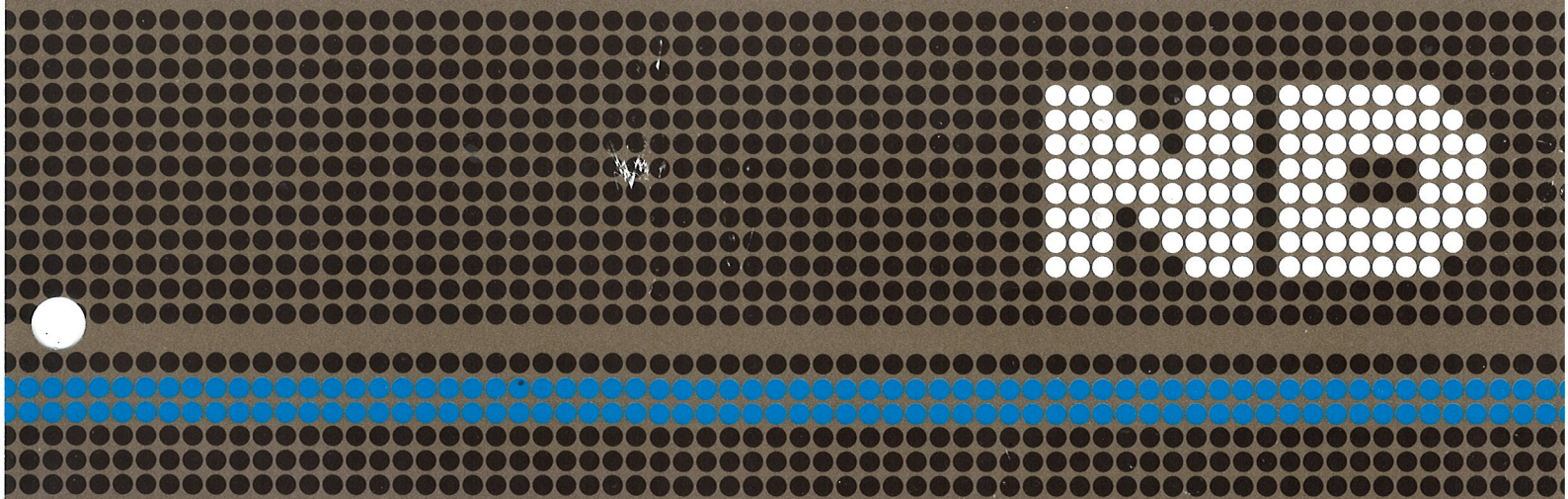


ISAM

Reference Manual

ND-60.108.6 EN



ISAM

Reference Manual

ND-60.108.6 EN

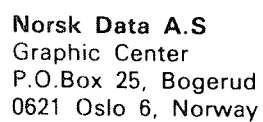
NOTICE

The information in this document is subject to change without notice. Norsk Data A.S. assumes no responsibility for any errors that may appear in this document. Norsk Data A.S. assumes no responsibility for the use or reliability of its software on equipment that is not furnished or supported by Norsk Data A.S.

The information described in this document is protected by copyright. It may not be photocopied, reproduced or translated without the prior consent of Norsk Data A.S.

Copyright © 1986 by Norsk Data A.S.

ND Indexed Sequential Access Method Manual
Publ.No. ND -60.108.6 EN



Manuals can be updated in two ways, new versions and revisions. New versions consist of a complete new manual which replaces the old manual. New versions incorporate all revisions since the previous version. Revisions consist of one or more single pages to be merged into the manual by the user, each revised page being listed on the new printing record sent out with the revision. The old printing record should be replaced by the new one.

New versions and revisions are announced in the Customer Support Information (CSI) and can be ordered as described below.

The reader's comments form at the back of this manual can be used both to report errors in the manual and to give an evaluation of the manual. Both detailed and general comments are welcome.

These forms and comments should be sent to:

Documentation Department
Norsk Data A.S
P.O. Box 25, Bogerud
0621 Oslo 6, Norway

Requests for documentation should be sent to the local ND office or (in Norway) to:

Graphic Center
Norsk Data A.S
P.O. Box 25, Bogerud
0621 Oslo 6, Norway

Preface:

THE PRODUCTS

This manual describes the ISAM (Indexed Sequential Access Method) file accessing software:

ND ISAM ND-210073 (version J)

ISAM ND-500 ND-210343 (version J)

THE READER

This manual will be of interest to programmers developing systems using the ISAM file accessing software.

PREREQUISITE KNOWLEDGE

The reader should have be able to write programs in a language which can access ISAM files.

THE MANUAL

This manual describes details of the ISAM file accessing facilities. It also contains descriptions of the utility programs, ISAM Service Program and ISAM Interactive, which are easy-to-use tools for accessing ISAM files.

RELATED MANUALS

The manuals for the programming languages - FORTRAN, PLANC, Pascal - are necessary if ISAM is to be used in programs written in any of these languages. Note that the details of accessing ISAM files from COBOL are documented in the COBOL Manual.

FORTRAN Reference Manual	ND-60.145
Pascal Reference Manual	ND-60.222
PLANC Reference Manual	ND-60.117
COBOL Reference Manual	ND-60.144

CHANGES FROM THE PREVIOUS VERSION

New features and calls are marked with lines in the margin in "Features of the ISAM system" (page 3), and in "Summary of ISAM calls" (page 7). These and other major changes are also marked in the text itself.

TABLE OF CONTENTS

Section	Page
1 INTRODUCTION	1
1.1 General ISAM data structure	2
1.2 Features of the ISAM system	3
1.3 An illustration of the use of an ISAM file	4
2 OVERVIEW OF ISAM CALLS	7
2.1 Summary of ISAM calls	7
2.2 Parameters in ISAM calls on the ND-500	9
3 ISAM FILE OPERATIONS	11
3.1 Initialize ISAM buffer (ISINI)	11
3.1.1 Declaration of group keys	14
3.2 Open file and associated indexes (ISOPF)	16
3.3 Close file and associated indexes (ISCLF)	18
3.4 Verify file consistency (ISVER)	19
3.5 Reset ISAM error flag (ISFLG)	20
3.6 Fetch file system error code (ISERR)	21
3.7 Check/restore data part consistency (ISCON)	22
3.8 Regenerate index tables (ISRGN)	23
3.9 Create/delete index table (ISIND)	24
4 ISAM RECORD OPERATIONS	25
4.1 Read next record (ISRNK,ISRVN)	25
4.2 Read record using key (ISRUK,ISRKV)	26
4.3 Write record (ISWRT,ISWRV)	27
4.4 Rewrite record (ISREW,ISRWV)	28
4.5 Delete record (ISDEL)	29
4.6 Delete record using key (ISDLK)	30
4.7 Start using key (ISTRK)	31
4.8 Remember current record (ISREM)	32
4.9 Restart current record (ISRES)	33
4.10 Lock next record (ISLCK)	34
4.11 Unlock locked records (ISUNL)	35
4.12 Read relative record (ISREL)	36
4.13 Get current record number (ISCUR)	37
4.14 Set read direction (ISDIR)	38
4.15 Flush out modifications (ISFLU)	39
5 ACCESSING ISAM FILES IN MULTIUSER MODE	41
5.1 Concurrency control	42
5.2 Definition of an ISAM process	44
5.3 Multiuser file requirements	45
5.3.1 Conversion of indexed files to continuous files	45
5.3.2 Changing logical unit numbers (ISLUN) used by the multiuser supervisor	46
5.4 Example of use of multiuser mode	47
6 ISAM SERVICE PROGRAM AND ISAM INTERACTIVE	49

Section		Page
7	FILE STATUS REPORTING	51
8	EXCEPTION HANDLING	53
9	CONSISTENCY OF THE ISAM FILES	55
10	LOADING AN ISAM PROGRAM	57
11	PERFORMANCE CONSIDERATIONS	59
12	ISAM EXAMPLE PROGRAMS	63
12.1	A FORTRAN program using ISAM	63
12.2	A Pascal program using ISAM	65
13	INDEX	67

The Indexed Sequential Access Method (ISAM) is especially suited to EDP applications where data records are stored and retrieved either randomly or in sequential order. Records may be retrieved in random order by the use of a record identifier called an index or key, or in sequential order of a key data field.

ISAM is call-oriented, ie. ISAM services are obtained by calling routines from the ISAM library.

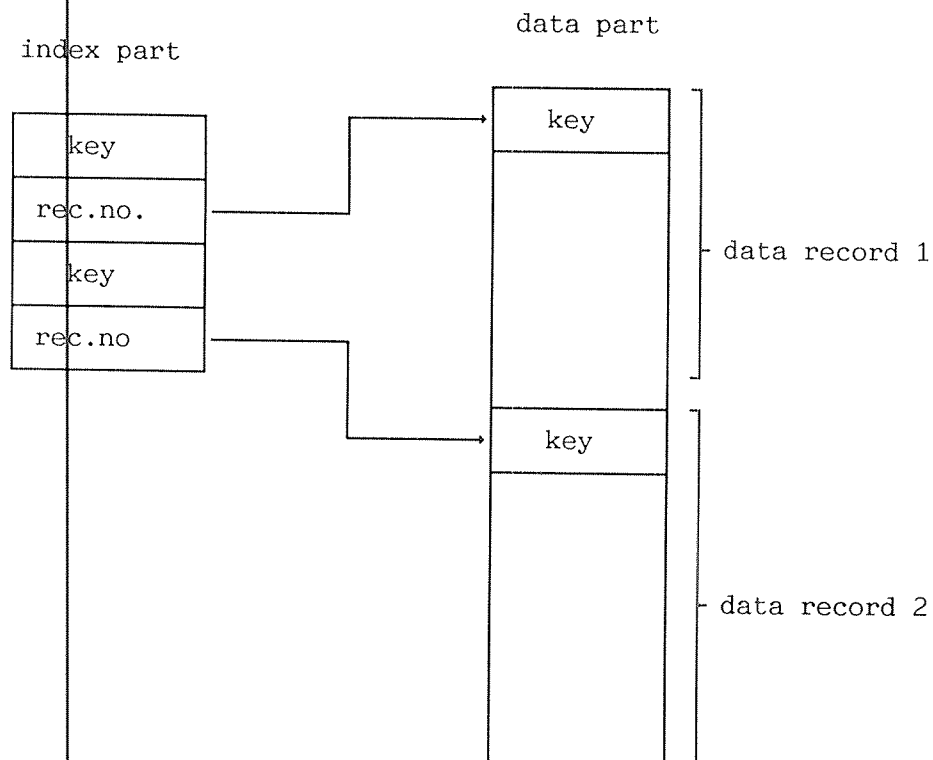
ISAM routines can be called from FORTRAN, Pascal and PLANC programs. Note that ISAM has been integrated into COBOL, and hence COBOL programs cannot use the routines as they are documented in this manual. The ISAM system consists of the following parts:

1. ISAM library (ISAM-1BANK, ISAM-2BANK for the ND-100 and ISAM-LIB for the ND-500) - all facilities which are callable from application programs to manipulate an ISAM file.
2. ISAM Service Program (ISAM-SERVICE) - an interactive utility program for general maintenance of entire ISAM files.
3. ISAM Interactive (ISAM-INTER) - an interactive utility program for manipulating the data in an ISAM file.
4. ISAM Multiuser Supervisor:
 - ND-100: ISAMRT - an RT program to supervise an ISAM multiuser process.
 - ND-500: IS-MULTI - a common link segment to supervise an ISAM multiuser process.

.1.1 General ISAM data structure

An ISAM file is usually located on two SINTRAN files. The index part is stored on one file, and the data part on another. Thus, the data and the data structure description are completely separated. The logical connection between these two files is shown below. The figure is simplified, but shows the principal structure.

Example: An ISAM file (with one key) containing two data records:



Note that the index and data parts may be placed in one SINTRAN file.

.1.2 Features of the ISAM system

1. The design is based on the ANSI-74 COBOL standard.
2. ISAM is a background oriented system.
3. Key length may be up to 240 bytes.
4. Up to 64 keys are allowed. All may have duplicates. Keys may overlap in the data record. All key values may be modified by the REWRITE call.
5. Group keys are allowed.
6. Maximum number of ISAM files within one program is 64.
7. Maximum number of data records is approximately 17,000,000, ie. there is no practical limitation.
8. Data record length can be fixed or variable. Maximum length of data record is 32 Kbytes.
9. ISAM may be run in four different modes:
 1. Single user - buffered input, buffered output.
 2. Single user - buffered input, immediate output.
 3. Multiuser - buffered input, immediate output, automatic unlock.
 4. Multiuser - buffered input, immediate output, manual unlock.
10. Records may be read in both ascending and descending order.
11. Records may be accessed either via index tables, or directly by the record number.
12. The consistency of an ISAM file can be verified - and, if necessary, reestablished - at runtime.
13. The ISAM file structure is dynamic, ie., index tables (keys) can be created/deleted at runtime.

.1.3 An illustration of the use of an ISAM file

Visualize yourself arriving at your office after having done battle with the everpresent, never-pleasant Morning Traffic Monster. You are already late for work, so imagine how delighted you would feel to find somebody's car sitting in your Very-Own, Officially-Assigned parking place. 8:10 A.M. and guess what? It's Decision Time once again! How are you going to deal with your problem?

Turn around and head for home, right?

Perhaps. But an equally satisfactory solution might be to call for the services of Supersystem, your installation's definitive answer for Every Conceivable Problem. Supersystem has a reputation for doing weird things; maybe this time it could help you to locate the owner of the car.

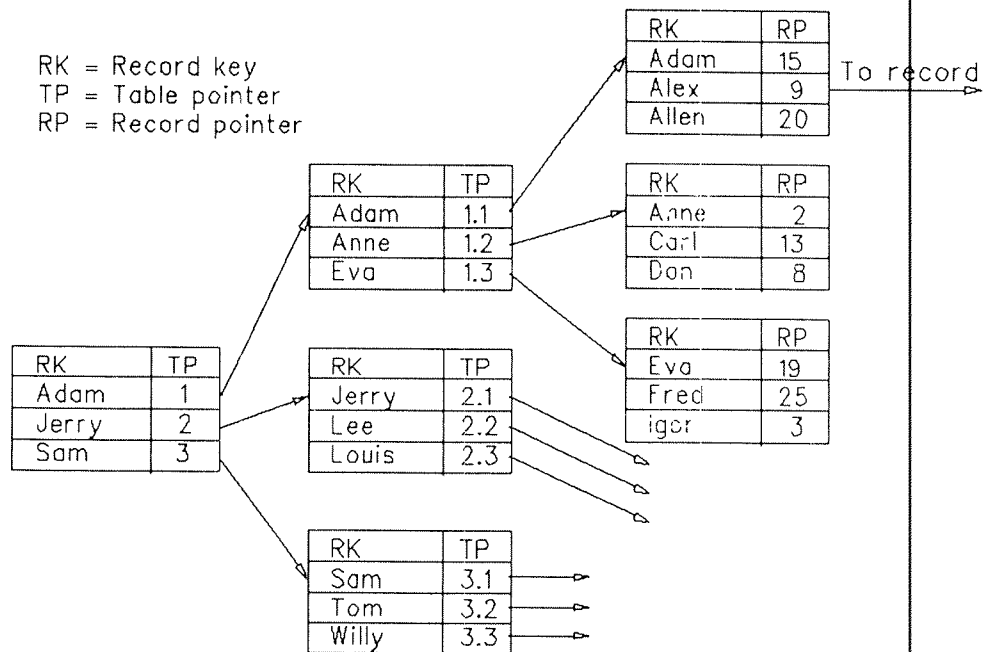
But first, you must ask Supersystem a few key questions;

1. Who out of 100 employees owns one of the 50 parked cars?
2. What is the owner's telephone extension?
3. What did you do to deserve this?

Within the wiring of Supersystem there is a record segment which looks like this:

unique key			
↓			
unique key		key with duplicates	
↓		↓	
Name	Tel.ext.	Car-reg.-no.	

Although everyone in your installation has a different name and can be reached by telephone, not everyone owns a car. For all those who do not, CAR-REG-NO contains a universal non-owner code, while for the car-owners themselves, CAR-REG-NO contains their vehicles' actual registration number. Match the registration number to NAME and your problem is solved. What is more, you are left with an example of one file with several keys, one of them duplicate.



The figure above depicts the ISAM search pattern employed to locate a single record on a direct-access device. The search proceeds through a tree-like structure of tables, each table having a limited capacity to store information concerning the record's location. When a table exceeds its capacity, another table, or level, is created automatically. Each table contains both keys and pointers. These pointers serve as guides to the next level within the search pattern. As new records are placed on or deleted from the file, all index tables within the table network are automatically updated.

.2 OVERVIEW OF ISAM CALLS

.2.1 Summary of ISAM calls

Note: All calls with a letter V at the end of their name (ISRVN, ISRVK, ISRVV and ISRVV), are for files with variable length records only.

INITIALIZE ISAM BUFFER

Call ISINI (<function>,<information>,<status>)

OPEN FILE AND ASSOCIATED INDEXES

Call ISOPF (<access-mode>,<file-id>,<file-name>,<status>)

CLOSE FILE AND ASSOCIATED INDEXES

Call ISCLF (<file-id>,<status>)

VERIFY FILE CONSISTENCY

Call ISVER (<file-id>,<status>)

CHECK DATA PART CONSISTENCY

Call ISCON (<file-id>,<restore>,<num-free-rec>,<status>)

REGENERATE INDEX TABLES

Call ISRGN (<file-id>,<status>)

RESET ISAM ERROR FLAG

Call ISFLG (<file-id>,<status>)

FETCH FILE SYSTEM ERROR CODE

Call ISERR (<error-code>)

CREATE/DELETE INDEX TABLES

Call ISIND (<file-id>,<key-id>,<code>,<status>)

READ NEXT RECORD

Call ISRNK (<file-id>,<record>,<status>)

Call ISRVN (<file-id>,<record>,<length>,<status>)

READ RECORD USING KEY

```
Call ISRUK (<file-id>,<key-id>, <record>, <key-disp>,  
           <status>)  
Call ISRKV (<file-id>,<key-id>,<record>, <key-disp>,  
           <length>,<status>)
```

WRITE RECORD

```
Call ISWRT (<file-id>,<record>,<status>)  
Call ISWRV (<file-id>,<record>,<length>,<status>)
```

REWRITE RECORD

```
Call ISREW <file-id>,<record>,<status>)  
Call ISRWV (<file-id>,<record>,<length>,<status>)
```

DELETE RECORD

```
Call ISDEL (<file-id>,<status>)
```

DELETE RECORD USING KEY

```
Call ISDLK (<file-id>,<record>,<key-disp>,<status>)
```

START USING KEY

```
Call ISTRT (<file-id>,<code>,<key-id>,<record>,  
           <key-disp>,<status>)
```

REMEMBER CURRENT RECORD NUMBER

```
Call ISREM (<file-id>,<status>)
```

RESTART CURRENT RECORD

```
Call ISRES (<file-id>,<status>)
```

LOCK NEXT RECORD

```
Call ISLCK
```

UNLOCK LOCKED RECORDS

```
Call ISUNL (<file-id>,<status>)
```

READ RELATIVE RECORD

```
Call ISREL (<file-id>,<record-no>,<record>,<status>)
```

GET CURRENT RECORD NUMBER

```
Call ISCUR (<file-id>,<record-no>,<status>)
```

SET READ DIRECTION

```
Call ISDIR (<file-id>,<direction>,<status>)
```

FLUSH OUT MODIFICATIONS

Call ISFLU (<file-id>,<status>)

.2.2 Parameters in ISAM calls on the ND-500

The following variables used as parameters in ISAM calls must be declared as 16 bit integer (INTEGER*2):

- The status returned from all ISAM calls.
- The INFO array used in the ISINI call.
- The file identification and the key identification.
- The access code used in the ISOPF call.

The following variables used as parameters in ISAM calls may be declared as either 16 bit or 32 bit integer arrays:

- The file name array used in the ISOPF call.
- The array used for each data record.

All the other variables used as parameters in ISAM calls must be declared as 32 bit integer (INTEGER or INTEGER*4).

It is extremely important to be aware of the above rules. If these are not followed, the results are unpredictable, and the process may enter ISAM's programmed trap (trap number 7635B).

.3 ISAM FILE OPERATIONS

.3.1 INITIALIZE ISAM BUFFER (ISINI)

Purpose.

ISAM subroutines use a buffer pool for data records and index information. This buffer pool must be initialized before any other operation. The same buffer pool holds a description of all files and all keys to be accessed during a run.

Syntax.

Call ISINI <function>,<information>,<status>)

Parameter description.

<function> single word, may take one of five values:

- | | |
|----------------------|---------------------------|
| * 0 max. no. of keys | <information> is used |
| 1 start definition, | <information> is not used |
| 2 new ISAM file, | <information> is used |
| 3 new key, | <information> is used |
| 4 end definition, | <information> is not used |

Functions marked with * may be omitted.

<information> array of up to five words; contents depend on the <function> value.

<function> = 0 maximum number of keys, default 6
word 1 contains maximum number of keys (up to 64)
word 2 contains zero

<function> = 1 start definition. The buffer is
cleared.

<function> = 2 new ISAM file
word 1 contains file identification
word 2 contains record length in number of bytes
word 3 contains -1 for variable length data
records
word 5 contains zero if data/indexes on separate
SINTRAN files > 0 number of pages
for the index part; data and
indexes are in the same SINTRAN
file.

<function> = 3 new key
word 1 contains file identification
word 2 contains key identification
word 3 contains key position, in bytes, within
record (first=0)
word 4 contains key length in bytes
word 5 contains duplicates allowed (NO=1, YES=0)

<function> = 4 switch on internal ISAM buffering. Note that this function must be called, once only, prior to the first ISAM OPEN call. However, more ISAM files may be defined (function 2 and 3) following the function 4 call.

<status> return parameter, single integer, may take the following values:

- 0 normal exit
- 1 invalid function code
- 2 file-identification not unique or record length not valid
- 3 new key rejected
- 4 too many files
- 6 buffer space insufficient

Rules.

1. The ISINI calls are the same whether they are used to declare a new file or an existing one.
2. The maximum number of files that may be declared in one program is 64.
3. The maximum number of keys for one file is 64, each of which may have duplicates. Keys may overlap in the data record.
4. The first key definition in a file is considered to be the PRIME KEY.
5. A practical limit for key lengths is 240 bytes. Note that ISAM will treat key values as a string of bits.
6. If variable length records are being used, the record length specified in ISINI (2,-,-) denotes how a logical record is split into physical records. All keys must lie in the first part of the record.
7. Maximum length of data records is 32 Kbytes. For large data records, the buffer length must be increased to accomodate at least one data record.
8. The buffer length provided by ISAM is 8 Kbytes for 1BANK systems and 24 Kbytes for 2BANK systems. To improve performance, this buffer may be increased; see Section 11, PERFORMANCE CONSIDERATIONS.

```
*****
* Declaration of the ISAM file SUPER-SYSTEM (ND-100 and ND-500) *
*****

      SUBROUTINE DECLARE
      INTEGER*2 INFO(5),IST
*
* Start definition
*
      CALL ISINI(1,INFO,IST)
      IF (IST.NE.0) GOTO 900
*
* Declare data file
*
      INFO(1) = "SU"                ;% File identification
      INFO(2) = 36                  ;% Record length in bytes
      INFO(3) = 0                   ;% Fixed record length
      INFO(5) = 0                   ;% Data/indexes on separate files
      CALL ISINI(2,INFO,IST)
      IF (IST.NE.0) GOTO 900        ;% Error
*
* Declare 3 keys (Name, Telephone and Car-number)
*
      INFO(1) = "SU"                ;% File identification
      INFO(2) = "NA"                ;% Key identification
      INFO(3) = 0                   ;% Key byte position (rel 0)
      INFO(4) = 20                  ;% Key byte length
      INFO(5) = 1                   ;% Duplicates not allowed
      CALL ISINI(3,INFO,IST)
      IF (IST.NE.0) GOTO 900        ;% Error

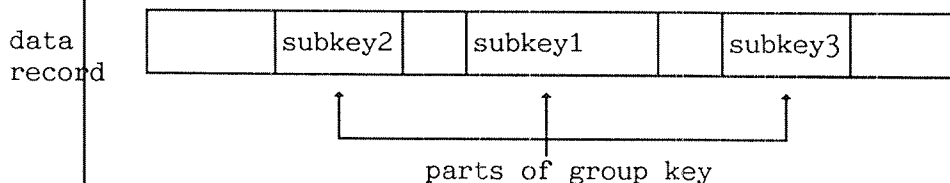
      INFO(1) = "SU"                ;% File identification
      INFO(2) = "TE"                ;% Key identification
      INFO(3) = 20                  ;% Key byte position (rel 0)
      INFO(4) = 4                   ;% Key byte length
      INFO(5) = 1                   ;% Duplicates not allowed
      CALL ISINI(3,INFO,IST)
      IF (IST.NE.0) GOTO 900        ;% Error

      INFO(1) = "SU"                ;% File identification
      INFO(2) = "CA"                ;% Key identification
      INFO(3) = 24                  ;% Key byte position (rel 0)
      INFO(4) = 12                  ;% Key byte length
      INFO(5) = 0                   ;% Duplicates allowed
      CALL ISINI(3,INFO,IST)
      IF (IST.NE.0) GOTO 900        ;% Error
*
* End of declaration, return to caller
*
      CALL ISINI(4,INFO,IST)
      IF (IST.NE.0) GOTO 900
      RETURN
*
* Error in declaration, stop execution
*
900   OUTPUT(1)'*** Error in ISAM declaration *** Status =',IST
      STOP
      END
```

.3.1.1 Declaration of group keys

Normally, key fields for use in indexes consist of one continuous part of a data record. Group keys, however, consist of up to 3 different parts of the data record. The parts may lie anywhere in the record and in any sequence. Other than the way the group key is constructed, it is used in exactly the same way for all index operations.

Layout of a group key in a record.



Declaration of a Subkey in an ISINI Call.

```

<function> = 3    new subkey
word 1 contains  file identification
word 2 contains  subkey identification. Subkeys must
                  use a negative integer
word 3 contains  subkey position, in bytes, within
                  record (first=0)
word 4 contains  subkey length in bytes
word 5 contains  duplicates allowed (NO=1, YES=0)
  
```

Declaration of a Group Key in an ISINI Call.

```

<function> = 3    new group key
word 1 contains  file identification
word 2 contains  key identification. Group keys must
                  use a negative integer
word 3 contains  subkey identifier, for first subkey
                  in group
word 4 contains  subkey identifier, for second subkey
                  in group
word 5 contains  subkey identifier, for third subkey
                  in group (or 0)
  
```

Rules.

1. All subkeys must be declared before all other key declarations for a file.
2. The subkey identification parameter described above, must use the values -1, -2, -3, and so on, for each consecutive subkey declaration.
3. The group key identification parameter described above, must use the values -101, -102, -103, and so on, for each consecutive group key declaration.

4. A subkey may not be used alone; it is allowed only as a part of a group key. However, the same part of the record might be declared separately as a normal key.
5. Subkeys, once declared, may be specified in any order within a group key declaration.
6. A group may consist of two or three subkeys. If there are only two, word five above must be set to zero.
7. A group key is unique only if it consists of unique subkeys.
8. A subkey, once declared, may be declared within several group keys.

.3.2 Open file and associated indexes (ISOPF)

Purpose.

To declare your intention to the system and associate the file identification to the actual file name. This must be done before attempting any record operation on a file. The first time the data file is opened, ISAM automatically creates the file which contains the index table.

Syntax.

Call ISOPF (<access>,<file-id>,<file-name>,<status>)

Parameter description.

<access> single word that may contain

"RM" Read access only

"WM" Write access only

"UM" Read and Write access

<file-id> single word, file identification

<file-name> array of words which contains the SINTRAN file name

<status> single word, input: run mode, output: status.

Rules.

1. The data part and the index part of an ISAM file may be placed on different directories. In this case the file name parameter in ISOPF must have the following layout:
(DIR1/DIR2:USER)FILENAME:TYPE
The data part will then be placed on the directory DIR1, the index part on DIR2.
2. The <file-id> must be known to ISAM; it is the same value that was declared with ISINI (2, <file-identification> ,<status>)
3. Initially the file must be empty, but it must exist; for this you may use:

@CREATE-FILE SUPER-SYSTEM : DATA, 0

The file which will hold the index tables must either be a non-existent file, or an existing file containing zero bytes. This file is named <file-name> :ISAM.
4. The first opening of the file must be done with <access> for write (WM) or read-and-write (UM).

5. If ISOPF returns status value 94, an error condition may exist in the ISAM file which has been opened. To check, call ISVER (verify consistency) before using the file again.
6. The four run modes are:
 0. Single user - buffered input, buffered output
 1. Single user - buffered input, immediate output
 2. Multiuser - buffered input, immediate output, automatic unlock
 3. Multiuser - buffered input, immediate output, manual unlock.

Example: (Note the space after the file name)

IST = 1

CALL ISOPF ("WM","SU","SUPER-SYSTEM:DATA ",IST)

.3.3 Close file and associated indexes (ISCLF)

Purpose.

To inform ISAM that file processing has been finished.

Syntax.

Call ISCLF (<file-id> , <status>)

Parameter description.

<file-id> single word, file identification
<status> return parameter, single word

Rules.

1. An ISAM file must be closed properly; otherwise some records and index tables may not be written out, causing inconsistencies between the data file and its associated indexes.

Example:

CALL ISCLF ("SU",IST)

.3.4 Verify file consistency (ISVER)

Purpose.

To verify consistency within the index and data parts of an ISAM file. See also the related ISCON call.

Syntax.

Call ISVER (<file-id>,<status>)

Parameter description.

<file-id> single word, file identification
<status> single word, return parameter

Rules.

1. The ISAM file must have been opened for READ or READ/WRITE access.

Example:

CALL ISVER ("SU",IST)

.3.5 Reset ISAM error flag (ISFLG)

Purpose.

To reset the ISAM error flag.

Syntax.

Call ISFLG (<file-id>,<status>)

Parameter description.

<file-id> single word, file identification

<status> single word, return parameter.

Rules.

1. The file must have been opened for READ/WRITE access.
2. This call should only be used after an ISVER call has shown the ISAM file to be consistent.

Example:

CALL ISFLG ("SU",IST)

.3.6 Fetch file system error code (ISERR)

Purpose.

To retrieve the file system error code if an ISAM status value 99 has been returned.

Syntax.

Call ISERR (<error-code>)

Parameter description.

<error-code> single word, return parameter

Example:

CALL ISERR (IERRCOD)

.3.7 Check/restore data part consistency (ISCON)

Purpose.

To check/restore the consistency within the data part of an ISAM file, ie. the variable length record consistency and the free list consistency (the free list is the list of deleted records). See also the related ISVER call.

Syntax.

Call ISCON (<file-id>,<restore>,<num-free-rec>,<status>

Parameter description.

<file-id>	single word, file identification
<restore>	single word, restore flag:
	0 - verify only
	1 - verify and restore
<num-free-rec>	single word, return parameter, number of deleted records on the ISAM file
<status>	single word, return parameter:
	"9A" - variable length record inconsistency
	"9B" - free list inconsistency
	"9C" - 9A and 9B

Rules.

1. The ISAM file must have been opened for READ or READ/WRITE access.
2. To assure full ISAM consistency, please run the ISVER call afterwards.

Example:

CALL ISCON("SU",1,NUMDELREC,IST)

3.8 Regenerate index tables (ISRGN)

Purpose.

To regenerate the index part from the data part. This call may be used to reestablish file consistency if the ISVER call indicates inconsistency.

Syntax.

Call ISRGN (<file-id>,<status>)

Parameter description.

<file-id> single word, file identification
<status> single word, return parameter

Rules.

1. The ISAM file must have been opened for READ/WRITE access.
2. Status = "22" means that one or more data records were rejected due to duplicate key. However, the data part has been read, and ISAM file consistency is assured.

Example:

CALL ISRGN ("SU",IST)

.3.9 Create/delete index table (ISIND)

Purpose.

To modify the ISAM file structure at runtime by creating or deleting keys

Syntax.

Call ISIND (<file-id>,<key-id>,<code>,<status>)

Parameter description.

<file-id> single word, file identification
<key-id> single word, key identification
<code> single word:

- 1 - create index table
- 2 - delete index table

Rules.

1. The ISAM file must have been opened for READ/WRITE access.
2. This call is not allowed in multi-user mode.

Example:

CALL ISIND ("SU",7,1,IST)

.4 ISAM RECORD OPERATIONS

.4.1 Read next record (ISRNX,ISRV)

Purpose.

To read a file sequentially (ascending or descending key order)

Syntax.

Call ISRNX (<file-id>,<record>,<status>) (FIXED)

or

Call ISRV (<file-id>,<record>,<length>,<status>)
(VARIABLE)

Parameter description.

<file-id> single word, file identification
<record> array of words must start on a word boundary
<length> single word, return parameter
<status> single word, return parameter

Rules.

1. This call implies that the file has been opened for READ or READ/WRITE access.
2. READ NEXT is valid only after an OPEN, START, REWRITE, DELETE, READ USING KEY or another READ NEXT call.
3. After an OPEN statement, READ NEXT will transfer the record with the lowest key value of the prime key. After a START call, READ NEXT will transfer the record with the key corresponding to the condition specified in the START call; the key may be the prime key or an alternate key.
4. <length> denotes record length in bytes.
5. ISRNX/ISRV reads data records in ascending key order by default. The order in which data records are being read may be switched between ascending and descending sequence by calling ISDIR.

Example:

```
INTEGER RECORD(18)
.
.
.
CALL ISRNX("SU",RECORD,IST)
```

.4.2 Read record using key (ISRUK,ISRKV)

Purpose.

To find a record using a given key and transfer the corresponding record to the user.

Syntax.

Call ISRUK (<file-id>, <key-id>, <record>, <dummy>,
 <status>) (FIXED)

or

Call ISRKV (<file-id>,<key-id>,<record>,<dummy>,
 <length>,<status>) (VARIABLE)

Parameter description.

<file-id> single word, file identification
<key-id> single word, key identification
<record> array of words, where the record to be
 transferred contains also the key to be
 searched
<dummy> single word, unused
<length> single word, return parameter single word,
<status> return parameter

Rules.

1. The file must have been opened for READ or READ/WRITE access.
2. Before the call, the record area must contain the key to be searched.
3. <length> denotes record length in bytes.

Example:

```
INTEGER RECORD(18)
.
.
.
CALL ISRUK("SU","CR",RECORD,24,IST)
```


4.3 Write record (ISWRT,ISWRV)

Purpose.

To transfer a record from the record area to the file,
and update the index tables

Syntax.

Call ISWRT (<file-id> , <record> , <status>) (FIXED)

or

Call ISWRV (<file-id>,<record>,<length>,<status>)
(VARIABLE)

Parameter description.

<file-id> single word, file identification
<record> array of words containing the record-data
<length> single word, input parameter
<status> single word, return parameter

Rules.

1. The file identified must have been opened for WRITE or READ/WRITE access.
2. Binary zeroes are accepted as the value for any key.
3. If duplicates are not allowed for some keys and the value of one of these keys already exists in the indexes, the "INVALID KEY" condition arises.
4. There is no constraint on the sequence of the keys - for example, the key-values do not have to be in ascending order for two consecutive WRITE statements.
5. <length> denotes record length in bytes.

Remark.

ISAM reuses the space previously occupied by deleted records.

Example:

INTEGER RECORD(18)

.
. .
. .
. .

CALL ISWRT ("SU",RECORD,IST)

4.4 Rewrite record (ISREW,ISRWW)

Purpose.

To modify the contents of the previously read record.

Syntax.

Call ISREW (<file-id> , <record> , <status>) (FIXED)

or

Call ISRWW (<file-id>,<record>,<length>,<status>)
(VARIABLE)

Parameter description.

<file-id> single word, file identification
<record> array of words containing the record data
<length> single word, input parameter
<status> single word, return parameter

Rules.

1. The file identified must have been opened for WRITE or READ/WRITE access.
2. The previously READ record must have been read into the user record area with a READ NEXT or READ USING KEY statement.
3. Key values may be changed during a REWRITE. If status 22 (duplicate key) occurs, no modification will be made.
4. <length> denotes record length in bytes. Note that this length need not be the same as in the previous READ call.

Example:

INTEGER RECORD(18)

.
.
.
.

CALL ISREW("SU",RECORD,IST)

.4.5 Delete record (ISDEL)

Purpose.

To inform ISAM that the current record should be removed from the ISAM file.

Syntax.

Call ISDEL (<file-id> , <status>)

Parameter description.

<file-id> single word, file identification single word,
<status> return parameter

Rules.

1. The file identified must have been opened for READ/WRITE access.
2. DELETE RECORD call must be preceded by a READ NEXT, READ USING KEY, READ RELATIVE, RESTART or START call. The record deleted is the one that was found by the previous call.

Remarks.

All index entries which pointed to the record are also deleted.

The space which the record occupied is made available and may be reused later when a WRITE call is executed.

Example:

INTEGER RECORD(18)

.
.
.
.

CALL ISDEL("SU",IST)

.4.6 Delete record using key (ISDLK)

Purpose.

To find and delete a record having the specified prime key.

Syntax.

Call ISDLK (<file-id> , <record> , <dummy> , <status>)

Parameter description.

<file-id> single word, file identification
<record> array of words, contains the key in correct place
<dummy> single word, unused
<status> single word, return parameter

Rules.

1. The file identified must have been opened for READ/WRITE access.
2. The prime key must be used to locate the record.
3. The key value must exist, otherwise an "INVALID KEY" condition arises.

Remarks.

All the indexes that pointed to the record are also deleted. The space that the record occupies is made available for reuse.

Example:

INTEGER RECORD(18)

.
. .
. .
. .

CALL ISDLK("SU",RECORD,0,IST)

.4.7 Start using key (ISTRT)

Purpose.

To find a record by comparing its key to a given value

Syntax.

```
Call ISTRT (<file-id>,<function>,<key-id>,<record>,  
           <dummy>,<status>)
```

Parameter description.

<file-id> single word, file identification
<function> single word, may take one of the following values:

1. Find record with key equal to value
2. Find record with key greater than or equal to value
3. Find record with key greater than value
4. Find record with key less than value
5. Find record with key less than or equal to value

<key-id> single word, key identification
<record> array of words, contains the key at its relative place
<dummy> single word, unused
<status> single word, return parameter

Rules.

1. The file identified must have been opened with READ or READ/WRITE access.
2. Even though no record is returned by this call, the record is made available and can be read by the READ NEXT call.
3. If duplicates of the specified key are allowed, the record with the lowest address is made available for the next READ NEXT or DELETE call.

Example:

```
INTEGER RECORD (18)  
CODE = 2  
CALL ISTRT("SU",CODE,"CR",RECORD,24,IST)
```

.4.8 Remember current record (ISREM)

Purpose.

To store the internal record number of the record currently being accessed, so that a series of sequential record accesses can be interrupted.

Note that in addition to the internal record number, all index information used to access data records is stored as well.

Syntax.

Call ISREM (<file-id>,<status>)

Parameter description.

<file-id> single word, file introduction
<status> single word, return parameter

Rules.

1. The file must have been opened with READ or READ/WRITE access.
2. This call may be used to interrupt a series of sequential data record read operations, then continue after some random record access on the same ISAM file.

Example:

CALL ISREM ("SU", IST)

.4.9 Restart current record (ISRES)

Purpose.

To resume record access at a point marked by the ISREM call.

Syntax.

Call ISRES (<file-id>,<status>)

Parameter Description.

<file-id> single word, file introduction
<status> single word, return parameter

Rules.

1. The file must have been opened with READ or READ/WRITE access.
2. The ISRES call will resume record accessing, corresponding to the most recent ISREM call. Note that at least one ISREM must have been made; otherwise results will be unpredictable.

Example:

CALL ISRES ("SU", IST)

.4.10 Lock next record (ISLCK)

Purpose.

To lock the next record to be made available by a read or write call.

Syntax.

Call ISLCK

Parameter description.

No parameters.

Rules.

1. This call is for ISAM in multiuser mode only. If it is called in another mode, it will be treated as a dummy call.
2. A locked record can be read from another program, but not modified, deleted or locked.
3. The ISLCK call should be placed right in front of the read or write call.
4. The lock-next-record mode will automatically be reset, regardless of the status after the read or write call.
5. For further information about ISAM run in multiuser mode, see chapter 5.

.4.11 Unlock locked records (ISUNL)

Purpose.

To unlock all records that have been locked on the file.

Syntax.

Call ISUNL (<file-id>,<status>)

Parameter description.

<file-id> single word, file identification
<status> single word, return parameter

Rule.

The call is effective for records locked by both manual-unlock and automatic modes.

Remarks.

We strongly advise you to use the automatic-unlock mode, to avoid the risk of deadlocks.

Example:

CALL ISUNL ("SU", ST)

.4.12 Read relative record (ISREL)

Purpose.

To read a data record directly, using its internal record number (relative to 1), rather than retrieving the record via indexes.

Syntax.

Call ISREL (<file-id>,<record-no>,<record>,<length>,
<status>)

Parameter description.

<file-id> single word, file identification
<record-no> double word, internal record number
<record> array of words, to receive the data record
<length> single word, return parameter containing
length in bytes
<status> single word, return parameter

Rules.

1. The file must have been opened with READ or READ/WRITE access.
2. If variable length records are being used, and a variable length record begins in the internal record requested, ISAM will retrieve the following internal records containing the variable length record.

Make sure that <record> is be large enough for the largest possible variable length record.
3. If the internal record requested does not contain the first part of a variable length record, error status "25" will be returned.

Remarks.

The ISREL call executes much faster than ISRNK or ISRUK to retrieve data records.

Example:

```
INTEGER*4 RECNO  
INTEGER RECORD (18),RECLN  
  
RECNO=123  
CALL ISREL ("SU",RECNO,RECORD,RECLN,IST)
```

.4.13 Get current record number (ISCUR)

Purpose.

To get the internal record number, relative to 1, of the current data record.

Syntax.

Call ISCUR (<file-id>,<record-no>,<status>)

Parameter description.

<file-id> single word, file identification
<record-no> double word, internal record number
<status> single word, return parameter

Rule.

1. The file must be open.

Remarks.

This call is likely to be useful in conjunction with ISREM and ISRES calls, which may be interrupting a series of ISREL calls.

Example:

```
INTEGER*4 RECNO  
CALL ISCUR ("SU",RECNO,IST)
```

.4.14 Set read direction (ISDIR)

Purpose.

To set the direction in which data records are read by the ISRNK call, ie. either read record with next ascending key value or read record with next descending key value.

Syntax.

Call ISDIR (<file-id>,<direction>,<status>)

Parameter description.

<file-id> single word, file identification
<direction> single word (1 = ascending, 2 = descending)
<status> single word, return parameter

Rules.

1. The file must have been opened for READ or READ/WRITE access.
2. The initial direction, set by ISOPF, of the ISRNK call, is ascending.
3. The direction remains in effect until either an ISDIR call is used to change it, or the ISAM file is closed.

.4.15 Flush out modifications (ISFLU)

Purpose.

To write modified parts of an ISAM file from the buffer onto the disk file(s). This is meaningful only in buffered I/O mode, and is used to increase the security and still have good performance.

Syntax.

Call ISFLU (<file-id>,<status>)

Parameter description.

<file-id> single word, file identification
<status> single word, return parameter

Rules.

1. The ISAM file must have been opened for WRITE or READ/WRITE access

Example:

CALL ISFLU ("SU",IST)

.5 ACCESSING ISAM FILES IN MULTIUSER MODE

When running ISAM in single-user mode, an ISAM file is opened from a program running on one terminal only.

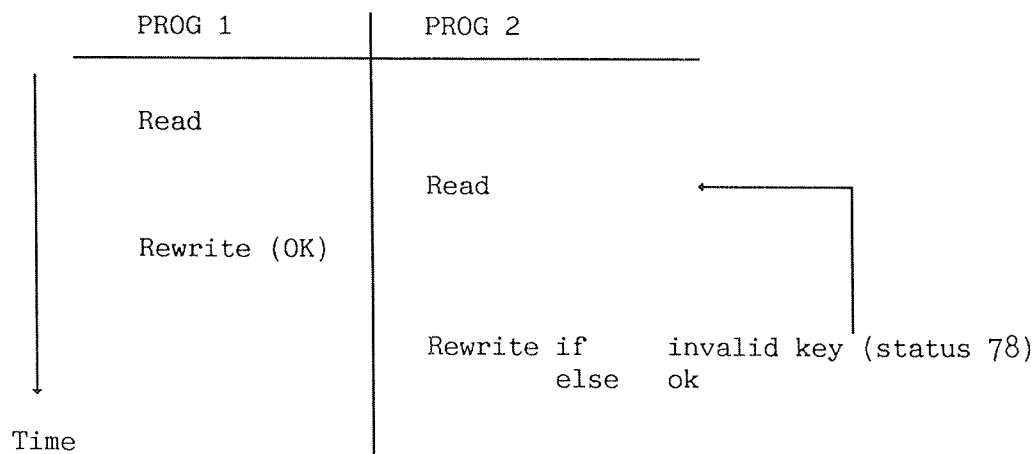
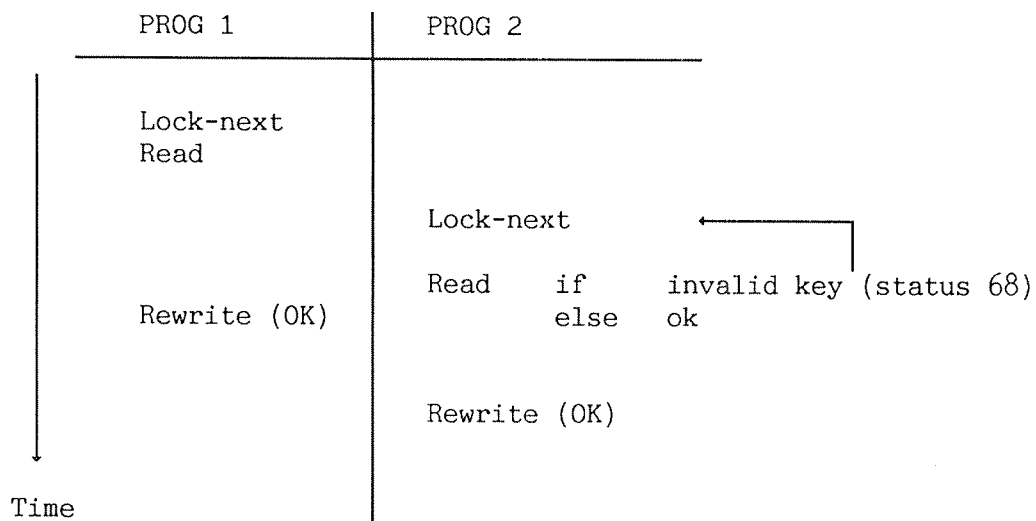
Multiuser mode allows one program to be running concurrently on several terminals, each accessing the same ISAM file. Further different programs may be running concurrently from several terminals, accessing the same ISAM file. In both cases the programs may even access the same record in the ISAM file without conflict.

As multiuser mode uses only immediate output, there is no need for logging and subsequent reprocessing, since the ISAM file will be consistent at any point in time.

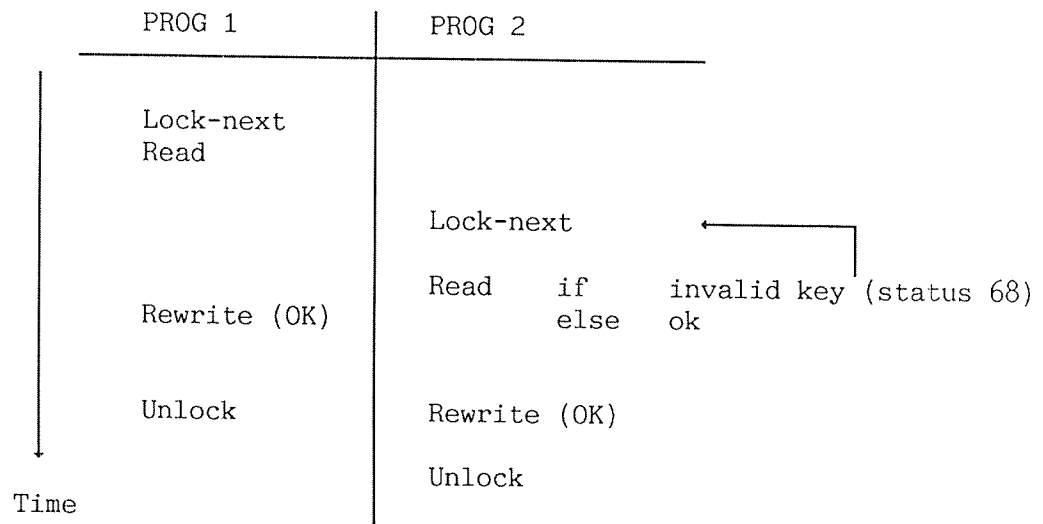
The ISAM file structure is identical whether single user or multiuser mode has been used on it.

.5.1 Concurrency control

In multiuser mode, several programs may wish to change the contents of the same record concurrently. This problem must be handled in some way to prevent any changes to data records being lost. The following example illustrates how this may be achieved.

Without locksWith automatic-unlock

With manual-unlock



Invalid key means that the two programs access the same record. A lock-next call will lock, for update, the record which is made current by the next read call. A locked record will normally be unlocked automatically when the record has been rewritten, when another record has been read or written on the same file, or when the file has been closed. If the record has been locked in manual-unlock mode, it will remain locked until explicitly unlocked by the program or until the file is closed.

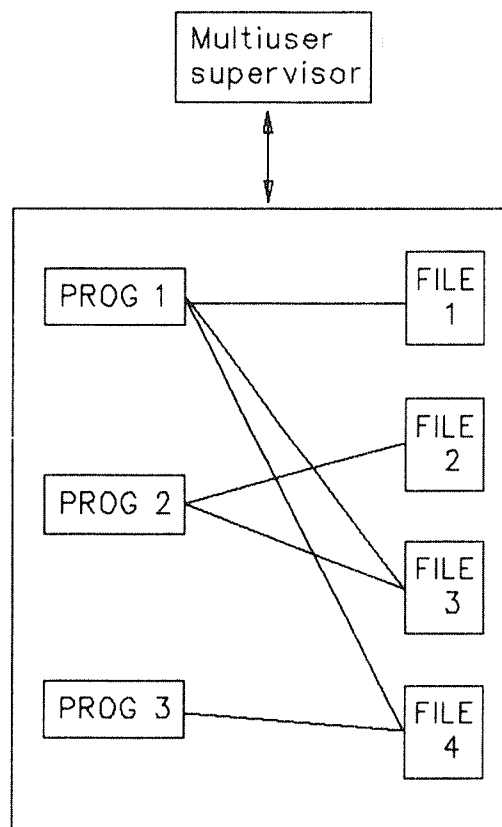
An application program must test status to determine whether it may change the contents of a record. Possibly the record has been locked by another program or has already been changed. If this is the case, then the application program must read the record again or try a new lock. The ISAM system does not do this internally, as deadlocks could easily arise.

5.2 Definition of an ISAM process

To handle the problem of several programs concurrently accessing the same ISAM file(s), or even the same record, an ISAM PROCESS is introduced.

An ISAM multiuser PROCESS consists of several programs and several files, supervised by an RT program (ISAMRT1) on the ND-100, and by a common link segment (IS-MULTI) on the ND-500. Within the process, the multiuser supervisor will keep track of which record is current (and eventually locked) for each program on each file belonging to the process.

An ISAM PROCESS may be illustrated as follows:



ISAM multiuser process

IMPORTANT NOTE: ND-100 and ND-500 programs must not run concurrently against the same ISAM files, or they may be corrupted.

.5.3 Multiuser file requirements

All ISAM files to be accessed in multiuser mode, both the index and data part, must be continuous SINTRAN files. The size of the index part (in SINTRAN pages) is found by using the ESTIMATE-INDEX-FILE-SIZE function in the ISAM SERVICE program. The size (in SINTRAN pages) of the data part is

$(\text{maximum number of records} * \text{record length} / 2048D) + 1$

ND-100: Load and start the ISAM multiuser supervisor ISAMRT before running any programs which access ISAM files in multiuser mode. This should be set up to occur automatically at MASTER CLEAR/LOAD time.

ND-500: Link the application programs to the common data segment IS-MULTI.

.5.3.1 Conversion of indexed files to continuous files

If existing files have been created prior to use of ISAM Multiuser facilities, then to access such files in multiuser mode, convert the files to SINTRAN continuous files by using the following procedure:

@CREATE-FILE	TEMPORARY:ISAM	<number of disk pages>
@CREATE-FILE	TEMPORARY:DATA	<number of disk pages>
@COPY-FILE	TEMPORARY:ISAM	ISAM-FILE:ISAM
@COPY-FILE	TEMPORARY:DATA	ISAM-FILE:DATA
@DELETE-FILE	ISAM-FILE:ISAM	
@DELETE-FILE	ISAM-FILE:DATA	
@RENAME-FILE	TEMPORARY:ISAM	ISAM-FILE:ISAM
@RENAME-FILE	TEMPORARY:DATA	ISAM-FILE:DATA

5.3.2 Changing logical unit numbers (ISLUN) used by the multiuser supervisor

ND-100:

In the multiuser mode, the internal devices 200 (octal) and 201 (octal) are used for communication between the multiuser supervisor and ISAM library routines.

If these internal devices have already been reserved, change the value of the variable ISLUN after loading the ISAM library. The multiuser supervisor must be changed as well; see the contents at the start of the file ISAMRT:MODE which is part of the installation procedure. Its use is described in the PD Sheet accompanying the product.

Example:

```
@BRF-LINKER
.
.
.
Br1: LOAD ISAM-2BANK
Br1: LOOK-AT-DATA ISLUN
202
.
Br1:
```

ND-500:

The ND-500 will not use internal devices at all. However, the semaphore 300 (octal) is used to synchronize the ISAM multiuser operations. If for some reason you have to use another semaphore, the following procedure can be used:

```
@LINKAGE-LOADER
.
.
.
N11: LOAD-SEGMENT ISAM-LIB
N11: DEFINE-ENTRY KK,301,D
N11: DATA-REFERENCE KK,ISLUN,D
N11: KILL-ENTRIES KK
.
.
.
```

5.4 Example of use of multiuser mode

```
*****
* The ISAM file SUPER-SYSTEM is opened in multiuser mode. Then the
* record having GARY BROOKER as name is read with lock. The CAR-NO
* field of that record is modified, and the record is written back.
* The program will run on the ND-100 and on the ND-500.
*****
```

```
PROGRAM ISAMEX
  INTEGER*2 FILNAM(10),RECORD(18),FILID,KEYID,ACCMODE,STATUS
  CHARACTER NAME*20,TELE*4,CAR*12,CFILNAM*20
  EQUIVALENCE (RECORD(01),NAME),(RECORD(11),TELE),
  -            (RECORD(13),CAR),(FILNAM(01),CFILNAM)

  * Declare the ISAM file (see chapter 3)
    CALL DECLARE

  * Open the ISAM file SUPER-SYSTEM (Note the space after the file name)
    CFILNAM = 'SUPER-SYSTEM:DATA '
    FILID   = "SU"
    ACCMODE = "UM"                                ;% Read-and-write access
    STATUS  = 2                                    ;% Multiuser mode, autom. unlock
    CALL ISOPF(ACCMODE,FILID,FILNAM,STATUS)
    IF (STATUS.NE."00") THEN
      WRITE(1,1) STATUS
1      FORMAT(X,'Error in ISAM OPEN. Status = ',A2)
      GO TO 999
    ENDIF

  * Read and lock the record having 'GARY BROOKER' as name
    NAME    = 'GARY BROOKER'
    KEYID   = "NA"
100    CALL ISLCK CALL ISRUK(FILID,KEYID,RECORD,0,STATUS)
    IF (STATUS.EQ."68") THEN                      ;% Record locked by another user
      CALL HOLD(1,2)                               ;% Wait 1 second, and try again
      GO TO 100
    ELSEIF (STATUS.NE."00") THEN
      WRITE(1,2) STATUS
2      FORMAT(X,'Error in ISAM OPERATION. Status = ',A2)
      GO TO 900
    ENDIF

  * Update the CAR-NO field of the record
    CAR = 'A-2' CALL ISREW(FILID,RECORD,STATUS) IF (STATUS.NE."00")
  THEN
    WRITE(1,2) STATUS
    GO TO 900
  ENDIF

  * Close the ISAM file and exit
900    CALL ISCLF(FILID,STATUS)
    IF (STATUS.NE."00") THEN
      WRITE(1,4) STATUS
4      FORMAT(X,'Error in ISAM CLOSE. Status = ',A2)
    ENDIF
999    END
```


.6 ISAM SERVICE PROGRAM AND ISAM INTERACTIVE

The ISAM Service Program is an easy-to-use interactive program which includes the following service functions:

1. Verify the consistency of an ISAM file, i.e., check that each element in the index part points to a data record, and that each data record has an element in the index part pointing to it.
2. Unload/load an existing ISAM file, eg. to rebuild an ISAM file in which VERIFY has found errors.
3. Build the index part from a flat SINTRAN file.
4. Estimate the size of the index part of an ISAM file. If the ISAM file does not exist, create it using ISAM-INTER.
5. Reset the error flag set if an ISAM file is not closed in a controlled way, i.e., through ISCLF.
6. Change current-record information contained in the ISAM multiuser supervisor, eg. in case of a terminal hang where the terminal has locked a record.
7. Modify the ISAM file structure:
 - create new keys
 - delete existing keys
 - define new sub-keys
 - change the record length

In order to enter this program, type

ISAM-SERVICE

The ISAM Interactive is an easy-to-use program to assist users in manipulating the data of ISAM files. This means that data may be stored/retrieved interactively without writing application programs.

In order to use this program, type

ISAM-INTER

Both these utility programs are intended to be self-explanatory. The command EXPLAIN-COMMAND will give a detailed explanation of each command.


```

.7 FILE STATUS REPORTING

```

Every ISAM call returns a status, a word containing two bytes.

Exceptions: The ISINI call will return a single integer value, the ISLCK call has no return status (see section 3.1 for details).

The following table is a summary of the possible status return values from each ISAM call.

Code	Meaning	I S O C R R W R D D T U R R E V F R C D C R I F P L N U R E E L R N E E R E L E U I O G N I F P X K T W L K T L M S R R G L R R N N D U
00	OK	x x
10	End of file	x x
21	Wrong sequence of calls	x x
22	Duplicates not allowed	x x
23	Record not found	x x
24	No more space on file	x x
25	Non-first part of var. record	x x
68	Record locked (see note)	x x
78	Record modified	x x
90	Multiususer Supervisor not ready	x x
92	Illegal ISAM run mode	x x
94	Error flag set	x x
95	File not initialized	x x
96	Key not initialized	x x
97	File access violation	x x
98	(See note)	x x
99	S-III file system error	x x
9A	Variable record inconsistency	x x
9B	Free list inconsistency	x x
9C	9A and 9B	x x

- Note:
- 1) File does not correspond to given description
 - 2) File already closed
 - 3) Illegal function code
 - 4) Read with lock
 - 5) File not consistent

Status from the ISINI call:

- 0 Normal exit
- 1 Invalid function code
- 2 File-id not unique or illegal record length
- 3 New key rejected
- 4 Too many files
- 6 Insufficient buffer space

.8 EXCEPTION HANDLING

Error messages from SINTRAN's file system are treated in a consistent way by ISAM.

ISAM will not terminate itself, but will return a status 99 (file system error) to the application program. The ISERR call may be used to retrieve the actual file system error code.

The only exception to this rule is the file system error message 75B, "NO MORE PAGES AVAILABLE FOR THIS USER", which may occur when using indexed SINTRAN files. In this case ISAM - in addition to the error message - will print the text "GIVE MORE PAGES, and Type CR TO CONTINUE" on the user's terminal and wait for input, thus giving the system supervisor time to give more space to the actual user. (This action is performed by an internal routine called IS75B in ISAM. If some other kind of action is wanted, the user is allowed to write his own routine IS75B. If so, this routine must be loaded before loading the ISAM Library).

If ISAM detects any fatal error, the internal buffer will be written to a file called ISAM-DUMP:DUMP. Note that this file will be automatically created if it does not already exist. For RT programs especially, this file should be created under user SYSTEM. A copy of this file should be sent to ND with an error report containing the name of the routine causing the dump.

.9 CONSISTENCY OF THE ISAM FILES

ISAM uses a paging/blocking technique to minimize I/O activity. In other words, when ISAM is run with buffered input/output, a record is not immediately transferred to the disk after a successful WRITE call. This has a clear advantage in that programs run much faster. However, if a program terminates before successfully closing its files, the contents of the data files and index files are unpredictable. Therefore, before using ISAM, backup procedures should exist.

The function UNLOAD/LOAD, in the program ISAM-SERVICE, may be used to reestablish consistency of an inconsistent ISAM file. The function VERIFY, available in the same program, will check for inconsistency.

It is strongly advisable to use SINTRAN's disable/enable-escape-function to protect ISAM files from being destroyed accidentally.

When running ISAM in buffered I/O mode, the escape function should be disabled if an ISAM file is opened for update. Otherwise, it is sufficient to disable the escape function when an updating ISAM call is executed.

.10 LOADING AN ISAM PROGRAM

Example of Loading on the ND-100:

```
@BRF-LINKER
*PROG-FILE MYPROGRAM:PROG
*LOAD      MYPROGRAM:BRF
*LOAD      ISAM-2BANK:BRF
*LOAD      FORTRAN-2BANK:BRF
*EXIT
```

Example of Loading on the ND-500:

```
@LINKAGE-LOADER
NLL:SET-DOMAIN <domain name>
NLL:LOAD-SEGMENT <program name>
NLL:LINK-SEGMENT IS-MULTI          %Needed if multiuser
NLL:LOAD-SEGMENT ISAM-LIB:NRF
NLL:LOAD-SEGMENT <language run-time library>
NLL:LOAD-SEGMENT EXCEPT-LIB:NRF
NLL:END-DOMAIN
NLL:EXIT
```


.11 PERFORMANCE CONSIDERATIONS

As stated earlier in Chapter 9, ISAM uses a paging/blocking technique. Block sizes are calculated at run-time and depend upon:

1. The size of the available buffer pool
2. The record length

As a rule of thumb, the bigger the buffer, the better the response-time (provided of course there is enough real memory available). The size of the buffer is initially 8 Kbytes (24 Kbytes in ISAM-2BANK). To expand the buffer you could write and load a MAC subprogram before load the ISAM library (ISAM-1BANK:BRF or ISAM-2BANK:BRF). The maximum size of the buffer is 62 Kbytes. The above comments apply to the ND-100 only.

Example (on the ND-100 only):

```
)9BEG
)9LIB ISIBL ISBBL ISBF1 ISBF2
)9ENT ISIBL ISBBL ISBF1 ISBF2
ISBF1,STZ*
ISIBL,STZ*
ISBBL, *+17777/      % OCTAL SIZE OF BUFFER (16K
BYTES)
ISBF2,STZ*
)FILL
)9END
)9EOF
)LINE
```

To compile the above program:

```
@MAC
)9ASSM ISAM-BUFFER, ISAM-BUFFER
)9EXIT
```

Modified loading procedure:

1-BANK:

```
@NRL
.
.
.
.
.
*X-LOAD ISAM-BUFFER
*LOAD ISAM-1BANK .
.
```

2-BANK:

```
@NRL
.
.
.
.
.
*SET-MODE DATA
*X-LOAD ISAM-BUFFER
*SET-MODE PROG
*LOAD ISAM-2BANK
.
.
```

Note: The warning message "MIXED ONE/TWO BANK ROUTINES", after the X-LOAD command, can be ignored.

Further hints:

1. By using ISAM on continuous files, access time can be reduced, but files of this type must have a fixed size that cannot be expanded by ISAM at runtime.
2. Use even-number record lengths.
3. Use even-number key lengths.
4. The size of the internal buffer does not influence performance of ISAM on the ND-500.
5. Run-time performance may be affected by a combination of the user program's choice of record length and the internal grouping of records into blocks for input/output operations.

Firstly, SINTRAN I/O, as used from ISAM, always uses blocks of 2 Kbytes. ISAM attempts to group records into blocks for I/O, as efficiently as possible, i.e. fill a 2K block with as many records as possible.

If fixed length records are being used, it is best to choose a record size which can be packed exactly into a 2K block, rather than a record size which leaves some unused space, or worse still, is just over 2K bytes, so that it requires another 2K byte block.

The best performance will be obtained with record sizes which are powers of 2, eg. 32, 64, 128, ... Worse performance would result from a record length of 65 bytes.

The same principle applies to variable length records. However, each variable length record has a 6 byte overhead which must be added to the record length when calculating record packing into blocks for I/O.

12 ISAM EXAMPLE PROGRAMS

12.1 A FORTRAN program using ISAM

```

*****
* This program is based on the ISAM file SUPER-SYSTEM
* described in chapter 1 and chapter 3.
* The purpose is to display information about GARY BROOKER.
* The program will run on the ND-100 and the ND-500.
*****

      PROGRAM ISAMEX
      INTEGER*2 FILNAM(10),RECORD(18),FILID,KEYID,ACCMODE,STATUS
      CHARACTER NAME*20,TELE*4,CAR*12,CFILNAM*20
      EQUIVALENCE (RECORD(01),NAME),
      -           (RECORD(11),TELE),
      -           (RECORD(13),CAR),
      -           (FILNAM(01),CFILNAM)

* Declare the ISAM file (see chapter 3)

      CALL DECLARE

* Open the ISAM file SUPER-SYSTEM (note the space after the file name)

      CFILNAM = 'SUPER-SYSTEM:DATA '
      FILID   = "SU"
      ACCMODE = "RM"      ;% Read access only
      STATUS  = 0         ;% Single-user mode, buffered I/O
      CALL ISOPF(ACCMODE,FILID,FILNAM,STATUS)
      IF (STATUS.NE."00") THEN
1        WRITE(1,1) STATUS
          FORMAT(X,'Error in ISAM OPEN. Status = ',A2)
          GOTO 999
      ENDIF

* Read and display the record having 'GARY BROOKER' as name

      NAME    = 'GARY BROOKER'
      KEYID   = "NA"
      CALL ISRUK(FILID,KEYID,RECORD,0,STATUS)
      IF (STATUS.NE."00") THEN
2        WRITE(1,2) STATUS
          FORMAT(X,'Error in ISAM OPERATION. Status = ',A2)
          GOTO 900
      ENDIF
      WRITE(1,3) NAME,TELE,CAR
3      FORMAT(' Name: ',A20,/, ' Tele: ',A4,/, ' Car: ',A11,/)

* Close the ISAM file and exit

900    CALL ISCLF(FILID,STATUS)
      IF (STATUS.NE."00") THEN
4      WRITE(1,4) STATUS
        FORMAT(X,'Error in ISAM CLOSE. Status = ',A2)
      ENDIF
999    END

```

```

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@
@@ Procedure for compiling, loading and executing ISAMEX @@
@@ (ND-100) @@
@@ @@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

```

```

@FORTRAN-100
SEPARATE ON
COMPILE ISAMEX,,ISAMEX
COMPILE DECLARE,,DECLARE
EXIT

```

```

@BRF-LINKER
*PROG-FILE ISAMEX
*LOAD ISAMEX
*LOAD DECLARE
*LOAD ISAM-2BANK
*LOAD FORTRAN-2BANK
*EXIT

```

```

@ISAMEX

```

```

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@
@@ Procedure for compiling, loading and executing ISAMEX @@
@@ (ND-500) @@
@@ @@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

```

```

@FORTRAN-500
COMPILE ISAMEX,,ISAMEX
COMPILE DECLARE,,DECLARE
EXIT

```

```

@LINKAGE-LOADER
NLL:SET-DOMAIN ISAMEX
NLL:LOAD-SEGMENT ISAMEX
NLL:LOAD-SEGMENT DECLARE
NLL:LOAD-SEGMENT ISAM-LIB
NLL:LOAD-SEGMENT FORTRAN-LIB
NLL:LOAD-SEGMENT EXCEPT-LIB:NRF
NLL:END-DOMAIN
NLL:EXIT

```

```

@ND ISAMEX

```

12.2. A Pascal program using ISAM

The first part of this example is equivalent to the declarations for the FORTRAN example in section 3.1.

```
( * ***** *)
(* Declaration of the ISAM file SUPER-SYSTEM (ND-100) *)
(* ***** *)

PROCEDURE DECLARE;
LABEL 900,999;
TYPE
  INFOTYPE = (ARRAY [1..5] OF INTEGER;
VAR
  INFO : INFOTYPE;
  IST : INTEGER;
PROCEDURE ISINI(V1 : INTEGER;W2: INFOTYPE;
                VAR V3: INTEGER); STANDARD;
BEGIN

(* Start definitions *)

  IST := 0;
  ISINI(1,INFO,IST);
  IF NOT (IST = 0) THEN GOTO 900;

(* Declare data file *)

  INFO[1] := ORD('S')* 256+ORD('U'); (* File identification *)
  INFO[2] := 36; (* Record length in bytes *)
  INFO[3] := 0; (* Fixed record length *)
  INFO[5] := 0; (* Data/indexes on separate files *)
  ISINI(2,INFO,IST);
  IF NOT (IST = 0) THEN GOTO 900; (* Error *)

(* Declare 3 keys (Name, Telephone and CAR-number) *)

  INFO[2] := ORD('N')* 256+ORD('A'); (* File identification *)
  INFO[3] := 0; (* Key byte position (rel 0) *)
  INFO[4] := 20; (* Key byte length *)
  INFO[5] := 1; (* Duplicates not allowed *)
  ISINI(3,INFO,IST);
  IF NOT (IST = 0) THEN GOTO 900; (* Error *)

  INFO[2] := ORD('T')* 256+ORD('E'); (* File identification *)
  INFO[3] := 20; (* Key byte position (rel 0) *)
  INFO[4] := 4; (* Key byte length *)
  INFO[5] := 1; (* Duplicates not allowed *)
  ISINI(3,INFO,IST);
  IF NOT (IST = 0) THEN GOTO 900; (* Error *)
```

```

INFO[2] := ORD('C')* 256+ORD('A'); (* File identification *)
INFO[3] := 24;          (* Key byte position (rel 0)      *)
INFO[4] := 12;          (* Key byte length      *)
INFO[5] := 0;          (* Duplicates allowed  *)
ISINI(3,INFO,IST);
IF NOT (IST = 0) THEN GOTO 900;          (* Error *)

(* End of declarations, return to caller *)

ISINI(4,INFO,IST);
IF NOT (IST = 0) THEN GOTO 900;
GOTO 999;

(* Error in declaration, stop execution *)

900:  WRITELN('*** Error in ISAM DECLARATION *** Status =',IST);
999:  END;
$EOF

(* ***** *)
(* This program example is based on the ISAM file SUPER-SYSTEM *)
(* described in chapters 1 and 3. *)
(* The purpose is to display information about GARY BROOKER. *)
(* The program will run on the ND-100. *)
(* ***** *)

PROGRAM ISAMEX(OUTPUT);
LABEL 900, 999;
TYPE
  RECTYPE = PACKED ARRAY [1..36] OF CHAR;
  NAMTYPE = PACKED ARRAY [1..20] OF CHAR;
VAR
  FILNAM : NAMTYPE;
  XRECORD : RECTYPE;
  FILID,KEYID,ACCMODE,STATUS,OKSTATUS : INTEGER;
  I : INTEGER;
  NAME : NAMTYPE;
  TELE : PACKED ARRAY [1..4] OF CHAR;
  CAR : PACKED ARRAY [1..12] OF CHAR;

$INCLUDE DECLARE
  PROCEDURE ISOPF(V1,V2 : INTEGER; V3 : NAMTYPE;
                  VAR V4 : INTEGER); STANDARD;
  PROCEDURE ISRUK(V1,V2 : INTEGER; V3 : RECTYPE;
                  V4 : INTEGER; VAR V5 : INTEGER); STANDARD;
  PROCEDURE ISCLF(V1 : INTEGER; VAR V2 : INTEGER); STANDARD;

BEGIN

```



```
(* Declare the ISAM file *)

DECLARE;
OKSTATUS := ORD('0')*256+ORD('0');

(* Open the ISAM file SUPER-SYSTEM *)
(* (Note the space after the file name) *)

FILNAM := 'SUPER-SYSTEM:DATA ';
FILID := ORD('S')*256+ORD('U');
ACCMODE := ORD('R')*256+ORD('M'); (* Read only *)
STATUS := 0; (* Single-user mode, buffered I/O *)
ISOPF(ACCMODE,FILID,FILNAM,STATUS);
IF NOT (STATUS = OKSTATUS) THEN
BEGIN
WRITELN('Error in ISAM OPEN. Status = ',STATUS);
GOTO 999;
END;

(* Read and display the record having 'GARY BROOKER as name *)

XRECORD := 'GARY BROOKER';
KEYID := ORD('N')*256+ORD('A');
ISRUK(FILID,KEYID,XRECORD,OSTATUS);
IF NOT (STATUS = OKSTATUS) THEN BEGIN
WRITELN('Error in ISAM OPERATION. Status = ',STATUS);
GOTO 900
END;

FOR I := 1 TO 20 DO NAME[I] := XRECORD[I];
FOR I := 21 TO 24 DO TELE[I-20] := XRECORD[I];
FOR I := 25 TO 36 DO CAR[I-24] := XRECORD[I];
WRITELN(' Name: ',NAME:20);
WRITELN(' Tele: ',TELE:4);
WRITELN(' Car : ',CAR:11);

(* Close the ISAM file and exit *)

900: ISCLF(FILID,STATUS);
IF NOT (STATUS = OKSTATUS) THEN BEGIN
WRITELN('Error in ISAM CLOSE. Status = ',STATUS);
999: END.
```

I N D E X L I S T

<u>Index term</u>	<u>Reference</u>
Buffer pool	11
Calls, summary	7
Check data part consistency	22
Close file	18
Concurrency	42
Consistency	
data part	22
file, verify	19
Continuous files	45
Create index table	24
Data part consistency, check/restore	22
Delete	
index table	24
record	29
record using key	30
Direction, read	38
Error code, fetch	21
Error flag, reset	20
Error messages	53
Example	
FORTRAN	13, 47, 63
loading	57
multiuser mode	47
Pascal	65
Features	3
Fetch file system error code	21
File	
consistency, verify	19
declaration, FORTRAN example	13
operations	11
requirements, multiuser	45
status	51
File system error code, fetch	21
Flush out modifications	39
FORTRAN example	13, 47, 63
Get current record number	37
Group key	14
Index table	
create/delete	24
regenerate	23

Index term	Reference
Indexed files	45
Initialize ISAM buffer	11
Internal devices	46
IRNV	25
IRNX	25
ISAM	
calls, summary	7
file operations	11
Interactive	49
multiuser process	44
Service program	49
ISCLF	18
ISCON	22
ISCUR	37
ISDEL	29
ISDIR	38
ISDLK	30
ISERR	21
ISFLG	20
ISFLU	39
ISIND	24
ISINI	11
ISLCK	34
ISOPF	16
ISREL	36
ISREM	32
ISRES	33
ISREW	28
ISRGN	23
ISRKV	26
ISRUK	26
ISRWW	28
ISTRT	31
ISUNL	35
ISVER	19
Key, value of	27
Limitations	3
Loading, example	57
Lock next record	34
Locking of records	42
Logical unit numbers	46
Maximum values	3
Messages	53
Multiuser	
file requirements	45
mode	42
mode, example	47
process	44

<u>Index term</u>	<u>Reference</u>
Next record, read	25
Open file	16
Parameters, on the ND-500	9
Parking	4
Pascal example	65
Performance	59
Prime key	12
Read	
direction, set	38
next record	25
record using key	26
relative record	36
Record	
delete	29
delete using key	30
lock	34
number, get current	37
read next	25
read relative	36
read using key	26
remember	32
restart	33
rewrite	28
unlock	35
write	27
Regenerate index tables	23
Relative record, read	36
Remember current record	32
Reset error flag	20
Restart current record	33
Restore data part consistency	22
Rewrite record	28
Set read direction	38
Start using key	31
Status codes	51
Subkey	14
Unlock locked records	35
Unlocking of records	42
Verify file consistency	19

<u>Index term</u>	<u>Reference</u>
Write record	27

SEND US YOUR COMMENTS!!!



Are you frustrated because of unclear information in this manual? Do you have trouble finding things? Why don't you join the Reader's Club and send us a note? You will receive a membership card — and an answer to your comments.

Please let us know if you

- * find errors
- * cannot understand information
- * cannot find information
- * find needless information

Do you think we could improve the manual by rearranging the contents? You could also tell us if you like the manual!



HELP YOURSELF BY HELPING US!!

Manual name: ISAM Reference Manual

Manual number: ND-60.108.6 EN

What problems do you have? (use extra pages if needed)

Do you have suggestions for improving this manual ?

Your name: _____ Date: _____

Company: _____ Position: _____

Address: _____

What are you using this manual for ? _____

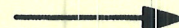
NOTE!

This form is primarily for documentation errors. Software and system errors should be reported on Customer System Reports.

Send to:

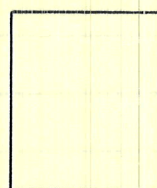
Norsk Data A.S
Documentation Department
P.O. Box 25, Bogerud
0621 Oslo 6, Norway

Norsk Data's answer will be found on reverse side



Answer from Norsk Data

Answered by _____ Date _____



Documentation Department
P.O. Box 25, Bogerud
0621 Oslo6, Norway

