following:

```
COMMANDS: KW)
KEY: 007#)
RECORD
FIELD #1: data)
```

If, subsequently, you want to physically delete that index entry and data record, do the following:

```
COMMANDS: KD)
KEY: 007#1)
LOGICAL DELETION? (Y OR N) N)
```

Note that if you don't tell INQUIRE that the key you are deleting is a duplicate, and give the occurrence number of the duplicate you want deleted, INQUIRE deletes the first occurrence of the key.

Whenever K modifies any other command (not write), you can use an approximate key, a generic key, or a duplicate key.

If you want to use an approximate key you can specify one or more characters of an actual key which you follow with the + character. INQUIRE returns the data record of the first index entry whose key value is equal to or greater than the actual characters you specify. For example, suppose you used the keys DAGGER, DAGMAR, and DAGWOOD in a single subindex. If you enter:

```
COMMANDS: KR)
KEY: DAGK+)
```

INQUIRE will return the data record for the index entry whose actual key is DAGMAR.

If you want to use a generic key you can specify one or more characters of an actual key which you follow with the - character. INQUIRE returns the data record of the first index entry whose key value exactly matches the actual key characters you enter, up to the length of the characters you enter. For example, again suppose the keys DAGGER, DAGMAR, and DAGWOOD are in a single subindex. If you enter:

```
COMMANDS: KR)
KEY: DAG-)
```

INQUIRE returns the data record for the index entry whose key is DAGGER. However, if you enter:

```
COMMANDS: KR)
KEY: DAGK-)
```

INQUIRE returns an error, since no key in this example has the first four characters DAGK.

If you want to use a duplicate key you should also include the occurrence number you want. If you do not, INQUIRE returns the first (zero) occurrence. For example, if you have defined a subindex under an index entry whose key is GEORGIA and all the keys in that subindex are duplicates of the key PLAINS, you can access the first (zero) occurrence of PLAINS by entering:

```
COMMANDS: K)
KEY: GEORGIA, PLAINS)
```

but, if you want to access the hundredth occurrence of PLAINS you could enter:

```
COMMANDS: K)
KEY: GEORGIA, PLAINS#99)
```

Similarly, you can use an approximate or generic key as the last level in a multilevel key definition. For example, if the subindex containing DAGGER, DAGMAR, and DAGWOOD, was defined under an index entry whose key is simply D, you could access the subindex by entering:

```
COMMANDS: K)
KEY: D, DAG+)
```

or by entering:

```
COMMANDS: K)
KEY: D, DAG-)
```

### Relative Motion

#### *Command Modifier*

R — This modifier specifies relative motion for accessing the desired entry. When you don't include K in the command string, you specify static relative motion. When K does appear in the command string, R specifies that the key is relative to the current subindex.

### Motion Control

#### *Command Modifiers*

←↑↓→ — You enter these modifiers using the corresponding cursor control keys on the console keyboard.

↑ — Specifies upward relative motion. This accesses the entry that owns the current subindex.

→ — Specifies forward relative motion. This accesses the next entry (relative to current position).

← — Specifies backward relative motion. This accesses the immediately-preceding entry (relative to current position).

↓ — Specifies down and forward relative motion. This accesses the first entry in the subindex owned by the current position entry.

### Suppress Data Base

*Command Modifier*

B You can use this modifier with read, write, and rewrite operations. It inhibits the retrieval of a data record on a read operation, and precludes writing a data record on a write or rewrite operation.

### Inverting

*Command Modifier*

I When you use it with the write operation, this modifier specifies inverting. An index entry is written for an existing data base record.

When you use it with a rewrite operation, this modifier specifies a linking operation. INQUIRE asks for a key; you specify an entry to which you want to link the data base record of the current position entry.

### Don't Set Current Position

*Command Modifier*

N This modifier specifies no position change. For an operation that normally sets the current position at the entry accessed, this command prevents setting a new current position.

### Include Partial Record

*Command Modifier*

P You can use this modifier with read, write, and rewrite operations. A read operation displays the partial record; a write or rewrite writes or modifies the partial record.

### Data Record Feedback

*Command Modifier*

A This modifier returns the octal value of the data record link, and the index level of the current position entry.

## INQUIRE FUNCTIONS

### Display List of INQUIRE Commands

*Inquire Command*

C Entering a command string of just C) displays all INQUIRE commands, as illustrated in the section on invoking INQUIRE.

### Change Output File

*Inquire Command*

O The output file is the file to which INQUIRE writes the commands and the results of INQUIRE operations. Normally, this is the $TTO file (i.e., the video terminal).

When you key the O command, INQUIRE asks you to name a new output file, which INQUIRE then creates. All subsequent results of INQUIRE commands are recorded in this file.

If you use the O command once, and want to return the output to the terminal, enter a second O command; when you're asked to name the output file, enter $TTO).

### Set Key and/or Record Formats

*Inquire Command*

F Normally, INQUIRE interprets the content of keys, partial records, and data records as ASCII characters, and the data record as a single field of 80 (or fewer) characters. If the file you are examining with INQUIRE does not conform to these conventions then you should use the F command so that INQUIRE will properly retrieve and display the keys, partial records, and data records.

You need define formats only once. INQUIRE will save the format specifications in a file you name so that whenever you examine the ISAM/DBAM file, you can specify the formats simply by naming the file containing the appropriate formats.

You set formats according to the dialogue shown in follows Table 4-1.

Table 4-1. INQUIRE Format Dialogue

| INQUIRE SAYS: | YOUR RESPONSE | ACTION |
|---|---|---|
| FORMATS IN FILE? (Y OR N) | Y | INQUIRE asks for the name of the file in which you have previously stored the desired formats. |
| | N | You will now specify key and/or record formats. |
| NUMBER OF LEVELS: | number | Enter the number of index levels in which you want the formats set. |
| LEVEL # n | | INQUIRE displays this message once for each index level you said you want. Each time this message appears, you define key, partial record, and data record formats for an index level. |
| KEY<br>NUMERIC? (Y OR N): | N | The keys for this index level are defined as ASCII value keys, so you're not asked the next two questions. |
| RADIX (8, 10, OR 16): | number | Enter the number for the desired radix. |
| BYTES (B) OR WORDS (W): | B | Each byte in the key field is interpreted as a numeric value according to the specified radix. |
| | W | Each word in the key field is interpreted as a numeric value according to the specified radix. |
| PARTIAL RECORD<br>NUMERIC? (Y OR N) | N | The partial records at this index level consist of ASCII characters, so you're not asked the next two questions. |
| RADIX? (8, 10, OR, 16) | number | Enter the number for the desired radix. |
| BYTES (B) OR WORDS (W): | B | Each byte in the partial record field is interpreted as a numeric value according to the specified radix. |
| | W | Each word in the partial record field is interpreted as a numeric value according to the specified radix. |

(Continued on the next page)

Table 4-1. INQUIRE Format Dialogue (Concluded)

| INQUIRE SAYS: | YOUR RESPONSE | ACTION |
|---|---|---|
| NUMBER OF FIELDS: | number | Enter the decimal number (1 or greater) corresponding to the number of fields in the record. You then define each record with the following questions: |
| FIELD # n | | INQUIRE displays this message before you define the format for each field in the record. |
| NUMERIC? (Y OR N): | N | The content of this field is in ASCII characters. You're not asked the RADIX and BYTES OR WORDS questions but you are asked the STARTING POSITION and LENGTH questions. |
| | Y | The content of this field is numeric, as you define by answering the next four questions. |
| RADIX? (8, 10, OR, 16): | number | Enter the number for the desired radix. |
| BYTES (B) OR WORDS (W): | B | Each byte in the field is interpreted as a numeric value according to the specified radix. |
| | W | Each word in the field is interpreted as a numeric value according to the specified radix. |
| STARTING CHARACTER POSITION (1-n): | number | Enter the decimal number of the byte in the record at which this field begins. |
| FIELD LENGTH, BYTES: | number | Enter the length, in bytes, of this field. |
| SAVE FORMATS IN FILE? (Y OR N): | Y | You can save the key, partial record, and data record formats defined during the current execution of INQUIRE and use them as input in subsequent executions of INQUIRE. INQUIRE will next ask you for the name of the save file. |
| | N | The formats you define are valid only for the current execution of INQUIRE, and you lose them when you dismiss INQUIRE or define new formats. |
| FILE NAME: | name | If you responded with Y to the SAVE FORMATS question, INQUIRE builds an RDOS file in which it stores the formats you specified. |

### Set Repeat Count

*Inquire Command*

#

You can use the repeat count feature when you want to examine several records at once. The keys for the records must all be at the same index level.

For example, if you enter the following command string:

```
COMMANDS: → #)
REPEAT COUNT: 4)
```

the results are:

```
RECORD
FIELD #1: JAMES BOND
FIELD #2: KIM PHILBY
FIELD #3: MATA HARI
FIELD #4: GARY POWERS
```

### Delete Line

*Inquire Command*

RUBOUT

You enter this command by pressing the RUBOUT key; it deletes the command string you've just typed in. For example:

```
COMMANDS: R W J RUBOUT)
COMMANDS:
```

You can also use RUBOUT to correct mis-entered information, in the same fashion as when you are at the CLI level.

END OF CHAPTER

# CHAPTER 5

# FILE COPY UTILITY

## INTRODUCTION

You'll be using ICOPY primarily to copy from a SAM source file to a SAM destination file and from or to an ISAM/DBAM file. A source file, the file copied from, can reside on disk or magnetic tape (either labeled or unlabeled) and need not be an INFOS file. A destination file, the one copied to, must be an INFOS file but need not exist when you invoke ICOPY.

If the destination file exists, it can reside on disk or magnetic tape (labeled or unlabeled) and, if it is a SAM or RAM file, you can specify that the records copied either overwrite the existing file, or be appended to it. If the destination file does not exist when you invoke ICOPY, it will automatically create a SAM file. You can specify the line printer as the destination file.

You can also direct ICOPY to format the records being copied to a SAM file for subsequent printer output. The system thus adds a line feed and carriage return to each record, and defines them as Data Sensitive records.

You can specify additional controls when copying from or to an ISAM/DBAM file. When an ISAM/DBAM file is the source, you can specify that either all or only a portion of the data base file be copied. (In both cases, no part of the index structure is copied; only the data base records.) You can copy all or part of the records for any single index level, or for as many levels as you want by specifying where the copy is to start and how many index levels you want ICOPY to descend.

An ISAM/DBAM file must already exist when you specify it as the destination file. Each time you invoke ICOPY, it copies to a single index level. If a subindex has not been defined when you invoke ICOPY to copy to it, you are asked to define it in the ICOPY dialogue. Whenever you copy to an ISAM/DBAM file, you are asked to specify the key, partial record, and data base record fields in the source record.

## INVOKING ICOPY

To invoke ICOPY, key in a CLI command line of the form:

```
ICOPY[/S][sourcefile[/L][/M]
        [destinationfile[/L][/M][/P]]]
        [logfile/A]
```

where:

```
/S  =  Short form of the utility
/M  =  Unlabeled magnetic tape file
/L  =  Labeled magnetic tape file
/P  =  Print format file
```

The more information you put on the command line, the fewer ICOPY prompts you'll have to respond to. As you can see, there is no switch to indicate that either the source or destination file is a disk file. Consequently, if you do not include /M or /L with either file name, you have, in effect, specified a disk file, unless you make a complete mag tape file specification (e.g., MT0:0) or unless the destination is the printer. For example:

```
ICOPY PARTS/M $LPT)
```

If the destination file is not the printer and you want the source records formatted for printing, use the /P switch with the destination file name. For example:

```
ICOPY PARTS/L INV/P)
```

If INV does not exist, ICOPY will create a SAM disk file; when the system copies the records it adds a line feed and carriage return to each record. You can later print INV by:

```
XFER INV $LPT )
```

If either the source or destination is an unlabeled tape file, you can use a device specifier on the command line. For example instead of entering:

```
ICOPY 0/M 0/M )
```

you can enter:

```
ICOPY MT0:0 MT1:0)
```

You can build a dialogue file for regularly-recurring copy operations. This saves you the effort of repeatedly responding to the ICOPY prompts each time you want to copy the same source to the same destination. You do this by making a log file of the original ICOPY command line. For example:

```
ICOPY PARTS/L SAVE LOG/A)
```

When you want to copy PARTS to SAVE again all you need do is delete SAVE and then enter:

```
ICOPY PARTS/L SAVE LOG/B)
```

If you don't want to delete SAVE then you can edit LOG to append the records to the file SAVE, and then enter:

```
ICOPY PARTS/L SAVE LOG/B)
```

This edit need be done only once.

If you do not include the source file and destination file names and local switches on the command line, you will get a preliminary dialogue (prior to the main dialogue described in Tables 5-2 and 5-3) whose questions and responses are described in Table 5-1. These are shown in the sequence in which they appear on the system console.

Table 5-1. ICOPY Preliminary Dialogue

| PROMPT | YOUR RESPONSE | DESCRIPTION |
|---|---|---|
| SOURCE FILE: | (name) | Enter the name of the source file. |
| MAG TAPE FILE? (Y OR N) | N or ) | The source file is on disk. |
| | Y | The source file is on magnetic tape, and you're asked the next question. |
| LABELED? (Y OR N) | Y | The source file is a labeled magnetic tape file. |
| | N | The source file is an unlabeled magnetic tape file. |
| DESTINATION FILE: | (name) | Enter the name of the destination file, or $LPT, to print the file. |
| MAG TAPE FILE? (Y OR N) | N or ) | The destination file is on disk. |
| | Y | The destination file is on magnetic tape, and you're asked the next question. |
| LABELED? (Y OR N) | Y | The destination file is a labeled magnetic tape file. |
| | N | The destination file is an unlabeled magnetic tape file. |
| FORMAT FILE FOR PRINTING? (Y OR N) | Y | ICOPY will add a line feed/carriage return to each source record copied to the destination file, so that you can subsequently print the destination file. |
| | N or ) | Records written to the destination file are not formatted for printing. |

## ICOPY SAM SOURCE FILE TO SAM DESTINATION FILE SUMMARY

After you invoke ICOPY (and complete the preliminary dialogue, if needed), it takes you through a further dialogue. The prompts it gives you in the dialogue depend on the nature of the source file, whether the destination file is new or old, and whether the files are on disk, tape, or the line printer.

Some prompts accept a default response (which is the carriage return) and some don't. If you default an entry, ICOPY displays the system-assigned value. If a default response is unacceptable, it repeats the prompt.

Tables 5-2 and 5-3 (which are each in alphabetical order) summarize the prompts and describe the responses to them. Table 5-2 contains source file prompts, and Table 5-3 destination file prompts.

Table 5-2. SAM SOURCE FILE PROMPTS

| PROMPT | YOUR RESPONSE | DESCRIPTION |
|---|---|---|
| BLOCK SIZE: | | This question is asked only when the source file is on magnetic tape, or when it is a non-INFOS disk file. |
| | number | Enter, in decimal, the size in bytes of a block of records. |
| | ) | The system default (80 bytes for tape, 512 bytes for disk) is used for block size. |
| CODE TRANSLATION ON INPUT? (Y OR N) | | This question is asked for all source files. Your response determines whether or not you're asked the remaining questions. |
| | N or ) | No code translation. |
| | Y | If you enter Y, indicating you want the source file records translated before they're written to the destination file, you'll be asked the following series of questions. |
| EBCDIC TO ASCII (Y OR N) | Y | ICOPY translates the source file records (recorded in EBCDIC) to ASCII before writing them to the destination file. |
| | N or ) | You do not want EBCDIC to ASCII translation, so you're asked the next question. |
| ASCII TO EBCDIC (Y OR N) | Y | ICOPY translates the source file records (recorded in ASCII) to EBCDIC before writing them to the destination file. |
| | N or ) | These responses mean you do not want ASCII to EBCDIC translation. |

(Continued on the next page)

Table 5-2. SAM SOURCE FILE PROMPTS (Continued)

| PROMPT | YOUR RESPONSE | DESCRIPTION |
|---|---|---|
| SELECTIVE FIELD TRANSLATION (Y OR N) | | This question is asked only if you specified code translation on input. |
| | N or ) | These responses mean you want the entire source record translated to the previously-selected code. |
| | Y | Entering Y indicates you only want certain fields of the source record translated to a different recording code. Thus, you're asked the next set of questions. |
| ***ENTER CARRIAGE RETURN TO TERMINATE LOOP*** | | |
| FIELD 1: STARTING CHARACTER POSITION (1-n): | number | The first character of the source record is number 1. Enter the decimal number of the character where the field to be translated begins. |
| LENGTH (CHARACTERS): | number | Enter, in decimal, the length of the field to be translated. |
| . . . | | The set of questions repeats until you enter ) to the prompt: STARTING CHARACTER POSITION (1-n): |
| ENABLE RUNTIME INITIALIZATION? (Y OR N) | | This question is asked only if the source file is a labeled magnetic tape file. |
| | Y | The system partially initializes each volume of the source file. |
| | N or ) | These responses imply that you have taken care of initializing each volume of the source file (which you can do with the LBINIT utility). |

(Continued on the next page)

Table 5-2. SAM SOURCE FILE PROMPTS (Continued)

| PROMPT | YOUR RESPONSE | DESCRIPTION |
|---|---|---|
| GENERATION NUMBER: | | This question is asked only if the source file is a labeled magnetic tape file. |
| | number | Enter the six-digit generation (and version) number of the source file recorded on the HDR1 label. |
| | ) | The system opens the first file whose name matches the specified source file name. |
| LABEL LEVEL: | | This is asked only if the source file is a labeled magnetic tape file. |
| | number | Enter the number 1, 2, or 3 that corresponds with the level number of the source file labels. |
| | ) | This response means the source file has level 1 labels. |
| LABEL TYPE (A=ANSI, I=IBM): | | This question is asked only if the source file is a labeled tape. |
| | I | The label type is IBM. |
| | A or ) | The label type is ANSI. |
| NUMBER OF BUFFERS: | | This question is asked for all source files. |
| | number | Enter the number of buffers you want used in reading the source file. |
| | ) | The number of buffers comes from the permanent file specification, or is defaulted to 1. |
| NUMBER OF VOLUMES: | | This question is asked for all source files on magnetic tape. |
| | number | Enter the number of volumes of the source file. |
| | ) | The source file has only one volume. |

(Continued on the next page)

Table 5-2. SAM SOURCE FILE PROMPTS (Continued)

| PROMPT | YOUR RESPONSE | DESCRIPTION |
|---|---|---|
| RECORD FORMAT: | | This question is asked for all source files for which there is no permanent file specification. |
| | F | Fixed-length records. |
| | V | Variable-length records. |
| | U | Undefined-length records. |
| | D | Data-Sensitive records. |
| | | You can't default this question. If you try, ICOPY repeats it. |
| RECORD LENGTH:<br>or<br>MAXIMUM RECORD LENGTH: | | This question is asked for disk and tape source files for which you specified fixed-length records, and for tape files with variable-length records. |
| | number | Enter, in decimal,:<br>1) the length (in bytes) of fixed-length records<br>or<br>2) the maximum length of variable-length records. |
| | ) | The variable-length record length is equal to block size. You cannot default this answer for fixed-length records. |
| REWIND ON CLOSE? (Y OR N) | | This question is asked for all tape source files. |
| | N or ) | For both labeled and unlabeled tape files the tape remains positioned<br>1) just before the next file on the tape, or<br>2) at the logical end-of-tape mark. |
| | Y | The tape, labeled or unlabeled, is rewound to the physical beginning-of-tape mark.<br>If the tape is labeled, you're asked the Runtime Release question. |
| RUNTIME RELEASE? (Y OR N) | N or ) | Entering either of these responses means you will take care of releasing each volume of the source file. |
| | Y | Entering Y directs the system to automatically release each volume of the source file. |

(Continued on the next page)

Table 5-2. SAM SOURCE FILE PROMPTS (Concluded)

| PROMPT | YOUR RESPONSE | DESCRIPTION |
|---|---|---|
| REWIND ON OPEN? (Y OR N) | | This question is asked for all source files on magnetic tape. |
| | Y | ICOPY rewinds the tape to the physical beginning-of-tape mark before searching for the start of the source file. |
| | N or ) | ICOPY searches for the source file from the current position on the tape. If found, the copy begins; if not found, it rewinds the tape automatically to the physical beginning-of-tape mark and ICOPY terminates abnormally. |
| SEQUENCE NUMBER: | | This question is asked only for labeled tape source files. |
| | number | Enter the four-digit file sequence number recorded in the file's HDR1 label. |
| | ) | Do not use the file's sequence number in the search for the source file. |
| VARIABLE LENGTH BLOCKS? (Y OR N) | | This question is asked for all source files on magnetic tape. |
| | Y | Use this response if:<br>1) you know the file has variable-length blocks, or<br>2) there is a possibility that the last block is short. |
| | N or ) | Use this response only if all blocks in the file are the same length. |
| VOLUME #1 ID: | | This question is asked for all source files on magnetic tape. |
| | volid | If the file is unlabeled, enter the RDOS tape unit specifier on which you mounted the volume.<br><br>If the file is labeled and you are using the runtime functions, enter the value in the corresponding VOL1 label. Whatever you enter must exactly match the VOL1 contents. |

Table 5-3. SAM DESTINATION FILE PROMPTS

| PROMPT | YOUR RESPONSE | DESCRIPTION |
|---|---|---|
| APPEND OUTPUT TO FILE? (Y OR N) | | This question is asked for existing disk and tape destination files. |
| | Y | This response positions the file to the logical end-of-file before any source records are copied to it. |
| | N or ) | These responses copy the source records to the destination file from the logical beginning-of-file. The last record copied becomes the end-of-file, and whatever formerly existed beyond that point is lost. |
| BLOCK SIZE: | | This question is asked for tape and new disk destination files. |
| | number | Enter, in decimal, the size (in bytes) of a block of records. |
| | ) | The system default (80 bytes for tape and 512 bytes for disk) is used. |
| ENABLE RUNTIME INITIALIZATION? (Y OR N) | | This question is asked only when the destination is a labeled magnetic tape file. |
| | Y | The system partially initializes each volume of the file. |
| | N or ) | These responses imply that you have initialized each volume of the destination file (which you can do with LBINIT). |
| FORCE END OF VOLUME ON OUTPUT? (Y OR N) | | This question is asked only if the destination is a multivolume file. |
| | N or ) | These responses mean the destination is a single volume file of sufficient size to contain all the source records you want copied to it. You control the number of records copied, and you can limit this number if you want. |
| | Y | This response means the destination is a user created multi volume file, and you want a limited number of records in each volume. With your Y response here, ICOPY asks you the Records Per Volume question. |

(Continued on the next page)

Table 5-3. SAM DESTINATION FILE PROMPTS (Continued)

| PROMPT | YOUR RESPONSE | DESCRIPTION |
|---|---|---|
| RECORDS PER VOLUME: | number | Enter, in decimal, the number of records you want in each volume of the destination file. |
| GENERATION NUMBER: | | ICOPY asks this question only when the destination is a labeled magnetic tape file. |
| | number | Enter the six-digit generation (and version) number recorded in the destination file's HDR1 label. If you do this, the system must find an exact match in a HDR1 label to successfully copy the named file. If you specify a generation number and it can't find the file, ICOPY rewinds the tape automatically to the beginning-of-tape mark. |
| | ) | ICOPY will not search for the destination file by generation number. |
| LABEL TYPE (A=ANSI, I=IBM): | A or ) | The file has ANSI labels. |
| | I | Enter I if the file has IBM labels. |
| LIMIT TOTAL NUMBER OF RECORDS? (Y OR N) | | This question is asked for all destination files. |
| | N or ) | Copy all records in the source file to the destination file. |
| | Y | This response means you want to copy a limited number of records, specified by Record Count, to the destination. |
| RECORD COUNT: | number | Enter, in decimal, the number of records from the beginning of the source file that you want copied. |
| MONITOR RECORD COUNT? (Y OR N) | | This question is asked for all destination files. |
| | Y | ICOPY will display the sequential number of each source file record on the video terminal as it copies the record to the destination. |
| | N or ) | ICOPY does not display the sequential number of each source record. |
| | | In either case, ICOPY will always give you the total number of records copied. |

(Continued on the next page)

Table 5-3. SAM DESTINATION FILE PROMPTS (Continued)

| PROMPT | YOUR RESPONSE | DESCRIPTION |
|---|---|---|
| NUMBER OF BUFFERS: | | This question is asked for all destination files. |
| | number | Enter, in decimal, the number of buffers you want ICOPY to use in writing to the destination file. |
| | ) | ICOPY will use the system-default number of buffers (1 for all tape files and 1 or 2 for disk files). |
| NUMBER OF VOLUMES: | | This question is asked for all destination files on magnetic tape. |
| | number | Enter, in decimal, the number of logical volumes of the destination file. |
| | ) | This response means the destination is a single volume file. |
| READ-AFTER-WRITE VERIFICATION? (Y OR N) | | This question is asked only for destination files on disk. |
| | Y | ICOPY will do Write Verification. If you get a verification failure, ICOPY displays a message indicating this. This probably means you have disk or system hardware problems; the data in the destination file is not reliable. |
| | N or ) | Both these responses mean that you don't want Write Verification. |
| RECORD FORMAT: | | This question is asked for tape and new disk destination files. |
| | F | Destination file records are fixed-length. |
| | V | Destination file records are variable-length. |
| | U | Destination file records are undefined-length. |
| | D | Destination file records are data sensitive. |

(Continued on the next page)

Table 5-3. SAM DESTINATION FILE PROMPTS (Continued)

| PROMPT | YOUR RESPONSE | DESCRIPTION |
|---|---|---|
| RECORD LENGTH:<br>or<br>MAXIMUM RECORD LENGTH: | number | This question is asked for disk and tape destination files for which you have already specified fixed-length records, and for tape files with variable-length records.<br>Enter, in decimal,<br>1) the length (in bytes) of fixed-length records; or<br>2) the maximum length (in bytes) of variable-length records. |
| | ) | Maximum record length (variable) is equal to block size. You cannot use this with fixed-length records. |
| REWIND ON CLOSE? (Y OR N) | | This question is asked for all tape destination files. |
| | N or ) | For both labeled and unlabeled tape files the tape remains positioned<br>1) just before the next file on the tape, or<br>2) at the Logical end-of-tape mark. |
| | Y | This rewinds the tape, labeled or unlabeled, to the physical beginning-of-tape mark.<br>If the tape is labeled, ICOPY asks you the Runtime Release question. |
| RUNTIME RELEASE? (Y OR N) | N or ) | These responses mean you will take care of releasing each volume of the destination file. |
| | Y | Entering Y directs the system to automatically release each volume of the destination file. |
| REWIND ON OPEN? (Y OR N) | | This question is asked for all destination files on magnetic tape. |
| | Y | Rewind the tape to the physical beginning-of-tape mark before searching it for the beginning of the destination file. |
| | N or ) | Search the tape from the current position, looking for the destination file. If found, begin the copy; if not found, ICOPY rewinds the tape automatically to the physical beginning-of-tape mark and terminates abnormally. |

(Continued on the next page)

Table 5-3. SAM DESTINATION FILE PROMPTS (Continued)

| PROMPT | YOUR RESPONSE | DESCRIPTION |
|---|---|---|
| SEQUENCE NUMBER: | | This question is asked only if the destination is a labeled tape. |
| | number | If you specify the four-digit file sequence number recorded in the file's HDR1 label the search for the destination file includes a comparison for sequence number. If ICOPY can't find the named file with a sequence number matching the one you enter, it rewinds the tape automatically to the physical beginning-of-tape mark and terminates abnormally. |
| | ) | ICOPY doesn't use the file's sequence number in searching for the destination file. |
| SPECIFY EXPIRATION DATE? (Y OR N) | | This question is only asked for labeled destination files. Remember that if a file has an expiration date recorded in its HDR1 label the file cannot be written to until that date has passed. |
| | N or ) | Do not write a new expiration date for the destination file to its HDR1 label. |
| | Y | You're asked the following three questions; ICOPY converts the date you enter to the HDR1 format, and records this in the file's HDR1 label. |
| MONTH: | number | Enter a number in the range 1-12. |
| DAY: | number | Enter a number in the range 1-31. |
| YEAR: | number | Enter a number in the range 68-99. |
| VARIABLE LENGTH BLOCKS? (Y OR N) | | This question is asked for all tape destination files. |
| | Y | If the last record ICOPY writes to the block does not completely fill the block, and if the next record would overflow the block, or if there is no next record, then ICOPY transfers the short block, as is, to the destination file. The next record starts at the beginning of a new block. |
| | N or ) | If the last record written to the block does not completely fill it, and if the next record would overflow it, or if there is no next record, then ICOPY fills the block with the pad character you have already specified before transferring it to the destination file. |

(Continued on the next page)

Table 5-3. SAM DESTINATION FILE PROMPTS (Concluded)

| PROMPT | YOUR RESPONSE | DESCRIPTION |
|---|---|---|
| VOLUME #1 ID: | volid | This question is asked for all tape destination files.<br><br>If the file is unlabeled, enter the RDOS Tape Unit Specifier on which the volume is mounted.<br><br>If the file is labeled, enter the volume name. Whatever you enter must exactly match the corresponding field in the VOL1 label. |

### SAM Source to SAM Destination Examples

*EXAMPLE 1 - Disk To Line Printer With Selective Field Translation*

```
ICOPY TEST1 $LPT LOG1/A)
 ****** DISK SOURCE FILE TEST1 ******
NUMBER OF BUFFERS: )
                   1
CODE TRANSLATION ON INPUT? (Y OR N) Y)
EBCDIC TO ASCII? (Y OR N) Y)
SELECTIVE FIELD TRANSLATION? (Y OR N) Y)
 ****** ENTER CARRIAGE RETURN TO TERMINATE LOOP ******
FIELD 1:
        STARTING CHARACTER POSITION (1-N): 1)
        LENGTH (CHARACTERS): 2)
FIELD 2:
        STARTING CHARACTER POSITION (1-N): )
 ****** LINE PRINTER OUTPUT ******
NUMBER OF BUFFERS: )
                   1
LIMIT TOTAL NUMBER OF RECORDS? (Y OR N) )
                                        N
MONITOR RECORD COUNT? (Y OR N) N)
5 RECORDS COPIED.
```

*EXAMPLE 2 - Disk To Disk With Selective Field Translation, Force End-Of-Volume, and Limit Total Records.*

```
ICOPY TEST TEST1 $LPT/A)
 ****** DISK SOURCE FILE TEST ******
NUMBER OF BUFFERS: )
                   1
CODE TRANSLATION ON INPUT? (Y OR N) Y)
EBCDIC TO ASCII? (Y OR N) )
                          N
ASCII TO EBCDIC? (Y OR N) Y)
SELECTIVE FIELD TRANSLATION? (Y OR N) Y)
 ****** ENTER CARRIAGE RETURN TO TERMINATE LOOP ******
FIELD 1:
        STARTING CHARACTER POSITION (1-N): 1)
        LENGTH (CHARACTERS): 2)
FIELD 2:
        STARTING CHARACTER POSITION (1-N): )
 ****** (NEW) DISK DESTINATION FILE TEST1 ******
RECORD FORMAT: F)
BLOCK SIZE: )
            512
NUMBER OF BUFFERS: )
                   1
RECORD LENGTH: 7)
READ-AFTER-WRITE VERIFICATION? (Y OR N) Y)
FORCE END OF VOLUME ON OUTPUT? (Y OR N) Y)
        RECORDS PER VOLUME: 20)
LIMIT TOTAL NUMBER OF RECORDS? (Y OR N) Y)
        RECORD COUNT: 40)
MONITOR RECORD COUNT? (Y OR N) N)
40 RECORDS COPIED.
```

*EXAMPLE 3 - Unlabeled Tape To Disk*

```
ICOPY MT0:0 TEST10 $LPT/A)
 ****** MAG TAPE SOURCE FILE MT0:0 ******
RECORD FORMAT: D)
VARIABLE LENGTH BLOCKS? (Y OR N) )
                                 N
BLOCK SIZE: )
          SYSTEM DEFAULT
NUMBER OF BUFFERS: )
                   1
REWIND ON OPEN? (Y OR N) Y)
REWIND ON CLOSE? (Y OR N) Y)
NUMBER OF VOLUMES: )
                   1
VOLUME #1 ID: MT0:0)
CODE TRANSLATION ON INPUT? (Y OR N) )
                                    N
 ****** (NEW) DISK DESTINATION FILE TEST10 ******
RECORD FORMAT: D)
BLOCK SIZE: )
           512
NUMBER OF BUFFERS: )
                   1
READ-AFTER-WRITE VERIFICATION? (Y OR N) )
                                        N
FORCE END OF VOLUME ON OUTPUT? (Y OR N) )
                                        N
LIMIT TOTAL NUMBER OF RECORDS? (Y OR N) )
                                        N
MONITOR RECORD COUNT? (Y OR N) N)
32 RECORDS COPIED.
```

*EXAMPLE 4 - Labeled Tape To Disk*

```
ICOPY $LPT/A)
SOURCE FILE: HOPE)
MAG TAPE FILE? (Y OR N) Y)
       LABELED? (Y OR N) Y)
DESTINATION FILE: P10)
MAG TAPE FILE? (Y OR N) )
                        N
FORMAT FILE FOR PRINTING? (Y OR N) )
                                   N
 ****** MAG TAPE SOURCE FILE HOPE ******
RECORD FORMAT: D)
VARIABLE LENGTH BLOCKS? (Y OR N) )
                                 N
BLOCK SIZE: )
          SYSTEM DEFAULT
NUMBER OF BUFFERS: )
                   1
SEQUENCE NUMBER: )
                 1
GENERATION NUMBER: )
                   0
LABEL TYPE (A=ANSI,I=IBM): )
                           A
LABEL LEVEL: 3)
ENABLE RUN TIME INITIALIZATION? (Y OR N) Y)
REWIND ON OPEN? (Y OR N) Y)
REWIND ON CLOSE? (Y OR N) Y)
       RUNTIME RELEASE? (Y OR N) )
                                 N
NUMBER OF VOLUMES: )
                   1
VOLUME #1 ID: P1)
CODE TRANSLATION ON INPUT? (Y OR N) )
                                    N
 ****** (NEW) DISK DESTINATION FILE P10 ******
RECORD FORMAT: D)
BLOCK SIZE: )
           512
NUMBER OF BUFFERS: )
                   1
READ-AFTER-WRITE VERIFICATION? (Y OR N) )
                                        N
FORCE END OF VOLUME ON OUTPUT? (Y OR N) )
                                        N
LIMIT TOTAL NUMBER OF RECORDS? (Y OR N) )
                                        N
MONITOR RECORD COUNT? (Y OR N) )
                               N
5 RECORDS COPIED.
```

*EXAMPLE 5 - Labeled Tape To Line Printer*

```
ICOPY LOG/A)
SOURCE FILE: HOPE)
MAG TAPE FILE? (Y OR N) Y)
        LABELED? (Y OR N) Y)
DESTINATION FILE: $LPT)
 ****** MAG TAPE SOURCE FILE HOPE ******
RECORD FORMAT: D)
VARIABLE LENGTH BLOCKS? (Y OR N) )
                                    N
BLOCK SIZE: )
            SYSTEM DEFAULT
NUMBER OF BUFFERS: )
                   1
SEQUENCE NUMBER: 1)
GENERATION NUMBER: 0)
LABEL TYPE (A=ANSI,I=IBM): A)
LABEL LEVEL: 3)
ENABLE RUN TIME INITIALIZATION? (Y OR N) Y)
REWIND ON OPEN? (Y OR N) Y)
REWIND ON CLOSE? (Y OR N) Y)
        RUNTIME RELEASE? (Y OR N) Y)
NUMBER OF VOLUMES: 1)
VOLUME #1 ID: P1)
CODE TRANSLATION ON INPUT? (Y OR N) )
                                    N
 ****** LINE PRINTER OUTPUT ******
NUMBER OF BUFFERS: )
                   1
LIMIT TOTAL NUMBER OF RECORDS? (Y OR N) )
                                        N
MONITOR RECORD COUNT? (Y OR N) )
                               N
5 RECORDS COPIED.
```

## ICOPY ISAM/DBAM DATA BASE FILE TO DESTINATION FILE SUMMARY

You can copy all or only part of an ISAM/DBAM Data Base file to a destination file. No part of an ISAM/DBAM Index File is ever copied. The destination file can be a new SAM disk file created during ICOPY, it can be any INFOS file you have previously created or it can be the line printer. At the end of the normal destination file dialogue (shown in Table 5-1) you are asked additional questions about where the copy is to begin and how many sub-indexes you want to descend.

The prompts for a Data Base to destination file copy are:

```
LIMIT DEPTH OF SOURCE FILE TRAVERSAL? (Y OR N)
NUMBER OF SUBINDEX LEVELS TO DESCEND:
SPECIFY STARTING POINT IN SOURCE FILE? (Y OR N)
***** ENTER KEY IN MULTILEVEL FORM *****
KEY:
```

These prompts are in addition to the normal source/destination file prompts, so you still can do other operations, like limiting the number of records copied and force end-of-volume, etc.

For example, given the following index structure, you can do all of the following copies (and several others).

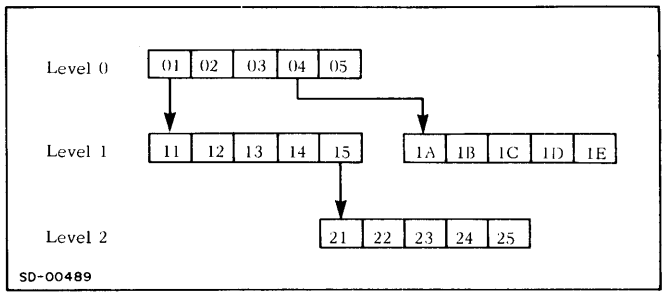


Figure 5-1. Index Structure for ICOPY Examples

1. If you want all the data records copied, enter:

```
LIMIT DEPTH OF SOURCE FILE TRAVERSAL? (Y OR N) )
                                               N
SPECIFY STARTING POINT IN SOURCE FILE? (Y OR N) )
                                                N
```

   The system does a pre-ordered traversal of the index, and copies the data records in the following sequence.

   01-11,12,13,14,15-21,22,23,24,25-02,03,04-1A,1B,1C,1D,1E-05

2. If you want the data records for index levels 0 and 1 copied, enter:

```
LIMIT DEPTH OF SOURCE FILE TRAVERSAL? (Y OR N) Y)
NUMBER OF SUBINDEX LEVELS TO DESCEND: 1)
SPECIFY STARTING POINT IN SOURCE FILE? (Y OR N) )
                                                N
```

   The records are copied in the following sequence.

   01-11,12,13,14,15-02,03,04-1A,1B,1C,1D,1E-05

3. If you want only the data records for the subindex defined under key 01 copied, enter:

```
LIMIT DEPTH OF SOURCE FILE TRAVERSAL? (Y OR N) Y)
NUMBER OF SUBINDEX LEVELS TO DESCEND: 0)
SPECIFY STARTING POINT IN SOURCE FILE? (Y OR N) Y)
**** ENTER KEY IN MULTILEVEL FORM ****
KEY: 01,11)
```

The records are copied in the following sequence.

11,12,13,14,15

As you can see, by limiting the number of index levels to descend, and by specifying a starting point for the copy, you can select the data records for a variety of meaningful subsets.

### DBAM Data Base to Labeled Tape Copy Example

```
ICOPY $LPT/A
SOURCE FILE: ACCTS
MAG TAPE FILE? (Y OR N)
                    N
DESTINATION FILE: HOPE
MAG TAPE FILE? (Y OR N) Y
       LABELED? (Y OR N) Y
FORMAT FILE FOR PRINTING? (Y OR N) Y
 ****** DISK SOURCE FILE ACCTS ******
NUMBER OF BUFFERS:
                  2
 ****** MAG TAPE DESTINATION FILE HOPE ******
APPEND OUTPUT TO FILE? (Y OR N)
                                N
VARIABLE LENGTH BLOCKS? (Y OR N)
                                N
BLOCK SIZE:
           SYSTEM DEFAULT
NUMBER OF BUFFERS:
                  1
SEQUENCE NUMBER: 1
GENERATION NUMBER: 0
LABEL TYPE (A=ANSI,I=IBM): A
LABEL LEVEL: 3
SPECIFY EXPIRATION DATE? (Y OR N) Y
MONTH: 03
DAY: 16
YEAR: 76
ENABLE RUN TIME INITIALIZATION? (Y OR N) Y
REWIND ON OPEN? (Y OR N) Y
REWIND ON CLOSE? (Y OR N) Y
       RUNTIME RELEASE? (Y OR N) Y
NUMBER OF VOLUMES: 1
VOLUME #1 ID: P1
FORCE END OF VOLUME ON OUTPUT? (Y OR N)
                                N
LIMIT TOTAL NUMBER OF RECORDS? (Y OR N)
                                N
MONITOR RECORD COUNT? (Y OR N) N
LIMIT DEPTH OF SOURCE FILE TRAVERSAL? (Y OR N) Y)
NUMBER OF SUBINDEX LEVELS TO DESCEND: 0)
SPECIFY STARTING POINT IN SOURCE FILE? (Y OR N) Y)
 ****** ENTER KEY IN MULTILEVEL FORM ******
KEY: 01, 15, 21)
5 RECORDS COPIED.
```

## ICOPY SOURCE FILE TO ISAM/DBAM FILE SUMMARY

You can copy the contents of any source file (INFOS or RDOS) to an ISAM/DBAM file you have previously created. In general, you will need one source file for each subindex in the structure. For example, if you want to copy into a DBAM file with the structure shown in Figure 5-2, you'll need to have six source files, and to invoke ICOPY six times.

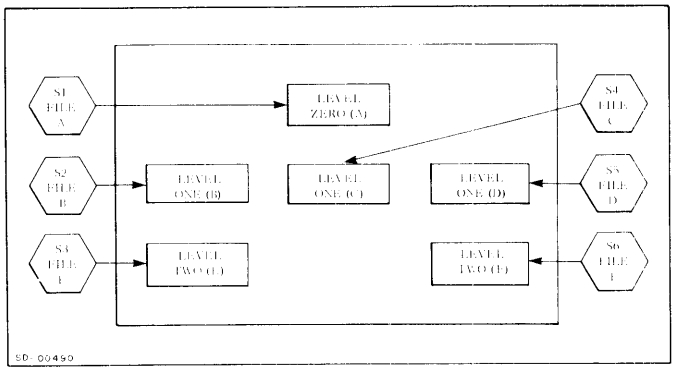


Figure 5-2. DBAM File Structure

Each time you invoke ICOPY and the destination is an ISAM/DBAM file, ICOPY asks you to specify the fields in the source records that contain the key, partial record and data base record. These fields need not be contiguous, and multiple fields can be concatenated into a single element (key, partial record, or data record).

An easy way to copy into a DBAM file is to create it with ICREATE, specifying the number of index levels and the key and partial record lengths for the level zero subindex. Then you can copy a source file into subindex level zero. After this, reinvoke ICOPY and specify the key in subindex level zero under which you want to copy a subindex. ICOPY will ask you to define the subindex, which it then creates.

At the end of the normal destination file dialogue (shown in Table 5-3), ICOPY asks you several questions about the fields in the source record for the key, partial record, and data record. These prompts are straightforward, as the following two examples show. The first example is designed to show how keys, data records, and partial records can be constructed from various fields in the source record. The second example is in three parts and shows three invocations of ICOPY which copy three source files to a single DBAM file whose index structure looks like that shown in Figure 5-3 at the end of the third copy:

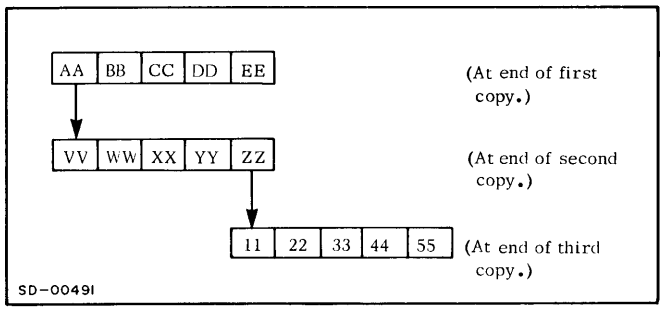


Figure 5-3. Index Structure for Source to ISAM/DBAM Copy, Examples 2, 3, and 4.

## Source to ISAM/DBAM Copy Examples

*EXAMPLE 1 - Building Keys, Data Records and Partial Records From Source File Records*

```
ICOPY $LPT/A
SOURCE FILE: P10
MAG TAPE FILE? (Y OR N)
                    N
DESTINATION FILE: TEST 15
MAG TAPE FILE? (Y OR N)
                    N
FORMAT FILE FOR PRINTING? (Y OR N)
                                  N
 ****** DISK SOURCE FILE P10 ******
NUMBER OF BUFFERS:
                  1
CODE TRANSLATION ON INPUT? (Y OR N)
                                  N
 ****** DISK DESTINATION FILE TEST15 ******
NUMBER OF BUFFERS:
                  2
READ-AFTER-WRITE VERIFICATION? (Y OR N)
                                  N
LIMIT TOTAL NUMBER OF RECORDS? (Y OR N)
                                  N
```

(Continued in next column)

```
CONCATENATE MULTIPLE FIELDS TO FORM KEY? (Y OR N) Y)
 ****** ENTER CARRIAGE RETURN TO TERMINATE LOOP ******
FIELD 1:
        STARTING CHARACTER POSITION (1-N): 1)
        LENGTH (CHARACTERS): 1)
FIELD 2:
        STARTING CHARACTER POSITION (1-N): 4)
        LENGTH (CHARACTERS): 2)
FIELD 3:
        STARTING CHARACTER POSITION (1-N): )
TRIM TRAILING BLANKS FROM KEYS? (Y OR N) Y)
DUPLICATE KEYS ALLOWED? (Y OR N) )
                                  N
SUPPRESS DATA BASE RECORDS? (Y OR N) )
                                  N
BUILD OUTPUT RECORD FROM SPECIFIC FIELDS? (Y OR N) Y)
 ****** ENTER CARRIAGE RETURN TO TERMINATE LOOP ******
FIELD 1:
        STARTING CHARACTER POSITION (1-N): 2)
        LENGTH (CHARACTERS): 3)
FIELD 2:
        STARTING CHARACTER POSITION (1-N): )
TRIM TRAILING BLANKS FROM RECORDS? (Y OR N) Y)
WRITE PARTIAL RECORDS? (Y OR N) Y)
CONCATENATE FIELDS TO FORM PARTIAL RECORD?(Y OR N) Y)
******ENTER CARRIAGE RETURN TO TERMINATE LOOP******
FIELD 1:
        STARTING CHARACTER POSITION (1-N): 1)
        LENGTH (CHARACTERS): 3)
FIELD2:
        STARTING CHARACTER POSITION (1-N): )
MONITOR KEYS? (Y OR N) N)
POSITION TO SUBINDEX IN DESTINATION FILE? (Y OR N) )
                                                  N
5 RECORDS COPIED.
```

*EXAMPLE 2 - Copy To Level 0 Subindex*

```
ICOPY TEST1 TESTER $LPT/A)
 ****** DISK SOURCE FILE TEST1 ******
NUMBER OF BUFFERS: )
                   1
CODE TRANSLATION ON INPUT? (Y OR N) )
                                    N
 ****** DISK DESTINATION FILE TESTER ******
NUMBER OF BUFFERS: )
                   2
READ-AFTER-WRITE VERIFICATION? (Y OR N) )
                                        N
LIMIT TOTAL NUMBER OF RECORDS? (Y OR N) )
                                        N
CONCATENATE MULTIPLE FIELDS TO FORM KEY? (Y OR N) )
                                                  N
FIRST BYTE POSITION TO BE USED AS KEY (1-N) 1)
LENGTH OF KEY, BYTES: 2)
TRIM TRAILING BLANKS FROM KEYS? (Y OR N) )
                                         N
DUPLICATE KEYS ALLOWED? (Y OR N) )
                                 N
SUPPRESS DATA BASE RECORDS? (Y OR N) )
                                     N
BUILD OUTPUT RECORD FROM SPECIFIC FIELDS? (Y OR N) )
                                                   N
FIRST BYTE POSITION TO BE INCLUDED IN OUTPUT RECORD (1-N): 1)
SPECIFY LENGTH OF RECORD TO BE COPIED? (Y OR N) Y)
       LENGTH, BYTES: 5)
TRIM TRAILING BLANKS FROM RECORDS? (Y OR N) )
                                            N
WRITE PARTIAL RECORDS? (Y OR N) )
                                N
MONITOR KEYS? (Y OR N) N)
POSITION TO SUBINDEX IN DESTINATION FILE? (Y OR N) )
                                                   N
5 RECORDS COPIED.
```

*EXAMPLE 3 - Copy To Level 1 Subindex*

```
ICOPY TEST2 TESTER $LPT/A
 ****** DISK SOURCE FILE TEST2 ******
NUMBER OF BUFFERS:
                  1
CODE TRANSLATION ON INPUT? (Y OR N)
                                N
 ****** DISK DESTINATION FILE TESTER ******
NUMBER OF BUFFERS:
                  2
READ-AFTER-WRITE VERIFICATION? (Y OR N)
                                    N
LIMIT TOTAL NUMBER OF RECORDS? (Y OR N)
                                   N
CONCATENATE MULTIPLE FIELDS TO FORM KEY? (Y OR N)
                                                N
FIRST BYTE POSITION TO BE USED AS KEY (1-N): 1
LENGTH OF KEY, BYTES: 2
TRIM TRAILING BLANKS FROM KEYS? (Y OR N)
                                  N
DUPLICATE KEYS ALLOWED? (Y OR N)
                               N
SUPPRESS DATA BASE RECORDS? (Y OR N)
                                 N
BUILD OUTPUT RECORD FROM SPECIFIC FIELDS? (Y OR N)
                                             N
FIRST BYTE POSITION TO BE INCLUDED IN OUTPUT RECORD (1-N): 1
SPECIFY LENGTH OF RECORD TO BE COPIED? (Y OR N) Y
       LENGTH, BYTES: 5
TRIM TRAILING BLANKS FROM RECORDS? (Y OR N)
                                          N
WRITE PARTIAL RECORDS? (Y OR N)
                                N
MONITOR KEYS? (Y OR N) N
POSITION TO SUBINDEX IN DESTINATION FILE? (Y OR N) Y)
 ****** ENTER KEY IN MULTILEVEL FORM FOR OWNER OF SUBINDEX ******
KEY: AA)
 ****** SUBINDEX MUST BE DEFINED ******
MINIMUM NODE SIZE: 512)
MAXIMUM KEY LENGTH: 2)
PARTIAL RECORD LENGTH: )
                       0
COMPRESSED KEYS? (Y OR N) )
                          N
SUBINDICES ALLOWED? (Y OR N) Y)
5 RECORDS COPIED.
```

*EXAMPLE 4 - Copy To Level 2 Subindex*

```
ICOPY TEST3 TESTER $LPT/A
 ****** DISK SOURCE FILE TEST3 ******
NUMBER OF BUFFERS:
                  1
CODE TRANSLATION ON INPUT? (Y OR N)
                                N
 ****** DISK DESTINATION FILE TESTER ******
NUMBER OF BUFFERS:
                  2
READ-AFTER-WRITE VERIFICATION? (Y OR N)
                                     N
LIMIT TOTAL NUMBER OF RECORDS? (Y OR N)
                                    N
CONCATENATE MULTIPLE FIELDS TO FORM KEY? (Y OR N)
                                                N
FIRST BYTE POSITION TO BE USED AS KEY (1-N): 1
LENGTH OF KEY, BYTES: 2
TRIM TRAILING BLANKS FROM KEYS? (Y OR N)
                                     N
DUPLICATE KEYS ALLOWED? (Y OR N)
                                  N
SUPPRESS DATA BASE RECORDS? (Y OR N)
                                  N
BUILD OUTPUT RECORD FROM SPECIFIC FIELDS? (Y OR N)
                                                   N
FIRST BYTE POSITION TO BE INCLUDED IN OUTPUT RECORD (1-N): 1
SPECIFY LENGTH OF RECORD TO BE COPIED? (Y OR N) Y
       LENGTH, BYTES: 5
TRIM TRAILING BLANKS FROM RECORDS? (Y OR N)
                                        N
WRITE PARTIAL RECORDS? (Y OR N)
                                  N
MONITOR KEYS? (Y OR N) N
POSITION TO SUBINDEX IN DESTINATION FILE? (Y OR N) Y)
 ****** ENTER KEY IN MULTILEVEL FORM FOR OWNER OF SUBINDEX ******
KEY: AA,ZZ)
 ****** SUBINDEX MUST BE DEFINED ******
MINIMUM NODE SIZE: 512)
MAXIMUM KEY LENGTH: 2)
PARTIAL RECORD LENGTH: )
                       0
COMPRESSED KEYS? (Y OR N) )
                          N
SUBINDICES ALLOWED? (Y OR N) )
                             N
5 RECORDS COPIED.
```

END OF CHAPTER

# CHAPTER 6

# PERMANENT FILE SPECIFICATION RETRIEVAL UTILITY

## INTRODUCTION

With this utility, you can retrieve the permanent file specification for any INFOS System disk file.

## INVOKING IFILE

To invoke IFILE, key the CLI command:

```
IFILE [/L] [filename]
```

Use the global switch /L when you want to output the specification to the line printer.

## EXAMPLES

IFILE/L)

```
FILE: INVOICE
ACCESS METHOD: ISAM/DBAM (INDEX FILE)
BLOCK SIZE: 1018 BYTES
NUMBER OF DATA BUFFERS: 4
LEVELS OF CURRENT POSITION: 5
NUMBER OF VOLUMES: 1
DATA BASE FILE: INVOICEDB
VOLUME 1: INVOICE
        VOLUME SIZE: 100 DISK BLOCKS (MAXIMUM)
        DISK BLOCKS USED: 2
        PAD CHARACTER (OCTAL CODE): 000
```

IFILE/L INVOICEDB)

```
ACCESS METHOD: ISAM/DBAM (DATA BASE FILE)
BLOCK SIZE: 512 BYTES
NUMBER OF DATA BUFFERS: 1
MAXIMUM RECORD LENGTH: 80 BYTES
NUMBER OF VOLUMES: 1
VOLUME 1: INVOICEDB
        VOLUME SIZE: 100 DISK BLOCKS (MAXIMUM)
        DISK BLOCKS USED: 1
        PAD CHARACTER (OCTAL CODE): 000
INDEX 1: ACCTS
```

END OF CHAPTER

# CHAPTER 7

# FILE RENAME UTILITY

## INTRODUCTION

With this utility, you can change the name of any INFOS System disk file. It automatically renames each volume of a multivolume file, and appropriately changes the system - generated Volume Definition and Index Definition files. You can also rename individual volumes of a file without renaming the file itself.

## INVOKING IRENAME

To invoke IRENAME, key in the CLI command:

```
IRENAME [/S] [/V] [present file name [new file name]]
        [log file/A]
```

When you use the global switch /S, the short version of IRENAME executes. When you use /V the utility displays the results of the rename operation.

## EXAMPLES

```
IRENAME/V $LPT/A)
PRESENT NAME: ACCTSDB)
        VOLUME 1: ACCTSDB
NEW NAME: INVOICEDB)
RDOS FILE ACCTSDB RENAMED INVOICEDB.
RDOS FILE ACCTSDB.VL RENAMED INVOICEDB.VL.
RDOS FILE ACCTSDB.IX RENAMED INVOICEDB.IX.
```

```
IRENAME/V ACCTS $LPT/A)
        VOLUME 1: ACCTS
NEW NAME: INVOICE)
RDOS FILE ACCTS RENAMED INVOICE.
RDOS FILE ACCTS.VL RENAMED INVOICE.VL.
```

```
IRENAME/S/V INVOICE ACCTS $LPT/A)
RDOS FILE INVOICE RENAMED ACCTS.
RDOS FILE INVOICE.VL RENAMED ACCTS.VL.
```

```
IRENAME/S/V INVOICEDB ACCTSDB $LPT/A)
RDOS FILE INVOICEDB RENAMED ACCTSDB.
RDOS FILE INVOICEDB.VL RENAMED ACCTSDB.VL.
RDOS FILE INVOICEDB.IX RENAMED ACCTSDB.IX.
```

END OF CHAPTER

# CHAPTER 8

# FILE DELETE UTILITY

## INTRODUCTION

With the IDELETE utility, you can delete any INFOS System disk file and one or more indexes from a multi-indexed DBAM file.

## INVOKING IDELETE

To invoke IDELETE, key the CLI command:

```
IDELETE [/S][/V][file name][log file/A]
```

If you name a data base file, IDELETE deletes all of its volumes and all volumes of its associated index files. If you name an index file, IDELETE gives you three options. Otherwise, IDELETE deletes only the volumes of the file you name.

When you include /S on the command line, the short version of IDELETE executes. This means that you automatically get option 1 if you name an index file.

Ordinarily you will want to use option 2 to delete a single index from a DBAM structure with multiple indexes. This option does a physical delete of each data base record associated with the index. Option 1 is faster, since it does not do the individual record deletions, but you should only use this form of deletion with a temporary index.

The /V switch provides you with visual confirmation (on the video terminal) that the proper files have been deleted.

## EXAMPLES

```
IDELETE/V $LPT/A)
FILF: ACCTS)
INDEX FILE--SELECT
1: DELETE INDEX ONLY
2: DELETE INDEX AND REFERENCED DATA RECORDS
3: DELETE DATA BASE FILE AND ALL ITS INDEX FILES
OPTION: 1)
RDOS FILE ACCTS DELETED.
RDOS FILE ACCTS.VL DELETED.
```

```
IDELETE/V ACCTS $LPT/A)
INDEX FILE--SELECT
1: DELETE INDEX ONLY
2: DELETE INDEX AND REFERENCED DATA RECORDS
3: DELETE DATA BASE FILE AND ALL ITS INDEX FILES
OPTION: 2)
RDOS FILE ACCTS DELETED.
RDOS FILE ACCTS.VL DELETED.
```

```
IDELETE/V ACCTS $LPT/A)
INDEX FILE--SELECT
1: DELETE INDEX ONLY
2: DELETE INDEX AND REFERENCED DATA RECORDS
3: DELETE DATA BASE FILE AND ALL ITS INDEX FILES
OPTION: 3)
RDOS FILE ACCTSDB DELETED.
RDOS FILE ACCTSDB.VL DELETED.
RDOS FILE ACCTS DELETED.
RDOS FILE ACCTS.VL DELETED.
RDOS FILE ACCTSDB.IX DELETED.
```

```
IDELETE/S/V ACCTS $LPT/A)
RDOS FILE ACCTS DELETED.
RDOS FILE ACCTS.VL DELETED.
```

```
IDELETE/V ACCTSDB $LPT/A)
RDOS FILE ACCTSDB DELETED.
RDOS FILE ACCTSDB.VL DELETED.
RDOS FILE ACCTS DELETED.
RDOS FILE ACCTS.VL DELETED.
RDOS FILE ACCTSDB.IX DELETED.
```

END OF CHAPTER

# CHAPTER 9

# LABELED MAGNETIC TAPE INITIALIZATION UTILITY

## INTRODUCTION

You can do either a full or partial initialization with LBINIT. On a full initialization, LBINIT writes dummy VOL1 and HDR1 labels and, if the file has ANSI labels, a dummy EOF1 label. On a partial initialization, it reads the VOL1 label and compares it with what you specify.

## INVOKING LBINIT

To invoke LBINIT, key in the CLI command:

```
LBINIT[/S][/A][/I][/F][/P]↑
    [dev specifier [vol id [owner id [level]]]]↑
    [log/A]
```

Global switch definitions are:

```
/S = Short form of LBINIT
/A = ANSI labels
/I = IBM labels
/F = Full initialization
/P = Partial initialization
```

Note the sequence of the remaining elements on the command line. This sequence corresponds to the sequence in which LBINIT asks for information; thus you must enter the elements in the sequence shown. You can truncate the list of elements from the right-hand side, but you cannot omit elements. You cannot, for example, enter:

```
LBINIT/A/F MT0 3
```

and expect to get level 3 labels. The utility assumes the "3" represents the vol id element.

To release the initialized tape volume, enter:

```
RELEASE vol id
```

## EXAMPLES

Examples 1 and 2 represent full ANSI level 1 initializations. Example 3 represents a partial ANSI level 1 initialization. Example 4 represents a partial initialization with ANSI level 1 labels, and it uses the short form of LBINIT. Example 5 represents an ANSI level 3 partial initialization. Example 6 represents a full IBM level 2 initialization.

Example 1:

```
LBINIT $LPT/A)
FULL(F) OR PARTIAL(P): F)
ANSI(A) OR IBM(I) LABELS: A)
DEVICE SPECIFIER: MT0)
VOLUME ID: ACCTS)
SPECIFY OWNER ID? (Y OR N) Y)
        ID: PAYROLL)
LABELING LEVEL NUMBER: 1)
```

Example 2:

```
LBINIT/A/F MT0 ACCTS PAYROLL 1 $LPT/A)
```

Example 3:

```
LBINIT $LPT/A)
FULL(F) OR PARTIAL(P): P)
ANSI(A) OR IBM(1) LABELS: A)
DEVICE SPECIFIER: MT0)
SPECIFY VOLUME ID? (Y OR N) Y)
        ID: ACCTS)
SPECIFY OWNER ID? (Y OR N) Y)
        ID: PAYROLL)
LABELING LEVEL NUMBER: 1)
```

Example 4:

```
LBINIT/S MT0 ACCTS PAYROLL $LPT/A)
```

Example 5:

```
LBINIT/A/P MT0 ACCTS $LPT/A)
SPECIFY OWNER ID? (Y OR N) Y)
        ID: PAYROLL)
LABELING LEVEL NUMBER: 3)
```

Example 6:

```
LBINIT/F/I MT0 ACCTS $LPT/A)
SPECIFY OWNER ID? (Y OR N) N)
LABELING LEVEL NUMBER: 2)
```

END OF CHAPTER

CUT ALONG DOTTED LINE

# DataGeneral

## Software Documentation Remarks Form

### How Do You Like This Manual?

Title _____ No. _____

We wrote the book for you, and naturally we had to make certain assumptions about who you are and how you would use it. Your comments will help us correct our assumptions and improve our manuals. Please take a few minutes to respond.

If you have any comments on the software itself, please contact your Data General representative. If you wish to order manuals, consult the Publications Catalog (012-330).

### Who Are You?

- ☐ EDP Manager
- ☐ Senior System Analyst
- ☐ Analyst/Programmer
- ☐ Operator
- ☐ Other _____

What programming language(s) do you use? _____

### How Do You Use This Manual?

*(List in order: 1 = Primary use)*

- ____ Introduction to the product
- ____ Reference
- ____ Tutorial Text
- ____ Operating Guide
- ____ _____

### Do You Like The Manual?

| Yes | Somewhat | No | |
|---|---|---|---|
| ☐ | ☐ | ☐ | Is the manual easy to read? |
| ☐ | ☐ | ☐ | Is it easy to understand? |
| ☐ | ☐ | ☐ | Is the topic order easy to follow? |
| ☐ | ☐ | ☐ | Is the technical information accurate? |
| ☐ | ☐ | ☐ | Can you easily find what you want? |
| ☐ | ☐ | ☐ | Do the illustrations help you? |
| ☐ | ☐ | ☐ | Does the manual tell you everything you need to know? |

### Comments?

*(Please note page number and paragraph where applicable.)*

### From:

Name _____ Title _____ Company _____

Address _____ Date _____

FOLD DOWN FIRST FOLD DOWN

FIRST
CLASS
PERMIT
No. 26
Southboro
Mass. 01772

## BUSINESS REPLY MAIL

No Postage Necessary if Mailed in the United States

Postage will be paid by:

**Data General Corporation**

Southboro, Massachusetts 01772

ATTENTION: Software Documentation

FOLD UP SECOND FOLD UP

CUT ALONG DOTTED LINE

DataGeneral

# users group

## Installation Membership Form

Name _____ Position _____ Date _____

Company, Organization or School _____

Address _____ City _____ State _____ Zip _____

Telephone: Area Code _____ No. _____ Ext. _____

### 1. Account Category

- ☐ OEM
- ☐ End User
- ☐ System House
- ☐ Government

### 2. Hardware

| | Qty. Installed | Qty. On Order |
|---|---|---|
| M/600 | | |
| C/350, C/330, C/300 | | |
| S/250, S/230, S/200 | | |
| S/130 | | |
| AP/130 | | |
| CS Series | | |
| N3/D | | |
| Other NOVA | | |
| microNOVA | | |
| Other (Specify) _____ | | |
| _____ | | |

### 3. Software

- ☐ AOS
- ☐ DOS
- ☐ SOS
- ☐ RDOS
- ☐ RTOS
- ☐ Other

Specify _____

### 4. Languages

- ☐ Algol
- ☐ DG/L
- ☐ Cobol
- ☐ ECLIPSE Cobol
- ☐ Business BASIC
- ☐ BASIC
- ☐ Assembler
- ☐ Interactive
- ☐ Fortran
- ☐ RPG II
- ☐ PL/1
- ☐ Other

Specify _____

### 5. Mode of Operation

- ☐ Batch (Central)
- ☐ Batch (Via RJE)
- ☐ On-Line Interactive

### 6. Communications

- ☐ RSTCP
- ☐ HASP
- ☐ RJE80
- ☐ SAM
- ☐ CAM
- ☐ 4025
- ☐ Other

Specify _____

### 7. Application Description

○ _____

### 8. Purchase

From whom was your machine(s) purchased?

- ☐ **Data General Corp.**
- ☐ Other

Specify _____

### 9. Users Group

Are you interested in joining a special interest or regional Data General Users Group?

○ _____

DataGeneral

Data General Corporation, Westboro, Massachusetts 01581, (617) 366-8911

FOLD FOLD

STAPLE STAPLE

FOLD FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 26 SOUTHBORO, MA. 01772

Postage will be paid by addressee:

Data General

ATTN: Users Group Coordinator

Southboro, Massachusetts 01772