3600 3800 COMPUTER SYSTEMS MSIO REFERENCE MANUAL

CONTROL DATA CORPORATION

3600 3800 COMPUTER SYSTEMS MSIO REFERENCE MANUAL

CONTROL DATA
CORPORATION

Revised—September, 1967
Pub. No. 60174800A

CONTENTS

INTRODUCTION

The 3600/3800 Mass Storage Input/Output system provides complete input/output control for mass storage devices. (Mass storage devices include all normally marketed disks installed on CONTROL DATA®3600 or 3800 computers.) The devices initially intended for implementation in this system are the 828 and 813/814 disk files. The system is designed to allow for the implementation of any new mass storage devices. At least two banks (65K) of memory are required for mass storage input/output.

All tape handling functions are performed by the 3600/3800 General Purpose Input/Output (GPIO) system. The Mass Storage Input/Output (MSIO) system consists of all routines necessary to perform input/output functions and record blocking and deblocking for mass storage devices. The GPIO system is called to perform input/output functions on physical devices other than mass storage.

This manual describes the MSIO features only; Appendix C contains a description of the areas where GPIO and MSIO are incompatible. A familiarity with the GPIO reference manual is advisable.

The MSIO environment includes the systems that use MSIO:

COBOL

DISK SORT III

and the system used by MSIO:

3600/3800 Drum SCOPE.

FILE STRUCTURE 1

Data placed on the mass storage devices is stored in logically related sets called files. A file consists of a set of logical records related to one subject or function. A logical record is a single unit of information, the smallest unit that may be specified by an I/O request. Logical records may be arranged in groups convenient for storage on a mass storage device. These groups, consisting of one or more logical records, are called physical records or blocks. Logical records may span blocks. The content, format, and inter-relationship of logical records, physical records, and files are defined by the user with macros and requests.

Before a file may be opened, it must be allocated space on a mass storage device. Space may be allocated in one or more segments where a segment is a contiguous section of disk storage. A device may contain one or more segments, but no segment may extend past the HI and LO limits assigned in the device label (1.4.1).

1.1 LOGICAL RECORDS

Logical records may be either fixed or variable in length and may be so described in five different ways. Any given file, however, must be composed of logical records described in the same manner.

1.1.1 FIXED LENGTH FORMAT

Logical records in this format are all of the same length.

1.1.2 VARIABLE LENGTH FORMAT

Logical records which vary in length may be defined in four ways:

Simple Variable Length Records

Logical records described in this manner require that the user define a key field which specifies the record length. This key field occupies the same position for all logical records in the file. Before the record is written, the value of the key field must be set to the number of 6-bit bytes in the logical record, including the key field.

Fixed Length with Trailer

This type of variable length logical record consists of a fixed length base and a variable number of fixed length trailer items. A key field is defined within the fixed portion of the logical record which specifies the number of trailer items. The key field occupies the same position within the record for every logical record in the file.

Record Mark

This type of variable length logical record is terminated by a record mark character (72_8) which is supplied by the user as a part of the record. The record mark character must occupy the last character position (bits 0-5) within a word.

Universal Record

With this type of variable length logical record, the user specifies an address which contains the length of the logical record as a write request parameter at the time the record is written. An MSIO record blocking routine prefixes each record with a 48-bit word containing a value equal to the number of 6-bit bytes in the record. This routine removes the prefix when the record is read. The prefix is carried in the buffer areas and not in the record area.

1.2 PHYSICAL RECORDS

When mass storage files are used, record blocks must be fixed in length and contain no preamble. An important consideration in specifying block size is the physical record limit for the device to be used. With the 828 disk, for example, the physical record limit is a 32-word sector. Therefore, a 33-word block would require two sectors on the disk. The next block begins with a new sector leaving 31 words associated with each block unused and unavailable. Each block is read with a single command.

Block sizes which are multiples of 32 words provide the most efficient disk usage; however, the user may prefer to define block size such that his records do not span blocks.

1.3 BUFFER AND RECORD AREAS

MSIO uses two types of intermediate storage areas in transferring data between storage devices and the computer: buffer areas and record areas.

A buffer area can contain one record block; a record area can contain one logical record. For variable length records, the record area must be large enough to contain the file's largest logical record. The buffer areas are defined by the user with the FILEDESC macro and the READFILE and RITEFILE requests. If both FILEDESC and READ/RITEFILE contain record area descriptions, READ/RITEFILE takes precedence.

Five variations are allowed on the number of record and buffer areas which may be defined:

- Two buffer areas and one record area—data is transferred from one buffer area to the record area/external device while the other buffer area is refilling. Records are blocked/deblocked as they are transferred to the external device/record area.
- Two buffer areas—data is transferred into/from one buffer area while records from the other area are being processed. Records are blocked and deblocked; however, records which span blocks and are larger than one block size may be partially inaccessible.
- One buffer area and one record area—records are blocked/deblocked as they are transferred to the external device/record area; but efficiency is reduced, since data from the buffer area must be completely emptied before the area can be refilled and transferred again.
- One buffer area—the user reserves space for one record block but no I/O buffering can take place. If a record spans blocks, the first portion of the record will not be available to the user.
- One record area—no blocking or deblocking is performed. Transmission to/from the external device is limited to the size of one block. If the file consists of records which are larger than the block size, the section of each record extending beyond the block size will be inaccessible.

Random and linked sequential accessing require the presence of at least one buffer area and one record area. With random accessing, MSIO cannot anticipate the next record address; therefore, no buffering will occur regardless of the number of buffer areas.

1.4 LABELS

MSIO uses two types of labels: device labels and file labels.

1.4.1 DEVICE LABELS

Each mass storage device has a label which uniquely identifies the device within the system. This label is written onto the device with a utility program when the system is installed.

The position of the device label is fixed for each type of mass storage equipment. The purpose of this label is to point to the label directory and to record and hold device allocation information.

Before any records on a file may be processed, space must be allocated for the file on the mass storage device. When allocation is to be made, the device label is searched for the smallest contiguous area which is large enough to satisfy the ALLOCATE request (3.1). If such an area does not exist, the largest available area is reserved as the first file segment, followed by the next largest and so on, unitl the request has been satisfied or the number of segments allocated exceeds the number specified in the request.

Each device has 18 allocatable areas; the HI, LO, next available sector, and unused sector are given for each such area. The size and type of area are installation dependent.

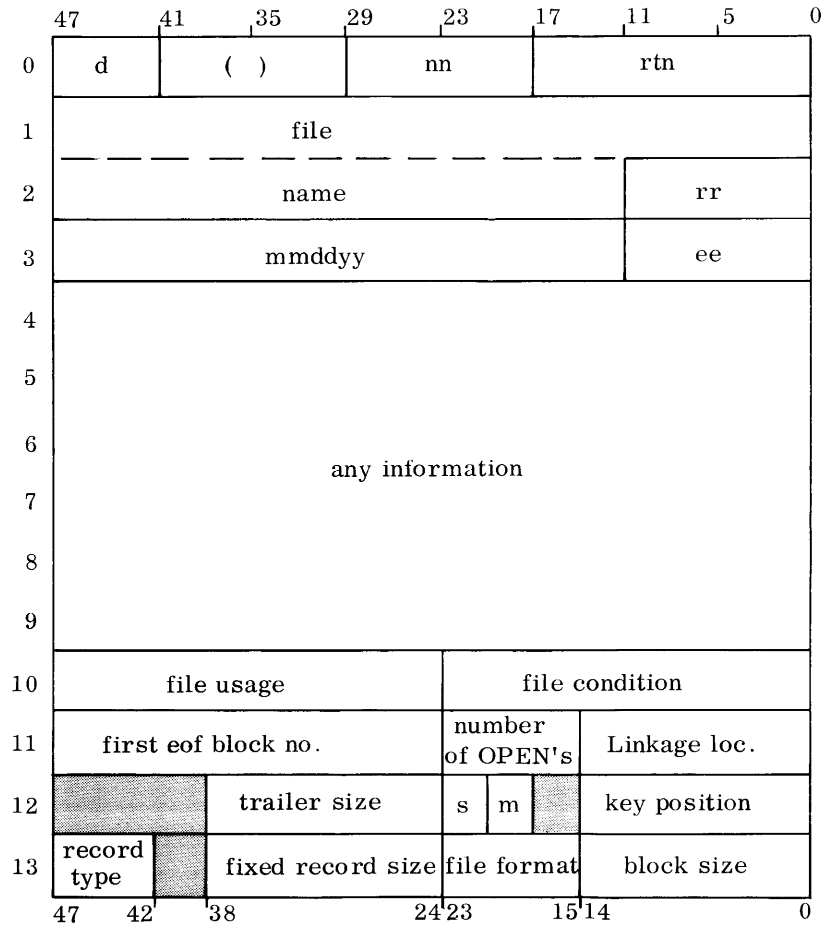
A device label contains the following fields:

Name	Function
Device name	Identifies device
Device address	Hardware address of device label
Directory address	Hardware address of label for label directory file
HI limit	Highest absolute address for each of 18 areas of device
LO limit	Lowest absolute address for each of 18 areas of device
Next available sector	Relative address of next available sector in each of 18 areas of device
Unused sectors	Number of released sectors in each of 18 areas of device

1.4.2 FILE LABELS

A file label identifies a file so that it may be referred to by input/output requests. When an OPENFILE request (3.2) is issued, MSIO refers to the label to locate the file and ready it for input/output operations. Labels for mass storage files are not attached to the file, but are stored in a directory file. MSIO allocates space and maintains the directory file. The user need not allow space for his file labels. A file label is required for all mass storage files.

File labels may be examined by the user through the after-beginning-label OWNCODE options (2.5). At which time, the label is in the user's record area (if available) or buffer area and may be changed. The first ten words describe tape files; the first 14 words describe mass storage files. (The whole label consists of 32 words but only the first 14 are available to the user; the remaining 18 words contain the security key, segment location information, and relevant SAK pointers used by MSIO.)


47 41 35 29 23 17 11 5 0
0 d ( ) nn rtn
1 file
2 name rr
3 mmddyy ee
4 5 6 7 8 9 any information
10 file usage file condition
11 first eof block no. number of OPEN's Linkage loc.
12 trailer size s m key position
13 record type fixed record size file format block size
47 42 38 24 23 15 14 0

The shaded areas of the File Label are not used.

FILE LABEL FIELDS

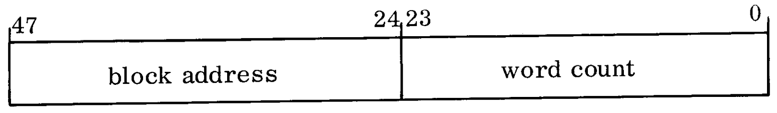
Word	Bit position	Code	Meaning
1	47-42	d	density; 2=200 bpi, 5=556 bpi, 8=800 bpi
1	41-30	( )	unique label identifier
1	29-18	nn	logical unit number; must be blank if file name is non-blank
1	17-0	rtn	retention code, 000 to 999. Number of days file is to be saved after date written. 999 specifies permanent retention; 000 specifies retention for the run/job only.
1,2	47-0 47-12	file name	14 alphanumeric characters; must be blank if nn is non-blank
2	11-0	rr	reel number, 01 to 99 (tape only)
3	47-12	mmddyy	date written; mm=month, dd=day, yy=year
3	11-0	ee	edition number, 01 to 99 or blank
10	47-24	file usage	cumulative count of the number of times the file has been opened. This count is reduced to zero when the file is re-ordered.
10	23-0	file condition	cumulative count of the number of record linkages
11	47-24	first EOF block number	relative block position of the end-of-file; if none, this field is set to zero
11	23-15	number of OPEN requests	number of FET's which indicate the file is currently open
11	14-0	linkage location	location of work in logical record that contains linkage information
12	38-24	trailer size	size of trailer portion, in 6-bit bytes
12	23-20	S	size of key field
12	19-18	M	mode of key (not used)
12	14-0	key position	position of key field within logical record
13	47-42	record type	type of fixed or variable-length logical records
13	38-24	fixed record size	size of fixed portion of logical record, in 6-bit bytes
13	23-15	file format	unused
13	14-0	block size	size of the physical record (block), in 6-bit bytes

1.5 FILE ENVIRONMENT TABLE

The File Environment Table (FET) describes each file processed by MSIO. It is constructed by MSIO at assembly time from user-submitted file description macros (Chapter 2). An entry for a file consists of 32 words in the format shown in Appendix A.

1.6 SEEK ADDRESS KEY

For each file, MSIO maintains several Seek Address Key (SAK) fields in the FET. These fields identify: the record currently in process, the next available record for writing, the previous record processed, the last record in the file, and the first record of the file. Each SAK field consists of a relative block address; and a word count field. A relative block address gives the position relative to the beginning of the file. For example, a file consisting of 25,000 blocks would have block addresses from 1 to 25,000. The word count field is relative to the beginning of the block; the first word in the block is 1. The format of the Seek Address Key is:



The SAK field is vital in determining the order in which records are accessed within a file. MSIO provides four types of file access: sequential, linked sequential, random, and sequentially indexed. Each record locating routine determines how many of the SAK fields it needs to keep.

1.7 FILE SECURITY

MSIO provides the facility to protect files and validate user requests to refer to files. Each user of a file must submit an access key with each request to use a file; this key is compared with a security key in the file label during OPENFILE processing. File access and security keys are determined at the time the system is installed; they supply the following options:

Reject	Access to this file is not permitted to this user
Read Only	Only read and position activities may be processed on this file by the user
Write Only	Only write and position activities may be processed on this file by this user

Read/Write	Both reading and writing activities processed, allowing this user to modify the data in this file
Master	Full access allowed this user for reading and writing this file and its associated file label data. Master access is required to alter file size (ALLOCATE).

A master access will be established for each user for the files he creates; an allocation request for a new file results in a master access.

1.8 MASSFILE

This Drum SCOPE control card is required for all jobs containing MSIO calls which refer to mass storage devices. The primary function of the MASSFILE card is to allocate memory for the file definition tables.

$^{7}_{9}$MASSFILE,k_1=d_1,k_2=d_2,. . . ,k_n=d_n

The k and d parameters may assume the values described below:

FILES=dd	This declaration is required; dd is a one- or two-digit decimal number defining the total number of files to be processed by the job. This parameter should be specified first and appear only once in any set of MASSFILE cards for a job.

The remaining declarations are optional and apply to one particular file. Each MASSFILE card may contain information about only one file; continuation cards are formed by repeating the FN declaration on every card of the set. Declarations on the MASSFILE card override those declared in the FET in the running program.

FN=dd	dd is a one- or two-digit decimal number defining an internal file number. Within a COBOL program all files are numbered consecutively from 1 through n as they appear in SELECT specifications within the ENVIRONMENT division. For COMPASS programs, the FN declaration is given as a parameter of the FILEDESC macro.
ID=name	name is an external file name or label identifier; it may contain from 1 to 14 characters, including blanks. A comma before the fourteenth character causes the filed to be completed with blanks.

EDT=dd	dd is a one- or two-digit decimal number defining a file edition number.
DATE=mmddyy	Date written; month, day, year; mmddyy must be six characters; for example, January 6, 1966 would be specified 010666.
DSP=SAVE	This declaration causes a file to have a retention code of zero, but to be retained until the end of the job before being released. If a file is not assigned a retention code and SAVE is not specified, the file is released at the end of the run.
DSP=ddd	ddd is a one- to three-digit decimal number defining the retention cycle for a file. 999 specifies permanent retention. Otherwise ddd is the number of days the file is to be saved from the date written. This declaration is given when creating the file.
BLOC=±num	num is a one- to eight-digit decimal number defining the number of blocks to be allocated. Plus indicates an addition to an existing allocation for the file, minus indicates reduction; omission of both indicates an initial allocation.
SEG=dd	dd is a one- or two-digit decimal number specifying how many segments allowed for the allocation. The maximum number of segments is 63; omission of the parameter, SEG=00, or any number larger than 63 causes the maximum to be used.

FILE DESCRIPTION MACROS 2

Label processing, record blocking and deblocking and other automatic features of MSIO require more information than can be contained in the calling sequence of each I/O request. To obtain this information, yet keep the I/O calling sequence a reasonable size, File Environment Tables (FET) are constructed.

The macros described below are used in constructing an FET. The FILEDESC macro reserves space for an FET and provides necessary information to be placed in the FET. FILEDESC is required and must appear first. LABELING is also required for mass storage files but may appear anywhere in the set of macros following FILEDESC. Additional macros are optional; when used, they must immediately follow FILEDESC but their order is unimportant.

2.1 FILEDESC

	10
loc	FILEDESC ($p_1, p_2, \ldots, p_{13}$)

loc File name which identifies the file; assigned to the FET entry and used in subsequent I/O requests.

p_1 Logical record length

Record type	Value
fixed length	number of 6-bit bytes
simple variable universal record mark	blank
trailer	number of 6-bit bytes in fixed portion of record

Legal values for this parameter are 17 to 32,760 or blank.

When used with GPIO, this parameter is the maximum logical record size in characters.

p_2 Record area; the address of an area in memory into which each record is to be moved for input or from which each record is to be obtained for output. If the READFILE and RITEFILE requests specify a record area, that area takes precedence over this parameter for the specific call.

p_3	Record block length; the length, in computer words, of the buffer area. This value must be large enough to accommodate one complete record block. Legal values are $3 \leq p \leq 4,095$.
p_4	Buffer address; the memory address of the first (or only) buffer area assigned to this file.
p_5	Alternate buffer address; the memory address of the alternate buffer assigned to this file, if any.
p_6	Mass storage indicator or logical unit number; 828 or 814 specify a disk file. Blank indicates any mass storage device may be chosen by MSIO for this file. For files stored in non-mass storage devices, this parameter indicates the logical unit number.
p_7	If p_6 indicates a mass storage device, p_7 indicates device name: 1-8 BCD characters = device name blank = device name unknown If p_6 indicates a magnetic tape unit, p_7 specifies recording density: LO = 200 bpi; HI = 556 bpi; HY = 800 bpi any other value = LO
p_8	For mass storage devices, this parameter specifies the internal file number. This parameter is required if MASSFILE cards (1.8) are used with this file. The internal file number must be unique for a scratch file that is to be closed and reopened. This number is the only check for reopening a scratch file since MSIO does not write scratch file labels on the disk. For magnetic tape units and low volume I/O devices, this parameter indicates recording mode: B binary other BCD
p_9	For mass storage files, this parameter indicates the type of logical record: F fixed length V simple variable length T fixed length with trailer fields R terminated by record mark U universal record For magnetic tape files, this parameter indicates the position of the file on a multi-file reel.
p_{10}	For mass storage devices, blank indicates standard error recovery procedures (Appendix B) will be used; any other value indicates standard error recovery procedures will be inhibited. For magnetic tape files, this parameter is unused.

p_{11}	For mass storage files, this parameter specifies the file security key (1.7). Key may be 1-8 BCD characters or blank.
	For magnetic tape files, this parameter is unused.
p_{12}	Optional file indicator; blank indicates a mandatory file; if MSIO cannot locate the file, SCOPE requests operator assignment of the file. Any other value indicates an optional file; if the file is not present, the first READFILE request will take the end-of-file exit.
	Optional files must have standard labels. This parameter is ignored for output files.
p_{13}	For mass storage files, this parameter specifies the location in memory of the routine used to locate a record on the file (Chapter 4).

RLOC.SEQ	sequential access by physical position
RLOC.LSQ	linked sequential access
RLOC.RND	random access using system randomizing
RLOC.SQI	sequential index access; index processed sequentially, file processed randomly according to SAK found as index entry

Any user-specified record locating routine may be used by specifying its entry point name.

For magnetic tape files, this parameter indicates whether records are to be written one at a time or blocked to the capacity of the buffer area.

Omitted	assumed unblocked
U	unblocked output, or same size logical and physical records
B	blocked output, or different size logical and physical records

2.2 LABELING

This macro is required for all mass storage files and for magnetic tape files with standard or non-standard header labels.

```
              10
            | LABELING (p1,p2,p3,p4,p5)
```

p_1	Label identification address required for mass storage files. Memory location of the file identification. The value in this location is compared against the file identification field in the label for input files. For output files, this value is written in the file identification field in the label. This field is assumed to contain 14 characters, left justified, and two trailing blanks.
p_2	Edition number address must be two BCD digits, left justified with blank fill. This parameter is optional.
p_3	Reel number address (magnetic tape only) must be two BCD digits, left justified with blank fill.
p_4	Date written address; mmddyy, month, day, year. This value is left justified with blank fill; it is used in locating the correct input file, but ignored for output files, as MSIO supplies the current date.
p_5	Retention cycle location required for mass storage files; three BCD digits, left justified with blank fill. This value is placed in an output file label; on input, it indicates whether a mass storage file is permanent or scratch.

blank or 000	= scratch file
1-998	= number of days file is to be retained
999	= file retained indefinitely

This parameter is ignored by GPIO when processing non-mass storage input files.

Both p_1 and p_5 are required for mass storage files; the other parameters are specified only when necessary to ensure that a file is unique or to provide file protection.

All label processing is initiated when a file is opened by the OPENFILE request macro (3.1). For mass storage files, this request refers to the FET to obtain the identifying information provided by the LABELING macro. This information is used in a search through the label directory to find the file and prepare it for processing. If only the file identification is specified in the LABELING macro, and it is not unique, MSIO selects the first file encountered with that identification.

If the retention code in p_5 is zero or blank, a search is made for the appropriate scratch file; otherwise, the given parameters are used to search the label directory for the correct file and to prepare for processing.

If the proper file cannot be found, a message is displayed on the console; and the following actions may be taken:

Request all labels on the directory to be displayed.

Request all labels bearing a given file name to be displayed.

Request to override the file identification parameters given in the program with new parameters.

Abandon the job.

If the file is defined in the FILEDESC macro as optional, no display is created. Instead, the FET will be set to cause the first READFILE operation to take the end-of-file return specified in the READFILE request or the OWNCODE macro.

MSIO provides no labels for scratch files. If a scratch file is to be closed and then reopened, it must have a unique internal file number (p_8 of FILEDESC) by which it may be identified.

2.3 VARIABLE

This macro provides information for variable-length logical records; it is not necessary for files defined as having fixed-length or universal records.

```
           10
          |VARIABLE (p1,p2,p3,p4)
```

p_1	Indicates the type of variable-length logical records. Blank indicates simple variable or fixed length with trailer. Any other value indicates variable-length logical records terminated by record marks in last character of word. In this case p_2 - p_4 are ignored.
p_2	Key field size from 0 to 7. 0 indicates the length of the key field as a full word binary key. 1-7 indicate the number of 6-bit bytes in a BCD key field.
p_3	Byte position of the first character of the key field within the logical record. If p_2=0, p_3 must be a multiple of 8 plus 1.
p_4	Trailer size: the number of 6-bit bytes contained in one occurrence of the trailer item.

If the record length varies by the occurrence of trailer fields, p_2 and p_3 refer to a field giving the number of occurrences of the trailer item. If a record is variable with trailer items, it is processed as if it were packed; the first trailer item begins in the character immediately following the fixed portion, the next trailer item in the character immediately following the first, and so forth. If the record is not a trailer type, p_2 and p_3 describe a field giving the number of 6-bit bytes in the record.

2.4 LINKMODE

This macro describes the linkage fields necessary for processing linked sequential or random files; it may be omitted when other types of access are specified.

```
                  10
   ______________________________________________
  |             |LINKMODE (p1,p2)
  |             |
```

p_1	Linkage key address; the character position within the fixed portion of the logical record of the first character of the linkage key. This field must always start at the beginning of a computer word; the values which it may assume are expressed by the formula $p_1 = 8k + 1$, where $k = 0,1,2,\ldots,n$.
p_2	Linkage type:

F	Forward linkage only
B	Backward linkage only
FB	Forward and backward linkage

The B and FB parameters are included for users who code and use their own record locating routines for backward linkage or forward and backward linkage. Forward and backward linkage words must be 48 bits each and occupy adjacent fields in the record, the forward linkage key precedes the backward linkage key.

This macro and the linkage fields it specifies, are required if INSERTM or DELETEM operations are performed on the file (3.9, 3.10).

2.5 OWNCODE

This macro is optional; it may be used to specify entry points, p_i, to user routines.

```
                  10
   ______________________________________________
  |             |OWNCODE (p1,p2,...,p12)
  |             |
```

	Routine to be entered
p_1	At the end-of-file on an input file; required for tape files only
p_2	Before MSIO beginning label processing for an input file

	Routine to be entered
p_3	After MSIO beginning label processing for an input file
p_4	Before MSIO ending label processing for an input file
p_5	After MSIO ending label processing for an input file
p_6	Upon occurrence of a parity error on an input file
p_7	Before MSIO beginning label processing for an output file
p_8	After MSIO beginning label processing for an output file
p_9	Before MSIO ending label processing for an output file
p_{10}	After MSIO ending label processing for an output file
p_{11}	Upon occurrence of a parity error on an output file
p_{12}	When an attempt is made to process an invalid linkage key, requesting a record outside of the file limits

All parameters except p_1 and p_{12} are entered by a Bank Return Jump at the first word of the user's routine. To continue processing, the user must specify a jump returning control to the entry point. Within the routine the user may: (1) ignore the condition and return to continue processing, (2) make an indication of the error and return to continue processing, (3) terminate the job.

Parameters p_1 and p_{12} do not generate a Bank Return Jump; control is returned to these entry points when the call is completed.

MSIO request macros may not be specified from within the user's OWNCODE routines unless the routine is an end-of-file or invalid key return (p_1 or p_{12}).

There are no ending labels in the MSIO system. Any before-ending and after-ending label procedures specified by p_4, p_5, p_9, and p_{10} will be executed during the CLOSFILE process, but no labels are produced.

2.6 RANDOM

This macro is required when random accessing is to be performed on the file, or when a new index is being created for sequentially indexed files. When accessing is random, parameters p_1 - p_3 must be specified; p_4 - p_5 may be specified if desired. Only p_1 is required when a new index is created for sequentially indexed files.

When random accessing is selected, MSIO provides a randomizing routine which includes a comparison between the nomenclature field in the record and the nomenclature field in memory. These fields are defined below.

```
          10
         |RANDOM (p1,p2,p3,p4,p5)
```

p_1	For random access, location of nomenclature field in memory to be compared against a nomenclature field in the record; must be beginning of a word. For sequentially indexed access, location of the SAK to be entered into the index file.
p_2	Relative character position within the record of first character of nomenclature field; must be beginning of a word. Values range from 1 to record size.
p_3	Length in characters of fields described by p_1, p_2, p_4. Value is counted from beginning of word whether a mask is specified or not.
p_4	Memory location of an optional mask field. If the user wishes to omit any characters or groups of characters from the nomenclature fields during the comparison, he may specify a mask field. This field contains zeros wherever unused characters occur, and one's elsewhere in the field. When a mask is specified, the randomizing routine takes the logical product of the mask and the p_2 field and then the logical product of the mask and the p_1 field. The comparison is made between the masked p_1 field and the masked p_2 field. p_4 must be the beginning of a word. If p_4 is not specified, the entire fields are used.
p_5	Number of contiguous characters from the beginning of the nomenclature to be used in randomizing. The characters are counted from the beginning of the field whether or not a mask is specified by p_4. If p_5 is not specified, the entire fields are used in combination with p_4 or separately; if neither is specified, the entire field is compared.

REQUEST MACROS 3

With the Mass Storage Input/Output system, the user performs input/output operations by calls to open, read, write, and close operations. The description of each operation indicates whether or not it applies to magnetic tape as well as to mass storage files. However, the user writing or reading tape files should consult the GPIO specifications.

In the MSIO system, each file is labeled and file storage space is allocated prior to any use of file storage space. With complete file labeling, the opening of a mass storage file is accomplished by first seeking out the label in the directory and from that label obtaining the information needed to open and use the file. The file description macros (Chapter 2) provide this information. The ALLOCATE macro is required, in addition, to allocate space to a file before it can be opened.

In each of the following descriptions, the file name is the symbolic reference to the File Environment Table which was specified in the FILEDESC macro for the file.

3.1 ALLOCATE

This request allocates space on a mass storage device for a file. Initial allocation for a file must be made before an OPENFILE request is issued. Since the size of an index file is a factor in randomizing, the allocation for an index file may not be changed, once specified.

ALLOCATE may not be issued from an interrupt subroutine.

The user may request physically contiguous space, or he may allow segmentation. When contiguous space request cannot be honored, a message is displayed, allowing the operator to release the requirement for contiguous space or terminate the program.

The space available for allocation is limited according to the p_6 and p_7 parameters of the FILEDESC macro. If a device name has been specified, only that device is searched for available space; if device type has been specified, all devices of that type are searched. If no device is specified, the search is limited to the device type used by the first segment.

```
                  10
                  ALLOCATE (p1,p2,p3,p4)
```

p_1	File name specified in the FILEDESC macro.
p_2	Memory location containing the number of record blocks to be allocated.
p_3	This parameter controls the action of ALLOCATE. A + indicates the existing allocation to be incremented by p_2 record blocks. A - indicates the existing allocation to be decremented by p_2 record blocks. Anything else indicates a new allocation. Allocation for an index file cannot be changed; it is necessary to release the file and create a new one.
p_4	Any digit from 1 to n indicates the number of segments into which the allocation may be divided. Blank or zero indicates as many segments as required by the space allocation on the device. The maximum number of segments allowed by MSIO is 63_{10}.

3.2 OPENFILE

OPENFILE initiates processing of the file named; this request must be specified prior to any file processing. It may not be issued from an interrupt subroutine.

```
                  10
                  OPENFILE (p1,p2)
```

p_1	File name specified in FILEDESC macro.
p_2	File type indicator; O output only file, I input only, B both input and output. For magnetic tape, this parameter must be either I or O.

A mass storage file may be opened any number of times within the same program as long as a different FET is supplied with each OPENFILE statement, or there is an intervening CLOSFILE request. For example, a file may be in an open state for both random and sequential accessing. The file may be opened only one time as an output file even though different FET's are supplied. Any number of FET's may open the file as input even though the file is already open as output; however, problems may arise if the user reads such a file while it is being altered. The user may check bits 18-23 to determine the type of other open request (see Appendix A).

The appropriate before-beginning-label OWNCODE is executed before the label is made available by OPENFILE. After-beginning-label owncodes may refer to the label in the record area, provided the user has master access to the file. If the file is I/O, an input OWNCODE is used.

OPENFILE does not establish the initial SAK for a file nor read any records from the data file. At the time of the OPENFILE request, the current SAK field is not altered; it will be zero unless the user has altered it following the last CLOSFILE request. After OPENFILE, the label will be in the record area or the primary buffer. The address of the label is stored in RECBEGIN in the FET. That is FET + 10_{10} in bits 17-0.

3.3 READFILE

This request transfers a logical record to the record area; or if no record area is specified, it indicates the location of the logical record in the buffer area by an entry in the FET. The first word address of the delivered record is always placed in a field called RECBEGIN in the FET table at FET + 10_{10} in bits 17-0.

A READFILE request followed by a RITEFILE request is, in effect, a replace; the logical records must be of the same length.

```
          10
|        |READFILE (p1,p2,p3,p4)
|        |
```

p_1	File name specified in the FILEDESC macro. (For tape files, only p_1 may be specified; the other parameters are ignored.)
p_2	Record area address; location into which the logical record is to be transferred from the buffer area. Optional; if specified, it takes precedence over any record area specified in the FILEDESC macro for this call only.
p_3	End-of-file return; entry point name of a routine to be executed upon detection of end-of-file. Optional; if specified, it takes precedence over any end-of-file return specified in the OWNCODE macro for this call only.
p_4	Return indicator; if non-blank, control is returned to the user upon initiation of the request. The user must determine when the action is complete by testing for a zero in bit 47 of the second word of the FET entry for the file.

An invalid key return occurs under the following conditions:

If the record requested is outside of the file limits

If the record requested is past the last record written

If a record requested by random accessing cannot be found (nomenclatures do not match) (4.3)

In every case, the invalid key exit specified in the OWNCODE macro is taken. If no invalid key routine is provided, the job is terminated.

3.4 RITEFILE

This request transfers a logical record from the record area, or, if no record area is specified, updates the index of the buffer area. If a record outside the file limits is sought, the invalid key exit specified in the OWNCODE macro is taken. If no invalid key exit has been specified when this condition occurs, the job will be abandoned.

```
          10
         |RITEFILE (p1,p2,p3,p4,p5)
```

p_1	File name specified in the FILEDESC macro.
p_2	Record area address: location from which the logical record is to be transferred into the buffer area. Optional: if specified, it takes precedence over any record area specified in the FILEDESC macro for this call only. Ignored by GPIO.
p_3	Logical record length; location in memory of the record length value. Required only for universal type logical records. Ignored by GPIO.
p_4	Return indicator; if non-blank, control is returned to the user upon initiation of the request. The user must determine when the action is complete by testing for a zero in bit 47 of the second word of the FET entry for the file. Ignored by GPIO.
p_5	Writecheck indicator; blank indicates normal write, non-blank indicates writecheck mode. The writecheck mode is described in the hardware manual for the particular disk used. Ignored by GPIO.

3.5 CLOSFILE

This request terminates the processing of a file with any one FET and prevents any subsequent accessing of that file until it is opened again. It may not be issued from an interrupt subroutine.

```
              10
             |CLOSFILE (p1,p2)
```

p_1	File name specified in the FILEDESC macro.
p_2	For a mass storage file, this parameter indicates modification of storage allocation for the file. If p_2 is blank, the storage allocation for the file remains unchanged. If non-blank, the allocation will be reduced to correspond to the actual size of the file. Only a user with master access to the file may alter storage allocation with this parameter. For magnetic tape files, this parameter specifies the rewind option; U indicates unload, R indicates rewind to loadpoint, anything else indicates no rewind.

Before- and after-ending-label OWNCODE is executed during CLOSFILE processing. If the file is I/O, input OWNCODE is used.

3.6 CLOSREEL

This request closes a reel of a magnetic tape file.

```
              10
             |CLOSREEL (p1,p2)
```

p_1	File name specified in the FILEDESC macro.
p_2	Rewind option; U indicates unload, R indicates rewind to loadpoint, anything else, no rewind.

CLOSREEL may only be specified for magnetic files; if specified for a mass storage file, an error diagnostic is printed and the job is terminated.

3.7 POSITION

This request alters the SAK fields in the FET in accordance with the information in parameters 2 and 3. This may result in a buffer transfer to the mass storage device. The mass storage access mechanism will be moved to the specified record block, if physically possible.

```
               10
|              |POSITION (p1,p2,p3)
|              |
```

p_1 File name specified in the FILEDESC macro.

p_2 The address of a field containing the record locator parameter for the type of record location used with this file. Value of locator depends on type of record locating routine used for access.

Routine	Value of p4
Sequential Linked Sequential	Current SAK
Random	Nomenclature
Sequentially Indexed	SIK (Sequential Index Key)

p_3 This parameter modifies p_2:

For sequential or linked sequential files, all values are significant and modify the SAK

For sequentially indexed files, all values are significant and modify the SIK

For random files, only blank is significant

blank	Value in the location specified by p_2 replaces any previous record locator for the file.
+	Value in the location specified by p_2 is used to increment the current SAK or SIK for the file. The file must be positioned already at some value before p_3 = + may be specified. This is accomplished with a READFILE, RITEFILE, or a POSITION request where p_3 has any value except +.
F	p_2 will be ignored and the SAK or SIK for the file will be set to the value necessary to access the first record in the file.
L	p_2 will be ignored and the SAK or SIK will be set to the value necessary to access the last record in the file.
A	p_2 will be ignored and the SAK or SIK will be set to the value necessary to access the next available record for writing.

3.8 LOOK

LOOK searches a file for a logical record which satisfies a condition defined by the parameters. The search begins with the current logical record and proceeds to the limit defined by p_2. A set of up to ten values (arguments) are compared with specific fields within the logical records in the file. When a logical record is found which satisfies the condition for the first value (major argument), the next value, if any, is compared with its specified field within the logical record, and so forth.

Control is passed to a location, dependent upon whether a logical record is found with fields which satisfy the condition for all the arguments specified.

LOOK requires that the file be open and an initial SAK be established.

The COBOL collating sequence is used for all comparisons.

```
          10
         |LOOK (p1,p2,...,p37)
```

p_1 File name.

p_2 Location containing number of logical records to be searched.

p_3 Type of search; if the file consists of fixed length logical records ordered by the arguments, a bisecting search can be performed. Otherwise, a sequential search through the file must be specified.

Both types of searches will locate the same logical record; if more than one logical record in the file satisfies the condition, the first one to appear on the file will be located.

Bisecting Search	Sequential Search	Function
EQS	EQL	Equality
LQS	LQL	Less than or equal
GQS	GQL	Greater than or equal

The bisecting search is used with sequential access files only; the sequential search is used with sequential, linked sequential, and sequentially indexed files.

p_4 Location to which control is transferred if the condition is satisfied.

p_5 Location to which control is transferred if the condition is not satisfied.

p_6 Location of area into which the record satisfying the condition is transferred. If omitted, the record address in the buffer area is returned in the FET. If a record area is omitted in both LOOK and the FET (FILEDESC parameter), the records may not span blocks for a successful LOOK operation.

p_7	Number of arguments, 1-10.
p_8	Memory location of the first (major) argument. The argument must be positioned in the word so that no shift is necessary before comparison with the logical record fields.
p_9	Length of first argument as number of 6-bit bytes. May be any value from 1 to 100, but the total length of all arguments must not exceed 100 6-bit bytes.
p_{10}	Byte position within the logical record of the first 6-bit byte of the search field. Legal values are 1 to n, where $n+p_6=$ logical record length in 6-bit bytes.
p_{11}-p_{37}	Same as p_8-p_{10}, applying to second through tenth arguments and search fields.

3.9 INSERTM

This request writes a logical record according to the rules of the RITEFILE request and provides linkage which will cause it to be logically inserted just before the record most recently retrieved from the file. Subsequent consecutive INSERTM requests cause the records they write to logically follow each other, all to precede the record most recently retrieved from the file.

INSERTM requires processing in linked sequential mode; the File Environment Table must contain the current SAK and the previous SAK; the current record is approached in a logically sequential manner, not by positioning.

```
                10
|              |INSERTM (p1,p2,p3,p4)
|              |
```

p_1	File name specified in the FILEDESC macro.
p_2	Memory location of the record to be inserted (optional).
p_3	Memory location of the record length value (for universal type logical records only).
p_4	Return indicator: if non-blank, control is returned to the user upon initiation of the request; if blank, control is retained until completion. To determine when the action is complete, the user must test for a zero in bit 47 of the second word of the FET.

3.10 DELETEM

This request generates the linkage which causes logical deletion of the record most recently retrieved from the file. Each subsequent DELETEM request causes logical deletion of the next succeeding record in the file.

DELETEM requires processing in the linked sequential mode. The FET must contain the current and previous SAK; the current record is approached in a logically sequential manner and not by positioning.

```
          10
         | DELETEM (p1)
```

p_1	File name specified in the FILEDESC macro.

3.11 INDXFILE

Files which are to be accessed in a random manner or by a sequential index require an index designating the locations of the logical records in the file. This index is written by MSIO when an output file is opened for processing in either of these modes, or the index may be created for an existing file with the INDXFILE request. The index is written as a separate file which must be opened and have space allocated to it.

An ATTACH macro (3.13) is required to link the data file and index file any time an index file is used. Only one index file can be attached to a data file through any one FET.

```
          10
         | INDXFILE (p1)
```

p_1	File name specified in the FILEDESC macro for the index file.

When an index is written for random access files, the user must use the RANDOM macro (2.6) to provide the nomenclature for the data file; all parameters of RANDOM are meaningful in this case. For random files, the current SAK of the data file is stored as the index entry; the position of the index entry (SIK) is developed by randomizing the data file nomenclature fields.

For sequentially indexed files, only p_1 of the RANDOM macro is specified. The contents of this field must be in SAK format. It is stored as the next sequential index entry in the index file.

3.12 SPACEM

This macro provides the user with a list of the remaining space available to the system for allocation of files.

```
              10
             |SPACEM (p1,p2)
```

p_1	Address (18 bits) of area where list of information is to be stored.
p_2	Number of devices desired.

p_1 must be at least p_2 cells in length. The number of sectors remaining will be stored as a 48-bit binary number where the space remaining on device n is stored in location: $p_1 + n - 1$.

Devices 1 through p_2 are examined and the remaining space is returned in the appropriate position of the p_1 area. If p_2 is zero, only device 1 will be examined.

If a device is not open, or if more devices are specified by p_2 than are available to the system, a zero is returned as the space remaining for that device.

On the return to the calling program from SPACEM, the A register will contain the following:

A Upper	Address of device name table.
A Lower	Address of MSIO key word for communication area.

The remaining bits in the A register are zero; both addresses may be processed as 18-bit addresses.

3.13 ATTACH

This request links the File Environment Tables of two files so that one file is used as an index for accessing the other file.

```
                10
       ┌───────┬──────────────────────────────
       │       │ATTACH (p1,p2)
       │       │
```

p_1	File name specified in the FILEDESC macro for file to be indexed.
p_2	File name specified in the FILEDESC macro for the index file.

Both files must have been opened prior to this request, and both files must have complete File Environment Tables.

The user may specify his own record locating routine for multi-level indexing; subsequent ATTACH macros may be issued as required. ATTACH may not be issued from an interrupt subroutine.

3.14 READBLOC/ RITEBLOC

These requests transmit one block according to the control word chain supplied; no deblocking is provided. The blocks will be read/written sequentially.

```
                10
       ┌───────┬──────────────────────────────
       │       │READBLOC
       │       │RITEBLOC  (p1,p2)
```

p_1	File name specified in the FILEDESC macro.
p_2	Memory location of control word chain.

The control word chain must be in the format described for a Drum SCOPE READ request. The total word count must be equal to the block size specified for the file. Normal Drum SCOPE restrictions apply. (See Drum SCOPE Reference Manual, Publication No. 60059200.)

If a parity error occurs with READBLOC or RITEBLOC, no automatic retry is provided.

The following requests, REWINDM, backspace, SKIPM, and MARKEFM may be used for tape simulation only, and the sequential mode of access must be used.

3.15 REWINDM

This request manipulates the SAK fields of the file such that the order of reference of the next logical record is the first logical record in the file. REWINDM may be only used for sequential files.

```
               10
   ┌──────────┬──────────────────────────────
   │          │ REWINDM (p1)
   │          │
```

p_1 File name specified in the FILEDESC macro.

3.16 BACKSPACE

Two backspace requests are available: backspace file (BSPFM) and backspace record (BSPRM). These requests cause MSIO to manipulate the SAK fields of the file such that the order of reference of the next logical record is changed. BSPFM positions the file at the first logical record in the file or the previous end-of-file block specified by the user. BSPRM backspaces one record block. If the request is issued when the current point of reference is not the beginning of the block, BSPRM will position the file at the first word in the same block. When already at the beginning of a block, the point of reference will be changed to the preceding block.

```
               10
   ┌──────────┬──────────────────────────────
   │          │ BSPFM (p1)
   │          │ BSPRM (p1)
```

p_1 File name specified in the FILEDESC macro.

Only files using sequential accessing may be backspaced. BSPRM should not be used when logical records span blocks.

3.17 SKIPM

This request effectively positions the file at the first record after the next user-supplied end-of-file. If there is no end-of-file, this request will take the invalid key return. This request is valid only with a file using sequential accessing.

```
               10
   ┌──────────┬──────────────────────────────
   │          │ SKIPM (p1)
   │          │
```

p_1 File name specified in the FILEDESC macro.

3.18 MARKEFM

This request marks the end-of-file; it should be used only to mark sets of data as end-of-file marks are not used by MSIO. If this request is given before the current block has been filled, the current block will be written as it exists in the buffer. An end-of-file mark requires one complete block.

```
             10
            |MARKEFM (p1)
```

p_1	File name specified in the FILEDESC macro.

This request is valid only with files using sequential accessing.

RECORD LOCATING ROUTINES 4

The technique of a record locating routine provides flexibility of file structure. A record locating routine forms the linkage between the user and the data records contained in a file. Four routines are provided by MSIO; the particular one to be used for a file is specified in the FILEDESC macro (p_{13}), and is designated within the File Environment Table. Only one type of routine may be used for any one file.

The user may prepare his own record locating routines; in this case he must specify the entry point to his routine in p_{13} of the FILEDESC macro.

MSIO provides the routine for the access desired if p_{13} is specified as any of the following:

p_{13} of FILEDESC	Record Locating Routine
RLOC.SEQ	Sequential
RLOC.LSQ	Linked sequential
RLOC.RND	Random
RLOC.SQI	Sequentially indexed

Seek Address Key

The controlling element for file processing is the Seek Address Key (SAK). Any record in a data file is accessible to the user through its SAK field. A record locating routine produces and maintains the particular SAK for the record being accessed. SAK format:

47 24		11 0
Block number	0	Relative word count

Range of block number is: $1 \le n \le$ file size

Range of the word count is: $1 \le n \le$ block size

Link Field

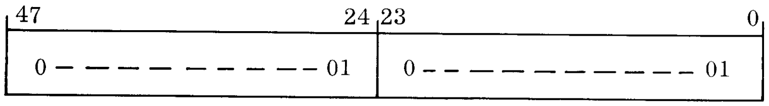
Linked sequential and random routines require a link field. This is a 48-bit field in SAK format which may be placed anywhere within a data record; but its position must be constant for all the data records in any one file. When this field is non-zero, MSIO has created linkage which will be followed as far as necessary to locate the proper record.

4.1 SEQUENTIAL

With sequential access, records are processed as they occur on the external device (input) or in the order they are produced (output); physical and logical sequence are the same.

For sequential access, the user must specify p_{13} of the FILEDESC macro as: RLOC.SEQ. Any record/buffer combination is legal within the restrictions specified for records and buffers (1.5).

Since the first block number is 1 and the first word in the file is 1, the initial SAK for a sequential file is:



As each record is accessed, the block and word count are incremented according to the size of the record and the number of records per block.

Sequential access is the only type of access for which the following tape simulation macros may be used:

REWINDM
SKIPM
BSPRM (Backspace Record)
MARKEFM (Mark End-of-File)
BSPFM (Backspace File)

4.2 LINKED SEQUENTIAL

Linked sequential access is used when the physical and logical sequence of records in a file are not necessarily the same. It is often used when a file is updated by deletion of old records or inclusion of new records with the DELETEM or INSERTM macros. When INSERTM or DELETEM are used with this mode, care must be taken that a POSITION request has not destroyed the previous SAK in the File Environment Table.

When linked sequential access is desired, the user must specify p_{13} of FILEDESC as: RLOC.LSQ
In addition, a record area and at least one buffer area must be specified. If the second buffer is given, it will be used only to hold out-of-sequence records.

When physical and logical sequences are not the same, the logical sequence is maintained by linkage data within the current logical record. The linkage data consists of a 48-bit word in SAK format within each record which indicates the next record in logical sequence. The field containing this linkage word is specified by the user with the LINKMODE macro (2.4); it may occur anywhere within a data record but its position within the record must be constant for all logical records in the file. The link field always appears in both the record and buffer areas. When the logical and physical sequences of a file are the same, this field will contain zero. The user is not required to preset the link field when creating a file.

4.3 RANDOM

Random access allows the user to refer to a logical record in a file by a symbolic key or name (nomenclature) within the record. MSIO uses a randomizing routine to determine the location in an index file of a word in SAK format which locates each data file record.

To use the random routine, the user must define and allocate an index file in addition to the data file being accessed. The index file and the data file are then attached with the ATTACH macro (3.13).

The user must specify p_{13} of the FILEDESC macro as RLOC.RND for both the data file and the index file when used for random access. Any other index references prior to a CLOSFILE are handled automatically by MSIO. (The creation of a new index file requires special procedures which are described later in this section.)

One buffer area and one record area must be specified for the data file; a second buffer will be ignored. Only one buffer area may be specified for the index file; anything else is ignored. The record type in the index file should be the same as that of the data file. When POSITION is specified for random files, p_3 must be blank.

The RANDOM macro (2.6) must be used to define both the position of the nomenclature field within each record of the data file, and the location of a corresponding nomenclature field in memory which will be randomized. The user must also define a field within the data file record in SAK format for linkage; this field is used by MSIO. The user defines this field with the LINKMODE macro (2.4). RANDOM and LINKMODE are used for the data file only.

The user should always specify an invalid key routine to be entered when reading a random file; the entry point to such a routine is defined by p_{12} of the OWNCODE macro (2.5).

When a file is processed randomly, the user must place the key name (nomenclature) for each record in the memory location defined by p_1 of RANDOM before he specifies a READFILE or a RITEFILE for that record. The nomenclature field in memory is randomized to determine the position within the index file of the index entry whether the record is being written or read. When reading, MSIO sets bit 47 of word 15 in the File Environment Table (Appendix A) when the record is found. If the record is not found,that bit is set to 0 and MSIO takes the invalid key return specified by p_{12} of OWNCODE.

The index file is maintained by MSIO and all reading and writing of the index is done automatically. The index file consists of one word entries in SAK format. The order of the entries depends on the nomenclature; the index file is written randomly. Once an index file is created, the size may not be changed; however, a new index file may be created by the user for a random data file. This might be desired if the existing index file is either too small requiring excessive linkage, or too large resulting in waste space. The following procedure is used to make a new index file for a random data file:

1. Define the data file with the FILEDESC, LINKMODE, and RANDOM macros; p_{13} of FILEDESC must be RLOC.SEQ.
2. Open the data file for I/O with the OPENFILE macro (p_2 = B).
3. Define the index file with FILEDESC (p_{13} = RLOC.RND); ALLOCATE, OPENFILE (p_2 = B), and ATTACH new index file.
4. Read first record (sequentially) from data file and pick up the nomenclature from the record.
5. Place nomenclature from record into nomenclature field in memory (defined by RANDOM macro).
6. Use INDXFILE macro (3.11) to place data file SAK field into a slot in new index file.
7. Read next record and return to step 5. Continue until end of file, then close both files.

4.4 SEQUENTIALLY INDEXED

Sequentially indexed access allows the user to read a data file in the order specified by an index. When a sequentially indexed file is written, MSIO automatically constructs an index file in the same order as the data file. When a sequentially indexed file is read, the index is processed sequentially and the data file is read according to the index. Each entry in the index file is the SAK for a particular record in the data file. The index file is distinct from the data file; the user must define this file and allocate storage for it.

This type of access is particularly useful for sorting. The sort tag or key together with the SAK field may be extracted from each record to produce a set of abbreviated records. These records are then sorted according to the tag or key producing a list of sorted SAK's.

It is recommended that one buffer area and one record area be specified for the data file; either may be used alone subject to the buffering restrictions (1.3). Only one buffer area may be specified for the index file; anything else is ignored.

A new index consisting of the sorted SAK's is then produced as follows:

1. Define the data file with the FILEDESC and RANDOM macros; p_{13} of FILEDESC must be RLOC.SQI.
2. Open the data file with the OPENFILE macro.
3. Define the index file with FILEDESC (p_{13} = RLOC.SQI), ALLOCATE and OPENFILE; ATTACH the new index file.
4. Use INDXFILE macro to place the SAK field into the next available consecutive slot in the new index file. The location of the SAK is specified in the RANDOM macro for the data file.
5. Continue to the end of the SAK list, then close both files.

FILE ENVIRONMENT TABLE FORMAT A

The format of areas in an FET which the user may wish to examine are listed below; areas not listed are reserved for internal use by MSIO.

Word	Bits	Usage
0	46	1 designates optional file indication
1	47	transmission in progress
	45	transmission error
	41-24	location of 14-character ident field
2	47	pointer advanced flag
	46	open or closed file
	45	end-of-file read on input
	41-24	location of edition number
	23-18	type of open found on label when opening 100000 = master 000001 = input 000010 = output 000100 = input/output
3	41-24	location of date-written value in label
	17-0	location of retention code value in label
4	38-24	word count of current record in record area
	17-0	location of record area (FWA)
5	17-0	location of primary buffer
6	17-0	location of secondary buffer
7	46	standard error recovery suppressed
	17-0	record locating routine entry point address
8	41-24	address of submitted nomenclature
9	41-40	10 = variable trailer items or 01 = universal format
	38-24	fixed record size in characters or zero if variable
	14-0	physical record size in words (maximum) (block size)
10	32-24	FN = file number (in binary)
	23-18	device type
	17-0	address of record (buffer or record area)
11	41-24	before-beginning-label on input entry point address
	17-0	before-beginning-label on output entry point address

Word	Bits	Usage
12	44-42	001 = input; 010 = output; 100 = input/output
	41-24	after-beginning-label on input entry point address
	17-0	after-beginning-label on output entry point address
13	41-24	before-ending-label on input entry point address
	17-0	before-ending-label on output entry point address
14	41-24	after-ending-label on input entry point address
	17-0	after-ending-label on output entry point address
15	47	record found indicator
	41-24	error on input entry point address
	17-0	after errors on output, entry point address
16	41-24	after end-of-file on input, entry point address
	17-0	invalid key entry point address
18	47	1 = record size determined by record mark
	38-24	size of OCCURS item (if used)
	23-21	number of characters in size descriptor (maximum = 7)
	14-0	location of key in record relative to beginning (character count)
19	47-0	device name
20	47-0	access key
21	35-24	relative position of link key in record
	23-0	file condition
22	47-45	allocation control
	32-24	number of segments allowable
	23-0	number of record blocks allocated for file
23	47	writecheck requested
	17-0	FET location of index file
24	47	call specified return before completion
	41-24	blocking routine entry address
	17-0	deblocking routine entry address
25	47-0	current SAK
26	47-0	SAK of next available record
27	47-0	SAK of last logical record
28	47-24	block number of last EOF accessed (sequential)
	23-0	block number of next EOF (sequential)
	47-0	if not sequentially accessed, first SAK
29	47-0	SAK of previous record
30	47-42	count of characters to be used when randomizing
	41-24	location in memory of mask to use with nomenclature
	23-18	size of nomenclature fields, character count
	17-0	relative position of nomenclature within record

ERROR MESSAGES AND RECOVERY PROCEDURE B

MSIO checks for errors at assembly and execution time to ensure the validity of each call to MSIO. Automatic procedures are available at the user's option to attempt recovery from data transmission errors.

ASSEMBLY TIME ERROR MESSAGES

An error detected at assembly time produces an assembly error from COMPASS by generating an illegal operation code with a remark describing the error condition. These error diagnostics are provided in addition to the normal error checking done by the COMPASS system.

The assembly diagnostics are printed in the form:

```
0 99        p      message
```

where 0 = the COMPASS error flag;
99 = the illegal operation code field;
p = the value of the parameter found in error.

0 99 (value of p_6) ERROR CONDITION—ILLEGAL DEVICE TYPE
FILEDESC macro submitted device type not 828 or 814 or an unknown device type

0 99 (value of p_6) ERROR CONDITION—BOTH DEVICE TYPE AND NAME USED
FILEDESC macro submitted both device type and device name

0 99 (value of p_1) ERROR—MASS FILE MUST BE LABELED
LABELING macro submitted NON—STND label for disk file

0 99 (value of p_1) ERROR—LINKMODE NOT USED FOR TAPES
LINKMODE macro submitted for tape file

0 99 (value of p_1) ERROR—RANDOM NOT USED FOR TAPE FILES
RANDOM macro submitted for tape files

0 99 (value of p_2) ERROR—ILLEGAL INPUT/OUTPUT KEY
OPENFILE macro submitted an I/O key not equal to I, O, or B

0 99 (value of p_3) ERROR—ILLEGAL SEARCH TYPE
LOOK macro submitted an unknown search type

EXECUTION TIME ERROR MESSAGES

In addition to error checking at assembly time, certain tests are made by MSIO each time a request for processing is made. When an error is discovered, a diagnostic is printed on the standard output unit and the job is terminated. The following diagnostics are printed in the form:

FILE-NAME

diagnostic

ALLOCATION PARAM. ERROR

initial allocation was specified for existing file; or an increase or decrease in allocation was requested when creating file.

BUSY FET CALLED

last operation for this FET is not yet completed; can occur when using return type calls to MSIO.

CHAIN ADDRESS ERROR

input/output chain refers to area outside of bounds.

CHAIN SIZE ERR.

for READ/RITEBLOC, the control word chain total word count is not equal to the block size.

DUPLICATE RANDOM RECORD

occurs when an index is constructed for random access file and an index entry already exists with identical nomenclature to the record presently being processed.

FET NOT BUSY AFTER CALL

MSIO system error.

FILE ALREADY OPEN

attempt made to open a file already opened.

FILE NOT OPEN

file must be opened prior to processing.

ILLEGAL ACCESS

an attempt was made to write on an input-only file, read an output-only file, allocate without master access, or there is security check error.

ILLEGAL PROGRAM LEVEL

an OPENFILE, CLOSFILE, ALLOCATE, or ATTACH request may not be specified from within an interrupt subroutine.

INT. SUB. -LOST RETURN

a call from within an interrupt subroutine has destroyed the return address to MSIO.

INVALID REQUEST

illegal call for this type access; for example, a REWIND request for a random-access file.

LOOK PARAMETER ERROR
: the number of characters in search field greater than 100; or the number of search fields greater than 10.

MORE THAN 63 SEGMENTS
: allocation for this file would take more than 63 segments.

MSIO BUSY
: an MSIO call was issued from an interrupt subroutine while a main program MSIO request was being processed.

NO EOF/INVKEY OWNCODE
: an error has occurred and no end-of-file or invalid key OWNCODE routine has been specified.

NO ERR. OWNCODE
: a parity error has occurred (input or output file) and no OWNCODE routine has been specified.

NO LABEL FOR FILE
: no label can be found in the label directory for this input file.

NO LABEL ID
: attempt made to open a file with no address given for file name.

NO MASSFILE CRD
: attempt made to open mass storage file without MASSFILE card.

NO RECORD AREA
: processing has been attempted in random or linked sequential mode and no record area was specified.

NO REC. LENGTH
: universal record format being used and no record length given for RITEFILE.

NOT ENOUGH SPACE ON DEV.
: not enough space available on the device for file allocation.

OUTPUT NOT ALLOWED
: write protect set in the label on opening the file.

PARITY ON LABEL
: parity error on the label; no reading or writing can be processed.

POS.-INS./DEL. ILLEGAL
: attempt made to insert or delete record in file but previous SAK not found; previous POSITION macro destroyed SAK.

PRIM. DEV. DOWN
: device to which first segment is allocated (primary device) unavailable.

SECONDARY DEVICE DOWN
device to which second through nth segment is allocated (secondary device) is unavailable. The operator may request continuation of processing or terminate the job (OPENFILE request). If continuation is requested and a READFILE request for the file is given, a message is printed: BLOCKS x TO y UNAVAILABLE. Accessing must be sequential and may be input only.

TOO MANY FILES
descriptor tables are full (MASSFILE card parameter too small), or files have too many segments.

TOO MANY SEGMENTS REQD.
allocation cannot be made within the number of segments alloted.

ZERO ALLOCATION
attempt was made to open a file without a prior ALLOCATE.

DATA TRANSMISSION ERROR RECOVERY

The user may request standard error recovery procedures through the FILEDESC macro. Recovery procedures for tape:

Reading:	Tape is backspaced and re-read three times.
Writing:	Tape is re-written three times at original position; if error persists, the tape is erased 6 inches, and the write procedure repeated. A series of three erase sequences and 16 writes are attempted.

If the error persists, the user's OWNCODE subroutine is executed, if specified. This subroutine is expected to perform necessary actions to continue or terminate the job. If no OWNCODE routine is specified, the operator is consulted. He can ignore the error and continue processing, repeat the recovery sequence, or terminate the job. If the job is terminated, appropriate messages are written on the standard output unit.

If standard error recovery procedures are requested for mass storage files, the record is re-read or re-written three times before the OWNCODE routine is entered or the operator is consulted for action.

MSIO/GPIO INCOMPATIBILITY C

1. Record mark; when a record is terminated by a record mark
   MSIO: must be last character (bits 0-5) of word
   GPIO: may be any character in record

2. FILEDESC macro; parameter p_1 designates logical record length

   MSIO: blank = simple variable length records
   universal records
   record mark records

   n = length in 6-bit bytes of fixed portion of trailer type variable length records, $17 \leq n \leq 32760$

   GPIO: n = maximum logical record size in characters

3. LABELING macro
   MSIO: p_1 required
   p_2 optional
   p_3 omitted
   p_4 optional, ignored for output
   p_5 required, if specified as zero or blank the file is assumed to be scratch

   GPIO: p_1 required
   p_2 optional
   p_3 optional
   p_4 optional
   p_5 optional, ignored for input

4. VARIABLE macro
   MSIO: p_1 blank = simple variable or fixed length with trailer
   non-blank = variable terminated by record mark; p_2-p_4 are omitted
   If the record is a simple variable, p_2 and p_3 give the size and position of a key field in each record holding the number of characters in the record; if the record has a trailer field, p_2 and p_3 describe a key containing the number of occurrences of the trailer for the record.
   p_4 contains the size of the trailer item

   GPIO: p_1 0 = simple variable; has key field
   1 = variable terminated by record mark, p_2-p_4 are omitted
   p_2 and p_3 always give the size and position of a key field in the record containing the number of characters in the record.
   p_4 size of trailer item, or number of 6-bit bytes if record is binary

5. OWNCODE

OWNCODE Exit Descriptions

OWNCODE exits are not serviced in the same manner by MSIO and GPIO. The exits are described below as they are handled by each I/O system.

p_1 MSIO - Exit (UBJP) when a READFILE request is made after the last logical record in a file has been read, or after the last logical record preceding an end-of-file block has been read. This exit occurs before p4 and p5 exits.
GPIO - Exit (BRTJ) when the end of an input file is reached. This exit occurs after p4 and p5 exits. If this exit is not specified, GPIO aborts the run at this point, with error message "IO ER. INCR USE 20".

p_2 Both - Exit (BRTJ) while processing an input OPENFILE request, before the beginning label (if any) is available to the user.

p_3 MSIO - Exit (BRTJ) while processing an input OPENFILE request before the beginining label is rewritten. The label is available to the user with master access, and certain fields may be modified. The location of the label is contained in item RECBEGIN (FET).
GPIO - Exit (BRTJ) while processing an input OPENFILE request. A standard label is available for examination in 9COMMON+1 after it is checked by GPIO. A non-standard label is stored in the record area and its location is defined by item RECAREA (FET). For a file with standard labels which is not the first file on a multi-file reel, the first label on the tape is checked by GPIO, stored at 9COMMON+1 and the label of the input file is treated as non-standard (read into the record area, unchecked).

p_4 MSIO - Exit (BRTJ) while processing an input CLOSFILE request, before the beginning label is rewritten and before the FET is cleared. The beginning label is not available to the user. MSIO provides no ending label.
GPIO - Exit (BRTJ) while processing a READFILE request: for standard labels, after the file mark preceding an end-of-file or end-of-tape label is read but before the label is available; for non-standard or no labels, after a file mark is read.

p_5 MSIO - Exit (BRTJ) while processing an input CLOSFILE request, after the beginning label is rewritten and the FET cleared. MSIO provides no ending labels.
GPIO - Exit (BRTJ) while processing a READFILE request: for standard labels, before the file mark following an end-of-file or end-of-tape label is read, the label is available for examination in 9COMMON+11; for non-standard or no labels, immediately after the p4 exit option is processed.

p_6 MSIO - Exit (BRTJ) on a read error after three reread attempts have been made. If no owncode routine is specified, a message indicating the error is typed. The operator is given the option of:

1. Abandoning the job (ABANDON)
2. Repeating the error recovery procedure (RETRY) or
3. Accepting the erroneous record (CONTINUE)

A logical input record may be reread by setting item POINTADV (FET) = 1 and reissuing the read request.
GPIO - Exit (BRTJ) on a read error after sixteen read attempts have been made. A message is typed indicating the error. If no owncode routine is specified, the operator is given the option of: ignoring the error; requesting a reread; dropping the physical record and printing it on the standard output; terminating the job.

p_7 MSIO - Exit (BRTJ) while processing an output OPENFILE request, before the beginning label is available to the user.

GPIO - Exit (BRTJ) while processing an output OPENFILE request. For standard labels, exit occurs before the first four label words are assembled by GPIO starting at 9COMMON+1. But the user may store in the last six label words at this time. A non-standard label will be written from the record area, defined by item RECAREA (FET), and should be assembled at this time. The entire record area is written.

p_8 MSIO - Exit (BRTJ) while processing an output OPENFILE request, before the beginning label is written. The label is available to the user and certain fields may be modified. The location of the label is contained in RECBEGIN (FET).

GPIO - Exit (BRTJ) while processing an output OPENFILE request, after the beginning label has been written.

p_9 MSIO - Exit (BRTJ) while processing an output CLOSFILE request, before the beginning label is rewritten and the FET is cleared. The beginning label is not available to the user. MSIO provides no ending label.

GPIO - Exit (BRTJ) while processing an output CLOSFILE or CLOSREEL request: for standard labels, before the file mark preceding the end-of-file or end-of-tape is written—the label is available for user modification in 9COMMON+11; for non-standard or no labels, before the file mark ending a file or tape is written.

p_{10} MSIO - Exit (BRTJ) while processing an output CLOSFILE request, after the beginning label is rewritten and the FET is cleared. MSIO provides no ending labels.

GPIO - Exit (BRTJ) while processing an output CLOSFILE or CLOSREEL request: for standard labels, after the file mark following an end-of-file or end-of-tape label is written; for non-standard or no labels, after the file mark ending a file or tape is written.

p_{11} MSIO - Exit (BRTJ) on a write error after three rewrite attempts have been made. If no owncode routine is specified, a message indicating the error is typed. The operator is given the option of: ABANDON, RETRY, or CONTINUE. (Same as input options.) A logical output record may be rewritten by saving item CNTSAK (FET) prior to the first write and restoring it prior to the rewrite. This operation can be done safely only if the file has been opened for I/O, or, if two buffers are defined and the largest logical record does not exceed the defined block size (buffer). On write parity errors, item CNTSAK (FET) may be saved when OWNCODE exit p11 is taken, then restored for a rewrite after returning to the main program.

GPIO - Exit (BRTJ) on a write error after 16 rewrite attempts are made, blanking tape each time. A message, indicating the error, is typed. At this point, the bad record has been erased and the tape is positioned at a new spot. If no owncode routine is specified, or upon return from an owncode routine, GPIO will again attempt to write the record. The write error procedure has been initialized so that if subsequent errors occur, 16 rewrites will be attempted as described above.

p_{12} MSIO - Exit (UBJP) when an attempt has been made to read, write or position beyond the bounds of a file. If this exit is not specified, MSIO types a message and aborts the run at this point.

OWNCODE Clarifications and Restrictions

BOTH - No I/O may be performed, using MSIO or GPIO, within the respective OWNCODE exit subroutines (p2 to p11).

GPIO - As far as GPIO logic is concerned, there are no multi-reel input files if non-standard or no labels are specified. The user must close and open (and check non-standard labels on) successive tapes as though they were separate files.
For tapes with standard label, closing and opening of reels is handled by GPIO and the p1 exit takes place only after the end-of-file label has been read.

GPIO - On output, when an end-of-tape mark is detected, GPIO issues a CLOSREEL (UNL) request, performing OWNCODE exits 9 and 10. On the next write request, GPIO opens a new tape, performing OWNCODE exits 7 and 8 and writing a label (standard or non-standard), then writes the logical record.

GPIO - Standard labels are read andwritten in BCD. Non-standard labels are read and written in the recording mode defined for the tape.

GPIO - The OWNCODE exit parameters for GPIO have been rearranged to conform with the OWNCODE macro description in the MSIO manual.

MSIO - For exit p3, there is no item in the FET table telling whether the user has master access or not. Therefore, the user might examine the label area defined by RECBEGIN (FET) to determine if MSIO has actually provided a label. The procedure should be to examine the first label word, bits 30-41 for a label identifier [( )]. The label will be provided whenever user record or buffer areas are large enough (at least 14_{10} words).

MSIO - The areas of the FET which are cleared during a CLOSFILE request (see p5 and p10) are the dynamic items which must be initialized in preparation for a possible re-open using the same FET. Items such as LINKFIELD, O/C, OPENTYP, I/O, ALLOCAT, DESLOC, CNTSAK, PREVSAK, ABLKCNT, BBLKCNT, etc., are cleared. Items set by the file description macros (FILEDESC, LABELING, etc.) are not cleared.

6. RITEFILE

MSIO:	p_2	location of record area
	p_3	location of record length value, used only for universal type logical records
	p_4	return indicator; if non-blank control is returned to user upon initiation of request
	p_5	writecheck indicator; blank = normal write, non-blank = writecheck mode

GPIO: p_2, p_3, p_4, and p_5 ignored by GPIO

7. READFILE

MSIO:	p_2	location of record area
	p_3	entry point of end of file return
	p_4	return indicator; if non-blank, control is returned to user upon initiation of request

GPIO: p_2, p_3, and p_4 are ignored by GPIO.

8. CLOSFILE

MSIO: p_1 file name specified in FILEDESC macro

p_2 modification indicator for storage allocation of mass storage file:

blank	allocation unchanged
non-blank	allocation reduced to size of file

lock indicator for non-mass storage files:

U	unload
R	rewind to loadpoint
other	no rewind

GPIO: p_1 rewind option:

RWD.	rewind tape
UNL.	rewind and unload tape
NOR.	no tape action

p_2 symbolic address of FET entry for this file

INDEX

CONTROL DATA

CORPORATION

COMMENT AND EVALUATION SHEET

3600/3800 MSIO

Reference Manual

Pub. No. 60174800A Revised - September, 1967

THIS FORM IS NOT INTENDED TO BE USED AS AN ORDER BLANK. YOUR EVALUATION OF THIS MANUAL WILL BE WELCOMED BY CONTROL DATA CORPORATION. ANY ERRORS, SUGGESTED ADDITIONS OR DELETIONS, OR GENERAL COMMENTS MAY BE MADE BELOW. PLEASE INCLUDE PAGE NUMBER REFERENCE.

FROM NAME: ______________________________

BUSINESS ADDRESS: ______________________________

NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

FOLD ON DOTTED LINES AND STAPLE

STAPLE STAPLE

OLD FOLD

FIRST CLASS
PERMIT NO. 8241

MINNEAPOLIS, MINN.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION

Documentation Department

3145 PORTER DRIVE

PALO ALTO, CALIFORNIA

OLD FOLD

STAPLE STAPLE

3600/3800 MSIO REFERENCE M ANUAL

CONTROL DATA
CORPORATION

CORPORATE HEADQUARTERS, 8100 34th AVE. SO., MINNEAPOLIS, MINN. 55440
SALES OFFICES AND SERVICE CENTERS IN MAJOR CITIES THROUGHOUT THE WORLD

Pub. No. 60174800A

Litho in U.S.A.