19.1 GENERAL

TOSS Data Management functions are implemented as a number of special tasks running on priority level 49.
There are at least two DM tasks per disk driver present in the system.

When a data file has been assigned it will be treated in a similar way to a common I/O device. That is, the file code of the data file is found in CDTAB or TTAB, together with the FWT address. The first part of the FWT has the same layout as a DWT. When a Data Management function is called, control is give to an activation handler (TIODM) within the monitor. This activation handler in its turn activates a DM task for the disk drive.

Each DM request locks all the files it uses and in this way only one user at a time can access a particular file. In other words, it is only when a DM file request is completed that the next DM request for that file can be handled.

19.2 ACTIVATION HANDLER (TIODM)

The activation handler (TIODM) is called in the same way as a driver via an address pointer in the FWT. It checks the order code and can also detect some error conditions, such as exclusive access conflicts and end-of-medium or end-of-file. For sequential and random access methods TIODM also computes the sector number and the relative offset within that sector of the record to be accessed. All the necessary information is stored in the FWT and the FWT address is the only parameter transferred at the final activation of the DM task.

In an indexed file structure only the data file is associated with a file code. Index files are identified internally via a file number. The FWT's of all files in a file structure are locked during a request. External requests to this structure are queued until FWT's are released. Internal DM requests from other DM tasks are not queued.

19.3 DATA MANAGEMENT TASK (DMTASK)

The DM Task contains all the physical I/O routines necessary to perform the various data management functions. Two DM tasks are defined for each disk driver.

DM requests that need data on more than one disk drive cause 'internal DM requests' from one DM task to another.

All I/O requests are made in wait mode. Because there are two DM tasks per driver a request for a drive which already has another request active can start its processing up to the point where it needs the disk, when it will be queued if the drive is still busy. There can thus be an overlap between processing and disk access.

The DM Task makes use of the common sector buffer pool in order to obtain a free sector buffer. If the requested sector is already in memory, this buffer is used and the sector does not need to be read into memory.

Apart from I/O handling, the DM Task also performs the following functions:-

* Moving records to/from user record area from/to sector buffer.

* Updating Exclusive Access buffers and Current Record Number area for the FWT.

* Setting Effective Length, Return Code, and Control Word in the user ECB area.

19.4 MONITOR TABLES

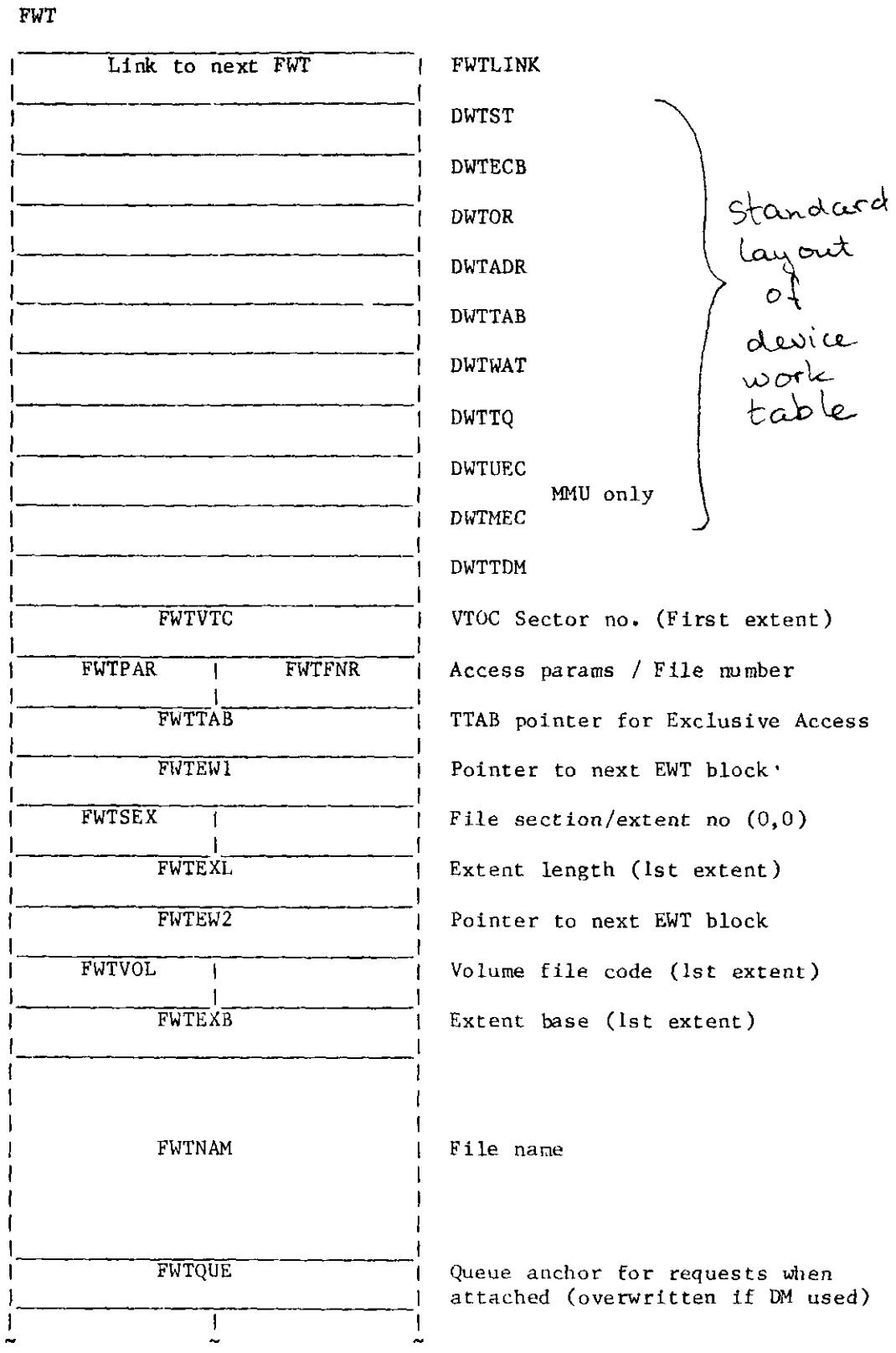
Initially all DM file codes are connected to a special FWT (in TIODM). When a file is opened, a free File Descriptor Block (FWT) is found by File Management, fetched from the FWT pool reserved at SYSGEN time, and the fields in memory are filled in.

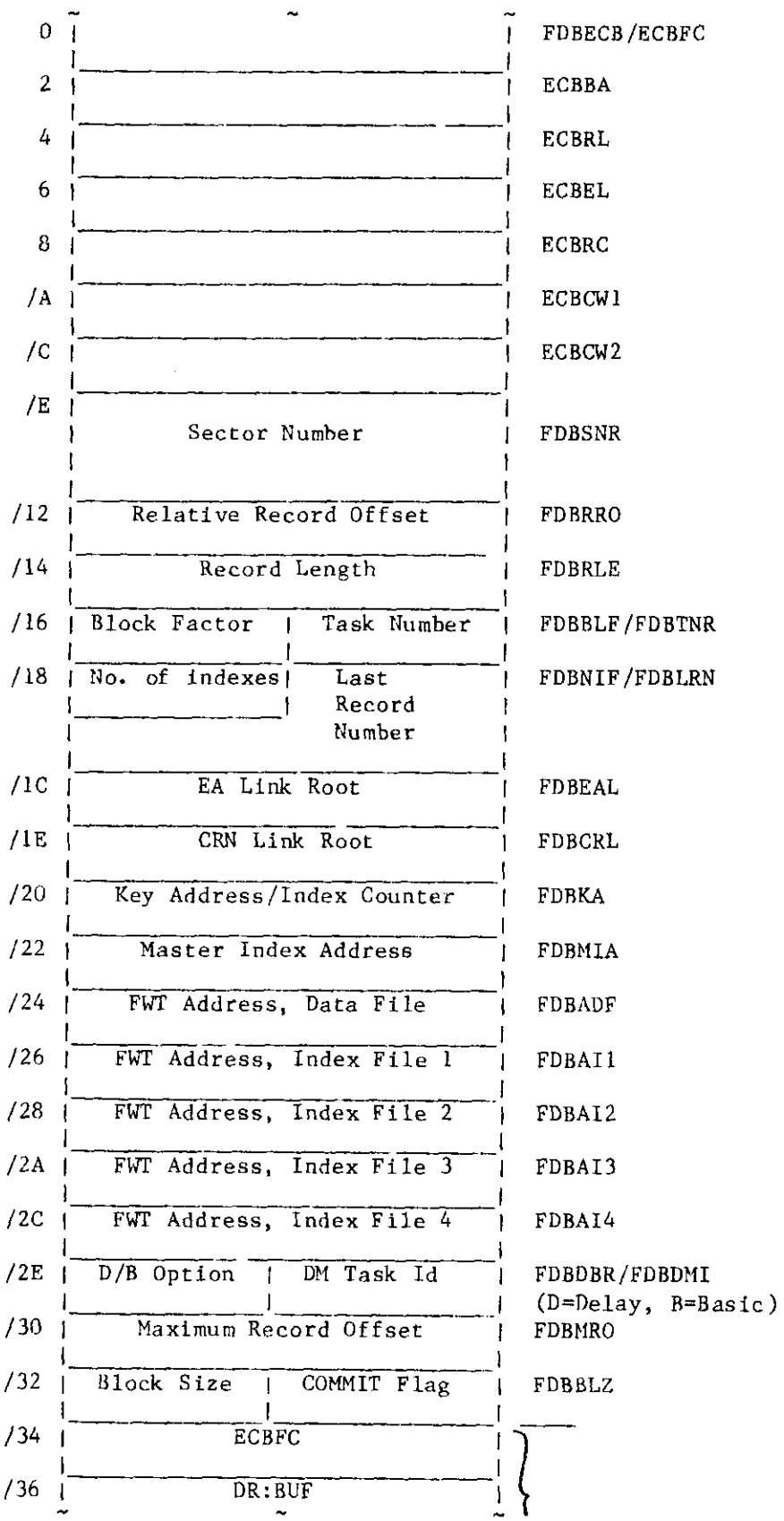
An entry, two words, is made in CDTAB (or TTAB) for the assigned file code. The file codes in CDTAB can be reached by all tasks in the system. Index files do not have file codes.

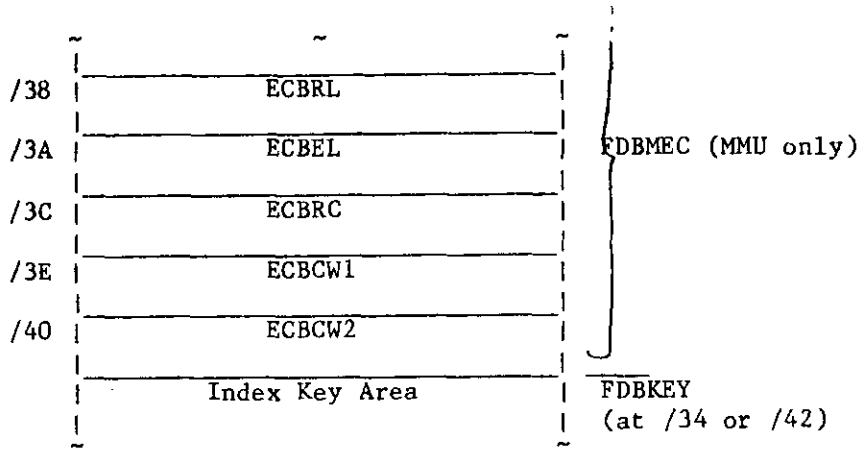
There are two different assign requests and one physical file can therefore be assigned twice with the same file code. The FWT is created at the first request and merely updated if a second request occurs. The CDTAB (or TTAB) is given a separate entry, two words for each request. The first character of the CDTAB/TTAB entry contains an index that identifies the corresponding assign type.

The Monitor Tables are released when a 'Close File' request is issued.

19.4.1 File Work Table




0
FDBECB/ECBFC
2
ECBBA
4
ECBRL
6
ECBEL
8
ECBRC
/A
ECBCW1
/C
ECBCW2
/E
Sector Number
FDBSNR
/12
Relative Record Offset
FDBRRO
/14
Record Length
FDBRLE
/16
Block Factor
Task Number
FDBBLF/FDBTNR
/18
No. of indexes
Last Record Number
FDBNIF/FDBLRN
/1C
EA Link Root
FDBEAL
/1E
CRN Link Root
FDBCRL
/20
Key Address/Index Counter
FDBKA
/22
Master Index Address
FDBMIA
/24
FWT Address, Data File
FDBADF
/26
FWT Address, Index File 1
FDBAI1
/28
FWT Address, Index File 2
FDBAI2
/2A
FWT Address, Index File 3
FDBAI3
/2C
FWT Address, Index File 4
FDBAI4
/2E
D/B Option
DM Task Id
FDBDBR/FDBDMI
(D=Delay, B=Basic)
/30
Maximum Record Offset
FDBMRO
/32
Block Size
COMMIT Flag
FDBBLZ
/34
ECBFC
/36
DR:BUF



FWTLINK This word links all physical FWT's which are reserved at system generation. Bit 15 (the use bit), if set to 1, indicates that the block is in use.

DWT Standard layout of device work table.

DWTTDM Contains the TTAB address of the calling task.

FWTPAR/FWTFNR /12

NV	B	NC	File Number
0	1	2 ... 7	8 ... 15

NV : 1 indicates 'New volume loaded' has been detected for this file. All requeste for this file code will be rejected withthe appropriate return code. To be able to access this fileagain, all tasks that have opened the file must close itand open it again successfully.

B : 1 indicates that physical write orders must be changed to basic write by File Management.

NC : Number of successful opens performed for file (max 63).

FWTEW1 Extent work table chain

0	EWTLNK	Link to next EWT block
2	EWTVOL	Volume file code
4	EWTEXL	Extent length

0	EWTLNK	Link to next EWT block
2	EWTVOL	Volume file code
4	EWTEXB	Extent base

ECB Layout of normal ECB.
ECBFC is the filecode of the disk (F0, F1, etc.).

FDBSNR Physical sector number; sector content is actually stored in the block buffer.

FDBRRO Relative offset of the record in the block buffer in bytes.

FDBRLE Record length in bytes.

FDBBLF Blocking Factor as specified with the Create file utility.

FDBTNR Number pointing to a word in TCTAB.
The entry contains the pointer to the current task table (TTAB). From TTAB the task identifier can be obtained.

FDBNIF Number of index files (0 when FWT belongs to an index file).

FDBLRN Last record number pointer (starts with one, 3 bytes long).

FDBEAL Exclusive Access link root.

FDBCRL Current Record Number link root.

FDBKA Key address/location in the data record.

FDBMIA Address of master index in memory.

FDBADF FWT address of the data file.

FDBAIn FWT address of index files 1, 2, 3, & 4 respectively.

FDBDBR D - 2 Means delay function included. A write on application level will result in an update in the block buffer. Later the buffer is written to disk. Extra block buffers are required, since one buffer per file is kept until the file is closed.
(Not used for CREDIT).

B - 1 Means basic write, no check is performed.
(Not used for CREDIT).

FDBDMI DM Task Identity.

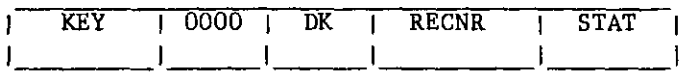
FDBMRO Maximum record offset in block buffer.

FDBBLZ Block Size.

FDBMEC User ECB (MMU only) copied to this area (standard layout).

FDBKEY Contains the record key for indexed access.
Size is defined at SYSGEN time.

19.4.2 Layout of a Record in an Index File



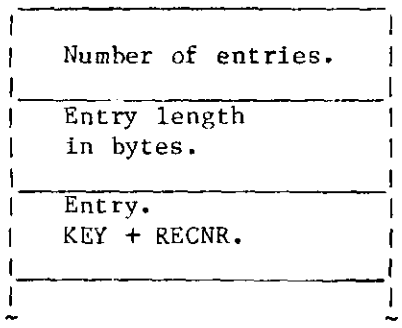
KEY Key = A string of 1 to N bytes, left adjusted and padded with blanks. It is used as a record identification.

DK Duplicate key.
One byte which contains a binary value specifying the minimum number of leading bytes in a key which are identical with the next key in the index file. When the next key is identical, this equals the key length.

RECNR Record number.
Three bytes containing the logical record number of the record in the data file, with corresponding key value.

STAT Status.
One byte indicating whether the record is used or free.
When used the value is X'FF'; when free it is X'00'.

19.4.3 Layout of a Master Index in Memory



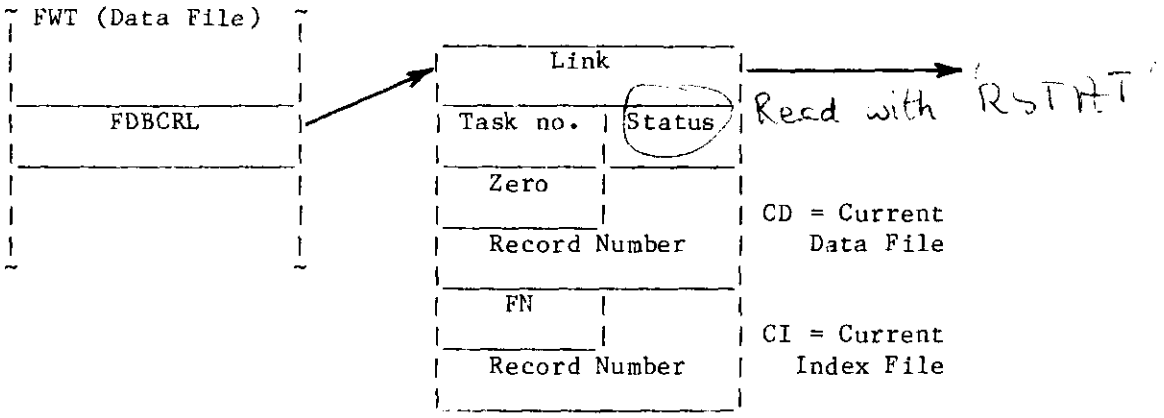
Number of Entries. One word containing a number indicating the entries in the master index.

Entry Length. One word containing a number indicating the number of bytes used per entry for the key and record number.

Entry. Each entry occupies 1-N bytes for the key, plus three bytes for the record number (Record number in index file).
The last entry will contain all ones in the key field.
When only one entry is present, the key value will be all ones, and the record number is set to one.

19.4.4 Current Record Number Handling

FN = File Number of Index File.
Task number is used as an index to TCTAB.



One CRN buffer (6 words) is allocated for each user task per file structure (data file with/without index file). Other data management requests from the same task to the same structure will use the same CRN buffer. The buffer is released after closing the file. When more than one task is using the same file structure, the CRN buffers are linked via the first word in the buffer.

LINK This word contains a pointer to next currency buffer. Zero indicates the end of the chain.

Task no. Contains in one byte a number which is a byte displacement from the second word in TCTAB (ie. the word following the length word) to a word containing the address of the TTAB for which this current record number is valid.

Record nr. Three bytes containing the record number in the data file.

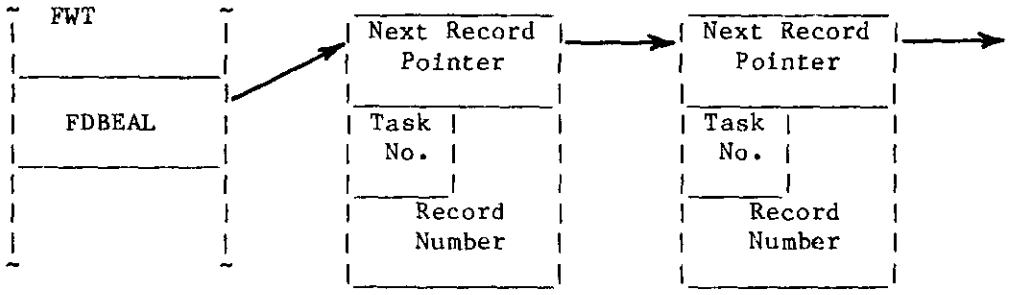
FN File number of the index file.

Record nr. Three bytes containing the record number in the index file.

19.4.5 Exclusive Access

Since files may be in simultaneous use by more than one task, it is important that these tasks are prevented from updating the same record at the same time, or otherwise interfering with each other. Therefore it is possible for a task to request Exclusive Access (EA) to a record or records in one or more files.

One EA buffer (3 words) is allocated for each record of the data file set under exclusive access. The EA buffers are linked via the first word in the buffer. The word FDBEAL (the EA Link Root) in the file descriptor block points to these chained EA buffers.



Whether or not EA is required is specified by the user at the time of issuing a Read Request, and EA is relinquished via Release Exclusive Access, which does not involve any physical I/O. EA is not applicable to Index files.

If an EA request for a record cannot be honoured because that record is already being accessed by a task having EA, then a Record Protected status reply will be returned to the requesting task. This is done in order to avoid deadlock situations. A deadlock may still occur if the requesting task, having received the Record Protected status reply, does not release EA on any other records to which it already has exclusive access.

Next Record Pointer. Pointer to the next EA block. When zero, the end of the chain has been reached.

Task no. Contains in one byte a number which is a byte displacement from the second word in TCTAB (ie. the word following the length word) to a word containing the address of the TTAB for which this record is set under exclusive access.

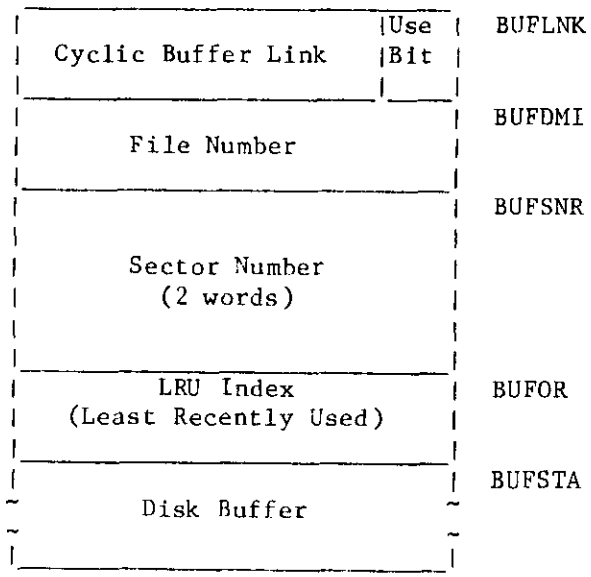
Record Number Three bytes containing the record numbers in the data file which are under exclusive access.

19.5 BUFFER MANAGEMENT

DM uses a common block buffer pool for all files and indexes. A 'Least Recently Used' algorithm is used to select buffers.

In the DMBUF module, data areas are allocated for these buffers, and routines to get and release a DM buffer are supplied.

By setting a constant (QNBUF), buffers can be allocated in DMBUF by SYSGEN, (2<QNBUF<16). The buffers will be linked together cyclically. Each buffer has a buffer header where the contents of the buffer are described.



The words BUFDMI and BUFSNR define one unique sector. The LRU index is kept in the buffer header to let buffers remain in memory as long as possible in order to minimize physical I/O. The word CURBUF contains a pointer to the current buffer.

BUFLNK This word contains a pointer to the next buffer in the cyclic chain. Bit 15 is used to indicate whether this buffer is in use or not.
Bit 15: 1 = buffer is in use; 0 = buffer is free.

BUFDMI File Number.

BUFSNR Contains the physical sector number, the contents of which are stored in the buffer.

BUFOR This index is incremented by one when a sector is found to be not present in memory. When the buffer is about to be used the index is reset. If a buffer has to be overwritten, the one with the highest LRU index is selected.

BUFSTA Start of the buffer area in which the sector contents is stored.

19.5.1 Get and Release a DM Buffer

As input parameter to the get buffer routine, a unique sector must be defined in registers A1 and A2 in the same way as in BUFDMI and BUFSNR.

If this sector's contents are already in a memory buffer, the buffer address is put in A3, the use bit in BUFLNK set, and a normal return taken.

If the sector searched for is not located in memory the oldest unused buffer is selected via the LRU index.
When this buffer is found, its use bit is set, and a skip return taken (i.e. return to normal return address + 2).

When no unused buffer is available, the system lights SOP lamps 10 and 11 and HALT.

The release buffer routine will reset the use bit in BUFLNK of the buffer address given in A3.

19.6 QUEUEING

There are four points where a queue may arise in Data Management Handling.

19.6.1 File Request Queue

Only one request at a time is handled for a particular file structure. If a file is busy the request is queued in the monitor before the activation handler (TIODM) is called.
Queueing is done using the principle 'First-In-First-Out' per level of calling task (the queue pointer is stored in the FWT, in DWTTQ.
The file request queue is dissolved when the DM request in progress is completed (TENDIO).

19.6.2 Pending Queue for DM Task

When a DM task is activated it may be busy with a job for another file. In that case the activation is put in the ordinary task pending queue (queue pointer is stored in TTAB for the task). The pending queue is dissolved upon an EXIT request for the DM task.

19.6.3 Device Queue for Physical I/O

If the disk drive is busy when a call for physical I/O is made, this request is queued in the monitor (queue pointer is stored in the DWT (DWTTQ) of the data management task. The device queue is dissolved when the running I/O request is completed. The device queue will never contain more than one element, since another DM request will be queued already in the DM task pending queue.

19.6.4 Open Queue

Open and Close orders are executed without overlap. This is accomplished through the DM task pending queue. Only one DM task is allowed to execute these orders.