TO: MSPM Distribution
FROM: D. L. Stone
SUBJECT: BF.6.01
DATE: 06/21/68

The attached copy of BF.6.01 represents the current state of the Tape DSM implementation. More detailed information and source files may be obtained from me.

Published: 06/21/68

Identification

Standard Tape DSM
Harlow Frick

Purpose

Operating in the user's working process, the tape DSM (Device Strategy Module) interfaces on one side with I/O system Requests from the user directed to magnetic tape; and on the other side with the tape DCM in the magnetic tape device manager process. There is also communication with the operator's process by means of requests made to the media request manager module.

Restrictions

The tape DSM currently contains the following restrictions:

1. All tapes are written in the Multics standard magnetic tape format.

2. Tapes to be read must have been written in Multics standard magnetic tape format.

3. The tape DSM will operate equally well with 7 or 9 track tapes and with either 800 or 556 BPI tapes. However, tape density is set by default to 800 BPI in tape registry files, thereby restricting users to 800 BPI. Also, it is the responsibility of the attachment module (BF.2.23) to insure that the tape reel is attached to a 7 or 9 track handler as required.

4. Ionames are unconditionally assigned default attributes as defined in BF.1.02 and elsewhere. For convenience they are listed below.

   Access mode is forward only.
   Data mode is logical, linear.
   Use mode is readable, rewritable, appendable.
   Write synchronization mode is asynchronous.
   Read synchronization mode is asynchronous.
   Workspace synchronization mode is synchronous.
   Read ahead and write behind limits are constants which are not adjustable by the user.
   Element size is a constant which is not adjustable by the user. It is currently set at 36 bits. The following two restrictions on element size are imposed by current implementation.

a) Element size is assumed less than 37 bits in the tape_dsm_detach procedure when filling a partial buffer with padding. This is an easily removed restriction.

b) Element size must not be a value which can cause an element to be split between two physical records (e.g., 8 bits or 3 words). It is uncertain whether it is worth while to remove this restriction.

5. No consideration is given to the insertion or removal of write rings. Information as to whether the write ring is inserted is, however, contained in the 144 bit status string returned to the user when return is made from the attach call.

6. Only the following calls will be recognized by the tape DSM:

attach
detach
read
write
seek
upstate
restart

Tape Registry Files

In order for a tape reel to be accessible to a user it must be described in a tape registry file. A tape reel is one of three general types, depending on the restriction_code in its registry file.

If the restriction_code is 0, the registry file is for an ordinary standard Multics tape. Ordinary standard Multics tapes are the only type available to ordinary users, and each has a tape registry file unique to that tape reel.

If the restriction code is 1, the registry file allows the user to have read and write access to whatever labeled standard Multics tape the operator mounts. The number of users able to specify tape registry files with a restriction_code of 1, will be extremely limited. For a tape reel attached with a restriction_code of 1, the tracks and density parameters in the registry file still apply, but prev_reel, next_reel, and reel_name have no meaning.

If the restriction_code is 2, physical mode is allowed (but not required). No kind of label check is performed by the DSM. There are no plans to allow this restriction_code initially.

Initially registry files may be fixed since the command to add or change tape registry files may not be implemented.

Each tape registry file has a data area containing profile information for the physical tape associated with that registry file. This data area is defined with the following declaration:

```
dcl 1 tape_profile based (profile_ptr),
    2 tracks fixed bin,          /*number of tracks(7 or 9)*/
    2 density fixed bin,         /*density(556 or 800)*/
    2 current_length fixed bin,
                                 /*number of data bits currently
                                   written on tape, exclusive of
                                   administrative data*/
    2 restriction_code fixed bin,
                                 /*0=standard multics format,
                                   perform standard label check
                                   1=standard multics format,
                                   perform label check by verify-
                                   ing only that some valid label
                                   exists.
                                   2=physical mode, don't perform
                                   any label check*/
    2 prev_reel char(32),        /*registry file name of previous
                                   reel if in a multireel sequence.
                                   If a null string this is the
                                   first reel.*/
    2 next_reel char(32),        /*registry file name of next reel
                                   if in a multireel sequence. If
                                   a null string this is the last
                                   reel.*/
                                 /*if only 1 reel, both prev_reel
                                   and next_reel must be blank
                                   character strings.*/
    2 reel_name char(32);        /*character string used to
                                   locate and visually identify
                                   this reel*/
```

Transaction Block Discipline

Note: This section assumes the reader is familiar with MSPM Section BF.2.20.

A buffer tb is allocated and attached to the end of the buffer tb main chain for every tape operation. If the tape operation is a read or write a buffer tbe is attached to the buffer tb. The tbe contains the following information:

1. The physical data transferred to/from the tape.

2. A count of the number of elements transferred, which is set by the DCM.

3. A code indicating whether data is for a write call or read call and whether the data read has been verified.

When the user makes a write call, each buffer tb for data associated with the write call is chained in a down chain from the users call. This is necessary since status changes may, and normally do, occur for the users write call after return from the call. The dsm updates the user's call status from buffer tb status in buffer tb's in the down chain from the users write call tb.

When the user makes a read call no down chain is attached to his call tb because his final status is returned to him by processing the buffer tb main chain before returning.

Per Ioname Base Extension Data

The following declaration is for the per_ioname base extension table used by the tape DSM. (The per_ioname base extension is described in BF.2.20).

```
dcl  1 pibe based(pibep),
     2 relp,
       3 next bit(18),          /*relp to next pibe*/
       3 last bit(18),          /*relp to last item in this
                                  structure*/
     2 read_out fixed bin,      /*read ahead limit in records*/
     2 write_out fixed bin,     /*write behind limit in records*/
     2 nout fixed bin,          /*current read_ahead/write_behind
                                  in records*/
     2 buffer_position fixed bin,
                                /*offset within buffer.data of next
                                  element to be filled*/
     2 physical_position fixed bin,
                                /*next physical record number to be
                                  transmitted to/from*/
     2 current_buffer bit(18),/*index of current buffer tb*/
     2 bufp bit(18),            /*relp for current buffer tbe*/
     2 event bit(70),           /*event for communication with
                                  device manager process*/
     2 id bit(36),              /*id of process which may use above
                                  event channel*/
     2 wf_written bit(52),      /*when the user issues the first
                                  read operation record no. 1 is
                                  unconditionally read and the time
                                  when it was written is stored
                                  here; for records to be accepted
                                  they must have been written
                                  later than this time*/
```

```
2 read_status bit(144),  /*if currently reading, the
                           status of the last read call
                           is stored here*/
2 brr_count fixed bin,   /*incremented when a bad record
                           is read, reset when a good
                           record is read*/
2 rt_count fixed bin,    /*incremented when a too large
                           record number is read if
                           brr_count ^= 0; reset when a
                           required record number is read
                           without errors*/
2 wc_flag fixed bin(1),  /*wait for completion flag; if
                           set, upstate wait for success-
                           ful completion of all transac-
                           tions before returning*/
2 read_flag fixed bin(1),/*current mode is reading if set*/
2 write_flag fixed bin(1);
                         /*current mode is writing if set*/
```

Hardware Errors

Write operations to write data records which terminate with parity or transfer timing error are not reflected to the user unless the error persists for 10 attempts to write the same record. Instead they are re-initiated, without first backspacing, per the write error recovery philosophy specified in BB.3.01. Parity or transfer timing errors on write operations to write eof records are ignored, however, in order to avoid the problem of backspacing over multiple eof records generated by parity or transfer timing errors when written.

Read operations which terminate with parity or transfer timing errors cause the following physical record to be read, in hope that an error occurred when the tape was written, that it was re-written, and the next record will therefore contain a good copy. If a good copy of the required record number is not found before locating larger numbered record or end of file, the tape is backspaced until a smaller numbered record is passed over, and then the operation is repeated. If this entire procedure is unsuccessful in 10 tries, unreadable data status is returned to the user. Program operation upon detection of read parity or transfer timing errors is detailed in the description of the isearch procedure. When unreadable data status must be returned to the user, the DSM abandons reading elements beyond those in the first encountered unreadable record.

The user may skip over unreadable data as follows:

1. First, process the good data (nelemt elements were transferred without error).

2. Calculate an offset to be added to the read_pointer.

   offset = nelemt + (9216/elsize)

3. Call seek in order to add offset to the read_pointer.

4. The result of adding the offset to the read_pointer sets the read_pointer to point at the first element of the next record number after the unreadable one. Therefore, a read call can now be executed to read starting with the first record after the unreadable one.

All hardware errors other than parity and transfer timing are considered fatal and are simply returned to the user in his status string.

Status String Format

The following bit assignment has been defined in addition to the definitions in BF.1.21. If bit 6 is set (serious or fatal error) then the following bits contain valid information.

Bits	Meaning
28-30	1 = write operation
	3 = order operation
	4 = read operation
31-36	If bits 28-30 = 3 then this field indicates the order code as listed in BF.6.02.
109-114	Major status
115-120	Sub status

Tape DSM Procedures

The tape dsm consists of 3 levels of procedures. The first level is the access segment which in the current implementation simply transfers to one of the following second level procedures:

```
tape_dsm_attach
tape_dsm_detach
tape_dsm_read
tape_dsm_write
tape_dsm_seek
tape_dsm_upstate
```

The above procedures then make calls to the following group of auxiliary, or third level, procedures:

```
tape_dsm_fread
tape_dsm_isearch
tape_dsm_bread
tape_dsm_order
tape_dsm_misc
tape_dsm_status_handler
tape_dsm_bseek
```

The remaining portions of this document describe the above listed procedures.

Tape_dsm_attach

This procedure is entered when the user calls tape_dsm$attach. Its purpose is to perform attach processing in order that subsequent read/write calls may be processed for the ioname being attached.

Upon receipt of an attach call, tape_dsm_attach does the following:

1. Initializes information in the per_ioname base table.

2. Calls attm$attach which performs standard attach processing as described in BF.2.23.

3. Calls ecm$create_ev_chn and ecm$give_access in order to establish interprocess communication with the media request module.

4. Calls mrm$put_request in order to initiate an operator request to mount the required tape reel, and waits until the required tape is mounted. (Described in BT.2.01).

5. Allocates and initializes the per-ioname base extension table.

6. Sets tape density to 800 or 556 BPI according to the density specified in the tape registry file.

7. Performs a label check per the following simplified steps.

   a. call fread in order to fetch record #0. If not blank tape status, go to c; else, go to b.

   b. Rewind; write tape label; write eof; rewind; go to a.

   c. If tape label not equal to label in tape registry file, error return.

   d. Forward space file.

8. Return.

Tape_dsm_detach

This procedure is entered when the user calls tape_dsm$detach. Upon receipt of a detach call tape_dsm_detach does the following:

1. Finishes tape I/O operations as follows:

   a. If the current operation mode is not writing, go to d.

   b. If there is a partial buffer, fill remaining buffer positions with "padding" and write the record.

   c. Write an end-of-reel sequence (consists of: EOF, EOR record, EOF, EOF; as specified in BB.2.01).

   d. Rewind and dismount.

   e. Wait until rewind is complete.

2. Calls mrm$put_request in order to initiate an operator request to return the tape reel to its permanent storage area.

3. Calls ecm$delet_ev_chn in order to destroy the operator communication event channel.

4. Calls attm$detach in which performs standard detach processing as described in BF.2.23.

5. Calls atm$delete_ioname with the delayed bit on.

6. Returns.

Tape_dsm_write

This procedure is entered when the user calls tape_dsm$write. Its function is to transfer the user's data to physical record blocks, initiate write operations as physical record blocks (buffer tbe's) are filled, and then return to the user after calling tape_dsm_upstate which processes any terminations which have occurred, and waits until writebehind is within limits. This procedure also initiates writing eof records after each 128th record.

Tape_dsm_read

This procedure is entered when the user calls tape_dsm$read. Its function is to transfer data from physical record blocks (buffer tbe's) to the area specified by the user. If the required data is not in a physical record block, the fread procedure is called in order to get it. Also, this procedure initiates allowable read ahead before returning to the user.

Tape_dsm_seek

This procedure is entered when the user calls tape_dsm$seek. After an attach call the user's read pointer is set to the first element on the tape, and his write pointer is set to the last element on the tape plus one. Also, the tape is physically positioned in front of the first data record.

The seek call allows the user to change the read or write pointers. It does not initiate any physical tape movement. Physical tape movement is initiated only by a read or write call.

Tape_dsm_bseek

When a read or write call arrives, a check is made to see if the tape is correctly positioned relative to the read or write pointer. If not, the tape is physically re-positioned as required by calling the bseek procedure. Or, more precisely, this procedure is called by tape_dsm_write/tape_dsm_read if the write/read pointer does not point to the element which will be written/read next. Its function is to cause the physical tape position and dsm transaction block status to be changed such that the next element transferred to/from tape will be specified by the write/read pointer. In some cases it is necessary to read tape in the process of executing write calls in order to properly position tape and/or partially fill

a write buffer. For example, if a user initiates writing at some point other than the beginning of tape, it is required to first read the tape in order to properly position it. If the user issues a seek to an element in the middle of a record, followed by a write, the tape must be re-positioned in front of the record containing the first element to be written, the unchanged part of the record must be read into the current write buffer, and the tape must be backspaced. It should be noted that this same strategy must be employed when the user starts writing at the end of a tape if the last data record on the tape had padding in it. All actual searching and reading operations will be performed by calling Fread procedure.

The bseek procedure is not yet implemented, or completely flowcharted, since it is not initially required. This causes the following restrictions:

1. Tapes must always be written starting at the beginning.

2. Tapes must be sequentially read starting at the beginning.

3. After writing a tape, the user must detach and attach before reading.

Tape_dsm_upstate

This procedure performs transaction block discipline, sets the 144 bit status string in the call sequence and then returns. If the current operational mode is reading, transaction block discipline is performed by calling fread. Fread will never initiate any read operations when called from upstate because it is called with the count argument = 0.

If current operational mode is writing, transaction block discipline is updated directly in this procedure. Recoverable write errors (i.e., parity or transfer timing error) are processed by upstate by re-issuing the write operation.

The upstate procedure is entered in one of the following ways:

1. User calls tape_dsm$upstate.

2. Tape_dsm_attach calls tape_dsm_upstate after writing a label record.

3. Tape_dsm_write always calls tape_dsm_upstate before returning to the user.

4. Tape_dsm_detach calls tape_dsm_upstate at detach time, if current mode is writing.

Restart Operation

The restart outer call is initiated by the overseer in order to restart I/O operations subsequent to a quit. It is implemented in the DSM by a call to the upstate procedure. At the time of this call it is likely that in progress I/O transactions were previously aborted by consequence of the user pressing his quit button. The upstate procedure, among other things, restarts I/O operations which were previously aborted.

Fread

This procedure is called in order to place verified data into buffers and set in_status. If count = 0, no attempt is made to place any verified data in buffers but in_status is set to the status of the last read operation. If count > 0, then count sequential verified data records are placed in adjacent buffer tbs, starting with the record number specified in physical_position (physical_position is maintained in the pibe).

Tape_dsm_isearch

This procedure issues a series of tape move operations followed by a read operation, waits for completion of the read operation if the wait flag in the call sequence is set, and then returns to the caller. This procedure is called by the fread procedure whenever a record number other than the one required is read.

Searching strategy is designed to be reasonably efficient for any repositioning action. The following operational steps illustrate the sequence in which logical situations are handled.

1. If the tape must be positioned more than 3/4 of the way back to the beginning from its present position, a rewind and forward search is initiated.

2. If the required record has eof (end of file) records between it and the current position, forward or backspace file commands are issued until no eof records intervene.

3. If the required record is forward on the tape and if the distance back from the next eof record is less than 1/2 the distance forward to the record, then a forward space file, backspace file (to backup over the eof record just forward spaced over) and a series of backspaces records are issued instead of issuing forward

space records. This strategy is applied because it reduces the number of commands sent to the dcm by at least 50%. The same strategy is applied when searching for records backward on the tape.

Duplicate record numbers may occur on the tape because when the tape is written, parity errors on write operations result in re-writing the same record until it is written without error, or until the record is written unsuccessfully 10 times. Duplicate record numbers can foil the attempt to read the required record number on the first attempt after positioning. However, if this occurs the fread procedure will discover that it has the wrong record number and call isearch a second time.

It is possible, but highly unlikely that isearch will not initiate reading the required record number when called a second time. However, Fread will persist and call isearch until the required record is finally read. Even if every record is duplicated a variable number of times, isearch will eventually find the correct record.

The effect of previously initiated read ahead is taken care of by initiating re-position commands only after waiting for all previously initiated commands to terminate and checking each read ahead record to determine whether it is the required record number. The waiting and checking is done in the fread procedure before calling the isearch procedure to initiate re-positioning operations.

Limits have been placed on the number of tape operations which will be initiated upon a single call to the isearch procedure for the following reasons:

1. It is not currently known if there is a maximum practical number of calls which the request queuer should queue.

2. The probability of correct positioning decreases as the number of positioning operations increases, and it therefore seems advisable to check the actual tape position occasionally when searching for a record.

Limits in the isearch procedure are currently set to a maximum of 10 forward or backspace files, when skipping past eof records, and a maximum of 20 forward or backspace records.

The maximum number of operations initiated by one call to the isearch procedure is 34 as illustrated by the following example:

ap = 8192	actual position	
nrp = 1369	next required position	
rewind		sets ap = 0
forward space	10 files	sets ap = 1280
forward space	1 file	sets ap = 1408
backspace	1 file	sets ap = 1408
backspace	20 records	sets ap = 1388
read	1 record	(reads record #1389)

Parity errors will, per current implementation, usually result in the following action:

1. Fread reads the required record with a parity error.
2. Fread increments brr_count to one and reads the next record.
3. Since the next record is not the required one, Fread calls isearch.
4. Isearch backspaces 3 records (2 to reposition in front of what would be the required record if there were no duplicates + brr_count, which is 1 in this example).
5. Isearch initiates a read operation, waits for completion, and returns.
6. Fread determines that the record read is not the required record number and so calls isearch a second time.
7. Isearch initiates a read operation, waits for completion, and returns (no repositioning is required because the required record number is next on the tape).
8. If no parity error occurs, fread accepts the record after making header and trailer data checks. If a parity error occurs on this second attempt, the retry procedure is repeated by continuing at step 2. After the 10th attempt fread gives up and return is made to the user with the fatal error bit set and with hardware status stored in status returned to the user.

Tape_dsm_bread

This procedure initiates a read call to the dcm after performing tb and tbe allocation and chaining. Also, it waits for physical completion before returning if the wait_flag is set. The tbx argument in the call sequence is set to the buffer tb index of the read call.

Tape_dsm_order

This procedure initiates an order call to the dcm after allocating and chaining a buffer tb. Also, it waits for physical completion before returning if the wait_flag in the call sequence is set. The tbx argument in the call sequence is set to the buffer tb index of the order call. The op argument in the call sequence is the operation code of the order call as listed in BF.6.02.

Tape_dsm_misc

This procedure contains the following entries:

1. Get_event - The purpose of this entry is to return as a return argument an event code which may be used by the tape dsm when calling the request queuer. The tape dsm may run in more than one process. An event code contains the process id of the process to be awakened. This procedure returns an event code which applies to the process currently running by comparing the process id of the current process to the process id part of the event which was returned the last time get_event was called. If equal, the same event is returned; if different a new event channel is created and the new event code is returned. This code should probably be placed in line or else another standard version of it should be called.

2. Order - This entry is no longer used. It should be deleted.

3. Error - This entry is called upon detection of status errors. It should be deleted when the standard error handler is implemented.

Tape_dsm_status_handler

This procedure is called when the DSM discovers status on an I/O transaction which it cannot handle. Tbx is the buffer tb containing the unrecoverable fatal error status. The following operations are performed:

1. Status in the buffer tb is moved to in_status.

2. The following additional information about the I/O transaction is placed in in_status.

bits 28 - 30	1 = write
	3 = order
	4 = read
bits 31 - 36	If an order call, these bits contain the order code as defined in BF.6.02.

3. The buffer tb chain is deleted.