7 THE KSAM 80 FILE MANAGEMENT SYSTEM

INTRODUCTION

KSAM 80 is a file management system, designed specifically for mini flexible disk microcomputer systems.

It was developed primarily for use in applications where large files are involved and fast random access is essential. Such applications include: inventory control, reservation systems, library systems, accounts receivable and bill of materials processing etc.

Files are always accessed dynamically using KSAM 80 commands; these are described in detail in section 7.3. Records are identified by a user-defined data field within the record called the primary key. This primary key must be unique. Additionally any number of fields may be designated as alternate keys for retrieval purposes. Files arranged in the order of an alternate key are called alternate files. For example, primary keys could be part numbers, account numbers or customer names (all these are unique identifiers), alternate keys could be locations, subjects, or age.

Records can be accessed randomly or sequentially by their primary key or by any of their alternate keys.

In addition KSAM 80 supports sequential movement through the file, and random access by partial key or relative record number.

Space is automatically allocated to the file when records are added, and reclaimed when records are deleted, so that KSAM 80 files are self reorganising, and any number of files can be processed simultaneously provided that sufficient buffer storage is available.

The KSAMUT utility package is also available as part of KSAM 80.

Note - This section refers to several examples named KSAM-xx. These are stored on the system disk.

7.1.1 KSAM 80 Files

A KSAM 80 file may extend to a maximum of three disks. The user specifies the number of disk volumes required at file creation time.

Each volume can be thought of as any file in the operating system. The file unit, name and type of each volume is defined by the user.

THE DATA SET

The data set contains the actual records created by the user. These records are arranged in ascending order on the basis of their primary keys. When records are added or deleted, KSAM 80 maintains this sequence.

Records are organized in data blocks, the block size is defined by the user at file creation time but it must be a multiple of a page, where a page equals 256 bytes.

The number of records per data block is derived by dividing the block length by the record length. Logical records may not be split between blocks, therefore any one block must contain an integral number of records. Any remaining bytes are left unused.

THE KEY SET

The key set contains various pointers to the data blocks and is maintained by KSAM 80. It can include up to 16 key blocks; the key block length is defined by the user and must be a multiple of a page.

THE HEADER

The header occupies one page of disk space and contains information about the structure and contents of the file such as the record length, key length, and key displacement. It is created from information supplied by the user at file creation time and maintained by KSAM 80.

ALLOCATION

KSAM 80 allocates space to a file in multiples of a page. Each data block occupies the number of pages specified at file creation time. The allocation routine searches for a free data block, if such a block exists it is used first, otherwise disk space is allocated at the end of the file.

7.1.2 Logical Records

Logical records form the largest part of a KSAM 80 file. Records are added to data base files by the user; KSAM 80 will add records to alternate files at the user's request. The records consist of fields, these are byte strings and may contain any value ASCII or binary.

Records are organized in data blocks, which must be multiples of one page. The number of pages in a data block is defined by the user at file creation time, but in most cases this can be changed later to achieve greater efficiency.

In this version of KSAM 80 all records of a given file must be of the same length. For data base files the record length is defined by the user and may vary from one byte to one data block. The alternate file record length is determined by the criteria outlined in section 7.1.1.

Records may not be split between blocks, therefore the minimum number of records in a block is one. The maximum number of records in a block is derived by dividing the block length (in bytes) by the record length, and taking the highest integer that is less than or equal to the quotient.

The user may specify that an area of unused block space be reserved for the later addition of records. Although this may result in wasted disk space, it can significantly improve the processing speed for applications where frequent additions to the file are made.

7.1.3 Keys

Any field within the record can be used as a means of randomly accessing the record; a field used for that purpose is called a key. Key fields may be up to 248 bytes in length.

For random retrieval the user must specify the key of the record required. For example, if the primary key of a customer record is the customer's account number, then that number must be supplied to retrieve the record by random access. Alternatively, if sequential retrieval is acceptable, the file can be read sequentially (forwards or backwards) until the record required is found.
The relative position of a record within the file may also be used for retrieval. This is a number in the range 0 to the number of records in the file minus one. The relative position of records will change as records are added or deleted.

PRIMARY KEYS

One of the record keys must contain a value that is unique, this key is used to identify the record and to distinguish it from other records of the same file; it is called the primary key. KSAM 80 records are always kept in primary key sequence.

The primary key must be declared by the user at file creation time, and can be up to 248 bytes long unless alternate key are also used. The length and displacement of the primary key must be the same for all records of the same file.

ALTERNATE KEYS

An alternate key is a field in the record that can be used as an alternative means of retrieving the record.

All fields in the record may be designated as alternate keys and, unlike primary keys, alternate key values need not be unique.

The length and displacement of an alternate key must be the same for all records of the same file. The sum of the length of an alternate key and the length of the primary key must not exceed 248 bytes.

A file where the records are ordered by an alternate key is called an alternate file.

7.1.4 Alternate Files

An alternate file is a KSAM 80 file in which the records are arranged in ascending order of the value of the alternate key field, and contain pointers to the corresponding records in the main data base file. Any number of alternate files may be created for the same data base file, each using a different alternate key. Since alternate files are also KSAM 80 files they can be read independently of their associated data base files.

The KSAM 80 File Management System

The records of an alternate file are called inversion records. There need not be a one-to-one correspondence between the data base records and their inversions.

Adding or deleting inversions is not performed automatically when a data base record is added or deleted. The user may therefore create inversions for only a selection of data base records. Similarly, a data base record may be deleted or altered without affecting its corresponding inversion records. If the user wishes to maintain a one-to-one correspondence then an inversion record must be added to every alternate file whenever a new data base record is added, and all inversion records must be deleted before the corresponding data base record is deleted.

If a data base record is modified so that the value of any alternate key field is altered, then all affected inversions must be deleted, and then added again after the data base record has been updated. This will ensure that the latest value of the alternate key field is stored in the alternate file, and that the data base record can still be accessed through the alternate key.

Inversion records are made up of two fields. The first contains the value of the alternate key field in the corresponding data base record. The second contains the primary key of the corresponding data base record.

The total length (in bytes) of the inversion record is therefore:

Length of alternate key field + length of DB primary key

Alternate files are also KSAM 80 files, and therefore inversion records must possess a unique primary key. Therefore the inversion record itself constitutes the primary (and only) key. Since inversion records include the unique primary key of the data base record, the primary key of the inversion record is also unique. The data base file can then be accessed directly using the information derived from alternate files.

The user does not have to supply the record length, primary key length, or primary key displacement when creating an alternate file, these are calculated by KSAM 80. The user merely has to specify the length and displacement of the field in the data base record that is to be used as an alternate key.

KSAM 80 will calculate the values as follows:

Inversion record length = length alternate key field
+ length of DB primary key

Inversion primary key length = Inversion record length

Primary key displacement = 0

The creation of alternate files provides the user with greater flexibility for accessing data, and is invaluable in applications that use report generation or inquiry programs.

7.1.5 The KSAFCB

The user must reserve a file control block for every KSAM 80 file, for alternate files as well as data base files.

This is described in detail in section 7.5.

7.2 CALLING KSAM 80

Depending on the function, two to five parameters are necessary when calling KSAM 80.

They contain:

- the command or operation code
- the KSAM 80 file control block for the file required
- the record address
- the file control block for the alternate file
- the alternate record

For example:

```
200 CODE%=&Hxx
210 CALL KSAM(CODE%,FCB$,REC$)
```

where:

The value of CODE% represents the KSAM command required, for example &HD3 is the code for the CREATE command.

FCB$ is the 128 byte file control block that contains information about the file to be accessed; this is described more fully in section 7.5. FCBALT$ is the equivalent parameter for an alternate file.

REC$ is the string that will contain the file record to be input or output or, in the case of the RENAME command, an additional command parameter. RECALT$ is the equivalent parameter for an alternate file.

Upon return from the CALL the command code field will contain a return code, a zero value always indicates a successful completion. A complete list of possible error codes is given in section 7.7.

7.3 THE KSAM 80 COMMANDS

This Chapter contains detailed descriptions of all the KSAM 80 commands. KSAM 80 commands are logically divided into the following four groups:

1. File commands, which treat the entire file as one unit
2. Sequential access commands (using primary key sequence)
3. Random access by primary key commands
4. Alternate key commands

A list of the KSAM 80 commands in each group is as follows:

File commands	- CREATE or CREATE ALTERNATE - CLOSE - OPEN
Sequential access commands	- READ PREVIOUS - READ CURRENT - READ NEXT - RETRIEVE KEY
Random access commands	- READ RANDOM - UPDATE - DELETE - READ NTH RECORD - READ APPROXIMATE - LOAD - ADD
Alternate key commands	- READ BY ALTERNATE KEY - READ FIRST - READ NEXT HOLDING - VERIFY INVERSION - ADD INVERSION - DELETE INVERSION

Commands in group 1 with the exception of CLOSE, expect the file to be closed. If it is not, an INVALID REQUEST error will occur. The CLOSE command expects the file to be open.

It is possible for a file to contain no records. This can happen when the only record on the file is deleted or immediately after the file is created.
When the file becomes EMPTY it does not disappear. The disk space previously occupied by the file remains and the header is still available; the file remains open until CLOSEd by the user.

All commands require a KSAM 80 File Control Block (referred to as KSAFCB) to be constructed by the user.

Some commands require additional information to be supplied by the user, in such cases the additional information needed is described under the appropriate command.

Since I/O errors are possible with every command they are not explicitly mentioned under error return codes. This also applies to all system related errors such as FILE NOT FOUND etc.

See section 7.6 for an alphabetical list of the commands with their codes and brief descriptions. Section 7.7 contains explanations of the error codes.

The command codes are given in brackets after each section after each section heading.

Note - Examples of all these commands are included on your system disk. At the end of each command, the file name of the example is given, so that you can run it on your P2500. When you use these examples, you should run the program KSAM.O first. This loads KSAM and creates a member KSAM P which will be merged automatically with all the programs KSAM.xx.

7.3.1 File Commands

Before a file can be accessed it must be created, and that is the function of the CREATE command. This command does not actually add any data records to the file (that is done by the ADD and LOAD commands); it only preformats certain fixed areas and allocates initial disk space to the file.

After a successful CREATE command the file is automatically closed by KSAM 80; it then exists in an EMPTY state until a record is ADDed or LOADed.

Before an existing file can be processed it must be opened using the OPEN command, and when the processing is complete it must be closed using the CLOSE command.

Note - It is very important to remember to close any file if records were added, deleted or updated.

To erase or rename a file, use the respective BASIC commands KILL or NAME.

The KSAM 80 File Management System

CLOSE (CO)

Calling format:

```
CODE%=&HCO
CALL KSAM(CODE%,FCB$)
```

Purpose:

To update the header on the disk to reflect the latest status of the file, and close all volumes.

Description:

All volumes of the file will be closed. This command is normally issued when file processing has been completed. Not issuing this command will result in loss of data if records have been added, updated or deleted from the file.

If a file is open it must be closed before a RENAME or ERASE operation is performed.

For further information see the operating system CLOSE command.

Use example KSAM - 12

CREATE (D3)

Calling format:

```
CODE%=&HD3
CALL KSAM(CODE%,FCB$)
```

Purpose:

To create a new and empty KSAM main file.

Description:

If successful the file header will be written on the first volume.

The following additional information must be placed in the standard KSAFCB (File Control Block):

- The number of disk volumes the file may extend to.
- The key block length (B) in pages, B cannot equal zero.
- The data block length (L) in pages per block, L cannot equal zero
- The unused space (U) in logical records; this must be less than the maximum number of logical records that will fit in a data block.
- The logical record length (R) in bytes, 0 R = L x 256
- The key length (K) in bytes, where 0 K + D R
- The key displacement (D) from the beginning of the record in bytes.

If there is an error then the file will not be created.

Use example KSAM - 1

The KSAM 80 File Management System

OPEN (DO)

Calling format:

```
CODE%=&HDO
CALL KSAM(CODE%,FCB$)
```

Purpose:

To make the file available for further processing.

Description:

The file must exist and all volumes of the file must be mounted.

If the operation is successful all volumes will be opened and the KSAFCB will be flagged to indicate that the file is open. The header information will be placed in the corresponding fields of the KSAFCB and the CRP will point to BOF.

If there is an error the file will not be opened.

Use example KSAM - 2
" " KSAM - 10

7.3.2 Sequential Access Commands

These commands do not require that a key be supplied by the user. Instead they use the current record pointer (CRP), which is described in section 7.5.2.

A standard KSAFCB is sufficient.

The READ PREVIOUS command will decrement the CRP unless it is pointing to BOF (Beginning Of File), in which case it will leave it unchanged.

The READ CURRENT and RETRIEVE KEY commands will not disturb the CRP.

The READ NEXT command will increment the CRP unless it is pointing to EOF (End Of File), in which case it will leave it unchanged.

READ PREVIOUS (C3)

Calling format:

```
CODE%=&HC3
CALL KSAM(CODE%,FCB$,REC$)
```

Purpose:

To move the previous available record from the file to the logical record area.

Description:

Unless the CRP (Current Record Pointer) is pointing to the first record, the CRP will be decremented by 1 and the record with the next lower primary key will be returned. If the CRP is pointing to the EOF then the last record on file will be returned.

When the CRP is pointing to the first record it will be decremented by 1 so that it points at the BOF; no data will be moved and a BOF error will be returned.

Use example KSAM - 4

READ CURRENT (C4)

Calling format:

```
CODE%=&HC4
CALL KSAM(CODE%,FCB$,REC$)
```

Purpose:

To move the current record from the file to the logical record area.

Description:

The record to which the CRP is pointing is moved to the logical record area. This command is normally used in conjunction with the random access commands which are described in the next section.

If the CRP is pointing to the BOF or EOF then the appropriate error code will be returned and no movement of data will take place.

Use example KSAM - 5

The KSAM 80 File Management System

READ NEXT (C5)

Calling format:

```
CODE%=&HC5
CALL KSAM(CODE%,FCB$,REC$)
```

Purpose:

To move the next available record from the file to the logical record area.

Description:

Unless the CRP (Current Record Pointer) is pointing to the last record, the CRP will be incremented by 1 and the record with the next higher primary key will be returned. If the CRP is pointing to the BOF then the first record on file will be returned.

If the CRP is pointing to the last record or to the EOF, an EOF error will be returned and no movement of data will take place.

Use example KSAM - 6

RETRIEVE KEY (C9)

Calling format:

```
CODE%=&HC9
CALL KSAM(CODE%,FCB$,REC$)
```

Purpose:

To move the primary key of the current record from the file to the primary key field of the logical record area.

Description:

This command is similar to the READ CURRENT command except that only the primary key of the current record is moved. The remaining part of the logical record area is not changed.

No example

7.3.3 Random Access Commands

These commands require a key value to be placed in the primary key field of the logical record area. The exception to this rule is the READ NTH RECORD command, which requires that a 16-bit binary number be placed in the first two bytes of the key in the logical record area.

The key or record number, and not the CRP, is used to locate the record required. On completion, successful or otherwise, the CRP is adjusted according to the rules described in section 7.5.2, so that sequential processing may follow any random command.

A standard KSAFCB is sufficient for all commands.

The following discussion applies to all random access commands except the READ NTH RECORD command. KSAM 80 will attempt to locate a record on file with a key that matches the key in the key field of the logical record area:

- If such a record exists the ADD and LOAD commands will fail with an INVALID KEY return code (the key on file must be unique); all other commands will proceed to perform their function. The CRP will then point to that record. If a DELETE command is successful, the CRP will move back by one record.

- If the given key is higher than any key on file the ADD and LOAD commands will add the record to the end of the file and set the CRP pointing to it. All other commands will fail with an end of file error code and the CRP will point to EOF.

- If the given key does not exist, and it is less than the highest key on file, the ADD and LOAD commands will add the record in the correct place according to the sequence of the keys; the READ APPROXIMATE command will retrieve the next higher key on file. The CRP will then be pointing at that record. All other commands will fail with an INVALID KEY error code. The CRP will then point to the next higher key on file.

The READ NTH RECORD command will attempt to locate the record by its sequence number rather than its key, and if found, the CRP will point to it. If not, an EOF error code will be returned and the CRP will point to EOF.

READ RANDOM (C6)

Calling format:

```
CODE%=&HC6
CALL KSAM(CODE%,FCB$,REC$)
```

Purpose:

To locate the record on file with the primary key that matches the key in the logical record area, and to move it to that area.

Description:

If the operation is successful, the record will be moved to the logical record area and the CRP will be pointing to it.

If there is an error no movement of data will take place. If the given key is higher than the highest key on file an EOF error will be returned. If the given key is less than the highest key on file but is non-existent an INVALID KEY error will be returned.

Use example KSAM - 12

The KSAM 80 File Management System

UPDATE (CA)

Calling format:

```
CODE%=&HCA
CALL KSAM(CODE%,FCB$,REC$)
```

Purpose: To modify a record

Description:

To locate the record on file with a primary key that matches the key in the logical record area, and to replace the contents with the contents of the logical record area.

If there is an error no movement of data will take place.

Use example KSAM - 8

DELETE (CB)

Calling format:

```
CODE%=&HCB
CALL KSAM(CODE%,FCB$,REC$)
```

Purpose:

To locate the record on the file with a primary key that matches the key in the logical record area, and remove it from the file.

Description:

If the record exists, it will be physically removed from the file and the space occupied by it will be reclaimed.

Empty data or key blocks are not physically removed from the file, their space is kept to be re-used by the same file. This means that the disk space allocated to the file will remain with that file even when some or all of its records have been deleted. If the only record of the file is deleted, then the file exists in an EMPTY state until a record is added or loaded.

After deletion the CRP will point to the previous record.

If an error occurs no record will be removed from the file.

Use example KSAM - 13

READ NTH RECORD (C8)

Calling format:

```
CODE%=&HC8
CALL KSAM(CODE%,FCB$,REC$)
```

Purpose:

To locate the record with a sequence number (sequence number 0, 1, 2 etc) that matches the number stored in the first two bytes of the primary key field of the logical record area, and move it to the logical record area.

Description:

The user must place a 16-bit unsigned integer in the first two bytes of the primary key field of the logical record area. This number is stored with its least significant byte first. The first record on file is considered to be record 0. For a successful retrieval the number must be in the range 0 to the number of records on file minus one.

If successful the record will be placed in the logical record area and the CRP will point to it.

If there is an error no movement of data will take place.

Use example KSAM - 7

READ APPROXIMATE (C7)

Calling format:

```
CODE%=&HC7
CALL KSAM(CODE%,FCB$,REC$)
```

Purpose:

To locate the first record on file with a primary key greater than or equal to the key in the logical record area, and to move it to the logical record area.

Description:

This is a very useful command if the exact record key is not known. Normally, it will only fail if the given key is higher than the highest on file. A partial key with the low order positions filled with zeros will either retrieve the desired record or the next higher key.

Usually sequential processing follows this operation. For example if a group of records exists for which the high order positions of the key are the same, a READ APPROXIMATE will retrieve the first of the group and the rest can be accessed sequentially.

When a record has been retrieved the CRP will point to it.

If the key supplied is higher than any on file then an EOF error occurs and the CRP is positioned at the EOF.

Use example KSAM - 5

The KSAM 80 File Management System

LOAD (C2)

Calling format:

```
CODE%=&HC2
CALL KSAM(CODE%,FCB$,REC$)
```

Purpose:

To add the record to the file, according to the rules of the ADD command, and attempt to preserve in each block the unused space specified at file creation time.

Description:

This command is of practical use only when adding successively higher keys to the end of the file, for example when building the file.

The only difference between the ADD and LOAD commands is that in such a case the unused space, specified when the CREATE command was issued, will be preserved.

LOAD behaves in exactly the same way as ADD, that is it attempts to fill the blocks completely, if the records are in a different sequence or if the LOAD command is used to insert records anywhere other than at the end of the file.

If there is an error the record is not added to the file and the contents of the file are not disturbed.

Use example KSAM - 3

ADD (C1)

Calling format:

```
CODE%=&HC1
CALL KSAM(CODE%,FCB$,REC$)
```

Purpose:

To add to the file the record found in the logical record area, in the correct sequence according to the primary key value of the logical record area.

Description:

The record is inserted in the correct position according to the primary key sequence; if the key is higher than any existing key then the record is added at the end of the file. The CRP will point to the new record.

The ADD command will disregard the unused space specified by the user and will attempt to fit as many records in a block as the block length permits. To preserve the unused space the user should add records using the LOAD command.

If there is an error, the record is not added to the file and the contents of the file are undisturbed. The position of the CRP can always be determined by issuing a READ CURRENT command.

If a record with the same key already exists, the operation will return an INVALID KEY error code, and the CRP will point to that record.

If a free block cannot be found, the operation will return a NO FREE BLOCKS error code, and the CRP will either point to the record with the next higher key or to EOF. The file should be closed and compacted (see section 7.4.3) or the number of volumes should be increased using the KSANV field in the KSAFCB (see section 7.5).

Use example KSAM - 2

7.3.4 Alternate Key Commands

Alternate key commands allow the user to create and maintain alternate files and access data bases through alternate record keys.

All alternate key commands involve two files: the data base file and one of its alternate files. Consequently there are two KSAFCBs involved in every command:

```
CODE%=&Hxx
CALL KSAM(CODE%,FCB$,REC$,FCBA$,RECA$)
```

will add an inversion record that corresponds to the current record in the data base file, to the alternate file specified by FCBA$.

The database must be open when any alternate key command is issued and, with the exception of the CREATE ALTERNATE command, the alternate file must also be open.

If an alternate key command is successful the code field will contain zero.

If an alternate key command fails the code field will contain the appropriate error code.

During the execution of an alternate key command an error can occur when accessing either the data base or the alternate file. If the error occurred on the data base record then the return code will range from C1H to CFH. If the error occurred on the alternate file then the return code will range from D1H to DFH.

Note that all KSAM 80 commands can be issued to an alternate file, and in that case KSAM 80 treats that file as a data base file so that if any errors occur the returned codes range from C1H to CFH.

CREATE ALTERNATE (D4)

Calling format:

```
CODE%=&HD4
CALL KSAM(CODE%,FCB$,DUMMY$,FCBA$)
```

Purpose:

To create a KSAM 80 file to be used as an alternate file for an existing data base file.

Description:

An entry is added to the directories of the volumes to which the file may extend. The header record is written on the first volume.

Use example KSAM - 9

READ BY ALTERNATE KEY (D8)

Calling format:

```
CODE%=&HD8
CALL KSAM(CODE%,FCB$,REC$,FCBA$,RECA$)
```

Purpose:

To move the data base record that corresponds to the current record on the given alternate file into the logical record area of the data base.

Description:

There are two possible reasons for an INVALID KEY error code to be returned:

- either there is no record on the data base with a primary key matching the primary key field on the current record of the alternate file; this usually indicates that the data base record has been deleted.

- or a record exists on the data base with the corresponding primary key but its alternate key value does not match the value of the alternate field on the current record of the alternate file; this is possibly due to modification of the alternate key field on the data base record.

  If an error does occur, no movement of data will take place. The CRP on the data base will behave in the same way as for the READ RANDOM command. The CRP on the alternate file remains unchanged.

Use example KSAM - 14

The KSAM 80 File Management System

READ FIRST (D9)

Calling format:

```
CODE%=&HD9
CALL KSAM(CODE%,FCB$,REC$,FCBA$,RECA$)
```

Purpose:

To move the first data base logical record, according to primary key sequence, that contains a given alternate key value to the logical record area of the data base.

Description:

The user must place the desired alternate key value in the alternate key field of the logical record for the appropriate alternate file before issuing this command.

If there is an error then no movement of data will take place. The CRP on the data base behaves in the same way as for the READ RANDOM command. The CRP on the alternate file will be adjusted as described in the READ APPROXIMATE command.

Use example KSAM - 12

The KSAM 80 File Management System

READ NEXT HOLDING (DA)

Calling format:

```
CODE%=&HDA
CALL KSAM(CODE%,FCB$,REC$,FCBA$,RECA$)
```

Purpose:

To move the next logical record of the data base that contains the same alternate key value as the current record of the given alternate file, to the logical record area of the data base.

Description:

The contents of the record area of the alternate file are used for the comparison. For example, if a READ FIRST (by alternate key) has been issued on an alternate file, then successful executions of READ NEXT HOLDING commands will access all logical records on the data base with the same alternate key value.

If an error occurs then no movement of data will take place. The CRP for the data base will be adjusted according to the rules of the read RANDOM command. The CRP for the alternate file will behave as described in the READ NEXT command.

Use example KSAM - 12

VERIFY INVERSION (D7)

Calling format:

```
CODE%=&HD7
CALL KSAM(CODE%,FCB$,REC$,FCBA$,RECA$)
```

Purpose:

To move the inversion record that corresponds to the current logical record of the data base, to the logical record area of the given alternate file.

Description:

This command attempts to locate the inversion of the current logical record of the data base on the given alternate file. If an inversion does not exist, an INVALID KEY error code will be returned.

If there is an error no movement of data will take place. The CRP for the data base will not be disturbed, and the CRP for the alternate file will be adjusted according to the rules of the READ RANDOM command.

Use example KSAM - 11

ADD INVERSION (D5)

Calling format:

```
CODE%=&HD5
CALL KSAM(CODE%,FCB$,REC$,FCBA$,RECA$)
```

Purpose:

To add an inversion to the given alternate file that will correspond to the current record of the data base.

Description:

If there is an error then the inversion will not be added, and the CRP for the data base will not be disturbed. The CRP for the alternate file will behave as described for the ADD command.

Use example KSAM - 10

DELETE INVERSION (D6)

Calling format:

```
CODE%=&HD6
CALL KSAM(CODE%,FCB$,REC$,FCBA$,RECA$)
```

Purpose:

To delete the inversion that corresponds to the current record of the data base from the given alternate file.

Description:
If there is an error then the inversion will not be deleted and the CRP for the data base will not be disturbed. The CRP for the alternate file will be adjusted according to the DELETE command.

Note - If a database record and the corresponding inversion have to be deleted, delete the inversion first.

Use example KSAM - 13

7.4 THE KSAMUT PROGRAM

Additional capabilities to those described in Chapter 3 are provided by the KSAMUT utility program. This program enables the user to perform certain file management tasks on KSAM 80 files.

The most frequently used function is CHANGE DISKS; this **must** be executed before and after any other KSAMUT function. A complete list of the available functions is as follows:

CHANGE DISKS	To change disks and initialize the operating system.
CREATE	To create a KSAM 80 data base or alternate file.
ERASE	To remove a KSAM 80 file from the directory.
COMPACT	To compact the file.
REORGANIZE	To reorganize the file.
STATUS	To print the file status.
RENAME	To rename a file.
COPY COMPACT	To copy the file in compacted form.
COPY REORGANIZE	To copy the file in reorganized form.

Any number of functions can be executed either on the same or on different files or disks during any one session.

7.4.1 Loading KSAMUT

The KSAMUT program can be loaded, simply by typing:

```
RUN "LKSAMUT"
```

The program will then display a menu, from which the user can select one function at a time.

If the selected function requires further information, the program will issue the appropriate prompts.

If a file has to be accessed, then the file unit, name and type must be provided; if the file extends to more than one volume, the file unit, name and type for all volumes must be provided. All volumes must be mounted.

7.4.2 Return Codes

All KSAM 80 and operating system return codes are possible.

If the execution of a function is successful, the program will respond with SUCCESSFUL, and will again display the choice of available functions.

If the function fails then the appropriate message is generated and the program awaits the next function.

7.4.3 Function Specifications

CHANGE DISKS

Purpose:

To change the disks and initialize the operating system.

Description:

This function must be executed if the user wishes to change disks.

After the completion of this function, the disk in drive A is the current disk.

CREATE

Purpose:

To create a KSAM 80 data base or alternate file.

Description:

A data base or alternate file can be created if the user follows the prompts and provides the necessary information.

For further information, see the KSAM 80 commands CREATE and CREATE ALTERNATE.

ERASE

Purpose:

To remove a KSAM 80 file from the disk directories.

Description:

If the file extends to more than one disk then all disks should be mounted.

If the function is successful, then all disk space occupied by the file will become available, and the file name will be removed from all the disk directories concerned.

For further information, see the KSAM 80 command ERASE.

COMPACT

Purpose:

To rewrite the original file, so that its blocks contain the maximum number of logical records.

Description:

The file name will remain the same. The new file may have fewer data or key blocks in use, but the disk space occupied by the compacted file will be the same as the original.

REORGANIZE

Purpose:

To rewrite the original file so that every block contains, if possible, the initial unused space specified at file creation.

Description:

The disk space occupied by the new file will be the same as the original file. A compact file can not be reorganized.

Note - In certain cases there may be an unreasonable delay in the operation of the COMPACT and REORGANIZE functions; in such cases it is recommended that COPY COMPACT and COPY REORGANIZE are used instead.

STATUS

Purpose:

To print the file status header information.

Description:

The following will be printed:

FILE NAME
RECORD LENGTH
INITIAL UNUSED SPACE
PAGES PER KEY BLOCK
PAGES PER DATA BLOCK
DATA BASE (or ALTERNATE FILE)
KEY LENGTH
KEY DISPLACEMENT
FIELD LENGTH IN DB (inversion files only)
FIELD DISPLACEMENT IN DB (inversion files only)
MAX RECORDS PER DATA BLOCK
FILE SIZE IN RECORDS
VOLUMES SPECIFIED
VOLUMES IN USE
KEY BLOCKS IN USE
DATA BLOCKS IN USE

RENAME

Purpose:

To rename a file.

Description:

The file to be renamed does not have to be a KSAM 80 file. This function does not need to have all volumes of a KSAM 80 file mounted to be executed.

COPY COMPACT

Purpose:

To copy the given file so that the blocks of the new file contain the maximum number of records.

Description:

The successful execution of this function will create a compact copy of the original file. The new file can be created on the same or different disks. It does not have to extend to the same number of volumes as the original file. If the new file is to extend to disks on which the original file still exists then their filenames must be different.

The number of data blocks, key blocks and the disk space occupied by the new file could be less than the original file.

COPY REORGANIZE

Purpose:

To copy the original file so that the blocks of the new file contain the initial unused space specified at file creation time.

Description:

This function is similar to the COPY COMPACT function. However, instead of compacting the blocks, it attempts to insert in each block the amount of empty space specified by the user at file creation time. The disk space occupied by the new file may be more than that of the original file.

Note - The COPY COMPACT and COPY REORGANIZE functions are the only functions that physically remove the free data and key blocks from the file.

7.5 FILE CONTROL BLOCKS

7.5.1 Setting Up An FCB

For each KSAM 80 file in the program, you must reserve an area to serve as the FCB for that file. The size of this area can be calculated from the following formula:

(128 + n x33) bytes

where n is the number of physical files (n = 1, 2 or 3).

This area is initialised by the user before the file is accessed and it is used to communicate information to KSAM 80 about the file. Some parts of it are also used by KSAM 80 for temporary storage and therefore they should not be altered by the user. Tampering with these fields may result in data being lost or made inaccessible.

The user must supply the following:

- The number of volumes the file should extend to.
- The key block length (in pages).
- The data block length (in pages).
- The amount of space to be left unused in each block. This may be equal to zero. If not, it must be in terms of logical records and must be less than the maximum number of records that fill a block.
- The logical record length in bytes (data bases only).
- The key length in bytes (data bases only).
- The displacement of the key from the beginning of the record (data bases only).
- The alternate key field length in the data base record, (alternate files only).
- The alternate key field displacement from the beginning of the data base record (alternate files only).

The user must also specify the address of KSAM 80 and the address of a buffer area.

The KSAM 80 File Management System

THE FILE IDENTIFICATION TABLE

The file identification table is used to communicate information about the various volumes of the file to KSAM 80.

Each volume is considered by the operating system in the same way as any other file, and as such every volume needs a 33-byte file identification block.

For single volume KSAM 80 files the address of this 33-byte area must be placed in the KSAFCB for the file.

For multi-volume files, a table of the file identification entries is constructed, with each entry 33 bytes long; the first entry corresponds to the first volume, the second entry corresponds to the second volume, etc.
Each 33 bytes entry consists of:
1 byte drive
"A"
"B"
"C"
"D"
8 bytes filename
3 bytes file extension
21 bytes binary zeros

All volumes of a given file must be mounted when the file is accessed. It is the user's responsibility to mount the disks on the correct drives.

THE LOGICAL RECORD AND BUFFER AREAS

The user must place the addresses of two areas, which are associated with the file in each KSAFCB.

The first area is called the BUFFER area and is used by KSAM 80 as a work area. It must not be altered by the user. The size of the buffer area is equal to the data block length of the file (in bytes) plus the key block length (in bytes). For example, if the data block length was defined at creation time as 4 pages, and the key block as 1 page, then the buffer size is 5 * 256 = 1280 bytes.
One buffer must be set aside for each KSAM 80 file in the program. Buffers may only be shared between files if the files are not open at the same time.

The second area is called the LOGICAL RECORD AREA and it is equal in size to the logical record length. This is where KSAM 80 will place the logical records read, and where the user will build records to be written to the file.
These addresses should be set before opening the file, but they may be reset at any time using the RESET BUFFER command. If the logical record address is set to binary zeros, KSAM assumes that the logical record is located immediately behind the buffer.

SETTING UP THE KEY FOR KEYED ACCESS

Since the logical record area is an image of the records on file, its key field is used to communicate the desired key for random access commands to KSAM 80. For example, if the user needs to retrieve the record with key 0123456789 then this value is placed in the key field of the logical record area.

The key required must be placed in the key field of the logical record area for the following commands:

- ADD
- DELETE
- LOAD
- READ RANDOM
- READ APPROXIMATE
- UPDATE
- READ FIRST WITH GIVEN ALTERNATE KEY

Although the key must normally be supplied by the user, in many cases the key will be placed in the correct position by KSAM 80 in a previous operation.

Sequential operations do not make use of the key field, they proceed strictly according to the logical sequence of the file. Sequential and random operations can be freely intermixed as processing needs dictate.

It is important that the user should be aware of the rules that govern the position of the current record pointer (CRP), which points at all times to the current record in the logical sequence. This is the subject of the next section.

7.5.2 The Current Record Pointer (CRP)

KSAM 80 maintains at all times a pointer to what is considered to be the current record in the logical sequence. The position of the CRP can always be determined by the READC command. Special cases are the beginning of file (BOF) and End of file (EOF) situations.

- A BOF condition occurs whenever the CRP points to a dummy record in front of the first logical record of the file. This is the case immediately after the file is opened. Reading backwards sequentially from the first record will also cause this to happen.

- EOF is reached when the CRP points to a dummy record beyond the last record on the file. This may happen when reading sequentially forward past the last record, or on any random command if the key is greater than any key on the file.

- Any successful READ, ADD or LOCATE command will cause the CRP to point to the successfully processed record.

- A successful DELETE command will cause the CRP to point to the previous record, in the primary key sequence, or to BOF if the first record of the file was deleted.

- Unsuccessful READ, UPDATE, or DELETE commands will position the CRP to the first record that has a key greater than the key specified. If the key is larger than any on the file, or if sequential reading past the last logical record was attempted, the CRP will point to EOF.

- An unsuccessful ADD command will position the CRP to a record in the file that has the same key as the one specified by the user, because primary keys must be unique. An exception to this rule is the case of an ADD command that fails because there is no space available on the disk; the CRP will then act as described in the previous paragraph.

- If the CRP is positioned at BOF and a READ PREVIOUS or READ CURRENT command is attempted, the CRP remains unchanged. This is also true if the CRP is positioned at EOF and a READ CURRENT or READ NEXT is attempted.

If an I/O error occurs during processing, the contents of the CRP become unreliable.

When in doubt issue a READ CURRENT command. This will return either the current record or a BOF/EOF/FILE EMPTY error code.

7.5.3 The KSAFCB Fields

The table below shows the structure of the KSAFCB and it is followed by explanations of the contents of each field mentioned in the table. The remainder of the KSAFCB is reserved for system use and should not be modified by the user.

FIELD	LENGTH	BYTE NUMBER (BASIC)	FIELD NAME
KSABA	2	7	Buffer address
KSAAI	2	9	Address of additional information
KSANV	1	11	Number of disk volumes
KSAFA	1	12	File attributes
KSAKB	1	13	Pages per key block
KSASB	1	14	Pages per data block
KSAUS	2	15	Initial unused space
KSARL	2	17	Logical record length
KSAPKL	1	19	Primary key length
KSAPKD	2	20	Primary key displacement
KSAAKL	1	22	Alternate key length
KSHAKD	2	23	Alternate key displacement

The KSAM 80 File Management System

The following list describes the information that should be included in each of the fields mentioned above. The numbers after the item names have the following meanings:

1) to be filled in always
2) to be filled in only for: CREATE and CREATE ALTERNATE
3) to be filled in only for: CREATE
4) to be filled in only for: CREATE ALTERNATE

KSABA 1) Must contain the address of an area used as a buffer by KSAM 80. This area should never be disturbed by the user.

KSAAI 2) This field may contain additional information that is required by the commands CREATE and CREATE ALTERNATE.

KSANV 1) This field is used to specify the number of volumes (between 1 and 3) to which the file may be extended; the default value is 1. It can be increased at any time, when the file is open, to allow the data to overflow to more disks than were originally specified. However, it must not exceed 3.

KSAFA 2) Contains the following attributes:
immediate (re-)write flag: set to on if KSAFA=1 or 3
suppress space management flag (empty blocks not to be re-used): set to on if KSAFA=2 or 3
both flags are off if KSAFA=0.

KSAKB 2) Must contain the number of pages (= sectors, each sector 256 bytes) per key block.

KSASB 2) Must contain the number of pages per data block.

KSAUS 2) Must contain the number of logical records per data block to be left empty when the file is built using the LOAD command; this must be less than the maximum number of logical records that will fit in a block.

KSARL 3) Must contain the length of a logical record as a 16-bit integer. This must be supplied when creating a data base file, but is ignored by KSAM 80 for an alternate file.

KSAPKL3) Must contain the length of the primary key as an 8-bit integer. This must be supplied for data base files, but is ignored for alternate files.

KSAPKD3) Must contain the displacement (in bytes) of the primary key field from the beginning of the logical record, expressed as a 16-bit integer. This must be supplied for data base files, but is ignored for alternate files.

KSAAKL 4) Must contain the length, as an 8-bit integer, of the field in the data base record that is to be used as an alternate key.

KSAAKD 4) Must contain the displacement (in bytes) of the alternate key field in the data base record. The same rules apply as described for the primary key displacement.

All other fields, apart from those described in this section, should not be disturbed by the user. However, they have to be cleared to zeros before accessing the file for the first time.

All of the parameters supplied at file creation time will be retrieved by KSAM 80 when the file is opened and placed in their corresponding fields as described above. The user may examine but not alter these values.

7.6 COMMAND CODES AND MEANINGS

The hexadecimal codes for KSAM 80 commands are given below:

CODE(HEX)	COMMAND
C0	CLOSE
C1	ADD
C2	LOAD
C3	READ PREVIOUS
C4	READ CURRENT
C5	READ NEXT
C6	READ RANDOM
C7	READ APPROXIMATE
C8	READ NTH RECORD
C9	RETRIEVE KEY
CA	UPDATE
CB	DELETE
D0	OPEN
D3	CREATE
D4	CREATE ALTERNATE
D5	ADD INVERSION
D6	DELETE INVERSION
D7	VERIFY INVERSION
D8	READ BY ALTERNATE KEY
D9	READ FIRST
DA	READ NEXT HOLDING

The KSAM 80 File Management System

The following list includes a brief description of the purpose of each command:

ADD	Adds a record to a data base file.
ADD INVERSION	Adds an inversion record to an alternate file.
CLOSE	Closes a file.
CREATE/ CREATE ALTERNATE	Creates a KSAM 80 file.
DELETE	Deletes a record from a data base file.
DELETE INVERSION	Deletes the inversion record (if any) that corresponds to the current data base record.
LOAD	Adds a record to a data base file and attempts to preserve the unused space in the block.
OPEN	Opens a file.
READ APPROXIMATE	Reads the first record whose primary key is greater than or equal to a given key.
READ BY ALT.KEY	Reads the data base record corresponding to the current inversion record.
READ CURRENT	Reads the current record.
READ FIRST	Reads the first data base record that contains the given alternate key value.
READ NEXT	Reads the next record in the primary key sequence.

The KSAM 80 File Management System

READ NEXT HOLDING	Reads the next data base record that contains the same alternate key value as the current inversion record.
READ NTH RECORD	Reads the nth record of the file.
READ PREVIOUS	Reads the previous record in the primary key sequence.
READ RANDOM	Reads a record randomly by primary key.
RENAME	Renames a file on the disk directory.
RETRIEVE KEY	Reads the primary key of the current record.
UPDATE	Replaces a given record.
VERIFY INVERSION	Reads the inversion record (if any) corresponding to the current data base record.

7.7 KSAM 80 ERROR AND RETURN CODES

7.7.1 Command Successful

If a KSAM 80 command is successful then the code field will contain zeros.

7.7.2 Command Fails

If a KSAM 80 command fails then the code field will contain the appropriate error code.

7.7.3 Error Codes

All operating system error codes plus KSAM 80 error codes are possible.

When alternate key commands are issued, two files are involved: the data base file and one of its alternate files. If an error occurs the user must be able to distinguish which file caused it. For this reason KSAM 80 returns a Cx error if the data base is at fault and a Dx error if the alternate file is causing the exception.

If a non-alternate key command is issued a Cx error will be returned.

The following is a list of the KSAM 80 errors and the appropriate codes for data base and alternate files:

ERROR TYPE	DATA BASE ERROR CODE	ALTERNATE FILE ERROR CODE
FILE EMPTY	0C1H	0D1H
BEGINNING OF FILE	0C2H	0D2H
END OF FILE	0C3H	0D3H
INVALID KEY	0C4H	0D4H
INVALID REQUEST	0C7H	0D7H
INVALID CREATE PARAMETERS	0C8H	0D8H
FILE EXISTS	0C9H	0D9H
KEY SET FULL	0CAH	0DAH
NO FREE BLOCKS	0CBH	0DBH
OPEN REQUEST FAILED	0CDH	0DDH
I/O ERROR	0CEH	0DEH

The KSAM 80 File Management System

FILE EMPTY

This error code is returned when an attempt is made to access a record of an empty file. A file is considered EMPTY if no records have been added or loaded, or if all its records have been deleted.

Note - After a CREATE or CREATE ALTERNATE command the file is empty and closed, it must therefore be opened before records can be loaded or added.

BEGINNING OF FILE (BOF)

This error code is returned, by the READP, READC or RETRV commands, when an attempt is made to read a record before the first record of the file.

For example, if the current record pointer (CRP) is pointing to the first record then a READ PREVIOUS command will result in a BOF error, similarly if an OPEN command is followed by a READ CURRENT or READ PREVIOUS command it will result in a BOF error.

END OF FILE (EOF)

This error code is returned when an attempt is made to access a record beyond the last record of the file.

INVALID KEY

This error code is returned by random access and alternate key commands.

When adding or loading records, this error indicates that a record with the same primary key already exists.

When reading, updating or deleting a record, this error indicates that there is no such record on file.

INVALID CREATE PARAMETERS

This error code indicates that the given creation parameters are incorrect; it can only be returned by the CREATE and CREATE ALTERNATE commands.

The description of the file to be created could contain any of the following errors:

- Key larger than the logical record.
- Key length and displacement larger than the record length.
- Key length equal to zero.
- Logical record length larger than data block length.
- Unused space per block, greater or equal to the maximum number of records per block.
- Zero pages per block.

FILE EXISTS

This error code is returned by a CREATE or CREATE ALTERNATE command. It indicates that a file with the given file name and type already exists on that disk.

KEY SET FULL

This error can only be returned by an ADD, LOAD or ADDALT command.

The maximum number of key blocks is 16, the KEY SET FULL error will occur if an attempt is made to allocate a key block to a file that already has 16 key blocks. The record causing the exception is not added to the file.

To continue it is advisable to close the file, compact it using the utility program and proceed by adding the last record again.

NO FREE BLOCKS

This error can only be returned by the ADD, LOAD or ADDALT command when an attempt is made to allocate a new data block to the file and there is no disk space available at the end of the file. The record concerned will not be added or loaded.

If the number of volumes cannot be increased, the file must be closed and compacted using the utility program to release any unused space.

OPEN REQUEST FAILED

This error can be returned by any command that attempts to open the file, for example OPENR, RENAMR, or ERASER.

It indicates that one of the files specified in the file table could not be opened (usually because it could not be found on the specified volume). In the case of multi-volume files it indicates that the first file specified in the table did not contain the file header page.

I/O ERROR

This error can be returned by any command that causes a disk access. It usually indicates hardware failure.

If no hardware error is involved, it may be that some file pointers were lost, causing the system to attempt a read operation beyond the end of the file. In this case the file may have to be restored from a backup.

7.8 SPACE CONSIDERATIONS

The following notation is used in this section:

I = Page length = 256 bytes
R = Logical record length, in bytes
B = Key block length, in bytes
L = Data block length, in bytes
K = Primary key length, in bytes
N = Maximum number of data blocks in the file
M = Maximum number of records in the file
C = Pages per data block = Blocking factor
D = Pages per key block
F = Unused space, in logical records
S = Buffer size, in bytes
P = Records per block

The relationships between these quantities should be:

- Block length must be a multiple of a page.
- Key block length: B = D * I
- Data block length: L = C * I
- Buffer size: S = L + B
- Key length K has a minimum of 248 bytes.
- Maximum number of data blocks: N = 16 * INT(B/(K+5))
- Record length R is fixed; records may not span blocks.
- Records per data block: P = INT(L/R)
- Maximum number of records: M = N * P
- Maximum number of volumes per file is four. All volumes must be on-line when processing.
- Duplicate keys are only allowed for alternate keys. Primary keys must be unique.