159
1-21-76

Dr-1919

ORNL-TM-5199

MASTER

The FBSAM Data Transmission Package for IBM 360/370 Computers

W. A. Rhoades

OAK RIDGE NATIONAL LABORATORY

OPERATED BY UNION CARBIDE CORPORATION • FOR THE U.S. ATOMIC ENERGY COMMISSION

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

Printed in the United States of America. Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road, Springfield, Virginia 22161
Price: Printed Copy $5.50; Microfiche $2.25

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the Energy Research and Development Administration, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

ORNL-TM-5199
UC-32

Contract No. W-7405-eng-26

Neutron Physics Division

THE FBSAM DATA TRANSMISSION PACKAGE FOR IBM 360/370 COMPUTERS

W. A. Rhoades

NOTICE
This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Energy Research and Development Administration, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

JANUARY 1976

OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37830
operated by
UNION CARBIDE CORPORATION
for the
U.S. ENERGY RESEARCH AND DEVELOPMENT ADMINISTRATION

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

THIS PAGE

WAS INTENTIONALLY

LEFT BLANK

Table of Contents

		Page No.
	ABSTRACT	v
	ACKNOWLEDGEMENT	v
	PROGRAM ABSTRACT	vi
I.	INTRODUCTION	1
II.	DATE SET SPECIFICATION	1
	A. Data Set Record Structure	1
	B. DD Statements for FBSAM	2
	C. The "K" Control Array	2
	D. System Block Space	3
	E. Logical Record Length Restriction	3
III.	PROCESSING A DATA SET	4
	A. Opening	4
	B. Writing	5
	C. Rewinding and Disposing	6
	D. Reading	7
	E. Test and Return	8
	F. Spacing	8
	G. Note-Point Operations	9
	H. Pointing by Block Number	10
	I. Cautions	10
	J. Constant Arguments	11
	K. Common Block	11
IV.	ERROR CONDITIONS	11
	A. Error Procedure	11
	B. Status Errors	12
	C. Other Errors Detected	13
V.	APPLICATIONS AND AVAILABILITY	14
IV.	DEMONSTRATION PROGRAM	14
App. A.	CONTENTS OF "K" CONTROL ARRAY	17
App. B.	SUMMARY OF ENTRIES AVAILABLE	18
App. C.	APPLICATION OF FBSAM TO THE FACT CODE	20
App. D.	LISTING OF FBSAM AND TEST PROGRAM	28

THIS PAGE

WAS INTENTIONALLY

LEFT BLANK

Abstract

The FBSAM subroutine package provides rapid movement of large blocks of data directly between user storage arrays and magnetic disk data sets. No buffer space is required. Large reductions in data transmission time and number of I/O requests can be obtained. Moves can be concurrent with each other and with computation. A form of random access is available. A demostration program provides a thorough test of all features. The package is intended for use by FORTRAN programmers with little or no assembler language experience. It should be operable on most IBM 360/370 machines. A sample application to an I/O-bound code is discussed. The use of three concurrent functions entirely eliminated the real time associated with data transmission tasks by allowing them to proceed concurrently with other tasks.

Acknowledgements

Essential assistance in the use of the KBSAM package, around which FBSAM was developed, was provided by R. P. Rannie and J. G. Sullivan of ORNL. L. M. Petrie contributed important improvement in the performance of the package. M. L. Gritzner, ORNL, provided the FACT program and assisted in its conversion to FBSAM I/O. F. R. Mynatt requested and provided guidance for this effort.

Program Abstract

1) Identification - FBSAM (FORTRAN Basic Sequential Access Method).

2) Function - FBSAM provides data transfer to and from disk storage devices with very high efficiency. No system buffers are required. Several data transfers can be processed concurrently, and these can be concurrent with computations. A form of random access is provided.

3) Method - The FBSAM program contains a number of entries which check for errors and invoke various assembly-language packages. The status data for a given data set are compacted into a single array.

4) Related Material - The package includes the KBSAM package developed by C. M. N. Cree[1] and improved by J. G. Sullivan at ORNL.[2] Important performance improvements were made by L. M. Petrie.[3]

5) Restrictions - The data sets manipulated by FBSAM are not compatible with the IBM FORTRAN input-output method.

6) Computers - FBSAM is operable on IBM 360/370 computers.

7) Running Time - The CPU time used is entirely negligible. Large logical records are transferred at the full disk rotation speed, without a rotational delay between blocks.

8) Programming Languages - FORTRAN 50%; Assembly Language 50%.

9) Operating System - IBM OS.

10) Machine Requirements - FBSAM has been used on, but is probably not limited to IBM 2314 and 3330 disks, and plug-compatible CDC hardware.

11) Authors - W. A. Rhoades, Oak Ridge National Laboratory, Oak Ridge, Tennessee 37830.

12) References -

[1]C. M. N. Cree, "BSAM Subroutine Package for OS/360, Version 1," IBM Contributed Program Library 360D-03.4.026, IBM Program Information Department, Hawthorne, New York 10532 (January 22, 1969).

[2]J. G. Sullivan, "KBSAM - Buffering Input-Output to Direct Access Devices on IBM/360 Computers," unpublished manuscript (June 30, 1972).

[3]L. M. Petrie, Union Carbide Corporation, Oak Ridge, Tenn. 37830.

13) Material Available - FBSAM source program, demonstration program, descriptive memo.

This abstract was prepared according to ANS 10.3 Standard.

I. INTRODUCTION

The FBSAM package is a collection of FORTRAN-callable routines which make many of the features of the Basic Sequential Access Method (BSAM) conveniently available to FORTRAN programmers. Data are moved directly between user storage areas and magnetic disk data sets without the use of additional buffer storage. The data moves can be carried out concurrently with each other and with computations. A powerful note-point method of random write and retrieval is available. Very small real and CPU time requirements for I/O are obtained.

Data to be moved must be stored in contiguous locations, and no FORTRAN FORMAT facility is allowed. The data sets produced are not compatible with normal FORTRAN I/O. Best efficiency is obtained using long logical records. The speed is slightly better than that using track overflow with FORTRAN sequential operations. Because chaining is used, secondary allocations of disk space are not permissible.

The FBSAM package makes use of the KBSAM package provided by J. G. Sullivan,[1] which was, in turn, based on earlier work done at IBM-Houston by C. M. N. Cree.[2] Some additional features not utilized by FBSAM can be found in those references. Background material on the subject of BSAM operations can be found in refs. 3 and 4. This memo should suffice for ordinary applications, however.

II. DATA SET SPECIFICATION

A. Data Set Record Structure

The data sets written by FBSAM do not follow the FORTRAN concept of records, segments, and blocks. Each physical record is written as an individual block, followed by an interrecord gap (IRG). While many blocks may be written with a single call to the FBSAM package, and we will speak of such a collection of blocks as forming a "logical record," they need not be read back with any regard to the logical record structure used in

in writing them. After each read, the read pointer is always positioned at the next block. Thus, a large number of blocks could be written with a single call to the package, and then the blocks could be read back individually, or vice versa. In the following discussion, "record" always means "logical record" and refers to a collection of one or more data blocks written or read with a single call to the FBSAM package.

B. DD Statements for FBSAM

The DD statement defining a data set for FBSAM follows the usual rules as to UNIT, DSN, DISP, VOL, DDNAME, and SPACE specification. DCB, if supplied, is ignored. The space field must, in general, contain enough continuous space in the primary extent for the data set.

As an example:

```
//FT01F001 DD UNIT=SYSDA,DISP=(NEW,DELETE),SPACE=(TRK,100, ,CØNTIG,RØUND)
//FT02F001 DD UNIT=(SYSDA,SEP=FT01F001),(DISP=NEW,DELETE),SPACE=(TRK,
     100,,CØNTIG,RØUND)
```

C. The "K" Control Array

The I/O activity for each data set is "controlled" by an array supplied by the user in which all status information for that data set is kept. The length is (40 + 5*NBL) full words, where NBL is the maximum number of blocks per logical record which may be queued together (more description of this later). The first 17 words contain FORTRAN-oriented parameters describing such things as the block size being used, number of bytes to be processed, error flags, etc. The next 23 words are assigned to the Data Control Block (DCB), which contains a machine-oriented description of the data set and its status. The remaining spaces are assigned to NBL data Event Control Blocks (DECB's). A DECB is set up for each block to be written. It contains, among other things, a status flag to indicate when the specified block I/O has been completed, and error flags. A detailed specification for the "K" array is given in Appendix A.

Once a data set has been opened into a "K" array, that array controls all I/O activity on that data set until the data set is closed. The array is given as an argument to the FBSAM package for any operation relating to the data set it controls. As many "K" arrays can be established as are required. For example, one array for an input data set and one for an output data set would be used in copying. Once a data set has been opened into a "K" array, that storage space must not be used for any other purpose until the data set has been closed. Failure to observe this rule will give rise to quite mystifying system errors.

D. System Block Space

Although FBSAM operations do not require buffer space, certain system blocks are still required, just as in FORTRAN. The space requirement depends on the number of DECB's actually given, which may be less than NBL if the logical records are short. The maximum requirement per data set is then approximately:

SYSTEM BLOCK SPACE $\leq$ NBL*0.2 K bytes

Track overflow can increase this requirement, but is not normally used with FBSAM.

E. Logical Record Length Restriction

If concurrent operations, i.e., two concurrent data transmissions or a transmission while computing are not required, there is no practical limit to the logical record length. If concurrency is required, the limit is 99 x the maximum block which will fit on a track (or 32,760 bytes if track overflow is used). For example:

99 x 7,294 $\simeq$ 700 K bytes (2314 disk)

99 x 13,030 $\simeq$ 1,200 K bytes (3330 disk)

99 x 32,760 $\simeq$ 3,000 K bytes (track overflow)

III. PROCESSING A DATA SET

A. Opening

A data set must be opened by a call to FBSAM prior to I/O operations. For example:

```
DIMENSION K(540)
DATA LUN,LBL,NBL,NERC/1,0,00,0/
CALL FBSAM(LUN,K,LBL,NBL,NERC)
```

where

LUN = the logical unit number of the data set to be used

K = the "K" control array, dimensioned at least as large as 40 + 5*NBL

LBL = the maximum block length. If 0 is supplied, a full track is used as the block length

NBL = the maximum number of DECB's which may be given together. 0 < NBL < 100. NBL has an important effect on performance, discussed later.

NERC = NERC1 + NERC3

NERC1 = 0 Undefined Length Blocks
1 Undefined Length Blocks with Track Overflow
2 Fixed Length Blocks
3 Fixed Length Blocks with Track Overflow

NERC3 = 0 follow error procedure if a filemark is encountered
= 100 set flag, but do not follow error procedure

Track overflow is normally not used with FBSAM. It defeats chaining, and usually results in longer execution time and more I/O requests. If the FPNTR routine (described later) is to be used to read records at random, the fixed-length option must be chosen.

If writing is to be done at random, NERC1=2 must be chosen, and LBL must be 0 or a number large enough that only one block can be fit onto a track. Otherwise, updating the first block on a track destroys the other data on that track.

The effect of NERC1=1 is to force each block written to the maximum size, LBL. Then if LBL is chosen large enough that only one such record will fit on a track, each block will occupy a separate track. When reading, only the number of bytes requested are read. An obscure error could result in writing. If an attempt is made to write the last words in the user's memory allocation, and if more words were required to fill the block, then a protection error could result. This has never happened in our applications, but the user should avoid such a situation.

The fixed-length option can lead to inefficiency if logical records are much shorter than LBL, and FBSAM should not be used for such transfers.

B. Writing

There are two basic methods of writing with FBSAM. Using the first method, the FWRCH entry allows the writing and checking of all blocks of a logical record. A single call to this entry is sufficient to transfer data to the data set. Control is not returned to the calling program until the transfer is completed. A value of NBL larger than 1 improves efficiency by queueing DECB's together. All of the tracks on a cylinder can be written with a single I/O request, and without a rotational delay between tracks, if the job is not interrupted. Increasing NBL beyond the number of tracks on a cylinder apparently has no effect using FWRCH. As an example:

```
DIMENSION X (300 000)
DATA N/1 200 00/
CALL FWRCH(K,X,N)
```

where

K = control array specified in call to FBSAM.

X = data array to be transferred

N = number of bytes to be transferred.

Using the second method, the FWRITE entry sets up the channel commands for writing a logical record, and then returns control to the calling program. Other calculations or data transmissions can be performed while the array is being transferred. The user must issue a call to FCHECK before the transferred data array is modified, and before another I/O instruction relating that that data set is ussed, or before the data spaces referred to as X are used by the program:

```
CALL FWRITE (K,X,N)
     '
     '
     '
(Other instructions not relating to this data set)
     '
     '
     '
CALL FCHECK (K)
```

An error procedure is followed if FWRITE is called with a value of N which would require the writing of more than NBL blocks in a record. This restriction does not apply to FWRCH. In a multi-programming environment FWRITE will often execute faster than FWRCH.

C. Rewinding and Disposing

When a data set has been read or written, it can be either repositioned by a call to FREW or FPASS, or disposed of by a call to FDISP. If FREW is chosen, the pointer is repositioned at the beginning of the data set, and the data set remains open for reading or rewriting. A call to FPASS has the same effect as FREW, except that the pointer is positioned at the end of the data set. If a call to FDISP is made, the data set is closed and disposed of according to DISP specification in the DD statement.

```
CALL FREW(K)
CALL FDISP(K)
CALL FPASS(K)
```

where

K = control array specified in call to FBSAM.

A call to FREW, FDISP, or FPASS must not be made if any call to FWRITE or FREAD has not been completed by a call to FCHECK. A call to FDISP must be made before the K array is reused for any purpose other than I/O events relating to this specific data set. After a call to FDISP, a new call to FBSAM must be issued before the data set can be used again. A call to FREW or FPASS may, but does not need to, precede a call to FDISP.

D. Reading

Two reading entries, FRDCH and FREAD, perform input tasks analogous to the FWRCH and FWRITE entries, and they use the same argument lists:

```
CALL FRDCH (K,X,N)
CALL FREAD(K,X,N)
      '
      '
      '
(Other instructions not relating to this data set)
      '
      '
      '
CALL FCHECK(K)
```

After calling FREAD, the user must issue a call to FCHECK before the data array being transferred is used, and before another I/O instruction relating to that data set is issued.

E. Test and Return

After a call to FWRITE or FREAD, the user may wish to check for completion of the I/O operation, but to continue with some other task if the operation is not completed. The FTEST entry allows this:

```
      CALL FWRITE (K,X,N)
   10 CONTINUE
      '
      '
      (Tasks not relating to this data set or the X array)
      CALL FTEST (K,L)
      IF (L.EQ.0) GO TO 10
   20 CONTINUE
```

This programming causes the program to work on the statements following 10 until the I/O is completed, then to proceed to 20. The arguments for FTEST are:

K = control array used in call to FBSAM

L = variable set to 0 if I/O is not complete
1 if I/O is complete

If the I/O is complete, FTEST performs the necessary checking, eliminating the need for a separate call to FCHECK.

F. Spacing

Routines are available for spacing a given number of blocks, forward (FORSPC) or backward (FBKSPC):

```
CALL FORSPC (K,M)
CALL FBKSPC (K,M)
```

where

K = control array supplied to FBSAM

M = number of blocks to be spaced over.

Spacing should not be attempted if an I/O request using FWRITE or FREAD has not been checked. Backspacing is subject to these limitations:

1) Backspacing after reading the last block does not work properly.

2) Backspacing must not be used with track overflow.

G. Note-Point Operations

Among the most useful features of the FBSAM package is the ability to note the position of a record, and then to return directly to that position at a later time. The NOTE function may be accomplished by a call to FNOTE, which also supplies some additional data (whether it is wanted or not). Then FPOINT moves the pointer to that position for reading or writing.

```
CALL FNOTE (K,MM,NOTE,NWR,KWR,LOF)
      '
      '
      '
CALL FPOINT (K,NOTE)
```

where

K = control array supplied to FBSAM

MM = no effect

NOTE = integer word into which is stored the relative position in the data set (TTR)

NWR = number of blocks transferred at last I/O

KWR = number of bytes transferred at last I/O

LOF = 0: no filemark has been encountered on this data set since last opened

1: a filemark has been encountered.

An alternate procedure can be used:

```
CØMMON/CØMSAM/LTRACK,NØTEP,KKSTA
NOTE = NOTEP
  '
  '
  '
  '
CALL FPOINT (K,NOTE)
```

The note-point feature can be superior to the FORTRAN direct access if a few long records whose length is difficult to predict are written. It is inferior if the storage space for the NOTE words is excessive.

H. Pointing by Block Number

If fixed-length blocks are chosen at opening time, the N<u>th</u> logical record can be pointed to as follows:

```
CALL FPNTR(K,N)
```

This eliminates the need to store the NOTE data. All logical records must have the same length when using this feature.

I. Cautions

A collected list of cautions should be provided, especially for the user who wishes to use the pointing feature.

1) First, the FBSAM data sets are not prewritten as are FORTRAN direct data sets. This saves time, but it is up to the user to avoid pointing to a block not yet written.

2) Writing a block destroys any other block on the same track.

3) After writing beyond the previous file mark on a data set, a call to FREW or FDISP must proceed a point operation in order to obtain a file mark. This same restriction applies to a new data set which has no file mark.

4) After writing, a point operation must precede a call to FREW, FDISP, or the reading routines if a file mark is not desired.

5) The K array must not be destroyed by an OVERLAY operation prior to closing its data set.

J. Constant Arguments

Although variables are shown as arguments in the illustrations, the only arguments changed by the package, in addition to the K array, of course, are the value of L in FTEST and the arguments of FNOTE. Thus, constants or expressions can be used for other arguments where appropriate.

A summary of entries and arguments is given in Appendix B.

K. Common Block

FBSAM keeps a small common block full of good things. It can be accessed as follows:

```
COMMON/COMSAM/LTRACK,NOTEP,KKSTA
```

LTRACK = longest block which can be written on a track

NOTEP = relative position of last record read or written

KKSTA = error status code as printed in the message described later.

If a call to FBSAM is made with LBL < 0, LTRACK will be set and a return made without opening a data set. This allows the user to use LTRACK in decision making prior to opening.

IV. ERROR CONDITIONS

A. Error Procedure

Our present experience indicates that most errors connected with FBSAM usage will be detected by the package. An error code will be printed, the K array will be edited, and then a call to SUBROUTINE ERRTRA is provided. The K array can be interpreted by referring to Appendix A.

If a full dump is wished, supply the following:

```
SUBROUTINE ERRTRA
DATA I/2 000 000 000/
DIMENSION J(1)
I=J(I)
STOP
END
```

B. Status Errors

An error code of greater than 1000 refers to a status error. If the code given is M, find:

M1 = M/1000 (integer)
M2 = M - M1*1000 (integer)

Then M2 indicates the entry code of the entry through which the invalid operation is attempted. The codes are listed in Appendix B. M1 is the prior status of the data set, which was unsatisfactory for the operation indicated by M2. The meanings of the status codes are as follows:

Status Code	Meaning
1	Opened, but not used
2	Written, but not checked
3	Read, but not checked
4	Checked
5	Rewound or passed
6	Permanently closed

The last digit of the entry code becomes the new status code after a successful operation. As an example, an error of 3002 indicates that an FWRITE was attempted without checking a previous read. A table of error-producing combinations of entry and status follows:

Status Error Table

("X" Indicates an Error)

Entry Code	Entry Name	Status Code 1 Open	2 Written	3 Read	4 Checked	5 Rewound or Passed	6 Closed
2	FWRITE		x	x			x
3	FREAD		x	x			x
74	FWRCH		x	x			x
84	FRDCH		x	x			x
14	FTEST						x
4	FCHECK						x
24	FNOTE	x	x	x			
34	FPNTR		x	x			x
64	FPOINT		x	x			x
44	FBKSPC	x	x	x			x
54	FORSPC		x	x			x
15	FREW		x	x			x
25	FPASS		x	x			x
6	FDISP		x	x			

C. Other Errors Detected

Certain other errors detected can be identified by inspecting the K array. These errors, grouped according to error code, are:

1. Code = 2,3

   a. Record is too long to process; K(NIO) > K(NBLK)

2. Code = 4,14,74,84

   a. I/O error was encountered; K(KERC) $\geq$ 10000

   b. Filemark was encountered; 1000 > K(KERC) $\geq$ 100

   c. Wrong record length processed; K(KLL) $\neq$ K(NERR)

3. Code = 54,64

   a. I/O error; K(KERF) $\geq$ 10000

   b. Filemark was encountered; K(KERF) $\geq$ 100

4. Code = 1

   a. Error in opening data set; K(KERO) > 0

V. APPLICATIONS AND AVAILABILITY

Application to a representative I/O-bound program is described in Appendix C. Although some of the I/O had to remain in FORTRAN for compatibility, the use of FBSAM led to a significant reduction in both real and CPU time usage. A demonstration program is listed and described in Appendix D, together with results from the program.

Appendix D includes a listing of FBSAM. FBSAM and the KBSAM assembler language package can be obtained from the Radiation Shielding Information Center, Box X, Oak Ridge, TN. 37830.

VI. DEMONSTRATION PROGRAM

After setting up control integers, the test program writes a data set on LUN1. While a given record is being written, the next record is being prepared. An integer code identifying each word uniquely is used. The position of each record is noted. A time analysis is given.

In Section 2, all the preceding blocks are read as a "super record." The first word of each record is edited to confirm the operation.

In Section 3, selective updating is illustrated.

In Section 4, all the data are read as a "super record" again, and the first-word edit given.

In Section 5, the data set is read backward, with the first word of each record saved again. The test-and-go function is used to allow a computer loop to continue until the I/O is completed.

In Section 6, the data set is written, checking implicitly.

In Section 7, the writing is repeated with explicit checking.

In Section 8, a complicated repositioning exercise is performed.

After the first "A" pass, Section 9 copies the data to a second data set and reverses the logical unit numbers.

Pass "B" repeats the above Sections 2 through 8 using the copy of the original data set. The operations in Sections 5 and 8 are done differently in pass B.

Three sets of passes are done on all, with the role of FT01F001 and FT02F001 reversing each time. The code is of no practical value, except that a deck which will survive its testing will probably be satisfactory for a comtemplated application!

REFERENCES

1. J. G. Sullivan, "KBSAM - Buffering Input/Output to Direct Access Devices on IBM/360 Computers," unpublished memorandum (June 30, 1972).

2. C. M. N. Cree, "KBSAM Version I, BSAM Subroutine Package For)S360," IBM Corporation, Hawthorne, N.Y., Program No. 360D-03.4.026 (January 1969).

3. "IBM System/360 Operating System: Supervisor and Data Management Macro Instructions," IBM Systems Reference Library Publication, Form C28-6647-4 (June 1970).

4. "IBM System/360 Operating System: Supervisor and Data Management Services," IBM Systems Reference Library Publications, Form C28-6646-3 (June 1970).

APPENDIX A

Contents of "K" Control Array

Position	Symbol	Use
1	K(1)	DDNAME (8 characters)
2	'	
3	K(KSTA)	Entry status code
4	K(KERO)	Opening error flag
5	K(KERC)	Checking error flag
6	K(KERB)	Backspacing error flag
7	K(KERF)	Forward spacing error flag
8	K(NOTE1)	"TTR" Position for first block
9	K(NOTE2)	"TTR" Position for last block
10	K(KLL)	Length of Logical record processed
11	K(LBLK)	Maximum block length
12	K(NBLK)	Maximum number of blocks per logical record
13	K(NIO)	Number of blocks processed
14	K(NSTA)	Prior status code
15	K(NERR)	Length of logical record requested
16	K(LIMK)	Length of K array
17	K(LERC)	Error override code
18	K(KDCB)	Data control block
'	'	'
'	'	'
'	'	'
'	'	'
40	'	'
41	K(KDECB)	Data event control block(s)
'	'	'
'	'	'
'	'	'
'	'	'
40+5*NBL	'	'

APPENDIX B

Summary of Entries Available

Entry Code	Entry Name	Comments
1	FBSAM(LUN,K,LBL, NBL,NERC)	Opens data set. LUN is logical unit number. K is 40 + 5*NBL word array. LBL is maximum block length (bytes). NBL is maximum blocks per record queued. NERC = NERC1 + NERC3*100 NERC1 is 0 for undefined block length 2 for fixed block length NERC3 is 0 for filemark error exit 1 to bypass filemark error exit
2	FWRITE (K,X,N)	Writes N bytes for X array. Must be followed by FCHECK or FTEST.
3	FREAD (K,X,N)	Reads N bytes into X array. Must be followed by FCHECK or FTEST.
74	FWRCH (K,X,N)	Writes N bytes from X array, checking as it writes.
84	FRDCH (K,X,N)	Reads N bytes into X array, checking as it reads.
14	FTEST (K,L)	Tests for completion of I/O. Returns flag: L is 0 if I/O is not complete 1 if I/O is complete If L=1 is returned, no call to FCHECK is needed.
4	FCHECK (K)	Tests for completion of I/O, and waits until it is done. Must be used after FWRITE or FREAD unless FTEST is used.
24	FNOTE(K,MM,NOTE, NWR,KWR,LOF)	Returns status information: MM has no effect NOTE is position of last block processed. NWR is set to number of blocks processed. KWR is set to number of bytes processed. LOF is 0 if no filemark encountered 1 if filemark encountered.
34	FPOINT(K,NOTE)	Moves pointer to position given by NOTE.
64	FPNTR(K,N)	Moves pointer to nth logical record.

Entry Code	Entry Name	Comments
44	FBKSPC(K,M)	Backspaces M blocks
54	FORSPC(K,M)	Forward spaces M blocks
15	FREW(K)	Repositions pointer at beginning of data set.
25	FPASS(K)	Repositions pointer at end of data set.
6	FDISP(K)	Closes data set according to DISP parameter.

APPENDIX C

Application of FBSAM to the FACT Code

A demonstration of the effects of BSAM I/O on an I/O-bound code, FACT, was made. FACT[C.1] is designed to convert boundary flux tapes from one DOT[C.2] problem for use in another DOT problem having different directional quadrature and spatial meshes. FACT also provides facility for folding the boundary flux together with boundary adjoint flux.

A flow chart for FACT using standard FORTRAN I/O is shown in Fig. C.1. The original code used the "implied-do" type of I/O typified by the following:

```
READ(NBØ)((BSØD(I,J),I=1,MMDIRØ),J=1,IJ)
```

Two trial runs of this code were made with a routine which measured the time consumption of 10 individual activities, in addition to the overall time. Real (wall clock) tume usage was measured in addition to CPU usage. Results from the two runs are shown in Table C.1. The jobs were run without interference from other jobs in the computer, and time reproducibility typically fell within a ±10% band. It should be noted that the "computational" activities included a small amount of incidental output.

In the next step, the sequential FORTRAN I/O was changed to the "short-list" type as shown:

```
CALL WARIN (BSØD,MMDIRØ*IJ,NBØ)
'
'
'
SUBROUTINE WARIN (X,N,M)
DIMENSION X(N)
READ (M) X
RETURN
END
```

This placed an important limitation on the use of the code. In one mode of operation, additional arrays were stored before BSØD in the same logical record. Using the "short-list" method, it is not possible to skip over those arrays without providing storage for them. It might have been possible to convert the direct access operation, but this was not attempted.

The "short-list" method had little effect on real time, as shown in the third column of the table, but it cut the CPU time consumed by the converted operations by about a factor of 20. (Times of less than a second are probably not very accurate due to timing uncertainties.)

The next step was to convert 4 of the 7 I/O activities to BSAM I/O, but without using the overlapping capability. The remaining 3 activities remained in FORTRAN to retain compatibility with DOT. The step in which the FORTRAN data set was copied to FORTRAN direct data set was replaced with a BSAM data set with note-point capability. The fourth column of the table shows these results. Further reductions were made in CPU time where BSAM was used, the real time used in writing and reading were cut by roughly 55% and 25%, respectively. The note-point method was vastly superior to the direct-access data set in performance. A small increase in clock time was due to programming changes to be explained later.

Finally, the program was reorganized to allow use of full overlapping, as illustrated in Fig. C.2. In the angular-mesh section, output of intermediate data set was overlapped with the reading of an input data set. In the spatial-mesh-change section, a "double buffering" technique involing dual storage arrays was used. Input, computation, and output were overlapped giving the capability for three concurrent activities. In the folding section, the flux and adjoint reading were overlapped. BUF1 and BUF2 are DSNAMES given to the two BSAM scratch data sets.

The striking improvement in real time consumption is evident from the last two columns of the table. The overlapped functions are "almost free" with respect to real time. Changes in CPU time were observed, but they are probably not significant in the statistical sense.

The small increase in CPU time between both BSAM trials and the FORTRAN trials was due to the programming required for the dual storage arrays.

Comparing overall BSAM performance with the "short-list FORTRAN" method, reductions of 54% in real time and 40% in CPU time were obtained. A decrease of 34% in I/O requests was obtained. Only the last two factors are used in the ORNL computer cost formula, but the real time can also be of concern in codes such as DOT, which monopolize core storage and/or magnetic tape and disk drive resources. The tests were performed prior to activating the chaining feature. Thus, even better performance would be obtained from the present packages.

The FBSAM package itself was not at all difficult to apply, and overlapping BSAM I/O with FORTRAN I/O was quite simple. Applying the double-buffering technique to the spatial-mesh-changing section, however, required considerable reprogramming.

The most serious limitation of the method appears to be the incompatibility with FORTRAN I/O, but no attempt to remove this limitation has been made. The FBSAM method shares with short-list FORTRAN I/O the limitation that it is not, in general, possible to pass a portion of a record without providing storage for it. An exception would be that an integral number of blocks at the beginning of a record can be passed using FBSAM.

Table C.1. FACT Timing Summary

Top number = real time (seconds)
Bottom number = CPU time (seconds)

	Implied Do			Short List and BSAM	Short List and BSAM
	Run #1	Run #2	Short List	No Overlap	Overlap
Clock 1	307.90	295.97	290.60	194.48	134.28
Overall Time	109.26	110.51	64.71	38.96	38.70
Clock 2	17.47	17.88	17.54	17.56	16.86
Change Directional Quadrature	16.97	17.11	17.38	17.10	16.44
Clock 3	39.72	36.54	⟨35.47⟩	⟨34.70⟩	⟨34.13⟩
Read Flux from FT01F001	9.31	8.92	⟨0.41⟩	⟨0.09⟩	⟨0.57⟩
Clock 4	26.17	24.20	⟨23.39⟩	[10.51]	[0.26]
Write Flux onto FT02F001 or BUF1	9.54	9.45	⟨0.56⟩	[0.13]	[0.01]
Clock 5	13.30	14.69	12.25	16.45	16.40
Change Spatial Mesh	13.08	13.90	11.95	15.87	15.89
Clock 6	33.93	25.09	⟨22.80⟩	[16.97]	[0.35]
Read Flux from FT02F001 or BUF1	9.07	9.30	⟨0.48⟩	[0.04]	[0.04]
Clock 7	21.75	27.58	⟨22.85⟩	⟨21.05⟩	⟨21.24⟩
Write Flux Onto FT03F001	9.63	9.35	⟨0.47⟩	⟨0.61⟩	⟨0.60⟩
Clock 8	83.81	84.54	84.15	[13.65]	[0.36]
Copy Flux Onto D.A. Data Set or BUF2	19.29	18.93	18.88	[0.18]	[0.10]
Clock 9	2.75	2.73	2.40	2.64	2.80
Flux-Adjoint Folding	2.70	2.68	2.38	2.56	2.83
Clock 10	31.41	34.25	32.92	[20.48]	[3.37]
Read Flux from D.A. Data Set or BUF 2	8.85	9.00	9.98	[0.03]	[0.05]
Clock 11	33.81	24.23	⟨33.55⟩	⟨36.74⟩	⟨34.99⟩
Read Adjoint from FT09F001	9.11	10.09	⟨0.29⟩	⟨0.51⟩	⟨0.47⟩
I/O Requests	4769	4770	4798	3152	3147

⟨ ⟩ Indicates use of short-list method.

[ ] Indicates use of BSAM method.

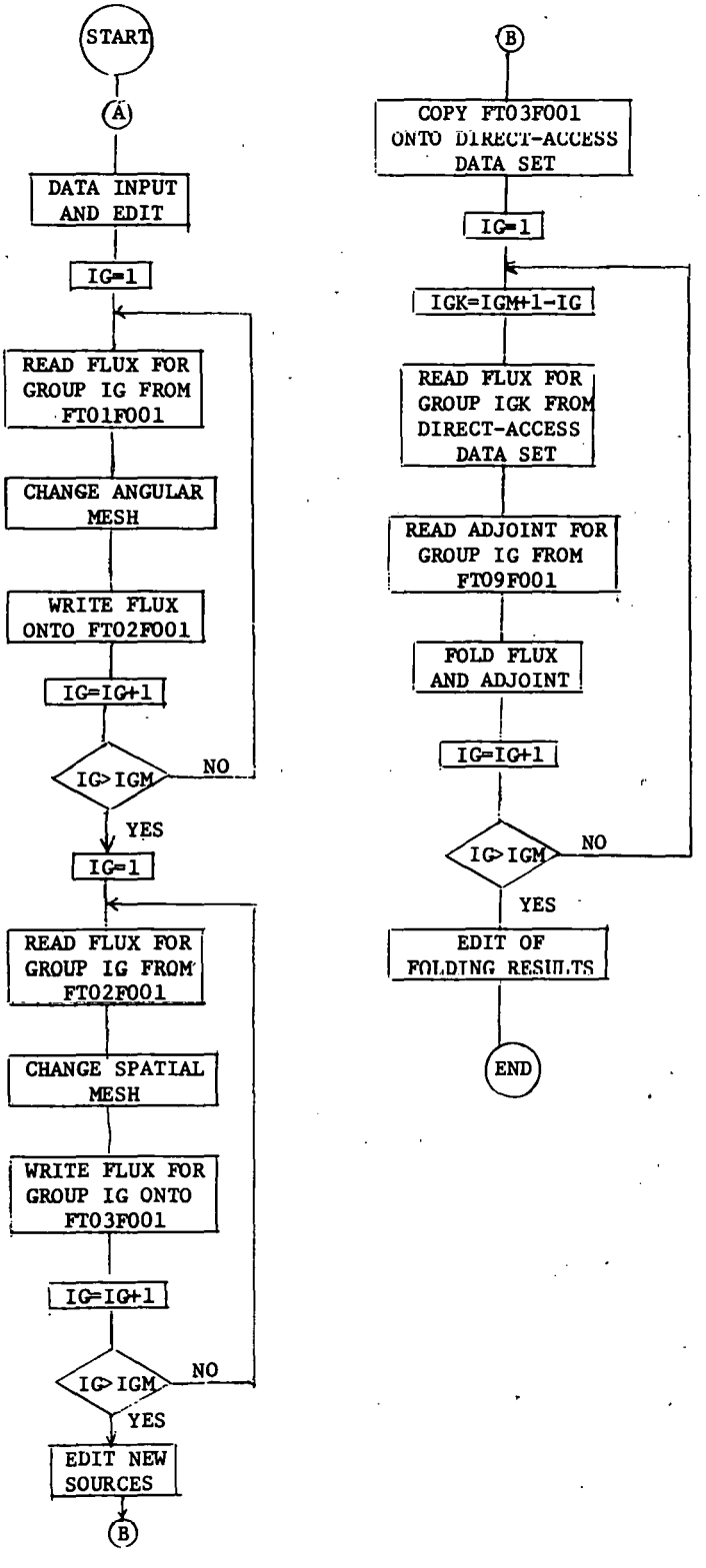

Fig. C.1. The FACT Code with FORTRAN I/O.

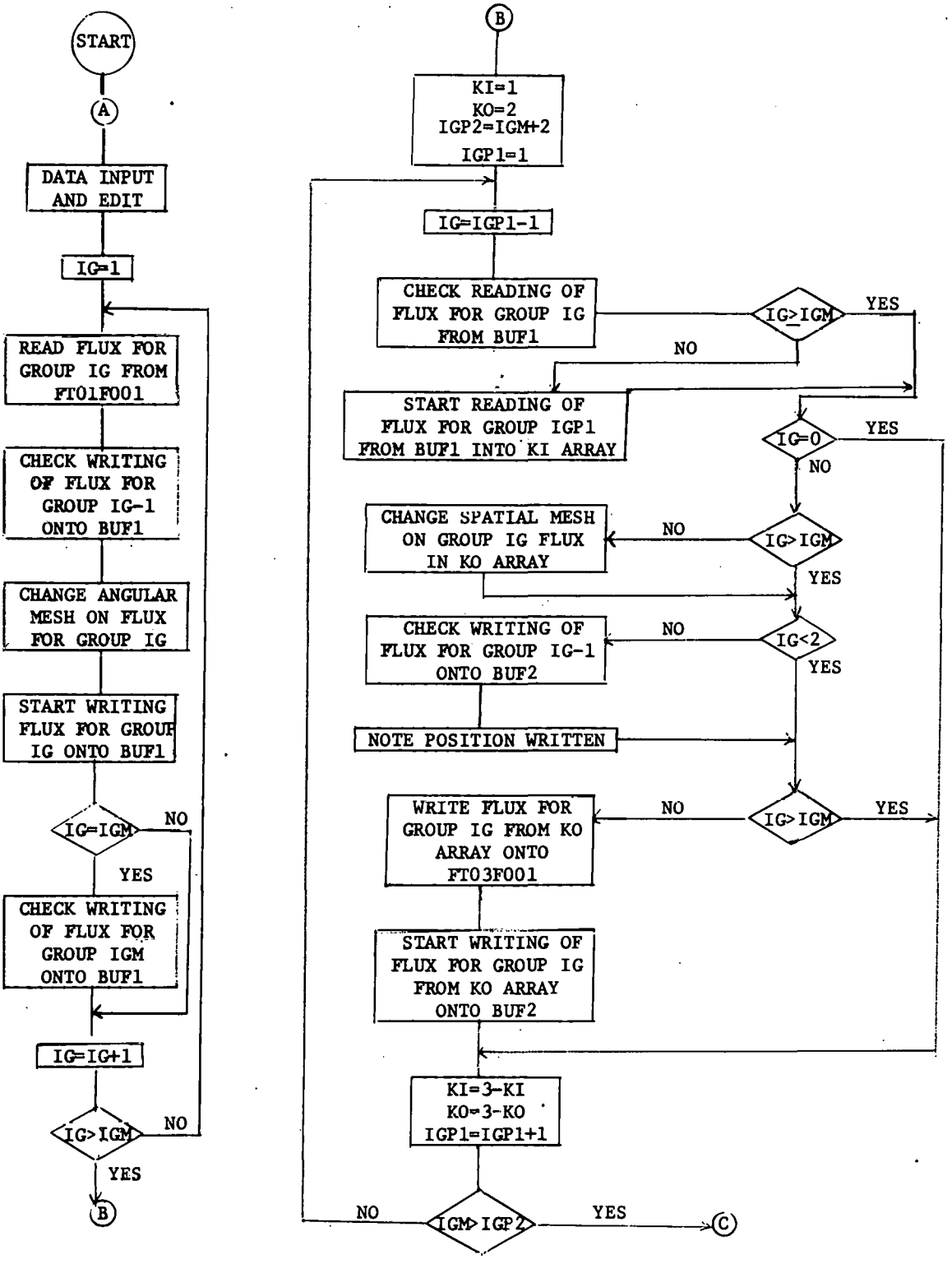

Fig. C.2. The FACT Code with BSAM I/O.

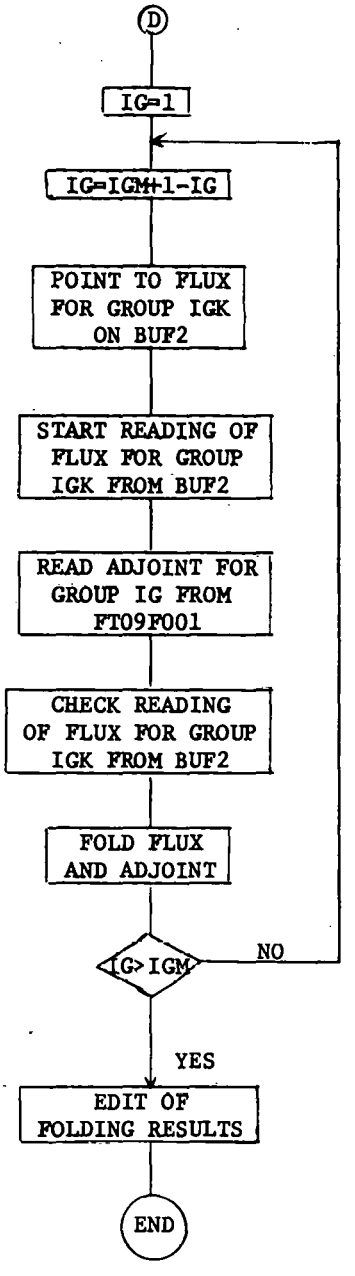
D
IG=1
IG=IGM+1-IG
POINT TO FLUX FOR GROUP IGK ON BUF2
START READING OF FLUX FOR GROUP IGK FROM BUF2
READ ADJOINT FOR GROUP IG FROM FT09F001
CHECK READING OF FLUX FOR GROUP IGK FROM BUF2
FOLD FLUX AND ADJOINT
IG>IGM
NO
YES
EDIT OF FOLDING RESULTS
END

Fig. C.2. (Cont'd.)

APPENDIX C

References

C.1. M. L. Gritzner, Neutron Physics Division, ORNL, unpublished Computer Code.

C.2. F. R. Mynatt, "A User's Manual for DOT - A Two-Dimensional Discrete Ordinates Transport Code with Anisotropic Scattering," Union Carbide Corporation, K-1694.

APPENDIX D

Listing of FBSAM and Test Program

```
FIRST WORD OF EACH RECORD
     1 *    72001      1901       1801       1701       1601 *     1501       1401       1301       1201      71101
    11 *     1001       901        801        701        601 *      501        401        301        201      70101

TIME SUMMARY (SECONDS) FOR SECTION   6

                 1          2          3          4          5          6          7          8          9         10
               OPEN    COMPUTE       TEST    RD/WRIT      CHECK       NOTE   RWND/PASS     FORSPC      BKSPC      POINT
REAL TIME=     0.0        0.0        0.0      1.410        0.0        0.0      0.340        0.0        0.0        0.0
CPU TIME =     0.0        0.0        0.0        0.0        0.0        0.0        0.0        0.0        0.0        0.0

TOTAL REAL TIME =      1.750

TOTAL CPU TIME  =      0.0

TIME SUMMARY (SECONDS) FOR SECTION   7

                 1          2          3          4          5          6          7          8          9         10
               OPEN    COMPUTE       TEST    RD/WRIT      CHECK       NOTE   RWND/PASS     FORSPC      BKSPC      POINT
REAL TIME=     0.0        0.0        0.0      0.030      0.530        0.0      0.340        0.0        0.0        0.0
CPU TIME =     0.0        0.0        0.0      0.020        0.0        0.0        0.0        0.0        0.0        0.0

TOTAL REAL TIME =      0.900

TOTAL CPU TIME  =      0.020

TIME SUMMARY (SECONDS) FOR SECTION   8

                 1          2          3          4          5          6          7          8          9         10
               OPEN    COMPUTE       TEST    RD/WRIT      CHECK       NOTE   RWND/PASS     FORSPC      BKSPC      POINT
REAL TIME=     0.0        0.0        0.0      0.180        0.0        0.0      0.200        0.0      1.900      0.020
CPU TIME =     0.0        0.0        0.0        0.0        0.0        0.0        0.0        0.0        0.0        0.0

TOTAL REAL TIME =      2.300

TOTAL CPU TIME  =      0.0

RESULTS OF SPACING EXERCISE
     1 *    72001      70101      72001      72001      72001 *     72001      66937       1301       1201      71101
```

```
* * * PROBLEM BEGAN ON 10/29/75 (302) AT TIME 11 HRS. 41.68 MIN. * * *

TIME SUMMARY (SECONDS) FOR SECTION   2

                  1       2       3       4       5       6       7       8       9      10
                 OPEN  COMPUTE    TEST  RD/WRIT   CHECK    NOTE  RWND/PASS  FORSPC   BKSPC   POINT
REAL TIME=       0.0     0.0     0.0     0.560    0.0     0.0     0.150    0.0     0.0     0.0
CPU TIME =       0.0     0.0     0.0     0.0      0.0     0.0     0.0      0.0     0.0     0.0

TOTAL REAL TIME =      0.710

TOTAL CPU TIME  =      0.0

FIRST WORD OF EACH RECORD
    1 *     70101       201       301       401       501 *       601       701       801       901      1001
 17501 *    71101      1201      1301      1401      1501 *      1601      1701      1801      1901     72001

TIME SUMMARY (SECONDS) FOR SECTION   3

                  1       2       3       4       5       6       7       8       9      10
                 OPEN  COMPUTE    TEST  RD/WRIT   CHECK    NOTE  RWND/PASS  FORSPC   BKSPC   POINT
REAL TIME=       0.0     0.0     0.0     0.530    0.0     0.0     0.0      0.0     0.0     0.0
CPU TIME =       0.0     0.0     0.0     0.0      0.0     0.0     0.0      0.0     0.0     0.0

TOTAL REAL TIME =      0.530

TOTAL CPU TIME  =      0.0

TIME SUMMARY (SECONDS) FOR SECTION   4

                  1       2       3       4       5       6       7       8       9      10
                 OPEN  COMPUTE    TEST  RD/WRIT   CHECK    NOTE  RWND/PASS  FORSPC   BKSPC   POINT
REAL TIME=       0.0     0.0     0.0     0.020    0.520    0.0     0.0      0.0     0.0     0.0
CPU TIME =       0.0     0.0     0.0     0.010    0.050    0.0     0.0      0.0     0.0     0.0

TOTAL REAL TIME =      0.540

TOTAL CPU TIME  =      0.060

FIRST WORD OF EACH RECORD
    1 *     70101       201       301       401       501 *       601       701       801       901      1001
 17501 *    71101      1201      1301      1401      1501 *      1601      1701      1801      1901     72001

TIME SUMMARY (SECONDS) FOR SECTION   5

                  1       2       3       4       5       6       7       8       9      10
                 OPEN  COMPUTE    TEST  RD/WRIT   CHECK    NOTE  RWND/PASS  FORSPC   BKSPC   POINT
REAL TIME=       0.0     0.0     0.0     0.0      2.200    0.0     0.0      0.0     0.0     0.0
CPU TIME =       0.0     0.0     0.0     0.0      0.0     0.0     0.0      0.0     0.0     0.0

TOTAL REAL TIME =      2.200

TOTAL CPU TIME  =      0.0
```

	1 OPEN	2 COMPUTE	3 TEST	4 RD/WRIT	5 CHECK	6 NOTE	7 RWND/PASS	8 FORSPC	9 BKSPC	10 POINT
REAL TIME=	0.520	0.0	0.0	6.160	0.0	0.0	0.0	0.0	0.0	0.0
CPU TIME =	0.050	0.0	0.0	0.100	0.0	0.0	0.0	0.0	0.0	0.0

TOTAL REAL TIME = 6.680

TOTAL CPU TIME = 0.150

```
FIRST WORD OF EACH RECORD
    1 *    70101      201      301      401      501 *      601      701      801      901     1001
 1750I *   71101     1201     1301     1401     1501 *     1601     1701     1801     1901    72001
TIME SUMMARY (SECONDS) FOR SECTION  5

                 1        2        3        4        5        6        7        8        9       10
               OPEN  COMPUTE     TEST  RD/WRIT    CHECK     NOTE  RWND/PASS   FORSPC   BKSPC    POINT
REAL TIME=     0.0      0.0    3.520      0.0      0.0      0.0      0.0      0.0      0.0      0.0
CPU TIME =     0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0

TOTAL REAL TIME =      3.520

TOTAL CPU TIME  =      0.0

FIRST WORD OF EACH RECORD
    1 *    72001     1901     1801     1701     1601 *     1501     1401     1301     1201    71101
   11 *     1001      901      801      701      601 *      501      401      301      201    70101
TIME SUMMARY (SECONDS) FOR SECTION   6

                 1        2        3        4        5        6        7        8        9       10
               OPEN  COMPUTE     TEST  RD/WRIT    CHECK     NOTE  RWND/PASS   FORSPC   BKSPC    POINT
REAL TIME=     0.0      0.0      0.0    0.570      0.0      0.0    0.750      0.0      0.0      0.0
CPU TIME =     0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0

TOTAL REAL TIME =      1.320

TOTAL CPU TIME  =      0.0

TIME SUMMARY (SECONDS) FOR SECTION   7

                 1        2        3        4        5        6        7        8        9       10
               OPEN  COMPUTE     TEST  RD/WRIT    CHECK     NOTE  RWND/PASS   FORSPC   BKSPC    POINT
REAL TIME=     0.0      0.0      0.0    0.010    0.550      0.0    0.570      0.0      0.0      0.0
CPU TIME =     0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0

TOTAL REAL TIME =      1.130

TOTAL CPU TIME  =      0.0

TIME SUMMARY (SECONDS) FOR SECTION   8

                 1        2        3        4        5        6        7        8        9       10
               OPEN  COMPUTE     TEST  RD/WRIT    CHECK     NOTE  RWND/PASS   FORSPC   BKSPC    POINT
REAL TIME=     0.0      0.0      0.0    0.750      0.0      0.0    0.040    4.660      0.0      0.0
CPU TIME =     0.0      0.0      0.0      0.0      0.0      0.0    0.020      0.0      0.0      0.0

TOTAL REAL TIME =      5.450

TOTAL CPU TIME  =      0.020

RESULTS OF SPACING EXERCISE
    1 *    72001    72001    72001    72001    72001 *    72001    66937     1301     1201    71101
TIME SUMMARY (SECONDS) FOR SECTION   9
```

```
* * * PROBLEM BEGAN ON 10/29/75 (302) AT TIME 11 HRS. 41.15 MIN. * * *

TIME SUMMARY (SECONDS) FOR SECTION   1

                  1       2       3       4       5       6       7       8       9      10
                 OPEN   COMPUTE   TEST   RD/WRIT CHECK   NOTE   RWND/PASS FORSPC  BKSPC  POINT
REAL TIME=      1.040   0.040   0.0     0.0     7.840   0.0     0.390   0.0     0.0     0.0
CPU TIME =      0.0     0.020   0.0     0.0     0.0     0.0     0.020   0.0     0.0     0.0

TOTAL REAL TIME =      9.310

TOTAL CPU TIME  =      0.040

NOTE ARRAY
     1 * 000000FF  00000100  00010100  00020100  00030100 * 00040100  00050100  00060100  00070100  00080100
    11 * 00090100  000A0100  000B0100  000C0100  000D0100 * 000E0100  000F0100  00100100  00110100  00120100

TIME SUMMARY (SECONDS) FOR SECTION   2

                  1       2       3       4       5       6       7       8       9      10
                 OPEN   COMPUTE   TEST   RD/WRIT CHECK   NOTE   RWND/PASS FORSPC  BKSPC  POINT
REAL TIME=      0.0     0.0     0.0     0.530   0.0     0.0     0.200   0.0     0.0     0.0
CPU TIME =      0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0

TOTAL REAL TIME =      0.730

TOTAL CPU TIME  =      0.0

FIRST WORD OF EACH RECORD
     1 *      101      201      301      401      501 *      601      701      801      901     1001
 17501 *     1101     1201     1301     1401     1501 *     1601     1701     1801     1901     2001

TIME SUMMARY (SECONDS) FOR SECTION   3

                  1       2       3       4       5       6       7       8       9      10
                 OPEN   COMPUTE   TEST   RD/WRIT CHECK   NOTE   RWND/PASS FORSPC  BKSPC  POINT
REAL TIME=      0.0     0.020   0.0     2.780   0.0     0.0     0.0     0.0     0.0     0.0
CPU TIME =      0.0     0.010   0.0     0.020   0.0     0.0     0.0     0.0     0.0     0.0

TOTAL REAL TIME =      2.800

TOTAL CPU TIME  =      0.030

TIME SUMMARY (SECONDS) FOR SECTION   4

                  1       2       3       4       5       6       7       8       9      10
                 OPEN   COMPUTE   TEST   RD/WRIT CHECK   NOTE   RWND/PASS FORSPC  BKSPC  POINT
REAL TIME=      0.0     0.0     0.0     0.0     0.750   0.0     0.0     0.0     0.0     0.0
CPU TIME =      0.0     0.0     0.0     0.0     0.030   0.0     0.0     0.0     0.0     0.0

TOTAL REAL TIME =      0.750

TOTAL CPU TIME  =      0.030
```

FIRST WORD OF EACH RECORD

1 *	72001	1901	1801	1701	1601 *	1501	1401	1301	1201	71101
11 *	1001	901	801	701	601 *	501	401	301	201	70101

TIME SUMMARY (SECONDS) FOR SECTION 6

	1 OPEN	2 COMPUTE	3 TEST	4 RD/WRIT	5 CHECK	6 NOTE	7 RWND/PASS	8 FORSPC	9 BKSPC	10 POINT
REAL TIME=	0.0	0.0	0.0	0.540	0.0	0.0	0.980	0.0	0.0	0.0
CPU TIME =	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

TOTAL REAL TIME = 1.520

TOTAL CPU TIME = 0.0

TIME SUMMARY (SECONDS) FOR SECTION 7

	1 OPEN	2 COMPUTE	3 TEST	4 RD/WRIT	5 CHECK	6 NOTE	7 RWND/PASS	8 FORSPC	9 BKSPC	10 POINT
REAL TIME=	0.0	0.0	0.0	0.0	0.680	0.0	1.380	0.0	0.0	0.0
CPU TIME =	0.0	0.0	0.0	0.0	0.040	0.0	0.010	0.0	0.0	0.0

TOTAL REAL TIME = 2.060

TOTAL CPU TIME = 0.050

TIME SUMMARY (SECONDS) FOR SECTION 8

	1 OPEN	2 COMPUTE	3 TEST	4 RD/WRIT	5 CHECK	6 NOTE	7 RWND/PASS	8 FORSPC	9 BKSPC	10 POINT
REAL TIME=	0.0	0.0	0.0	0.380	0.0	0.0	0.040	2.520	0.0	0.0
CPU TIME =	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

TOTAL REAL TIME = 2.940

TOTAL CPU TIME = 0.0

RESULTS OF SPACING EXERCISE

1 *	72001	72001	72001	72001	72001 *	72001	66937	1301	1201	71101

```
* * * PROBLEM BEGAN ON 10/29/75 (302) AT TIME 11 HRS. 40.88 MIN. * * *

TIME SUMMARY (SECONDS) FOR SECTION   2

                  1       2       3       4       5       6       7       8       9      10
                 OPEN  COMPUTE    TEST  RD/WRIT   CHECK    NOTE  RWND/PASS  FORSPC   BKSPC   POINT
REAL TIME=      0.0     0.0     0.0     0.670     0.0     0.0     0.180     0.0     0.0     0.0
CPU TIME =      0.0     0.0     0.0     0.0       0.0     0.0     0.010     0.0     0.0     0.0

TOTAL REAL TIME =      0.850

TOTAL CPU TIME  =      0.010

FIRST WORD OF EACH RECORD
     1 *    70101      201      301      401      501 *      601      701      801      901     1001
 17501 *    71101     1201     1301     1401     1501 *     1601     1701     1801     1901    72001

TIME SUMMARY (SECONDS) FOR SECTION   3

                  1       2       3       4       5       6       7       8       9      10
                 OPEN  COMPUTE    TEST  RD/WRIT   CHECK    NOTE  RWND/PASS  FORSPC   BKSPC   POINT
REAL TIME=      0.0     0.0     0.0     2.250     0.0     0.0     0.0     0.0     0.0     0.0
CPU TIME =      0.0     0.0     0.0     0.0       0.0     0.0     0.0     0.0     0.0     0.0

TOTAL REAL TIME =      2.250

TOTAL CPU TIME  =      0.0

TIME SUMMARY (SECONDS) FOR SECTION   4

                  1       2       3       4       5       6       7       8       9      10
                 OPEN  COMPUTE    TEST  RD/WRIT   CHECK    NOTE  RWND/PASS  FORSPC   BKSPC   POINT
REAL TIME=      0.0     0.0     0.0     0.0       0.910   0.0     0.0     0.0     0.0     0.0
CPU TIME =      0.0     0.0     0.0     0.0       0.0     0.0     0.0     0.0     0.0     0.0

TOTAL REAL TIME =      0.910

TOTAL CPU TIME  =      0.0

FIRST WORD OF EACH RECORD
     1 *    70101      201      3C1      401      501 *      601      701      801      901     1001
 17501 *    71101     1201     13C1     1401     1501 *     1601     1701     1801     1901    72001

TIME SUMMARY (SECONDS) FOR SECTION   5

                  1       2       3       4       5       6       7       8       9      10
                 OPEN  COMPUTE    TEST  RD/WRIT   CHECK    NOTE  RWND/PASS  FORSPC   BKSPC   POINT
REAL TIME=      0.0     0.0     0.0     0.020     6.000   0.0     0.0     0.0     0.0     0.0
CPU TIME =      0.0     0.0     0.0     0.020     0.0     0.0     0.0     0.0     0.0     0.0

TOTAL REAL TIME =      6.020

TOTAL CPU TIME  =      0.020
```

	1 OPEN	2 COMPUTE	3 TEST	4 RD/WRIT	5 CHECK	6 NOTE	7 RWND/PASS	8 FORSPC	9 BKSPC	10 POINT
REAL TIME=	0.670	0.0	0.0	11.970	0.0	0.0	0.0	0.0	0.0	0.0
CPU TIME =	0.020	0.0	0.0	0.080	0.0	0.0	0.0	0.0	0.0	0.0

TOTAL REAL TIME = 12.640

TOTAL CPU TIME = 0.100

```
FIRST WORD OF EACH RECORD
    1 *     7010:      201       301       401       501 *      601       701       801       901      1001
 17501 *    7110:     1201      1301      1401      1501 *     1601      1701      1801      1901     72001

TIME SUMMARY (SECONDS) FOR SECTION    5

                  1       2       3       4       5       6       7       8       9      10
                 OPEN  COMPUTE    TEST  RD/WRIT   CHECK    NOTE  RWND/PASS  FORSPC   BKSPC   POINT
REAL TIME=      0.0     0.0     2.990   0.040     0.0     0.0     0.0     0.0     0.0     0.0
CPU TIME =      0.0     0.0     0.0     0.020     0.0     0.0     0.0     0.0     0.0     0.0

TOTAL REAL TIME =       3.030

TOTAL CPU TIME  =       0.020

FIRST WORD OF EACH RECORD
    1 *    72001     1901      1801      1701      1601 *     1501      1401      1301      1201     71101
   11 *     1001      901       801       701       601 *      501       401       301       201     70101

TIME SUMMARY (SECONDS) FOR SECTION    6

                  1       2       3       4       5       6       7       8       9      10
                 OPEN  COMPUTE    TEST  RD/WRIT   CHECK    NOTE  RWND/PASS  FORSPC   BKSPC   POINT
REAL TIME=      0.0     0.0     0.0     0.730     0.0     0.0     0.500   0.0     0.0     0.0
CPU TIME =      0.0     0.0     0.0     0.0       0.0     0.0     0.0     0.0     0.0     0.0

TOTAL REAL TIME =       1.230

TOTAL CPU TIME  =       0.0

TIME SUMMARY (SECONDS) FOR SECTION    7

                  1       2       3       4       5       6       7       8       9      10
                 OPEN  COMPUTE    TEST  RD/WRIT   CHECK    NOTE  RWND/PASS  FORSPC   BKSPC   POINT
REAL TIME=      0.0     0.0     0.0     0.0       0.680   0.0     0.350   0.0     0.0     0.0
CPU TIME =      0.0     0.0     0.0     0.0       0.0     0.0     0.020   0.0     0.0     0.0

TOTAL REAL TIME =       1.030

TOTAL CPU TIME  =       0.020

TIME SUMMARY (SECONDS) FOR SECTION    8

                  1       2       3       4       5       6       7       8       9      10
                 OPEN  COMPUTE    TEST  RD/WRIT   CHECK    NOTE  RWND/PASS  FORSPC   BKSPC   POINT
REAL TIME=      0.0     0.0     0.0     0.290     0.0     0.0     0.050   0.0     5.000   0.010
CPU TIME =      0.0     0.0     0.0     0.0       0.0     0.0     0.0     0.0     0.050   0.0

TOTAL REAL TIME =       5.350

TOTAL CPU TIME  =       0.050

RESULTS OF SPACING EXERCISE
    1 *    72001    70101     72001     72001     72001 *    72001     66937      1301      1201     71101

TIME SUMMARY (SECONDS) FOR SECTION    9
```

```
* * * PROBLEM BEGAN ON 10/29/75 (302) AT TIME 11 HRS. 40.37 MIN. * * *

TIME SUMMARY (SECONDS) FOR SECTION   1

                 1      2       3      4       5      6       7       8      9      10
                OPEN  COMPUTE  TEST  RD/WRIT  CHECK  NOTE  RWND/PASS  FORSPC  BKSPC  POINT
REAL TIME=     0.820   0.040   0.0    0.060   3.030  0.0    0.510    0.0     0.0    0.0
CPU TIME =     0.040   0.030   0.0    0.040   0.070  0.0    0.0      0.0     0.0    0.0

TOTAL REAL TIME =      8.460

TOTAL CPU TIME  =      0.180

NOTE ARRAY
    1 * 000000FF  00000100  00010100  00020100  00030100 * 00040100  00050100  00060100  00070100  00080100
   11 * 00090100  000A0100  000B0100  000C0100  000D0100 * 000E0100  000F0100  00100100  00110100  00120100

TIME SUMMARY (SECONDS) FOR SECTION   2

                 1      2       3      4       5      6       7       8      9      10
                OPEN  COMPUTE  TEST  RD/WRIT  CHECK  NOTE  RWND/PASS  FORSPC  BKSPC  POINT
REAL TIME=     0.0     0.0     0.0    0.530   0.0    0.0    0.100    0.0     0.0    0.0
CPU TIME =     0.0     0.0     0.0    0.040   0.0    0.0    0.0      0.0     0.0    0.0

TOTAL REAL TIME =      0.630

TOTAL CPU TIME  =      0.040

FIRST WORD OF EACH RECORD
    1 *     101     201     301     401     501 *     601     701     801     901    1001
 17501 *   1101    1201    1301    1401    1501 *    1601    1701    1801    1901    2001

TIME SUMMARY (SECONDS) FOR SECTION   3

                 1      2       3      4       5      6       7       8      9      10
                OPEN  COMPUTE  TEST  RD/WRIT  CHECK  NOTE  RWND/PASS  FORSPC  BKSPC  POINT
REAL TIME=     0.0     0.0     0.0    0.630   0.0    0.0    0.0      0.0     0.0    0.0
CPU TIME =     0.0     0.0     0.0    0.0     0.0    0.0    0.0      0.0     0.0    0.0

TOTAL REAL TIME =      0.630

TOTAL CPU TIME  =      0.0

TIME SUMMARY (SECONDS) FOR SECTION   4

                 1      2       3      4       5      6       7       8      9      10
                OPEN  COMPUTE  TEST  RD/WRIT  CHECK  NOTE  RWND/PASS  FORSPC  BKSPC  POINT
REAL TIME=     0.0     0.0     0.0    0.0     0.610  0.0    0.0      0.0     0.0    0.0
CPU TIME =     0.0     0.0     0.0    0.0     0.020  0.0    0.0      0.0     0.0    0.0

TOTAL REAL TIME =      0.610

TOTAL CPU TIME  =      0.020
```

```
FIRST WORD OF EACH RECORD
     1 *    72001      1901       1601       1701       1601 *     1501       1401       1301       1201      71101
    11 *     1001       901        601        701        601 *      501        401        301        201      70101

TIME SUMMARY (SECONDS) FOR SECTION    6

                  1          2          3          4          5          6          7          8          9         10
                 OPEN    COMPUTE       TEST    RD/WRIT      CHECK       NOTE   RWND/PASS     FORSPC      BKSPC      POINT
REAL TIME=      0.0        0.0        0.0        0.600      0.0        0.0        0.430      0.0        0.0        0.0
CPU TIME =      0.0        0.0        0.0        0.010      0.0        0.0        0.020      0.0        0.0        0.0

TOTAL REAL TIME =      1.030

TOTAL CPU TIME  =      0.030

TIME SUMMARY (SECONDS) FOR SECTION    7

                  1          2          3          4          5          6          7          8          9         10
                 OPEN    COMPUTE       TEST    RD/WRIT      CHECK       NOTE   RWND/PASS     FORSPC      BKSPC      POINT
REAL TIME=      0.0        0.0        0.0        0.040      0.560      0.0        0.350      0.0        0.0        0.0
CPU TIME =      0.0        0.0        0.0        0.020      0.0        0.0        0.010      0.0        0.0        0.0

TOTAL REAL TIME =      0.950

TOTAL CPU TIME  =      0.030

TIME SUMMARY (SECONDS) FOR SECTION    8

                  1          2          3          4          5          6          7          8          9         10
                 OPEN    COMPUTE       TEST    RD/WRIT      CHECK       NOTE   RWND/PASS     FORSPC      BKSPC      POINT
REAL TIME=      0.0        0.0        0.0        0.400      0.0        0.0        0.040      0.0        3.280      0.020
CPU TIME =      0.0        0.0        0.0        0.0        0.0        0.0        0.0        0.0        0.020      0.0

TOTAL REAL TIME =      3.740

TOTAL CPU TIME  =      0.020

RESULTS OF SPACING EXERCISE
     1 *    72001      70101      72001      72001      72001 *     72001      66937       1301       1201      71101
```

```
* * * PROBLEM BEGAN ON 10/29/75 (302) AT TIME 11 HRS. 40.20 MIN. * * *

TIME SUMMARY (SECONDS) FOR SECTION   2

                  1       2       3       4       5       6       7       8       9      10
                 OPEN  COMPUTE    TEST  RD/WRIT   CHECK    NOTE  RWND/PASS  FORSPC   BKSPC   POINT
REAL TIME=       0.0     0.0     0.0     0.580    0.0     0.0     0.150    0.0     0.0     0.0
CPU TIME =       0.0     0.0     0.0     0.0      0.0     0.0     0.0      0.0     0.0     0.0

TOTAL REAL TIME =      0.730

TOTAL CPU TIME  =      0.0

FIRST WORD OF EACH RECORD
    1 *    70101      201      301      401      501 *      601      701      801      901     1001
17501 *    71101     1201     1301     1401     1501 *     1601     1701     1801     1901    72001

TIME SUMMARY (SECONDS) FOR SECTION   3

                  1       2       3       4       5       6       7       8       9      10
                 OPEN  COMPUTE    TEST  RD/WRIT   CHECK    NOTE  RWND/PASS  FORSPC   BKSPC   POINT
REAL TIME=       0.0     0.0     0.0     0.580    0.0     0.0     0.0      0.0     0.0     0.0
CPU TIME =       0.0     0.0     0.0     0.0      0.0     0.0     0.0      0.0     0.0     0.0

TOTAL REAL TIME =      0.580

TOTAL CPU TIME  =      0.0

TIME SUMMARY (SECONDS) FOR SECTION   4

                  1       2       3       4       5       6       7       8       9      10
                 OPEN  COMPUTE    TEST  RD/WRIT   CHECK    NOTE  RWND/PASS  FORSPC   BKSPC   POINT
REAL TIME=       0.0     0.0     0.0     0.0      0.520   0.0     0.0      0.0     0.0     0.0
CPU TIME =       0.0     0.0     0.0     0.0      0.0     0.0     0.0      0.0     0.0     0.0

TOTAL REAL TIME =      0.520

TOTAL CPU TIME  =      0.0

FIRST WORD OF EACH RECORD
    1 *    70101      201      301      401      501 *      601      701      801      901     1001
17501 *    71101     1201     1301     1401     1501 *     1601     1701     1801     1901    72001

TIME SUMMARY (SECONDS) FOR SECTION   5

                  1       2       3       4       5       6       7       8       9      10
                 OPEN  COMPUTE    TEST  RD/WRIT   CHECK    NOTE  RWND/PASS  FORSPC   BKSPC   POINT
REAL TIME=       0.0     0.010   0.0     0.0      2.470   0.0     0.0      0.0     0.0     0.010
CPU TIME =       0.0     0.0     0.0     0.0      0.0     0.0     0.0      0.0     0.0     0.0

TOTAL REAL TIME =      2.490

TOTAL CPU TIME  =      0.0
```

	1 OPEN	2 COMPUTE	3 TEST	4 RD/WRIT	5 CHECK	6 NOTE	7 RWND/PASS	8 FORSPC	9 BKSPC	10 POINT
REAL TIME=	0.600	0.0	0.0	5.500	0.0	0.0	0.0	0.0	0.0	0.0
CPU TIME =	0.020	0.0	0.0	0.080	0.0	0.0	0.0	0.0	0.0	0.0

TOTAL REAL TIME = 6.100

TOTAL CPU TIME = 0.100

```
FIRST WORD OF EACH RECORD
    1 *    70101      201      301      401      501 *      601      701      801      901     1001
 1750 *    71101     1201     1301     1401     1501 *     1601     1701     1801     1901    72001

TIME SUMMARY (SECONDS) FOR SECTION    5

                  1        2        3        4        5        6        7        8        9       10
                 OPEN  COMPUTE     TEST  RD/WRIT    CHECK     NOTE  RWND/PASS   FORSPC    BKSPC    POINT
REAL TIME=      0.0      0.0    5.340      0.0      0.0      0.0      0.0      0.0      0.0      0.0
CPU TIME =      0.0    0.020    0.010      0.0      0.0      0.0      0.0      0.0      0.0      0.0

TOTAL REAL TIME =      5.340

TOTAL CPU TIME  =      0.030

FIRST WORD OF EACH RECORD
    1 *    72001     1901     1801     1701     1601 *     1501     1401     1301     1201    71101
   11 *     1001      901      801      701      601 *      501      401      301      201    70101

TIME SUMMARY (SECONDS) FOR SECTION    6

                  1        2        3        4        5        6        7        8        9       10
                 OPEN  COMPUTE     TEST  RD/WRIT    CHECK     NOTE  RWND/PASS   FORSPC    BKSPC    POINT
REAL TIME=      0.0      0.0      0.0    0.770      0.0      0.0    0.350      0.0      0.0      0.0
CPU TIME =      0.0      0.0      0.0      0.0      0.0      0.0    0.020      0.0      0.0      0.0

TOTAL REAL TIME =      1.120

TOTAL CPU TIME  =      0.020

TIME SUMMARY (SECONDS) FOR SECTION    7

                  1        2        3        4        5        6        7        8        9       10
                 OPEN  COMPUTE     TEST  RD/WRIT    CHECK     NOTE  RWND/PASS   FORSPC    BKSPC    POINT
REAL TIME=      0.0      0.0      0.0      0.0    0.530      0.0    0.370      0.0      0.0      0.0
CPU TIME =      0.0      0.0      0.0      0.0      0.0      0.0    0.030      0.0      0.0      0.0

TOTAL REAL TIME =      0.900

TOTAL CPU TIME  =      0.030

TIME SUMMARY (SECONDS) FOR SECTION    8

                  1        2        3        4        5        6        7        8        9       10
                 OPEN  COMPUTE     TEST  RD/WRIT    CHECK     NOTE  RWND/PASS   FORSPC    BKSPC    POINT
REAL TIME=      0.0      0.0      0.0    0.250      0.0      0.0    3.070    2.610      0.0      0.0
CPU TIME =      0.0      0.0      0.0      0.0      0.0      0.0    0.020      0.0      0.0      0.0

TOTAL REAL TIME =      5.930

TOTAL CPU TIME  =      0.020

RESULTS OF SPACING EXERCISE
    1 *    72001    72001    72001    72001    72001 *    72001    66937     1301     1201    71101

TIME SUMMARY (SECONDS) FOR SECTION    9
```

```
* * * PROBLEM BEGAN ON 10/29/75 (302) AT TIME 11 HRS. 39.78 MIN. * * *

TIME SUMMARY (SECONDS) FOR SECTION   1

                  1       2       3       4       5       6       7       8       9      10
                 OPEN  COMPUTE    TEST  RD/WRIT  CHECK    NOTE  RWND/PASS  FORSPC   BKSPC   POINT
REAL TIME=      1.460   0.060   0.0     0.0     1.490   0.0     0.670   0.0     0.0     0.0
CPU TIME =      0.050   0.020   0.0     0.010   0.010   0.0     0.020   0.0     0.0     0.0

TOTAL REAL TIME =      3.680

TOTAL CPU TIME  =      0.110

NOTE ARRAY
    1 * 000000FF  00000100  00010100  00020100  00030100 * 00040100  00050100  00060100  00070100  00080100
   11 * 00090100  000A0100  000B0100  000C0100  000D0100 * 000E0100  000F0100  00100100  00110100  00120100

TIME SUMMARY (SECONDS) FOR SECTION   2

                  1       2       3       4       5       6       7       8       9      10
                 OPEN  COMPUTE    TEST  RD/WRIT  CHECK    NOTE  RWND/PASS  FORSPC   BKSPC   POINT
REAL TIME=      0.0     0.0     0.0     0.560   0.0     0.0     0.170   0.0     0.0     0.0
CPU TIME =      0.0     0.0     0.0     0.020   0.0     0.0     0.0     0.0     0.0     0.0

TOTAL REAL TIME =      0.730

TOTAL CPU TIME  =      0.020

FIRST WORD OF EACH RECORD
    1 *      101      201      301      401      501 *      601      701      801      901     1001
 17501 *     1101     1201     1301     1401     1501 *     1601     1701     1801     1901     2001

TIME SUMMARY (SECONDS) FOR SECTION   3

                  1       2       3       4       5       6       7       8       9      10
                 OPEN  COMPUTE    TEST  RD/WRIT  CHECK    NOTE  RWND/PASS  FORSPC   BKSPC   POINT
REAL TIME=      0.0     0.020   0.0     0.560   0.0     0.0     0.0     0.0     0.0     0.020
CPU TIME =      0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.020

TOTAL REAL TIME =      0.600

TOTAL CPU TIME  =      0.020

TIME SUMMARY (SECONDS) FOR SECTION   4

                  1       2       3       4       5       6       7       8       9      10
                 OPEN  COMPUTE    TEST  RD/WRIT  CHECK    NOTE  RWND/PASS  FORSPC   BKSPC   POINT
REAL TIME=      0.0     0.0     0.0     0.0     0.520   0.0     0.0     0.0     0.0     0.0
CPU TIME =      0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0

TOTAL REAL TIME =      0.520

TOTAL CPU TIME  =      0.0
```

```
COMPILER OPTIONS - NAME=  MAIN,OPT=00,LINECNT=60,SIZE=0000K,
                   SOURCE,EBCDIC,NOLIST,NODECK,LOAD,NOMAP,NOEDIT,ID,NOXREF
                C - - - - - FTEST PROGRAM TO DEMONSTRATE FBSAM PACKAGE                FTST0010
                C           W.A.RHOADES, ORNL (02/28/75)                              FTST0020
ISN 0002                COMMON /COMSAM/ LTRACK,NCTEP,KKSTA                            FTST0025
ISN 0003                COMMON /COMN/ K(540,2), NOTE(100), MOTE(100), Z(146000)       FTST0030
ISN 0004                DIMENSION LUN1(2), LUN2(2), L(8190,2)                         FTST0040
ISN 0005                LOGICAL*1 Z                                                   FTST0050
ISN 0006                EQUIVALENCE (K(1,1),K1), (K(1,2),K2), (L(1,1),Z(1))           FTST0060
ISN 0007                DATA NLOOP, NPASS, NERO /100,3,000/, LUN1, LUN2 /1,0,2,0/     FTST0070
                      1 ,   NBLMX, NBLREC, LBLMX /99,01,07000/, NBT /140000/          FTST0080
                C                                                                     FTST0090
                C - - - - -  NBLMX = MAX NO. BLOCKS PERMITTED WITH SINGLE CALL        FTST0100
                C            MUST BE .LT. 100                                         FTST0110
                C           NBLREC = MAX NO. BLOCKS PER LOGICAL RECORD                FTST0120
                C            LBLMX = LONGEST BLOCK PERMITTED                          FTST0130
                C              NBT = TOTAL NO. BYTES PROCESSED                        FTST0140
                C             NREC = NO. LOGICAL RECORDS PROCESSED                    FTST0150
                C             LREC = LENGTH OF LOGICAL RECORD, BYTES                  FTST0160
                C            LRECW = LENGTH OF LOGICAL RECORD, WORDS                  FTST0170
                C              NWD = TOTAL NO. WORDS PROCESSED                        FTST0180
                C           LTRACK = TRACK LENGTH                                     FTST0190
                C - - - NBT/LREC SHOULD BE AN INTEGER                                 FTST0205
                C - - - IF NBLREC.NE.1, POINT OPERATIONS GIVE ANOMALOUS RESULTS       FTST0200
                C                                                                     FTST0210
ISN 0008                NREC = (NBT-1) / (LBLMX*NBLREC) + 1                           FTST0220
ISN 0009                LREC = (NBT-1) / NREC + 1                                     FTST0230
ISN 0010                LRECW =(LREC-1)/4 + 1                                         FTST0240
ISN 0011                NWD = (NBT-1)/4 + 1                                           FTST0250
ISN 0012                CALL FBSAM (LUN1,K,-1,0,0)                                    FTST0260
ISN 0013                NERC = NERO                                                   FTST0270
ISN 0014                IF(LBLMX.GT.LTRACK) NERC = 1                                  FTST0280
ISN 0016                DO 900 MPASS=1,NPASS                                          FTST0290
ISN 0017                M = 1                                                         FTST0300
ISN 0018                N = 2*MOD(MPASS+M,2)                                          FTST0310
                C - - - - - M=1/2 ON ALTERNATE HALVES OF A PASS, BYPASSES WAIT IN SCN. 5FTST0320
                C           N=0,2,2,0,0,2 ON ALTERNATE HALVES OF A 3-PASS PROBLEM     FTST0322
                C           N CONTROLS OPENING MODE, REPOSITIONING FOR AND EXECUTION OF FTST0324
                C           SECTION 8                                                 FTST0326
ISN 0019                LUN1(2) = NERC + N                                            FTST0330
ISN 0020                CALL TIMSET                                                   FTST0340
ISN 0021                CALL TIMSTR(0)                                                FTST0350
                C - - - - - 1. FILL ARRAY WITH INTEGERS, WRITE ON LUN1, AND REFILL ARRAYFTST0360
                C             WITHOUT WAITING.                                        FTST0370
ISN 0022             10 CALL TIMON(1)                                                 FTST0380
ISN 0023             15 CALL FBSAM (LUN1, K, LBLMX, NBLMX, LUN1(2) )                  FTST0390
ISN 0024                CALL TIMON(2)                                                 FTST0400
ISN 0025                NIP = NREC + 1                                                FTST0410
ISN 0026                NK = 1                                                        FTST0420
ISN 0027                DO 18 I=1,NIP                                                 FTST0430
ISN 0028                CALL TIMON(4)                                                 FTST0440
ISN 0029                IF(I.GT.1) CALL FWRITE (K,L(1,3-NK),LREC)                     FTST0450
ISN 0031                CALL TIMON(2)                                                 FTST0460
ISN 0032                IF(I.LT.NIP) CALL SETR (L(1,NK),LRECW,LRECW,I)                FTST0470
ISN 0034                IF(I.EQ.1) GO TO 18                                           FTST0480
ISN 0036                CALL TIMON(5)                                                 FTST0490
ISN 0037                CALL FCHECK(K)                                                FTST0500
ISN 0038                CALL TIMON(6)                                                 FTST0510
```

```
ISN 0039          CALL FNOTE (K,2,NOTE(I-1),NBLP,LRECP,LEOFP)                FTST0520
ISN 0040       18 NK = 3 - NK                                                FTST0530
ISN 0041          CALL TIMON(7)                                              FTST0540
ISN 0042          CALL FREW(K)                                               FTST0550
ISN 0043          CALL TIMSUM                                                FTST0560
ISN 0044          CALL SKPEDZ (NOTE,NREC,1,'0NOTE ARRAY',11)                 FTST0570
            C                                                                FTST0580
            C - - - - - - 2.  READ ALL DATA AS A SUPER-RECORD                FTST0590
ISN 0045       20 CALL TIMON(4)                                              FTST0600
ISN 0046          CALL FRDCH(K,L,NBT)                                        FTST0610
ISN 0047          CALL TIMON(7)                                              FTST0620
ISN 0048          CALL FREW(K)                                               FTST0630
ISN 0049          CALL TIMSUM                                                FTST0640
ISN 0050          CALL SKPEDI (L,NWD,LRECW,'0FIRST WORD OF EACH RECORD',26)  FTST0650
            C                                                                FTST0660
            C - - - - - - 3. REVISE LAST, MIDDLE, AND FIRST BLOCKS           FTST0670
ISN 0051       30 MOTE(1) = NBT/LREC                                         FTST0680
ISN 0052          MOTE(2) = MOTE(1)/2 + 1                                    FTST0690
ISN 0053          MOTE(3) = 1                                                FTST0700
ISN 0054          DO 32 I=1,3                                                FTST0730
ISN 0055          CALL TIMON(10)                                             FTST0740
ISN 0056          CALL FPOINT (K,NOTE(MOTE(I)))                              FTST0750
ISN 0057          CALL TIMON(2)                                              FTST0760
ISN 0058          CALL SETR (L,LRECW,LRECW,MOTE(I)+700)                      FTST0770
ISN 0059          CALL TIMON(4)                                              FTST0780
ISN 0060       32 CALL FWRCH (K,L,LREC)                                      FTST0790
            C     WARNING - - - USE OF KTCLOS AFTER WRITE PRODUCES A FILEMARK FTST0798
ISN 0061          CALL TIMON(10)                                             FTST0800
ISN 0062          CALL FPOINT (K,NOTE)                                       FTST0810
ISN 0063       34 CALL TIMSUM                                                FTST0820
            C                                                                FTST0830
            C - - - - - - 4.  READ ALL DATA AS A SUPER-RECORD                FTST0840
ISN 0064       40 CALL TIMON(4)                                              FTST0850
ISN 0065          CALL FREAD (K,L,NBT)                                       FTST0860
ISN 0066       43 CALL TIMON(5)                                              FTST0870
ISN 0067          CALL FCHECK (K)                                            FTST0880
ISN 0068          CALL TIMSUM                                                FTST0890
ISN 0069          CALL SKPEDI (L,NWD,LRECW,'0FIRST WORD OF EACH RECORD',26)  FTST0900
            C                                                                FTST0910
            C - - - - - - 5. READ ARRAY BACKWARD, STORING RESULTS            FTST0920
            C                AND TESTING FOR COMPLETION                      FTST0930
ISN 0070       50 MOTE1 = MOTE(1)                                            FTST0940
ISN 0071          MOTE2 = MOTE(2)                                            FTST0950
ISN 0072          NK = 1                                                     FTST0960
ISN 0073          DO 52 I=1,MOTE1                                            FTST0970
ISN 0074          CALL TIMON(10)                                             FTST0980
ISN 0075          CALL FPOINT (K,NOTE(MOTE1-I+1) )                           FTST0990
ISN 0076          CALL TIMON(4)                                              FTST1000
ISN 0077          CALL FREAD (K,L(1,NK),LREC)                                FTST1010
ISN 0078       54 DO 56 J=1,NLOOP                                            FTST1020
            C - - - - - - ACTIVITIES CAN BE CARRIED OUT UNTIL I/O IS COMPLETED FTST1030
            C     SKIP ON A PASS (M.GT.1)                                    FTST1040
ISN 0079          IF(M.GT.1) GO TO 55                                        FTST1050
ISN 0081          CALL TIMON(3)                                              FTST1060
ISN 0082          CALL FTEST (K,LL)                                          FTST1070
ISN 0083       56 IF(LL.GT.0) GO TO 55                                       FTST1080
ISN 0085       55 CALL TIMON(5)                                              FTST1090
ISN 0086          CALL FCHECK(K)                                             FTST1100
```

```
ISN 0087          CALL TIMON(2)                                          FTST1110
ISN 0088          NOTE(I )= L(1,  NK)                                    FTST1120
ISN 0089       52 CONTINUE                                               FTST1130
ISN 0090          CALL TIMON(10)                                         FTST1140
ISN 0091          CALL FPOINT (K,NOTE(1))                                FTST1150
ISN 0092          CALL TIMSUM                                            FTST1160
ISN 0093          CALL SKPEDI(MOTE,MOTE1,1,'0FIRST WORD OF EACH RECORD',26)  FTST1170
            C                                                            FTST1180
            C - - - - - - 6. WRITE ALL DATA AS A SUPER-RECORD            FTST1190
ISN 0094       60 CALL TIMON(4)                                          FTST1200
ISN 0095          CALL FWRCH (K,L,NBT)                                   FTST1210
ISN 0096          CALL TIMON(7)                                          FTST1220
ISN 0097          CALL FREW(K)                                           FTST1230
ISN 0098       68 CALL TIMSUM                                            FTST1240
            C                                                            FTST1250
            C - - - - - - 7. WRITE ALL DATA AS A SUPER-RECORD            FTST1260
ISN 0099       70 CALL TIMON(4)                                          FTST1270
ISN 0100          CALL FWRITE (K,L,NBT)                                  FTST1280
ISN 0101       73 CALL TIMON(5)                                          FTST1290
ISN 0102          CALL FCHECK (K)                                        FTST1300
ISN 0103          CALL TIMON(7)                                          FTST1310
ISN 0104          IF(N.EQ.0) CALL FREW(K)                                FTST1320
ISN 0106          IF(N.GT.0) CALL FPASS(K)                               FTST1330
ISN 0108          CALL TIMSUM                                            FTST1340
            C                                                            FTST1350
            C - - - - - - 8. MOVE TO LAST BLOCK, DEMONSTRATE SPACING ACCORDING TO M  FTST1360
ISN 0109       80 CALL TIMON(8)                                          FTST1370
ISN 0110          IF(LREC.GT.LTRACK) GO TO 86                            FTST1380
ISN 0112          IF(N.EQ.0) CALL FORSPC (K,MOTE1-1)                     FTST1400
ISN 0114          CALL TIMON(0)                                          FTST1410
ISN 0115          IF(N.GT.0) CALL FBKSPC (K,1)                           FTST1420
ISN 0117          NRD = LREC                                             FTST1430
ISN 0118          NRD = 18                                               FTST1440
ISN 0119       82 CALL TIMON(4)                                          FTST1460
            C - - - - - - READ  LAST RECORD                              FTST1470
ISN 0120          CALL FRDCH (K,MOTE(1),NRD)                             FTST1480
ISN 0121          IF(N.GT.0) GO TO 83                                    FTST1485
            C      WARNING - - - BACKSPACE AFTER READ OF END OF LAST RECORD,  FTST1495
            C                    ALSO, POINT FOLLOWED BY BACKSPACE HAS NO EFFECT  FTST1497
ISN 0123          CALL TIMON(10)                                         FTST1498
ISN 0124          CALL FPOINT (K,NOTE(MOTE1))                            FTST1510
ISN 0125          GO TO 84                                               FTST1515
ISN 0126       83 CALL TIMON(9)                                          FTST1520
ISN 0127          CALL FBKSPC (K,MOTE1)                                  FTST1530
ISN 0128       84 CALL TIMON(4)                                          FTST1540
            C - - - - - - READ FIRST RECORD                              FTST1550
ISN 0129          CALL FRDCH (K,MOTE(2),NRD)                             FTST1560
ISN 0130       85 CALL TIMON(10)                                         FTST1570
ISN 0131          IF(N.EQ.0) CALL FPOINT(K,NOTE(MOTE1))                  FTST1580
ISN 0133          IF(N.GT.0) CALL FPNTR(K,MOTE1)                         FTST1582
ISN 0135          CALL TIMON(4)                                          FTST1590
            C - - - - - - READ  LAST RECORD                              FTST1600
ISN 0136          CALL FRDCH (K,MOTE(3),NRD)                             FTST1610
ISN 0137       86 CALL TIMON(7)                                          FTST1620
ISN 0138          CALL FDISP (K)                                         FTST1630
ISN 0139          CALL TIMSUM                                            FTST1640
ISN 0140          CALL SKPEDI (MOTE,3,1,'0RESULTS OF SPACING EXERCISE',28)  FTST1650
            C                                                            FTST1660
```

```
                C - - - - - - 9. COPY DATA FROM LUN1 TO LUN2, AND USE LUN2 FOR THE TESTS  FTST1670
   ISN 0141          90 IF(M.GT.1) GO TO 900                                             FTST1690
   ISN 0143             M = M + 1                                                        FTST1700
   ISN 0144             N = 2*MOD(MPASS+M,2)                                             FTST1710
   ISN 0145             LUN2(2) = NERC + N                                               FTST1720
   ISN 0146             CALL TIMON(4)                                                    FTST1730
   ISN 0147             CALL FCOPY (K, L, LREC, LUN1, LUN2)                              FTST1740
   ISN 0148             DO 91 I=1,2                                                      FTST1760
   ISN 0149             LUN = LUN1(I)                                                    FTST1770
   ISN 0150             LUN1(I) = LUN2(I)                                                FTST1780
   ISN 0151          91 LUN2(I) = LUN                                                    FTST1790
   ISN 0152          92 CALL TIMON(1)                                                    FTST1800
   ISN 0153             CALL FBSAM (LUN1, K, LBLMX, NBLMX, LUN1(2) )                     FTST1810
   ISN 0154             CALL TIMSUM                                                      FTST1820
   ISN 0155             CALL TIMSET                                                      FTST1830
   ISN 0156             GO TO 20                                                         FTST1840
   ISN 0157         900 CONTINUE                                                         FTST1850
   ISN 0158             STOP 1                                                           FTST1870
   ISN 0159             END                                                              FTST1880

*OPTIONS IN EFFECT*      NAME=  MAIN,OPT=00,LINECNT=60,SIZE=0000K,

*OPTIONS IN EFFECT*      SOURCE,EBCDIC,NOLIST,NODECK,LOAD,NOMAP,NOEDIT,ID,NOXREF

*STATISTICS*     SOURCE STATEMENTS =      158 ,PROGRAM SIZE =      3608

*STATISTICS*  NO  DIAGNOSTICS GENERATED

****** END OF COMPILATION ******                                  105K BYTES OF CORE NOT USED
```

```
COMPILER OPTIONS - NAME=  MAIN,OPT=00,LINECNT=60,SIZE=0000K,
                   SOURCE,EBCDIC,NOLIST,NODECK,LOAD,NOMAP,NOEDIT,ID,NOXREF
ISN 0002          SUBROUTINE FBSAM (LUN,K,LBL,NBL,NERC)                    FBSM0010
            C - - - FORTRAN - BSAM I/O LINK FOR DIRECT I/O UNDER PROGRAMMER CONTROL.FBSM0020
            C       W.RHOADES, ORNL, 03/10/72                              FBSM0030
            C - - - NDD = 8-BYTE DDNAME FOR DATA SET TO BE OPENED (LEFT ADJUSTED).FBSM0040
            C       LUN = LOGICAL UNIT NUMBER                              FBSM0045
            C       K   = 740 + 5*NBL) - WORD ARRAY FOR STORING CONTROL AND STATUS FBSM0050
            C             INFORMATION FOR THIS DATA SET.                   FBSM0060
            C       LBL = BLOCK LENGTH IN BYTES, LBL.LE.32760.             FBSM0070
            C             = 0 DEFAULTS TO MAXIMUM TRACK LENGTH             FBSM0071
            C             .LT.0 SETS LTRACK AND RETURNS                    FBSM0072
            C       NBL = MAX. NO. BLOCKS PER LOGICAL RECORD.              FBSM0080
            C       NERC= 0/100  END OF FILE NOT ALLOWED / ALLOWED.        FBSM0090
            C             + 0/10   OPEN FOR OUTPUT-INPUT / UPDATE          FBSM0092
            C             +0/1/2/3 RECFM=U/UT/F/FT                         FBSM0094
            C                                                              FBSM0100
ISN 0003          COMMON /COMSAM/ LTRACK,NOTEP,KKSTA                       FBSM0110
ISN 0004          DIMENSION NDD(2),K(45),Y(5),IOPTNS(2)                    FBSM0115
ISN 0005          INTEGER LTT*2(2)                                         FBSM0120
ISN 0006          LOGICAL*1 X(1), NFOUR(4), KFOUR(4), ZERO, RECFM(4)       FBSM0122
ISN 0007          EQUIVALENCE (XX,IX)                                      FBSM0123
ISN 0008          EQUIVALENCE (NONE,NFOUR(1)), (KONE,KFOUR(1))             FBSM0124
ISN 0009          DATA KONE, KTWO / 'FT00', 'F001' /, ZERO / ZF0 /, KLOS /'CLOS'/ FBSM0126
ISN 0010          DATA KSTA,KERO,KERC,KERB,KERF,NOTE1,NOTE2,KLL /3,4,5,6,7,8,9,10/ FBSM0130
                 1 ,   LBLK,NBLK,NIO,NSTA,NERR,LIMK,LERC /11,12,13,14,15,16,17/ FBSM0140
                 2 ,   KDCB,KDECB /18,41/                                  FBSM0150
ISN 0011          DATA NOU,NE   /6,18/, IOPTNS /55,52/, RECFM /ZC0,ZE0,Z80,ZA0/ FBSM0160
            C                                                              FBSM0162
            C - - - OPENS DATA SET FOR EITHER READING OR WRITING           FBSM0163
ISN 0012        2 NONE = LUN / 10                                          FBSM0165
ISN 0013          KFOUR(3) = NFOUR(4) + ZERO                               FBSM0166
ISN 0014          NONE = LUN - NONE*10                                     FBSM0168
ISN 0015          KFOUR(4) = NFOUR(4) + ZERO                               FBSM0170
ISN 0016        3 IF(K(2).NE.KTWO) GO TO 4                                 FBSM0171
ISN 0018          K(2) = K(1)                                              FBSM0172
ISN 0019          K(1) = KONE                                              FBSM0173
ISN 0020          GO TO 970                                                FBSM0174
ISN 0021        4 K(2) = KTWO                                              FBSM0175
ISN 0022          K(1) = KONE                                              FBSM0176
ISN 0023          GO TO 1                                                  FBSM0177
            C                                                              FBSM0178
ISN 0024          ENTRY      FUPEN (NDD,K,LBL,NBL,NERC)                    FBSM0180
ISN 0025          K(1) = NDD(1)                                            FBSM0182
ISN 0026          K(2) = NDD(2)                                            FBSM0190
ISN 0027        1 LIM  = 40 + 5*NBL                                        FBSM0200
ISN 0028          DO 5 I=3,LIM                                             FBSM0210
ISN 0029        5 K(I) = -16                                               FBSM0220
ISN 0030          CALL KBLCK (K,1,I,LTRACK)                                FBSM0222
ISN 0031          IF(LBL.LT.0) GO TO 92                                    FBSM0224
ISN 0033       11 K(KSTA) = 1                                              FBSM0230
ISN 0034          K(NBLK) = NBL                                            FBSM0245
ISN 0035          K(LBLK) = LBL                                            FBSM0246
ISN 0036          IF(LBL.EQ.0)  K(LBLK) = LTRACK                           FBSM0247
ISN 0038       10 CALL KOPEN ( K(KDCB),IOPTNS(MOD(NERC,100)/10+1),K,9252,K(KERO), FBSM0260
                1  K(NBLK),K(LBLK),RECFM(MOD(NERC,2)+1) )                 FBSM0262
ISN 0039       12 K(LIMK) = LIM                                            FBSM0290
ISN 0040          K(LERC) = NERC                                           FBSM0300
```

```
ISN 0041          K(NOTE2) = 255                                          FBS 0305
ISN 0042          IF(K(KERO).GT.0) GO TO 970                              FBSM0310
ISN 0044          GO TO 980                                               FBSM0320
            C                                                             FBSM0330
ISN 0045          ENTRY FWRITE (K,X,N)                                    FBSM0340
            C - - - WRITES N BYTES FROM ARRAY X INTO DATA SET CONTROLLED BY K.  FBSM0350
ISN 0046       20 K(KSTA) = 2                                             FBSM0360
ISN 0047          GO TO 22                                                FBSM0370
ISN 0048          ENTRY FREAD (K,X,N)                                     FBSM0380
            C - - - READS N BYTES FROM DATA SET CONTROLLED BY K INTO ARRAY X.  FBSM0390
ISN 0049          K(KSTA) = 3                                             FBSM0400
ISN 0050       22 IF(K(NSTA).NE.1 .AND. K(NSTA).NE.4 .AND. K(NSTA).NE.5) GO TO 960  FBSM0410
ISN 0052          K(NERR) = 0                                             FBSM0420
ISN 0053          K(NIO) = (N-1) / K(LBLK) + 1                            FBSM0430
ISN 0054          IF(K(NIO).GT.K(NBLK)) GO TO 970                         FBSM0440
ISN 0056          KLBLK = K(LBLK)                                         FBSM0450
ISN 0057          IE = KDECB                                              FBSM0460
ISN 0058          INTRY = K(KSTA)                                         FBSM0470
ISN 0059          DO 24 I=1,N,KLBLK                                       FBSM0480
ISN 0060          LL = MAX0 (NE,MIN0(KLBLK,N+1-I))                        FBSM0490
ISN 0061          IF(MOD(K(LERC),10).GT.1 .AND. K(KSTA).EQ. 2) LL=MAX0(LTRACK,KLBLK)FBSM0495
ISN 0063          IF(K(KSTA).EQ. 2) CALL KWRITE (K(IE),K(KDCB),X(I),LL)   FBSM0500
ISN 0065          IF(K(KSTA).EQ. 3) CALL KREAD  (K(IE),K(KDCB),X(I),LL)   FBSM0510
ISN 0067          K(NERR) = K(NERR) + LL                                  FBSM0515
ISN 0068       24 IE = IE + 5                                             FBSM0520
ISN 0069          GO TO 980                                               FBSM0530
            C                                                             FBSM0540
ISN 0070          ENTRY FWRCH (K,X,N)                                     FBSM0550
            C - - - WRITES N BYTES FROM X ARRAY ONTO DATA SET CONTROLLED BY K.  FBSM0560
            C         CHECKS AFTER EACH BLOCK.  NOTES AFTER FIRST AND LAST.  FBSM0570
ISN 0071       25 K(KSTA) = 74                                            FBSM0580
ISN 0072          GO TO 26                                                FBSM0590
ISN 0073          ENTRY FRDCH (K,X,N)                                     FBSM0600
            C - - - READS  N BYTES FROM X ARRAY ONTO DATA SET CONTROLLED BY K.  FBSM0610
            C         CHECKS AFTER EACH BLOCK.  NOTES AFTER FIRST AND LAST.  FBSM0620
ISN 0074          K(KSTA) = 84                                            FBSM0630
ISN 0075       26 IF(K(NSTA).NE.1 .AND. K(NSTA).NE.4 .AND. K(NSTA).NE.5) GO TO 960  FBSM0640
ISN 0077          K(NERR) = 0                                             FBSM0650
ISN 0078          K(NIO) = 0                                              FBSM0660
ISN 0079          K(KLL) = 0                                              FBSM0670
ISN 0080          K(KERC) = 0                                             FBSM0680
ISN 0081          IE = KDECB                                              FBSM0682
ISN 0082          KLBLK = K(LBLK)                                         FBSM0690
ISN 0083          DO 27 I=1,N,KLBLK                                       FBSM0700
ISN 0084          LL = MAX0 (NE,MIN0(KLBLK,N+1-I))                        FBSM0710
ISN 0085          IF(MOD(K(LERC),10).GT.1 .AND. K(KSTA).EQ.74) LL=MAX0(LTRACK,KLBLK)FBSM0715
ISN 0087          IF(K(KSTA).EQ.74) CALL KWRITE (K(IE),K(KDCB),X(I),LL)   FBSM0720
ISN 0089          IF(K(KSTA).EQ.84) CALL KREAD  (K(IE),K(KDCB),X(I),LL)   FBSM0730
ISN 0091          K(NERR) = K(NERR) + LL                                  FBSM0731
ISN 0092          IF(IE.LT.K(LIMK)-4 .AND. I.LT.N-KLBLK) GO TO 29         FBSM0732
ISN 0094          DO 34 J=KDECB,IE,5                                      FBSM0734
ISN 0095          CALL KCHECK (K( J),NER,LL)                              FBSM0740
ISN 0096          K(NIO) = K(NIO) + 1                                     FBSM0760
ISN 0097          K(KLL) = K(KLL) + LL                                    FBSM0770
ISN 0098          IF(NER.GT.0) K(KERC) = K(KERC) + 100**NER               FBSM0780
ISN 0100          IF(NER.GT.K(LERC)/100) GO TO 970                        FBSM0790
ISN 0102          IF(K(KERC).GT.0) GO TO 28                               FBSM0800
ISN 0104       34 CONTINUE                                                FBSM0804
```

```
ISN 0105          IE = KDECB                                                FBSM0806
ISN 0106          GO TO 27                                                  FBSM0807
ISN 0107       29 IE = IE + 5                                               FBSM0809
ISN 0108       27 CONTINUE                                                  FBSM0810
ISN 0109       28 K(NOTE1) = K(NOTE2)                                       FBSM0822
ISN 0110          CALL KNOTE (K(KDCB),K(NOTE2))                             FBSM0826
ISN 0111          IF(K(KLL).NE.K(NERR) .AND. K(LERC)/100.EQ.0) GO TO 970    FBSM0830
ISN 0113          GO TO 980                                                 FBSM0850
            C                                                               FBSM0860
ISN 0114          ENTRY FTEST (K,M)                                         FBSM0870
            C - - - CHECKS FOR COMPLETION OF I/O CONTROLLED BY K ARRAY.     FBSM0880
            C         N IS SET TO 1 IF I/O IS COMPLETED. OTHERWISE N=0.     FBSM0890
ISN 0115       31 K(KSTA) = 14                                              FBSM0900
ISN 0116       30 IF(K(NSTA).EQ.1 .OR. K(NSTA).EQ.4 .OR. K(NSTA).EQ.5) GO TO 36 FBSM0910
ISN 0118          IF(K(NSTA).LT.2 .OR. K(NSTA).GT.3) GO TO 960              FBSM0920
ISN 0120          M = 0                                                     FBSM0930
ISN 0121          IF(K(KDECB+5*NIO-5).EQ.0) GO TO 990                       FBSM0940
ISN 0123       36 M = 1                                                     FBSM0950
ISN 0124          GO TO 40                                                  FBSM0960
            C                                                               FBSM0970
ISN 0125          ENTRY FCHECK (K)                                          FBSM0980
            C - - - WAITS FOR COMPLETION OF ARRAY CONTROLLED BY K.          FBSM0990
ISN 0126       41 K(KSTA) = 4                                               FBSM1000
ISN 0127       40 K(KLL) = 0                                                FBSM1020
ISN 0128          K(KERC) = 0                                               FBSM1030
ISN 0129          IF(K(NSTA).EQ.1 .OR. K(NSTA).EQ.4 .OR. K(NSTA).EQ.5) GO TO 980 FBSM1040
ISN 0131          IF(K(NSTA).LT.2 .OR. K(NSTA).GT.3) GO TO 960              FBSM1050
ISN 0133          IE = KDECB                                                FBSM1060
ISN 0134          KNIO = K(NIO)                                             FBSM1070
ISN 0135          K(NIO) = 0                                                FBSM1080
ISN 0136          DO 44 I =1,KNIO                                           FBSM1090
ISN 0137          CALL KCHECK (K(IE),NER,LL)                                FBSM1100
ISN 0138          K(NIO) = K(NIO) + 1                                       FBSM1110
ISN 0139          K(KLL) = K(KLL) + LL                                      FBSM1120
ISN 0140          IF(NER.GT.0) K(KERC) = K(KERC) + 100**NER                 FBSM1130
ISN 0142          IF(NER.GT.K(LERC)/100) GO TO 970                          FBSM1140
ISN 0144       44 IE = IE + 5                                               FBSM1150
ISN 0145       45 K(NOTE1) = K(NOTE2)                                       FBSM1160
ISN 0146          CALL KNOTE (K(KDCB),K(NOTE2))                             FBSM1170
ISN 0147          IF(K(KLL).NE.K(NERR) .AND. K(LERC)/100.EQ.0) GO TO 970    FBSM1180
ISN 0149          GO TO 980                                                 FBSM1200
            C                                                               FBSM1210
ISN 0150          ENTRY FNOTE (K,M,NOTE,NWR,KWR,LOF)                        FBSM1220
            C - - - ENTERS INTO NOTE THE POSITION OF FIRST /  LAST BLOCK OF PRECEDINFBSM1230
            C         I/O CONTROLLED BY K, IF N = 1/2. NWR BECOMES NO. BLOCKS PROCESSEFBSM1240
            C         IN LAST I/O. KWR BECOMES NO. OF BYTES PROCESSED.      FBSM1250
            C         LOF = NO. OF FILE MARKS ENCOUNTERED.                  FBSM1260
ISN 0151       51 K(KSTA) = 24                                              FBSM1270
ISN 0152       50 IF(K(NSTA).LT.4) GO TO 960                                FBSM1280
ISN 0154          NOTE = K(NOTE1)                                           FBSM1290
ISN 0155          NWR = K(NIO)                                              FBSM1310
ISN 0156          KWR = K(KLL)                                              FBSM1320
ISN 0157          LOF = (K(KERC) + K(KERF) + 16) / 100                      FBSM1330
ISN 0158          GO TO 990                                                 FBSM1340
            C                                                               FBSM1350
ISN 0159          ENTRY FPNTR (K,NOTE)                                      FBSM1356
ISN 0160          K(KSTA) = 34                                              FBSM1358
ISN 0161          IF(MOD(K(LERC),10).LT.2) GO TO 970                        FBSM1360
```

```
ISN 0163          I  = ( NOTE-1 )*K(NIO)                                    FBSM1366
ISN 0164          CALL KBLCK ( K, K(LBLK), NBPT, LTRACK )                   FBSM1368
ISN 0165          LTT(1) = I /NBPT                                          FBSM1372
ISN 0166          LTT(2) = 256*( I -NBPT*LTT(1)+1 )                         FBSM1376
ISN 0167          CALL KPOINT ( K(KDCB), LTT )                              FBSM1380
ISN 0168          GO TO 978                                                 FBSM1384
            C                                                               FBSM1388
ISN 0169          ENTRY FPOINT (K,NOTE)                                     FBSM1395
ISN 0170          ENTRY FPNTSP (K,NOTE)                                     FBSM1400
            C - - - SETS I/O DEVICE TO BEGIN NEXT W/R AT NEXT RECORD BEYOND FBSM1410
            C         POSITION INDICATED BY POINT.                          FBSM1420
ISN 0171          K(KSTA) = 64                                              FBSM1430
ISN 0172       60 IF(K(NSTA).NE.1 .AND. K(NSTA).NE.4 .AND. K(NSTA).NE.5) GO TO 960 FBSM1440
ISN 0174       62 K(NOTE1) = K(NOTE2)                                       FBSM1442
ISN 0175          K(NOTE2) = NOTE                                           FBSM1443
ISN 0176          IF(K(NOTE2).LT.0) GO TO 970                               FBSM1447
ISN 0178          CALL KPOINT (K(KDCB),K(NOTE2)+1)                          FBSM1460
ISN 0179          GO TO 980                                                 FBSM1540
            C                                                               FBSM1550
ISN 0180          ENTRY FBKSPC (K,N)                                        FBSM1560
            C - - - BACKSPACES N BLOCKS.                                    FBSM1570
            C         WARNING * * * NOTES ARE USELESS AFTER BACKSPACE       FBSM1572
ISN 0181       66 K(KSTA) = 44                                              FBSM1580
ISN 0182          IF(K(NSTA).LT.4 .OR. K(NSTA).GT.5) GO TO 960              FBSM1590
ISN 0184          K(KERB) = 0                                               FBSM1600
ISN 0185          DO 64 I=1,N                                               FBSM1610
ISN 0186          CALL KBSP (K(KDCB),NER)                                   FBSM1620
ISN 0187       64 K(KERB) = K(KERB) + NER                                   FBSM1630
ISN 0188          IF(K(KERB).GT.0) GO TO 970                                FBSM1640
ISN 0190          GO TO 978                                                 FBSM1650
            C                                                               FBSM1660
ISN 0191          ENTRY FORSPC (K,N)                                        FBSM1670
            C - - - FORWARD SPACES N BLOCKS.                                FBSM1680
ISN 0192       71 K(KSTA) = 54                                              FBSM1690
ISN 0193       70 IF(K(NSTA).NE.1 .AND. K(NSTA).NE.4 .AND. K(NSTA).NE.5) GO TO 960 FBSM1700
ISN 0195          K(KLL) = 0                                                FBSM1710
ISN 0196          K(KERF) = 0                                               FBSM1720
ISN 0197          DO 78 I=1,N                                               FBSM1730
ISN 0198          CALL KREAD (K(KDECB),K(KDCB),Y,12)                        FBSM1740
ISN 0199          CALL KCHECK (K(KDECB),NER,LL)                             FBSM1750
ISN 0200          K(KLL) = K(KLL) + LL                                      FBSM1770
ISN 0201          IF(NER.GT.0) K(KERF) = K(KERF) + 100**NER                 FBSM1780
ISN 0203          IF(NER.GT.K(LERC)/100) GO TO 970                          FBSM1790
ISN 0205          IF(K(KERF).GT.0) GO TO 77                                 FBSM1800
ISN 0207       78 IE = IE + 5                                               FBSM1810
ISN 0208          I = N                                                     FBSM1820
ISN 0209       77 K(NIO) = I                                                FBSM1830
ISN 0210          K(NOTE1) = K(NOTE2)                                       KBSM1833
ISN 0211          CALL KNOTE (K(KDCB),K(NOTE2))                             FBSM1840
ISN 0212       75 GO TO 980                                                 FBSM1850
            C                                                               FBSM1860
ISN 0213          ENTRY FREW (K)                                            FBSM1870
            C - - - REPOSITIONS DATA SET CONTROLLED BY K AT BEGINNING OF DATA FBSM1880
ISN 0214       83 K(KSTA) = 15                                              FBSM1900
ISN 0215          K(NOTE1) = K(NOTE2)                                       FBSM1910
ISN 0216          K(NOTE2) = 255                                            FBSM1912
ISN 0217       84 LL = 144                                                  FBSM1918
ISN 0218          GO TO 80                                                  FBSM1920
```

```
ISN 0219          ENTRY FPASS (K)                                          FBSM1930
            C - - - REPOSITIONS DATA SET CONTROLLED BY K AT END OF DATA    FBSM1940
            C         END OF DATA SET.                                     FBSM1950
            C - - - WARNING***THIS WILL WRITE AN END OF FILE IF LAST OPERATION WAS  FBSM1952
            C                  A WRITE AND CHECK                           FBSM1954
ISN 0220       85 K(KSTA) = 25                                             FBSM1960
ISN 0221          LL = 176                                                 FBSM1970
ISN 0222       80 IF(K(NSTA).EQ.5) GO TO 980                               FBSM1980
ISN 0224          IF(K(NSTA).NE.1 .AND. K(NSTA).NE.4) GO TO 960            FBSM1990
ISN 0226          CALL KTCLOS (K(KOCB),LL)                                 FBSM2000
ISN 0227          IF(LL.EQ.176) GO TO 978                                  FBSM2008
ISN 0229          GO TO 980                                                FBSM2010
            C                                                              FBSM2020
ISN 0230          ENTRY FDISP (K)                                          FBSM2030
            C - - - CLOSES I/O ON DATA SET CONTROLLED BY K, AND DISPOSES OF DATA SET  FBSM2040
            C         ACCORDING TO DISP PARAMETER ON DD CARD.              FBSM2050
ISN 0231       91 K(KSTA) = 6                                              FBSM2060
ISN 0232       90 IF(K(NSTA).EQ.6) GO TO 980                               FBSM2070
ISN 0234          IF(K(NSTA).NE.1 .AND. K(NSTA).LT.4) GO TO 960            FBSM2080
ISN 0236          CALL KCLOSE (K(KOCB),128)                                FBSM2090
ISN 0237       92 K(2) = KLOS                                              KBSM2095
ISN 0238          GO TO 980                                                FBSM2100
            C                                                              FBSM2110
ISN 0239      960 KKSTA  =  K(KSTA) + 1000*K(NSTA)                         FBSM2120
ISN 0240          GO TO 974                                                FBSM2130
ISN 0241      970 KKSTA  = K(KSTA)                                         FBSM2140
ISN 0242      974 KLIMK = MIN0(1000,K(LIMK))                               FBSM2150
ISN 0243          WRITE (NOU,972)KKSTA,(K(I),I=1,2),(K(I),I=1,KLIMK)       FBSM2160
ISN 0244      972 FORMAT ( '0* * * ERROR IN FORTRAN - BSAM PACKAGE. CODE=', I20 /,  FBSM2170
                 1          '0        DDNAME = ', 2A4, '  K ARRAY EDIT = = =' //  FBSM2180
                 2          ( 2( 4X, 4Z12 ) )    )                        FBSM2190
ISN 0245          CALL ERRTRA                                              FBSM2200
ISN 0246      978 K(NOTE1) = -16                                           FBSM2204
ISN 0247          K(NOTE2) = -16                                           FBSM2206
ISN 0248      980 K(NSTA) = MOD(K(KSTA),10)                                FBSM2210
ISN 0249      990 RETURN                                                   FBSM2220
ISN 0250          END                                                      FBSM2230
```

```
*OPTIONS IN EFFECT*      NAME=  MAIN,OPT=00,LINECNT=60,SIZE=0000K,

*OPTIONS IN EFFECT*      SOURCE,EBCDIC,NOLIST,NODECK,LOAD,NOMAP,NOEDIT,ID,NOXREF

*STATISTICS*     SOURCE STATEMENTS =      249 ,PROGRAM SIZE =      9312

*STATISTICS*  NO  DIAGNOSTICS GENERATED

****** END OF COMPILATION ******                                  85K BYTES OF CORE NOT USED

*STATISTICS*  NO  DIAGNOSTICS THIS STEP
```

ORNL-TM-5199
UC-32

INTERNAL DISTRIBUTION

1-3. L. S. Abbott
4. A. A. Brooks
5. R. L. Childs
6. C. E. Clifford
7. G. W. Cunningham
8. W. W. Engle, Jr.
9. T. B. Fowler
10. W. B. Gardner
11. H. Goldstein (consultant)
12. N. M. Greene
13. F. C. Maienschein
14. F. R. Mynatt
15. J. V. Pace, III
16. L. M. Petrie
17. R. P. Rannie
18-31. W. A. Rhoades
32. D. E. Simpson
33. J. G. Sullivan
34. D. R. Vondy
35. J. E. White
36. G. E. Whitesides
37. A. Zucker
38. P. F. Fox (consultant)
39. W. W. Havens, Jr. (consultant)
40. A. F. Henry (consultant)
41-42. Central Research Library
43. ORNL Y-12 Technical Library
44-45. Laboratory Records Department
46. Laboratory Records RC
47. ORNL Patent Office
48-98. RSIC

EXTERNAL DISTRIBUTION

99. M. Gritzner, Science Applications, Inc., Clinton Building, Huntsville, Alabama 35805
100. USERDA Oak Ridge Operations, Research and Technical Support Division, P. O. Box E, Oak Ridge, Tennessee 37830: Director
101-258. For distribution as shown in TID-4500 Distribution Category UC-32, Mathematics and Computer.