

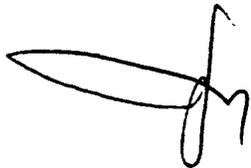
09/17/80

Copies for:

- Don McBride
- Owen Townsend
- Charles Gibbs
- Jim Condon
- Dean Arychuk
- Leo Rodgers
- Richard Coughlan

Attached is a copy of the memory size requirements for OS/3 Release 7.0 and the System/80. Note that the amount of resident memory is up to 120K.

More precise memory sizing will be available in the future when the SRD is available.





INTER-OFFICE MEMORANDUM

Copy to  
Don McBride  
+  
Owen Townsend

TO: Gilbert Dy  
LOCATION: Vancouver Branch

FROM: Roger Wainwright

LOCATION & DATE: H.O. - June 6, 1980.

CARBONS:

DEPARTMENT: Customer Support Services

SUBJECT: OS/3 MIRAM

Some considerable while ago Owen Townsend asked me if I could get him a working paper on OS/3 MIRAM. Well here it is at last.

Because it's dated 1/6/78 there may be some inaccuracies, but if you're looking at the thing from the point of view of getting a better insight than can be gleaned from standard manuals, I have no doubt it'll serve a useful purpose.

(I'm assuming that you won't be coding an assembler program to use the imperatives since, as far as I know, the assembler interface to this access method is not currently supported;)

Good luck!

*Roger Wainwright*

R. Wainwright.

/sww  
Encl.



## OS/3 WORKING PAPER

COMPONENT: Multi Indexed Random Access Method (MIRAM)	AUTHOR: Goldberg, Willis
COMPONENT NO:	DATE: 1/6/78
SUBJECT: Component Product Software Description	REV. NO: 1
SECTION NO:	
SPECIFIC CARBONEES:	

## ABSTRACT:

This paper describes a separate new disk access method for OS/3. This method is specifically designed to include support for .ANS'74 COBOL (relative and indexed I/O) and also to support the IRAM file structure.

Appendix A - glossary of terms

Appendix B - DTF layout; DSECT label definitions; DTF field definitions

Changes in this revision are denoted by a vertical line in the right-hand margin.



TABLE OF CONTENTS

1.0	INTRODUCTION
1.1	Scope
1.2	Purpose
2.0	COMPONENT DESCRIPTION
3.0	HARDWARE REQUIREMENTS
4.0	INTERFACE REQUIREMENTS
4.1	Related Software Components
4.2	Data Base
4.3	Operator Interface
4.4	User Interfaces
4.4.1	Declarative Macro
4.4.2	Imperative Macros
5.0	FUNCTIONAL DESCRIPTION
5.1	File Format
5.2	Function and Subroutine Procedures
6.0	PERFORMANCE
7.0	COMPATIBILITY
8.0	CONVERSION
9.0	DOCUMENTATION AND SUPPORT
10.0	RESTRICTIONS
11.0	MAINTAINABILITY AND RELIABILITY
12.0	RELATED DOCUMENTS
13.0	STANDARDS DEVIATIONS
APPENDIX A GLOSSARY	
APPENDIX B DTF LAYOUT	



## 1.0

## INTRODUCTION

MIRAM (enhanced IRAM) provides additional facilities beyond those of IRAM, which is described in Working Paper #675. It is a complete disk access method, based on data records that do not move from original placement location; and based on direct addressing by use of file relative record number.

## 1.1

Scope

MIRAM is a comprehensive disk access method in which a single processor provides the essential operations for SAM, DAM, and multi-key ISAM processing; including capabilities for deletion, variable length records and duplicate keys.

The processor will be able to process files created by IRAM, but programs which interface with IRAM will have to undergo changes in order to interface with MIRAM due to its new declarative and imperative macro architecture. MIRAM will be able to create files which can be accessed by IRAM as long as the resulting files involves no functionality which IRAM does not provide (e.g. deletion, multi-key, variable length records).

## 1.2

Purpose

MIRAM is developed for the support of disk access for RPG in the IBM System/3 manner, for ANS'74 COBOL relative and indexed files, and for projected use by SUL and Library handling programs.

2.0

COMPONENT DESCRIPTION

A single processor handles all functions, permitting the user to intermix input and output, sequential and random, keyed and unkeyed operations. Within a single job, the user can employ all functions.

The file may be used for strictly non-indexed purposes, for indexed and non-indexed combined, or as an index facility alone, independent of data. Up to five separate index structures can be requested. Key sizes may range from 1 through 80 bytes. Keys may be individually specified as allowing for duplication, and allowing for change during update. A duplicate key series is returned sequentially in FIFO order (first in first out).

Means are provided to establish a position in the file, from which sequential retrieval can be requested. Once established, this position can be disestablished, changed, or held constant during digressions into random operations. The "held" position is unchanged by output, update, delete, and random retrieve with hold.

The position is changed by success (or undefined by failure) during operations of select, sequential retrieve, and random retrieve without hold.

When the user elects to have variable length records, the first 4 bytes of his record must be dedicated to control purposes. Bytes 1 and 2 must contain the record size in binary. For example, a value of 44 would leave 40 bytes for user data.

If the user wishes to employ the delete facility or intermix keyed and unkeyed records, he must call for a record control byte (RCB). For variable length records, if the RCB is elected, byte 3 of the record would be used by the system as the RCB. For fixed length records, if the RCB is elected, the user must predicate his data buffer size on the one byte larger record slot size. However, his specification of record size is not to include the RCB. MIRAM provides a new declarative macro and a new set of record handling imperative macros. The imperative macros are more specific than those of IRAM, because they do not depend on mode settings placed in the DTF. The basic operations are the same as before, with deletion added.

- . Output a new record.
- . Input an existing record.
- . Select a position for sequential.
- . Update an existing record.
- . Delete an existing record.

The index-only facility can be used to force several index entries to point to a single data record, or to cause indexing to a data record not containing the key(s) on which it is indexed. The index-only functions operate only on the index entry, which consists of the key and 3 byte pointer.

### Output

When deletion capability is elected, the processor has new functionality for the random output of new records. If the user directs a record to a point beyond current file end, any resulting gap will be filled with void records. If he directs a record to point short of file end the operation will be rejected if a non-void record is found at that point.

In MIRAM, the user may output records to a file that contains index records or index entries, directing that they not be indexed. Such records will be marked as unkeyed. Either keyed or unkeyed records may be directed to selected positions, or to end of file. There is a switch that can be set to cause a follow up to a record output that interrupts sequential input. Follow up consists of re-retrieving the last retrieved sequential record.

If there is keyed output, there will be a report as to the duplication of keys that has resulted.

For output, the new record must always be provided in a work space. After output, the relative record number of the newly placed record is available to the user.

### Input

The input macro provides for choosing random or sequential, and keyed, index-only or unkeyed. There is also a choice for random with-hold, which will prevent the loss of a current sequential position. If keyed or index-only input is used, there will be a report stating that the next record of the key set has/has not a duplicate key.

The user is required to forecast the use of the input record: for information, for changes, or for total replacement. For total replacement, the record will not be moved from the buffer to workspace. When "information" is forecast, update or delete is not allowed to follow. Recognizable void records are bypassed in sequential; treated as no-finds in random.

A successful random-without-hold can be followed by sequential progress to records beyond.

#### Select

New functionality has been added to the selection process. BOF and EOF are now available. Other specifications are equal (EQ), greater than (GT), and greater or equal (GE). Failure to satisfy the request will result in a no-find report, whether caused by an empty file or other reasons.

If the request is select-by-key or index-only, success will be accompanied by a report stating that the record beyond has/has not a duplicate key in the set referenced.

By a special means, the user may select by key, using only the n leftmost bytes of argument, where n is less than declared key size.

Changing the key of reference is done thru the SELECT macro.

#### Update

MIRAM update of keyed records is considerably more complex than IRAM, since keys are permitted to change. This may require that index adds and deletions be effected.

Variable records are permitted in MIRAM. Record size may be changed during update, but not to exceed slot size, nor to fall below the size that will encompass all keys.

### Delete

The function to delete is new. It consists of marking the record as void, and of also voiding any index entries pointing to it.

### Erase

The function to erase is new. The user can erase an entire file, thus simulating the INIT specification on the //LFD job control statement. In addition, for files which contain only unkeyed records, the user can erase all records starting with a specified relative record number.

## 3.0

HARDWARE REQUIREMENTS

A MIRAM file must reside on from one to eight disk packs of the same type.

A program using one or more MIRAM files (and in addition any IRAM files) must be linked to one of the two resident processing modules. There is the "maximum" module which will permit all functions, and also a "non-indexed" module which can be linked in when index operations will not be performed.

D3\$M111 - maximum module

D3\$M000 - non-indexed module

For each file, the program must provide at least the following:

Register Save Area	<u>72</u>
DTF Area	(approx. 400)
Record Work Area	SLOT SIZE
Key Argument Area <sup>1</sup>	LARGEST KEY SIZE+3 (6 MINIMUM)
Seek Address Area <sup>2</sup>	4
Contiguous Buffer Area:	
Min index buffer <sup>1</sup>	256 <sup>3</sup>
Min data buffer	256 <sup>3</sup>

1. Required only if keyed or index-only operations are to be performed.
2. Always required.
3. All buffers must be multiples of 256 bytes. See BFSZ keyword description for minimum value determination. The buffer areas must start on a half-word boundary.

4.0 INTERFACE REQUIREMENTS

The MIRAM processor system utilizes OS/3 Transient Management for activation of selected processing functions, and the System Access Technique (SAT) for all I/O requirements and to maintain a device independent mode of operation.

All user interfaces are maintained through the DTFMI (Define the File Multi-Index) declarative macro, and selected imperative macro instructions. These interfaces are detailed in Section 4.4.

Operator communication is maintained through output of error and status information to the operator console and/or system message log.

4.1 Related Software Components

MIRAM requires file initialization and termination procedure supplied via the file OPEN and CLOSE transient facilities. These functions are initiated by the users OPEN and CLOSE imperative macro instructions.

Linkage to the SAT processor is provided through the DTF table; the address of the SAT processor is established during file initialization.

4.2 Data Base

All files to be accessed by the MIRAM processor system must have been created by the MIRAM or IRAM processors. Files created under any other OS/3 access methods are not compatible with MIRAM processing requirements; MIRAM files are not accessed through other OS/3 access methods, except for IRAM, and then only if the file was created with functionality provided by IRAM.

4.3

Operator Interface

Operator communication will be limited to the display, by the MIRAM system, of appropriate error/status messages. All communication is supplied by the OS/3 DMS message handling routine. These messages are outlined in the OS/3 User Guide (UP-8068, current version, Appendix B).

4.4

User Interfaces

4.4.1

Declarative Macro (DTFMI)

The DTFMI macro instruction is provided for definition of file characteristics. The following list outlines all of these keyword parameters, and detailed descriptions of each keyword follow it, with additional keyword spellings provided in parentheses. (Appropriate PNOTE's will be generated if errors are detected in keyword processing).



PARAMETERS:

- ACCESS=EXC - Exclusive use of the file is requested. No other access of the file will be granted once it is dedicated to the requesting DTF.
- EXCR - Exclusive Read use of the file is requested. The DTF declares itself as the exclusive update, add owner of the file but will allow it to be shared with others performing read functions.
- SRDO - Shared Read-Only access to the file is requested. The DTF declares itself as a reader but will only tolerate other readers to have access to the file.
- SRD - Shared Read access to the file is requested. This declaration identifies an intention to perform only read access to the file. It indicates a willingness to share the file with any other type of access (read, update or add).
- ~~SUPD~~ - Shared Update access to the file is requested. This specification identifies an intent to update the file but declares that it will not be extending it. The file can be shared by other reader DTFs.
- SADD - Shared Add access to the file is requested. The DTF declares an intention of extending the file. The file may be shared with other readers.
- BFSZ=n  
(BLKSIZE  
BKSZ) - Specifies the size of the data buffer in the file, where n is the size, in bytes. This keyword is always required. Size must be at least 256 as well as a multiple of 256.

The minimum value can be determined as follows:

- If the slot length is less than or equal to 256 and evenly divisible into 256, the size is 256.
- If the slot length is greater than 256 and a multiple of 256, the size is equal to the slot length.
- If the slot length is not evenly divisible into 256 and not a multiple of 256, the size can be calculated by adding 255 to the slot length and rounding up to the next multiple of 256.

- EOFA=symbol  
(EOFADDR) - Specifies the address of a routine the user has coded to handle end-of-data for a sequential by key or consecutively processed file, where symbol is the symbolic address to which data management transfers control on sensing the end of data.

PARAMETERS: (Cont'd)

- ERRO=symbol - Specifies the address of the user's error-handling routine to which Data Management transfers control for all conditions of error or exception to exact performance of the requested function. When Data Management transfers control, filenameC contains information on the reasons for the error. (See UP-8068, Data Management User Guide Table B-1 for error messages, and Table B-3 under DTFIS for significance of bits in filenameC). If omitted, control returns to the user inline.  
(ERROR)
- INDA=symbol - Specifies the location in main storage in which index blocks are processed during keyed operations, where symbol (address) is the location. Must be half-word aligned. The length of the area is specified by the INDS keyword. This area must immediately precede the primary I/O buffer (IOA1). In order for index operations (keyed or index only) to be permitted, all of the index related keywords must be specified: INDA, INDS, KARG, and KEY1. If any are missing, it will be assumed that index operations were not intended to be employed, and no index operations will be permitted. (See KEYn keyword for single exception to this rule.)  
(INDAREA)
- INDS=n - Specifies length of index area in main storage (INDA keyword), where n is the length, in bytes. The length must be at least 256 bytes and in addition, a multiple of 256. Required for all index operations.  
(INDSIZE)
- IOA1=symbol - Required to specify the location of the I/O area where symbol (address) is the location. Must be half-word aligned. Must be greater than or equal to 256 bytes, a multiple of 256, and consistent with the BFSZ specification. Must immediately follow the index buffer (INDA) if it is specified. Must immediately precede the secondary I/O buffer (IOA2) if it is specified, unless index operations are not to be performed. A file which can perform index operations must have all buffers contiguous.  
(IOAREAL)
- IOA2=symbol - Specifies the location of additional I/O area, where symbol (address) is the location. Must also be halfword aligned and of the same size as the required area specified by the IOA1 keyword. If index operations can be performed, this buffer must immediately follow the primary I/O buffer (IOA1). Use of a secondary buffer is only permitted when performing sequential output (keyed or unkeyed) or unkeyed sequential input operations.  
(IOAREA2)

IORG=(r) - Required to specify the general register to be used to  
(IOREG) point to the current record when the user is not referencing records in the work area, where r is the number of general register. Registers 2 through 12 are available. Either IORG or WORK must be specified, but not both. (If both specified, WORK will be used).

KARG=symbol - Specifies the field in the user's program where he will  
(KEYARG) place the keys to effect retrieval of records, where symbol (address) is the location of this field. The length of the KARG area is equal to largest key length plus 3 (6 minimum). Required for all index operations.

$$\text{KEY}_n = \left( s, [1], \left[ \left\{ \begin{array}{l} \text{NDUP} \\ \text{DUP} \end{array} \right\} \right], \left[ \left\{ \begin{array}{l} \text{NCHG} \\ \text{CHG} \end{array} \right\} \right] \right)$$

Specifies one of up to 5 keys for an indexed file ( $1 \leq n \leq 5$ ). Permitted size(s) is 1 through 80 bytes. Location (1) specifies the number of bytes preceding the key. If location is omitted,  $\emptyset$  assumed for fixed records, 4 for variable. DUP specifies that duplicate keys are allowed (NDUP indicates they are not allowed and is the default). CHG specifies that key can change during update (NCHG indicates it cannot and is the default.) Required for all index operations unless user wishes to "accept" the key specifications that were employed to create the file. In that case, no KEY<sub>n</sub> specifications should be present.

LOCK=NO - Specifies that the file lock applied to a lockable file at OPEN time be set for read-only and that no output functions be allowed to the file. If omitted from the DTF for a lockable file, a write-only lock is set, and no other task may have access to the file while it is open under this DTF. Ignored if specified for a non-lockable file.

(A lockable file is one which has been assigned the 6-character prefix to the file ID, using the LBL job control statement).

MODE = SEQ - DTF is set for sequential operations should corresponding positional parameter be defaulted on OUTPUT or INPUT macros. (Default case).

RAN - Random operations  
RANH - Random with hold (if appropriate) operations

- OPTN=YES  
(OPTION) - Specifies that the sequentially processed file is an optional file: one the user anticipates will not invariably be present for every program execution. When specified for file not allocated to a device by the job control DVC statement, transfers control to the user's EOFA routine on the first issue of an input function or inline and with no error upon issuing an output function.
- PROC = KEY - DTF is set for keyed operations (index and data) should corresponding positional parameter be defaulted on OUTPUT, INPUT, or SELECT macros. (Default case)
- UNK - Non-keyed operations (data only)
- INDO - Index only operations
- RCB=NO - This specification only applies to files which are being newly created and it indicates that each record is not to contain a record control byte. Therefore, the DELETE macro will not be permitted. (The default is that each record will contain an RCB.) At close time, the format label will be marked to indicate whether or not the RCB is present. For existing files, the format label indication will override this DTF specification. The RCB is also necessary in order to create a mixed file (e.g., one which contains unkeyed records as well as keyed or index only records).
- RCFM=FIX  
(RECFORM) - Specifies that fixed length records will be used. This is the default case should the keyword be omitted.
- VAR - Specifies that variable length records will be used. The record size specification will pertain to a slot size where the first 4 bytes of the slot are overhead, and the first data byte is the fifth of the slot.
- RCSZ=n  
(RECSIZE) - Specifies the length of each record, where n is the length measured in bytes. This keyword is always required. (If variable records, specify the maximum size).

The record size specification should include the 4 byte overhead required for variable length records but should not include the 1 byte RCB required for the delete or intermix (i.e., keyed and unkeyed records) capabilities. (If the RCB is requested along with variable length record support, the third byte in the 4 byte overhead will be used as the RCB.)

- SKAD=symbol - Specifies the location in the user's program into which he loads the relative disk address for use in processing files by relative record number. The form of the record address is a 4 byte value. The first record is relative record #1. This keyword is always required.  
(SEEKADR)
- VRFY=YES - Specifies that Data Management is to check parity of output records after they have been written to disk. Necessarily increases execution time for output functions by about one rotation period per block. If bad parity is detected, Data Management sets the output parity check flag (byte 2, bit 2) in filenameC and transfers control to the user's error routine or to him inline. If omitted, no output parity verification will be done.  
(VERIFY)
- VMNT=ONE - Specifies that the file is to be processed with only one volume online at any time. A file which is created in this manner must be processed likewise and files can only be processed with one volume online at a time if they were created that way. Non-keyed random operations will not be permitted.
- WORK=YES - Specifies to Data Management that the user will be processing input or output records in a work area and not in the I/O area. The IORG Keyword cannot be specified when the WORK Keyword is specified. The address of the work area is specified with each issue of the appropriate macro. Required for all output and keyed update and delete functions.  
(WORKA)

4.4.2 Imperative Macros

All functional capabilities are initiated by issuing the appropriate imperative macro instruction. Imperative macros are supported to perform file initialization and termination, I/O processing, and dynamic file table modification. All error and exception conditions are reported to the user as defined by the ERRO keyword parameter.

4.4.2.1 OPEN Macro

The OPEN imperative macro initializes the data file and DTF table for subsequent processing. Standard labels are processed and validated; the DTF table is validated and completed for subsequent file access. If the DTF supports index operations, OPEN establishes KEY1 as the initial key of reference.

FORMAT:

label	OPEN	{	filename-1 (1)	}	[.....,filename-n]
			1		

positional parameter 1: always required.

REPLIES:

Reports of unsuccessful completion are:

- Invalid DTF
- Invalid DTF specification
- Illegal Record Size
- Illegal Block size
- Illegal key specifications
- Open issued to an opened file
- FCB not found/invalid
- Format - 1 label not found
- Partition invalid for specified DTF

## 4.4.2.2

CLOSE Macro

Upon completion of file processing, the CLOSE macro is issued to complete and/or terminate processing of the file. All standard file labels are created or updated. Further access of the file is inhibited.

FORMAT:

label	CLOSE	{	filename-1	{	[ ,.....,filename-n ]	}
			(1)			
			1			

positional parameter 1: always required

REPLIES:

Reports of unsuccessful completion are:

NONE

## 4.4.2.3

FEOV Macro

The FEOV macro provides the capability to terminate processing on the current volume of the file for files processed with only one volume online at any time (see VMNT keyword parameter). If the FEOV macro is issued for a file with all volumes mounted the macro is ignored.

FORMAT:

label	FEOV	{	filename	}
			(1)	
			1	

positional parameter 1: always required

When FEOV is issued, the current volume is closed and a mount message is issued requesting that the next volume of the file be mounted. The new volume of the file is opened for processing and subsequent macros continue processing on the new volume.

## REPLIES:

Reports of unsuccessful completion are:

Hardware error accessing FCB or ERB

## 4.4.2.4

OUTPUT Macro

The OUTPUT macro provides for placement of a new record in a file.

## FORMAT:

label	OUTPUT	$\left\{ \begin{array}{c} \text{filename} \\ (1) \\ 1 \end{array} \right\}$	,	$\left\{ \begin{array}{c} \text{workarea} \\ (\emptyset) \\ \emptyset \end{array} \right\}$	,	$\left[ \left\{ \begin{array}{c} \text{UNK} \\ \text{KEY} \\ \text{INDO} \end{array} \right\} \right]$	,	$\left\{ \begin{array}{c} \text{SEQ} \\ \text{RAN} \\ \text{RANH} \end{array} \right\}$
-------	--------	---	---	---	---	--	---	---

positional parameters 1 and 2: always required

positional parameters 3 and 4: The concept of a "long-form" and "short-form" macro will be introduced here. The long-form implies that certain optional parameters are all specified (e.g., for the OUTPUT macro, parameters 3 and 4). The short-form implies that none of these special optional parameters are specified. If a macro has more than one long-form parameter, and if one is specified but not all, the macro will not be expanded. If the short-form is employed, the defaults will be obtained from indicators within the DTF. These indicators can be set by use of the PROC and MODE keywords of DTFMI. They can be changed by use of the PROC and MODE keywords of the APPLY macro. Use of the long-form of the macro will also change these indicators in accordance with the long-form parameter specifications.

- . UNK - record is not to be indexed. (Primarily for files without keyed records or index-only entries, but if the RCB exists, may be used to place a non-indexed record in a file which contains keyed records and/or index-only entries.
- . KEY - record is to be indexed according to the key(s) of the file specification. There will be one index entry (which points to the record via a relative record number) for each key in the file.
- . INDO - an index entry (which consists of a key and 3 byte pointer) will be added to the file. Both the key and pointer must be supplied. (Positional parameter 4 is ignored).
- . SEQ - record is to be placed at end-of-file and its record number made available to the user.
- . RAN - record is to be placed in a relative slot according to the record number given in SKAD. However, this operation is sensitive to the presence or absence of the RCB. When present, an attempt to overlay an existing record will be rejected, with an error report. Also, placement beyond file end will cause any gap created to be filled with void records. When absent, these two services are not available.
- . RANH - (same as RAN specification).

The execution of OUTPUT does not affect the current sequential position or current reference key.

If the user has caused setting of the OUTF action switch (see APPLY macro), there is a follow up to the new record placement; consisting of reverting to the current sequential record and making that record available again.

REPLIES:

Reports of unsuccessful completion are:

- Illegal record size
- Illegal key value
- Overlay of existing record (if record control byte present)
- Undefined sequential positions, and OUTF requested
- Insufficient file space

If the operation is successful, status will note the keys where legal duplication has occurred. Also, the record number (of the new record) will be placed in the seek address field.

## 4.4.2.5

INPUT Macro

The INPUT macro makes a record available for processing. It also permits the user to state his intentions with respect to subsequent processing.

## FORMAT:

[ label ] INPUT { filename  
(1)  
1 }, { workarea  
(Ø)  
Ø }, [ INF  
MOD  
REP ], [ UNK  
KEY  
INDO ], { SEQ  
RAN  
RANH

positional parameter 1: always required

positional parameter 2: if specified, the record will be moved from the buffer to the area; otherwise, the record will be pointed to, in the buffer, by a specified register. Must be specified if there is intent to update a keyed record or when retrieving via the INDO specification.

positional parameter 3: optional specification; default is INF

- . INF - indicates an intent to retrieve the record for information purposes only; no intent to update or delete the record.
- . MOD - indicates an intent to modify the record in part. It is assumed that the user wants to inspect the record before changing it. Workarea or the I/O register will be employed according to user specification.
- . REP - indicates an intent to replace the entire record. The record will not be moved into the workarea, as it is assumed that the user already has a replacement in the workarea which cannot be overlaid.

positional parameters 4 and 5: long-form parameters (see positional parameters 3 and 4 under OUTPUT macro description).

- . UNK - calls for unkeyed retrieval
- . KEY - calls for keyed retrieval. Record is retrieved based on a specified key argument.

- . INDO - calls for retrieval of an index entry (key and pointer) based on a specified key argument.
- . SEQ - calls for a sequential access based on current sequential position (keyed sequential if KEY or INDO specified; next higher record number if UNK specified). Current sequential position will be modified.
- . RAN - calls for a random access per an argument provided by the user (in KARG for KEY or INDO; in SKAD for UNK). Current sequential position will be modified.
- . RANH - same as RAN specification except that the current sequential position will be held (not modified).

## REPLIES:

Reports of unsuccessful completion are:

Required work area not provided  
Record not found  
End of file reached  
Sequential position undefined.

If the operation is successful, status will show whether or not the succeeding record has a duplicate key in the current reference set. Status will also show whether or not the acquired record is a keyed record.

The record number (of the acquired record) will be placed in the seek address field.

## 1.4.2.6

SELECT Macro

The SELECT macro prepares for making records available in sequential order by key or by record number. It can also be used to change the key of reference.

## FORMAT:

```
[label] SELECT { filename } , { EQ } , { PKEY } , { UNK }
                { (1) } , { GT } , { KREF } , { KEY }
                { 1 } , { GE } , { } , { INDO }
                { } , { BOF }
                { EOF }
```

positional parameter 1: always required

positional parameter 2: required unless KREF specification in positional parameter 3 is used (in which case, positional parameter 2 is optional).

- . EQ - a no-find is returned unless an equal-key record is found, or a non-void record is found at the given location.
- . GT - a find is reported if a non-void record can be found with value greater than the given key or record number.
- . GE - a find is reported if either EQ or GT is satisfied.
- . BOF - for UNK, operates as a GE request with SKAD=1. For KEY or INDO, operates as a GE request with KARG=Ø.
- . EOF - for UNK operates as a request for highest numbered record; for KEY or INDO as a request for highest keyed record or index entry.

positional parameter 3: not required.

- . PKEY - selection is based on n leading bytes of the KARG space, n < key size. When this parameter is used, register Ø must be preset with the value of n. Cannot be specified in conjunction with the UNK specification. If not specified in conjunction with KEY or INDO, the full key will be used.

- . KREF - indicates that the key of reference is to be changed. Register  $\emptyset$  must be preset with a value  $n$ , where  $1 \leq n \leq 5$ . In addition,  $n$  must not exceed the number of keys in the file. This specification can be used in conjunction with positional parameter 2, if first, a key of reference change and then, a sequential preparation is desired.

positional parameter 4: long-form parameter. (See positional parameters 3 and 4 under OUTPUT macro description).

- . UNK - preparation is to be based on relative record number. Except for BOF and EOF, preparation is further based on the value given in SKAD.
- . KEY - preparation is to be based on the current reference key. Except for BOF and EOF, preparation is further based on the value given in KARG.
- . INDO - (same as KEY specification)

#### REPLIES:

If the select operation is unsuccessful, a no-find is reported.. The current sequential position becomes undefined, precluding a following sequential input. (Reference to an empty file also produces the no-find). If the operation is successful, the record number (of the record pointed to) will be placed in the seek address field, and for a SELECT which employs the index, the key (of the record pointed to) will be placed in the key argument field.

## 4.4.2.7

UPDATE Macro

The UPDATE macro causes the most recently retrieved record to be updated.

## FORMAT:

```
[label] UPDATE { filename } , { workarea }
                (1)
                1
                (Ø)
                Ø
```

positional parameter 1: always required

positional parameter 2: workarea must be used in all cases where the existing record is keyed or an index-only entry.

## REPLIES:

Reports of unsuccessful completion are:

Record was obtained for information  
 Illegal record size  
 Illegal key value

## 4.4.2.8

DELETE Macro

The DELETE macro is used to void the record most recently acquired. Marks the subject record as void, and voids any index entries pointing to the record. Cannot be used for files without the RCB.

## FORMAT:

```
[label] DELETE { filename }
                (1)
                1
```

positional parameter 1: always required

## REPLIES:

None

4.4.2.9 ERASE Macro

The ERASE macro is used to erase an entire file or to erase part of a file (which contains only non-indexed records) starting from a given relative record number. The RCB is not required to perform these functions.

## FORMAT:

```
[label]      ERASE      { filename } , { ALL
                        (1)   }      { PART }
                        1
```

positional parameter 1: always required

positional parameter 2: always required

- . ALL - Causes the entire file to be discarded. Can be used on file which contains any kind of record (e.g., keyed, unkeyed, index-only).
- . PART - Causes all records, beginning with a user specified record number (in SKAD), to be discarded. The record, which corresponds to the record number in the seek address field, and all records whose record numbers are greater in value, will be discarded. This form of the ERASE macro can be issued against a file which contains only unkeyed records (i.e., no keyed records or index only entries).

## REPLIES:

Reports of unsuccessful completion are:

Invalid Macro error (if ERASE PART is specified and file contains keyed records or index entries).

## 4.4.2.10

APPLY Macro

The APPLY macro is used to apply changes to the DTF, which will have an effect on subsequent processing. The changes will be effective until changed by another APPLY macro call. Code will be generated in line at assembly time.

## FORMAT:

```
[label]  APPLY  { filename } , { IORG=
                    (1)      }   { WORK=
                    1        }   { OUTF=
                                }   { MODE=
                                }   { PROC=
                                }
```

positional parameter 1: always required

positional parameter 2: always required

- . IORG=(r) - Will either change the I/O register being employed or change the DTF from workarea to I/O register mode. (See IORG keyword in DTF description for additional information).
- . WORK=YES - Will change the DTF from I/O register to workarea mode. (See WORK keyword in DTF description for additional information).
- . OUTF=YES - Will set an indicator such that following an add to the file, the last record retrieved will be read back in order to duplicate the conditions which existed before the add. (OUTF=NO will turn off the indicator).
- . MODE= { SEQ  
          RAN  
          RANH } - Will change the DTF indicators which are used for the short-form macros. (See MODE keyword in DTF description, and positional parameters 3 and 4 under OUTPUT macro description).
- . PROC= { UNK  
          KEY  
          INDO } - Will change the DTF indicators which are used for the short-form macros. (See PROC keyword in DTF description, and positional parameters 3 and 4 under OUTPUT macro description).

5.0

FUNCTIONAL DESCRIPTION

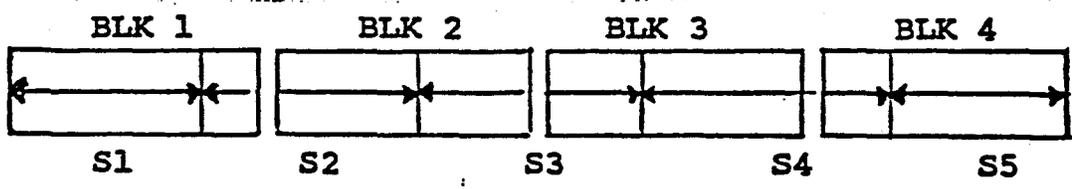
MIRAM provides two processing modules. The modules are reentrant; so they are limited to modifying core locations in the file DTF area, and user areas that are defined in the DTF.

There are also several transients that are called as their services are required. In general, these are used to perform services that are infrequently needed.

5.1

File Format

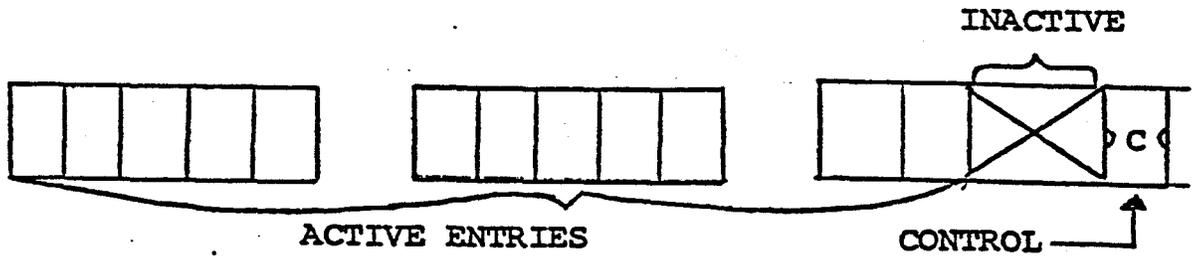
All MIRAM files consist of two partitions, a data and an index partition. Initially, there will be no allocations made to either of the partitions. The first output function which references a partition, will cause an initial allocation made to that partition. For example, if the first record output to the file is unkeyed, the data partition only will at that time be extended to receive an allocation. Then, if a keyed record or index only entry is output to the file, the index partition will be extended to receive an allocation.



The data partition consists of 256 byte unkeyed physical blocks. User record slots are required to be of uniform size, and the size chosen is not required to conform to the physical block size. Consequently, it is possible for records to span physical block boundaries as illustrated by the above diagram.

The index partition has 256 byte keyed blocks. In both partitions, the processing programs make use of the SAT facility for transferring several physical blocks with a single access.

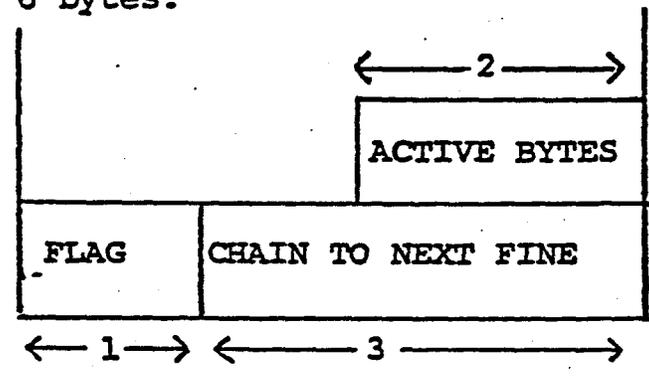
Part of the work in the index partition is done by hardware key search, and part by multi-block transfers. The fine level of index is treated as a chain of multi-sector blocks, not formatted for search. A three-sector fine block is diagrammed below:



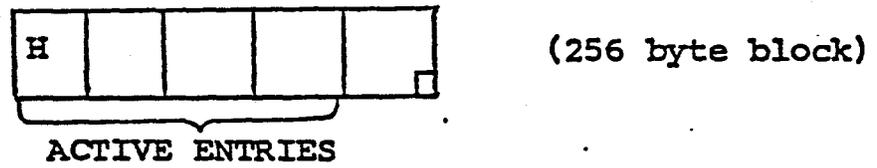
5.1

File Format (Cont'd)

Each active entry consists of a key plus a 3-byte pointer which gives the file relative record number of a data record. The number of active entries varies from block to block. The control area consists of three fields, totalling 6 bytes:



The coarse and mid levels of index are formatted for hardware key search. Areas subject to search may have insufficient entries to fill out a track. Hence, there must be suitable dummy entries to prevent false hits. A partially filled coarse/mid sector is diagrammed below:



For hardware search, the high key entry of the sector must be in first position. Remaining entries are in descending sequence. The final byte of the sector is used to contain the current number of active bytes.

When there are multiple keys, each has its own coarse level in the index partition. All hardware searchable sectors must have a front key area equal in size to the longest key of the group. For shorter keys, the storage space is filled out with appended FF bytes.

For handling of keys where duplicates are allowed, in the coarse and mid levels of index, an extra byte between the key and pointer is used.

## 5.2 Function and Subroutine Procedures

### 5.2.1 Addition of New Keyed Records

This function performs two actions; placement of an index entry in fine level index, and placement of the new data record at the end of the data string. The processor first assures sufficient index space, then tests for orderly/disorderly load, and calls in the indicated transient. The transient handles all index modification, and returns to the processor. Processor coding handles the placement of the data record, using the same code as for placement in a non-indexed case.

If there are multiple keys, the processor follows another path. On this path, there is no checking for a high key situation. The processor calls n times on the transient that will add anywhere. If all keys are added successfully, the path then leads to processor code which appends the new data record to the data string.

If an illegal duplicate key is found, a transient is called to undo the part of the process already done. This deletes any index entries already made.

### 5.2.2 Random Retrieval

This function retrieves a record by key or by relative record number. All coding for this action is resident in the processor. If it is keyed retrieval, a subroutine conducts the index search. At conclusion, the main line coding uses the reported relative record number to retrieve the desired record.

Random retrieval can be followed by an update or delete function.

### 5.2.3 Preparation for Sequential Retrieval

The SELECT instruction allows the user to set the low limit of a range of records to be retrieved in key sequence or consecutive sequence. If key sequence is demanded, a transient is called to perform the key search.

SELECT does not provide a record to the user. Instead, the user's first input function provides the first record of the range.

#### 5.2.4 Sequential Retrieval

After an input function, the user may issue an update or delete function. Else he may issue another input function, passing on without update. Sequential input coding is resident. While in a sequential mode, the user may also request that a new record be added. This requires that Data Management perform the add, then revert to the sequential mode as though there had not been this disturbance. This action is performed in a transient, because of the large amount of coding required.

#### 5.2.5 Deleting a Record

Coding to mark the record is resident. Any required index modification is transient.

#### 5.2.6 Updating a Record

Coding to update a record is resident. Any required index modification is transient.

#### 5.2.7 Erasing all or part of a file

Coding to erase is resident.

## PERFORMANCE

Every effort has been made to provide best possible performance under the constraints imposed by small buffer sizes.

First priority has been assigned to performance during retrieve operations. Consecutive retrieve is accomplished without reference to index, and takes advantage of any extra buffer space provided by the user. Random keyed retrieve coding is part of the resident keyed modules, to avoid burdening each retrieve with a transient call. Keyed search from coarse to fine level is expected to cost 3 accesses for most records of a sizeable file, and 2 accesses for the remainder.

By the design chosen, operations that add new records to the file maintain the index in usable form. This eliminates the cost of an index sort at program termination. However, the record-by-record cost of this method is greater than the immediate cost of placing a new record without index maintenance. It is believed that the index maintenance method will show good results, particularly when a file is subject to growth by daily addition of records.

Disorderly load is essentially a random add process where the user gives a series of add commands without interspersing commands that would require writing an incomplete data buffer to free the space for other use. Thus, there is some performance advantage in the concerted series of adds.

Orderly load is estimated to be better than twice as fast as disorderly.

This results from elimination of search for the place to put an index entry, and from the concerted series advantage mentioned.

## 7.0 COMPATIBILITY

MIRAM must provide file and record handling services for disk files such that OS/3 COBOL can provide the functions available to the user of ANS'74 COBOL. MIRAM must also be able to access and create IRAM files.

## 8.0 CONVERSION

Other software components requiring modifications are COBOL, SORT, DATA UTILITIES. These components must be capable of accepting a user's specifications for a MIRAM file and providing the interfaces to process the file according to the user's wishes. They must be modified to produce suitable MIRAM DTF tables, and to provide suitable imperative calls.

## 9.0 DOCUMENTATION AND SUPPORT

Sections added to the User Guide UP-8068 and the Programmer Reference UP-8159, will describe the MIRAM facility, and explain its use.

Program listings and flowcharts will be maintained to assist in program support.

## 10.0 RESTRICTIONS

- Search keys may not exceed 80 bytes.
- Buffers must start on half-word boundaries.
- Search keys may not contain any FF bytes on fixed sector disks unless these disks have the "binary key" feature.

## 11.0 MAINTAINABILITY AND RELIABILITY

OS/3 Physical IOCS is the agency that detects and attempts to correct hardware errors during disk reference. When an uncorrectable error occurs, this fact is reported through the chain: PIOCS to SAT to MIRAM to the user of MIRAM. At termination of a user program, a CLOSE ALL transient is called, to close any files found to be still open. Proper closing of MIRAM files is vitally necessary when there has been sequential loading or sequential retrieval with updating. In these cases, MIRAM will frequently be in a delayed-write status, where changes in content of the buffer in main memory have not yet been written to disk.

## 12.0

RELATED DOCUMENTS

IRAM CPSD (W.P. #675), PD A-43058 (ISAM) and PD A-43061 (SAT) may be read for background information on error handling, multi-partition files, and VTOC. However, they currently contain no reference to MIRAM as such.

- Otherwise, there are the IBM System 3 manuals listed below:

GC21-7571-2 Disk Concepts and Planning Guide  
GC21-7512-6 Control Program Reference Manual  
GC21-7562-2 Model 10 Disk System  
SC21-7504-5 System 3 RPG II Reference Manual  
SC21-7595-0 System 32 RPG II Reference Manual

## 13.0

STANDARDS DEVIATIONS

This component is designed to provide compatibility of services to ANS'74 COBOL and IBM System 3. Deviations from any applicable Sperry Univac Standards are not established at this time. Adoption of the solidly packed data string with records spanning physical block ends is not a usual practice in Univac Data Management Systems. However, this is not known to violate a Sperry Univac Standard.

The method was adopted in order to provide conservation of disk space in the System 3 manner. It also helps in support of the System 3 concept of handling a file with different size buffers at different times - a concept that negates the more common concept of a fixed size logical block.



APPENDIX A



1375 DOCUMENTATION COMPONENT: MODULE: PAGE:

Glossary of Terms:

A.

appender string

A string of records that is enlarged only by placing a new record at end-of-string.

B.

block

The portion of a file transferred into or out of main storage by a single access.

block splitting

A technique for maintenance of inserter strings. When insertion into a block causes overflow, a fresh block is chained into sequence, and the records are "split" between the two blocks.

buffer

An area in main storage for handling a block of data. Must not be smaller than the blocks to be handled.

C.

Consecutive sequence

The sequence in which records of a string are originally passed to Data Management by the user. In some cases, differs from ascending key sequence.

coarse level

The level of a hierarchical index system that has the least number of entries, which subdivide the file into large sections.

D.....

direct addressing

Retrieving a specific block or record from disc storage by a single access, using numeric values given in a field.

E.....

extent

A set of contiguous tracks on disc assigned exclusively to one file. Several extents may be required to provide space enough for a file.

F.....

field

One or more contiguous characters, normally comprising a single unit of information.

file

A delimited storage space having an identifying filename; useful for subdividing the entire data mass into manageable groups. Also, the data residing in such a storage space.

fine level

The level of hierarchical index systems that has the greatest number of entries, providing the most detailed subdivisions of the file.

I.....

inserter string

A string of records that may be enlarged by placing a new record between existing records.

M.

mid level

Any level of hierarchical index system that falls between the coarse and fine levels.

P.

partition

A file subdivision, which is required to have uniform block specifications. OS/3 data management provides partition-relative block addressing, and individual partition extension capabilities.

pointer

A field containing a value for direct addressing.

R.

record

The collection of contiguous characters designated by the user to data management as such, for handling as a unit. Record size must not exceed block size.

S.

slot

A filing space that may or may not be occupied by a record. Slots are uniform size, number consecutively from 1 to n. For variable size records, slot length is maximum record size + 4. For fixed size records, if there is an RCB, slot length is record size + 1; otherwise, slot length is record size.

string of records

A series of records having exclusive use of the blocks occupied, and retrievable in sequence by a series of "get"

instructions.

V.

volume

The largest physical unit for data storage, such as a tape reel or disc pack.

APPENDIX B  
(to be supplied)



## SPECIFICATIONS DISTRIBUTION

ATKINS, J.	E8-126
BENEDETTO, J.	1C1-NE/6
BEJARANO, R.	A-2
BRENNAN, J.	A-2
BURDICK, W.	M8-152 (COVER ONLY)
CAROSELLI, J.	A-2
FAZAH, P.	M8-153
FINK, W.	E8-126
FREEDMAN, B.	M11-119
GROSSMAN, R.	E2-128B
HARTSEL, R.(3)	M8-152
HEITNER, J.	E2-128B (COVER ONLY)
HUNT, W.	M11-111
HOGAN, F.	A-2
HUX, R.	A-4 (COVER ONLY)
JEANS, C.	A-2
LADSON, R.	A-4
LINDINGER, J.	E8-126
LUDWIG, R.	M8-149
MACKSON, D.	E2-s28B (COVER ONLY)
MC FADDEN, M.(2)	M8-152
PRATT, R.	M11-111
REYNOLDS, W.(2)	M11-107
SCHWALM, E.	C2-NW/15
SCOTT, B.	E2-128B
SWEENEY, J.	M11-111
THEODORE, M.	M11-111
MASCIANTONIO, M.	A-2
OS/3 DEVELOPMENT(7)	M8-142



INTERCOMMUNICATION

cc Distribution

FROM NAME & EXT: F. Buttschardt X3019

LOCATION & DATE: Blue Bell, 10/24/77

DEPARTMENT & M.S.: OS/3 Software Development

SUBJECT: OS/3 RELEASE 6.0  
UNIT SPECIFICATIONS

The following unit specification is being submitted for your review and approval:

### MULTI-INDEXED RANDOM ACCESS (MIRAM)

Please document your comments on the standard Documentation Review Form (sample enclosed) and/or in the review document itself. In the latter case, reference your comments on the Documentation Review Form.

A design review meeting is scheduled for this facility on 11/9/77 in CLASSROOM #2 AT 2 P.M.

Due to the limited nature of the changes described by this specification, no design review meeting will be scheduled.

The purpose of the design review meeting will be to discuss the functions and interfaces of the enhancements. It will not be directed toward the implementation techniques except as they relate to the interfaces with other components.

Please submit all comments to D. WINSTON by 11/10/77. After consideration of these comments and those raised in the design review, the specification will be updated and formal approval will be requested.

Those persons whose names are asterisked on the Distribution List are expected to respond. Others on the list are invited to respond if they desire to do so.

F. Buttschardt

F. Buttschardt, Manager  
OS/3 Software Development

FB/lor  
Enclosure



## INTERCOMMUNICATION

TO: P. Fazah  
 F. Hogan  
 R. Hux  
 R. Ludwig  
R. F. Frankson

FROM (NAME & EXT): F. Buttschardt 3019

LOCATION & DATE: Blue Bell

DEPARTMENT & M.S.: OS/3 Development

CC:

SUBJECT: OS/3 RELEASE 6.0  
 SPECIFICATION APPROVAL

The following unit specification is being submitted for your final review and approval:

## MIRAM

This specification was previously submitted for review by your department and comments that were returned to us have been reviewed and, where appropriate, incorporated into this version. The remaining comments have been addressed directly with the reviewers.

Please sign the attached approval form and forward to T. Gannon by: 1/27/78.

Questions regarding this specification should be directed to:  
E. LUDT.

F. Buttschardt

F. Buttschardt, Manager  
 OS/3 Software Development

FB/lor

Distribution\*

R. Boos  
 R. Bejarano  
 J. Benedetto  
 J. Lindinger  
 W. Reynolds  
 M. McFadden (2)  
 D. Mackson  
 E. Schwalm  
 J. Sweeney

OS/3 Development  
 \*Specification attached



Le:

Page:

MIRAM

Document No.	Original Issue Date	Revision No.	Revision Date	Type of Documentation
0-854		1	1/6/78	Rel. 6.0 Unit Specification

SPECIFICATION APPROVAL

---

P.G. Fazah, Director  
Software Integrity and Services

---

F. Lee Hogan, Director  
Software Design and Control

---

R.E. Hux, Director  
Languages

---

R.M. Ludwig, Director  
Basic Systems

---

R. F. Frankson  
Major Systems

