TO: MSPM Distribution
FROM: P. G. Neumann
SUBJ: Standard Tape DCM
DATE: 07/26/67

The attached document BF.6.02 presents the standard tape device control module (DCM). BF.6.02 obsoletes the document BF.6.10, Interface Specifications for the Tape Controller Interface Module, (TCIM) 1/27/67. The TCIM no longer exists, but is roughly subsumed functionally by the current tape DCM. The tape device strategy module (DSM) will appear shortly as BF.6.01.

Published: 07/26/67
(Supersedes: BF.6.10, 01/27/67)

Identification

Standard Tape Device Control Module (DCM)
C. D. Olmsted

Purpose

The Tape DCM is at the end of any sequence of I/O System outer calls relating to tape I/O. It does the minimum amount of processing that will accomplish the I/O service and still permit maximum flexibility. Explicit calls are provided which implement all tape primitive commands plus entries which process interrupts and update status.

Introduction

For maximum flexibility the DCM operates workspace synchronously only. All other operations are asynchronous. It must, therefore, be modular with respect to starting and finishing I/O transactions. One module, the initiator, will be called to initiate transactions. It does this logically by creating a data base called the transaction block extension (TBE). This data base is a part of the I/O system data base structure (BF.2.20) and contains information which is request dependent and which is used both in implementing and cleaning up after the transaction. The initiator also contains the entries for certain control calls (attach, upstate, etc.). These are dealt with in a straightforward way after which the initiator always makes a call to the second module, the housekeeper, before it returns.

The housekeeper is the heart of the DCM. It actually implements requests and handles completed transactions. Before we describe it, therefore, it will be advantageous to discuss the general logic of tape I/O.

The implementation of tape transactions is more complicated than for other devices because, while each request is directed to one of the several tape drives (the ultimate devices), there is only one tape controller which has available one or two GIOC channels. Thus the DCM must keep track of the requests set up for the various drives. For a two channel controller it must also choose a channel.

Handling completed transactions is also complicated by the multiplicity of tape drives. In the most usual case the completion of the transaction would be discovered by a wakeup signal from the interrupt handler which would have received a hardware interrupt of some sort signifying the completion of the transaction on the device. The wait coordinator would then be able to call the dispatcher which would decode the signal and send an upstate call to the DCM. This chain of calls would ensure proper communication (copying of status, etc.) between the device manager process and the working process.

However, in the case of tapes the interrupts will be associated, not with a specific drive, but with the channel through which the drive is run. Thus a simple upstate call cannot be made because the dispatcher cannot direct it to any specific device; only the DCM keeps track of tape drive activity.

Furthermore, in the interests of efficiency, a policy is maintained of minimizing the number of interrupts and channel terminations so as to maximize channel activity. This dictates that the channel be kept in multiple peripheral instruction mode and be stopped only when necessary as in the case of errors, certain tape commands, rewind initiation, etc. Because of this minimization policy there will be requests whose completion can be determined only by noticing that their corresponding DCWs have been executed without error. This is strictly the domain of the DCM and the GIOC interface module (GIM).

Thus all hardware events having to do with tape must be routed directly to the DCM. The housekeeper then determines which transactions are complete, posts status for them, and generally cleans up (deallocates storage, sets switches, etc.). In the case of a transaction which is completed but which is not signalled as a hardware event, the posted status will be a fixed standard one. After this the housekeeper <u>simulates</u> a hardware event for each drive which it has found to have completed a transaction. This will eventually precipitate upstate calls directed for each such drive. The DCM, upon reception of the upstate call will note that the hardware event has been realized and, since all the housekeeping has been done previously, return with the appropriate status.

When the transaction completions have been dealt with, then the requests which have been set up by the initiator are implemented. Appropriate changes to a pseudo-DCW list are made by means of calls to the GIM. This is continued until all requests have been implemented or until list space is used up. Unimplemented requests are left queued pending future calls.

The housekeeper also deals with special cases such as rewinding tapes and transactions resulting in errors. Rewinding tapes are singular in that the request completion may come before the actual rewind is complete. A special interrupt signals the physical rewind completion but does not specify which tape has rewound. Therefore the DCM must determine which drive(s) has rewound.

In handling errors the housekeeper is unsympathetic. All outstanding requests for the drive in error are deleted and no more are accepted until the error is acknowledged by a special call to the DCM.

The calls to the DCM are as shown in Figure 1 where the jagged lines indicate signals and the numbers are explained as follows:

1. The DSM signals for an I/O call.

2. The wait coordinator calls the dispatcher,

3. which calls the driver

4., 5. which calls the DCM at the initiator

6. which calls the housekeeper.

   Control is returned back along the line to the wait coordinator.

   When a hardware interrupt occurs

7. the interrupt handler signals a tape interrupt.

8. The wait coordinator calls the housekeeper

9. which signals simulated hardware events

10. the wait coordinator makes upstate calls through the dispatcher,

11. and the driver,

12., 13. to the DCM.

Attachment

The attach call is passed on from the tape device strategy module (DSM). At this stage most of the attach work (validation, authorization, etc.) will have been done. The DCM, however, must account for those features which are particular to the specific device (tape drive). The call is

```
     call attach (ioname, type, device_name, mode, status,
                  pibptr);
```

where

ioname	is the attached ioname,
type	must be "tape_dcm",
device_name	will be a drive identification, i.e., a registry file name such as "drive_8",
mode	can be any combination of "P", "R", "W", "A",
status	returned status string,
pibptr	is a pointer to the base of the per-ioname data base (PIB) which is provided by the I/O switch.

The only processing done with the <u>type</u> and <u>mode</u> arguments is error checking. If they are not as given above, then the attach call is rejected. Otherwise <u>pibptr</u> is used to access the PIB. In the PIB area, storage is allocated for the PIB extension which has the following declaration.

```
     dcl 1 pib_ext based (p),

         2 chain,
           3 next_ext bit (18),
           3 last_word bit (18),
         2 tape_no fixed bin (17),
         2 channel fixed bin (17),
         2 attach_substatus bit (6),
         2 error_switch bit (1),
         2 rewind_ptr ptr,
         2 device_event_name bit (70),
         2 signal_set_sw bit (1);
```

The first entry, chain contains two relative pointers. These are a standard part of every PIB extension. One points to the next extension, if any (the tape DCM will have only one extension), and the other points to the last entry in the structure (i.e. signal_set_sw). The remaining items will be explained in later parts of this document.

The device name is looked up in a table of global information which contains the physical drive numbers corresponding to the device names. This number is entered in tape no in the PIB extension. The entry channel is set to zero. It will be filled in with one of the two tape channel device indices when a request is made to use the attached device. Substatus for the drive is got by means of a GIM call. This substatus will be stored in pib_ext.attach_ substatus and will be used along with a major status of "subsystem ready" to manufacture status for transactions which are completed with neither an error nor a status event to signal the completion. If at this time the major status is not "subsystem ready", then a default minor status is stored. At every status request which results in a major status of "subsystem ready" attach substatus will be updated. Eventually the substatus will indicate whether the tape drive is 7 or 9 track and whether the reel is write protected or not. Also indicated will be the "tape at leader" status. This, however, will be assumed to be off, since because of backspacing operations it would be prohibitively complex and time consuming to maintain it. In order to check "tape at leader" status the DCM must make a separate "request status" call.

The remaining quantities in the PIB extension are initialized by setting error switch off, rewind ptr null, signal set sw off, and device event name by means of the get_hardware entry in the dispatcher. This event identification will be used by the housekeeper for simulating hardware events for individual tape drives.

The pibptr is stored in a global drive information table which is indexed by physical drive number and which contains information which is more relevant to the attached drive than to the ioname. Before attachment pibptr should be null. If not the attach call is rejected. Also in the drive information table is a pointer which indicates any uninitiated requests. This is initialized as a null pointer.

Status is returned as "transaction logically and physically complete and successful" and "status reporting complete".

Calls to the Initiator

The data transfer calls are

```
         read
call           (ioname, workspace_ptr, word_count, read_count,
                status, pibptr);
         write
```

where

<u>ioname</u> has been previously attached,

<u>workspace ptr</u> points to the work space to be read into / written from

<u>word count</u> is the number of words to be read / written (element size is fixed at 36),

<u>read count</u> is the number actually read. It is ignored because of the asynchronous nature of the DCM.

<u>status</u>, <u>pibptr</u> are as before.

To implement the remaining tape primitives and also to provide other services a general call is provided.

```
call order (ioname, op_group, op_ptr, status, pibptr);
```

where

<u>ioname</u> has been previously attached,

<u>op group</u> is a character string which identifies the group in which the operation will fall. There are two groups: op_group = "drive_op" which specifies physical hardware operations and op_group = "switch_set" which specifies software mode and switch changes.

<u>op ptr</u> is a pointer to an integer which is a symbolic op code. The meaning of the code in its operation group is given below. In the drive-op group, where each corresponds to a tape controller command:

code	meaning
0	request status
1	reset status
2	forward space record
3	forward space file
4	backspace record
5	backspace file
6	erase (8 1/2 inches)
7	write EOF mark
8	set high density
9	set low density
10	set file protect
11	rewind
12	rewind and unload.

In the switch_set group:

code	meaning
1	set read/write binary mode
2	set read/write 9 track mode
3	set read/write BCD mode
4	reset error switch in PIB extension.

The default read/write mode is binary.

<u>status</u>, <u>pibptr</u> are as before.

<u>Request Handling in the Initiator</u>

A request is any read, write, or "drive_op" order call. Its processing begins with the allocation of a transaction block extension (TBE) in the PIB area. The TBE includes request dependent information such as the op code of the tape drive operation and the expected (if all goes well) major status at the termination of the operation. A pointer to the TBE is stored in the transaction block (TB), a data base which is allocated automatically at each outer call by the I/O switch. The TB contains, among other things, the returned status, various switches which indicate completion of the transaction (BF.20.02), and the pointer to the TBE.

These TBs are threaded together in a list, the top and bottom of which are pointed to by entries in the PIB. Thus the external data structure with which the DCM is concerned looks like Figure 2.

Request Handling in the Housekeeper

Once this data is set up, the request is implemented with a call to the housekeeper which, among other things, makes the appropriate GIOC interface module (GIM) calls. Except when there are no outstanding requests (pseudo list inactive) this will entail extending the currently executing pseudo list by means of a change call to the GIM. The pseudo DCW for a read call will be set to cause an external signal in order to fetch and draw attention to the DCW residue and the read count. If there are no outstanding requests at all then there will also be a connect call. A list of pointers, the <u>request pointer list</u>, is maintained in parallel with the pseudo list and has one entry for every pseudo DCW. The entries are null except for those corresponding to the last pseudo DCW of a request. In this entry is put the index of the corresponding transaction block, and the corresponding <u>pibptr</u>. Obversely, for purposes of cross reference, the index of this pseudo DCW is placed in the TBE.

If there are no requests outstanding for the particular device, then the channel whose list contains fewer unprocessed pseudo-DCWs is chosen and the channel name entered in the PIB extension. The number of unprocessed pseudo-DCWs is maintained as a running count which is incremented as new pseudo-DCWs are added to the list and decremented whenever a transaction is terminated. If there is a request, then subsequent requests must, of course, be sent through the same channel. Thus if pib_est.chan is not null then it is used to determine the channel.

It is possible that there will be no room on the proper one or both pseudo-lists. In this case the uninitiated request pointer in the drive information table is examined. If it is not null, then it points to the TB of the first such uninitiated request. It is assumed that subsequent requests are also uninitiated. If the pointer is null then it is replaced with a pointer to the current TB. In either case the arguments of the call are stored in the TBE. Control is then returned to the caller.

It is also possible that a change$list call to the GIM which alters an active list may occur too late and the channel be already stopped. When such a situation is signalled by the GIM the channel must be restarted by issuing another connect call to the GIM.

Data Bases

The declaration for the PIB extension has already been given. The remaining data bases are declared here and explained briefly.

The transaction block extension declaration is

```
dcl 1 tbe based (p);

    2 chain,

      3 next_ext bit (18),        /* standard chain relative...*/

      3 last_word bit (18),       /* ... pointers as in PIBE */

    2 opcode bit (6),             /* tape controller op code */

    2 exp_stat bit (4),           /* expected status for error
                                     free transaction */

    2 workspace_ptr ptr,          /* pointer to data work space */

    2 word_count,                 /* number of words to read
                                     or write */

    2 list_index fixed bin (17),  /* index of last pseudo DCW
                                     for this transaction */

    2 pib_relp bit (18);          /* back pointer to PIB */
```

The global data base (a read only segment which is provided during initialization) which supplies the names of the channels and the drives is called the tape names table and has the following declaration:

```
     dcl 1 controllers (MAX) based (p),
         2 channel_name(2) char (32),
         2 drive_name(16) char (32);
```

The entries are, respectively, the symbolic names of the 2 tape channels and the 16 tape drives. MAX is the number of tape controllers in the installation. If a controller has only one channel, channel_name(2) is null.

The drive information table and the request pointer list are combined in

```
     dcl 1 drive_request_info (MAX),

         2 rqest_ptr_list(2),

           3 rqest_ptr(LISTLGTH) bit (18),

           3 pibptr (LISTLGTH) ptr,

         2 drive_info(16),

           3 pibptr ptr,

           3 uninit_rqest bit (18);
```

where rqest ptr(i) is zero unless the ith pseudo DCW is the last of a block of pseudo DCWs corresponding to a request. In this case rqest ptr(i) is the index of the transaction block for the request and pibptr(i) gives the PIB with which the request is associated. In the drive information table pitptr(i) points to the PIB of the ioname attached to drive i. uninit rqest(i) is the index of the TB of the first uninitiated request for drive i.

The Housekeeper

Whenever the DCM gets control for any reason, the housekeeper is called. This routine tends to all transactions which have been completed since the last call to the housekeeper. The flow chart in figure 3 gives its structure.

Status and the list index corresponding to the DCW which caused the status event are gotten by means of a request status call to the GIM. This list index is compared with the list index corresponding to the last request terminated by the housekeeper. All requests in between (but not including) are considered implicitly terminated and the standard termination procedure is called for them.

The termination procedure does the following

1. stores status in the TB,

2. sets the completion bit on,

3. deallocates the TBE by a call to tbm$remove (see BF.2.20),

4. signals a simulated hardware event for the device and sets the signal set switch in the PIB on. This step is omitted if the signal set switch is already on.

5. If there is an error, call the error routine.

After dealing with the implicitly terminated requests the housekeeper examines the status for the request causing the status event. Special interrupts are noted for later processing. If there was a terminate signal the request is checked to see if it initiated a rewind. If so the rewind pointer in the PIB extension is set and a "request status" pseudo DCW is set up in a special list. This list will be used in processing the special interrupts which signal termination of tape rewinds.

If the status event is an external signal, we test for special interrupt. If there is one, then the channel is stopped and a rewind has completed. Control is transferred immediately to the rewind handler. Otherwise we know that a read request has been completed since we set up an external signal only for read operations. The GIM is called to get and process the tally residue and the read_count is encoded in the status string. Then for either status event, termination is called for the request which caused the event. Termination is not called, however, if the request initiated a rewind. Control is then returned to the beginning and status is again obtained from the GIM.

This loop is terminated when the GIM call indicates that there is no more status. At this time special interrupts (if any) are processed. The special pseudo list is started (provided that the channel has stopped) and a loop on the request$status call is entered to wait for the completion of the request status. When the status is returned the substatus is either "device busy" (tape still rewinding) or "tape at leader". In the latter case the termination procedure is called for the request which caused the tape rewind. This routine is repeated for each drive which is rewinding.

The last housekeeping job is setting up uninitiated requests. This includes left over requests and requests recently set up by the initiator. A non-null pointer in the drive information table will point to the TB of the first such request. If there is now room on the proper list, the request is set up with GIM calls and, if the channel is inactive, it is restarted. This is repeated for all subsequent uninitiated requests until either all requests have been initiated, or until available pseudo list space is again exhausted. In the former case the drive information table entry for the drive is set null. In the latter it is replaced with a pointer to the TB of the next uninitiated request. If the pointer is set null, then uninitiated requests will be sought on other drives and the above procedure repeated. Whenever the housekeeper is unable to initiate all requests it leaves an index containing the drive number whose uninitiated requests are to be processed at the next call to the housekeeper. This ensures impartial queueing of uninitiated requests.

Error Handling

When the termination procedure finds an error (status is unexpected) all I/O for the affected drive is abandoned. This means that all requests for the drive are aborted, i.e.

1. Using the request pointer list and the list index in the TBE, the pseudo DCW lists which had been set up for these requests are replaced with null-ops and the pointers in the request pointer list are set to null.

2. The TBEs are deallocated.

3. The abort bit is set on and the hold lower bit is set off in the TB.

4. The error switch is set in the PIB extension.

5. The uninitiated request pointer in the drive information table is set null.

6. Rewind_ptr is set to null.

Non hardware errors are sent back to the DSM. They include

1. Attach rejects because of improper argument sequence or type, redundant ioname, a non-null pibptr (drive not detached).

2. Request rejects because of improper argument sequence or type undefined software opcode.

3. Detach rejects because of improper argument sequence or type.

The Upstate Call

This call will result from having simulated a hardware event in the housekeeper. Its only function is to force a normal sequence of calls so that the driver can copy status for completed transactions. The signal set switch in the PIB is set off, the hold bits in all TBs which show completion are set off, and control is returned to the caller.

Detachment

The detach call frees a previously attached drive. This will cause the same behavior as does an error on the drive attached to ioname except that the error switch is not put on. Further, all entries for the drive in the drive information table are set to null.

Initialization

Upon receipt of the first attach call after system initialization, the DCM will initialize its tables (the request pointer table and drive information table) and make define list calls to the GIM for its various channels.

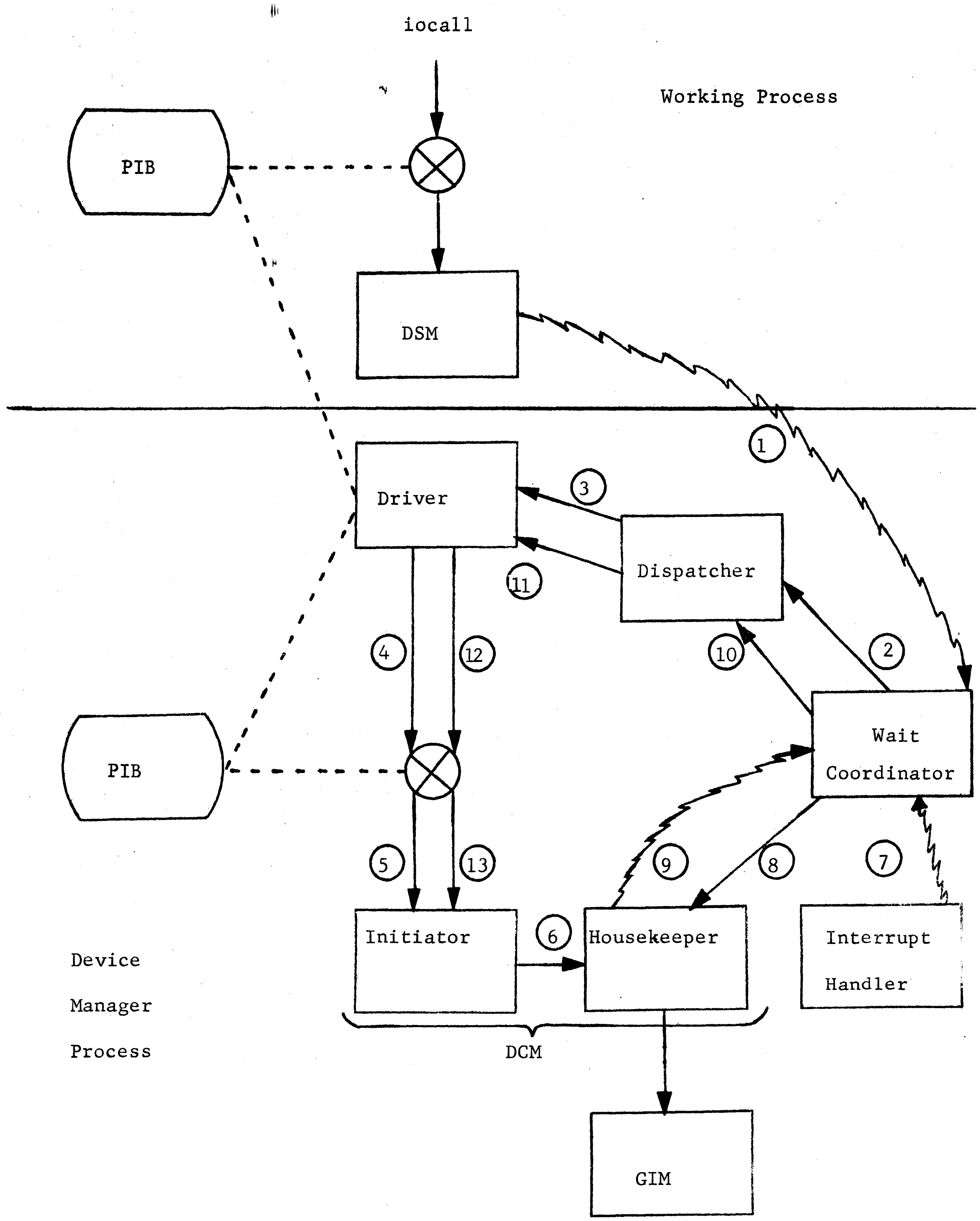


FIGURE 1: Calls to the Initiator and the Housekeeper

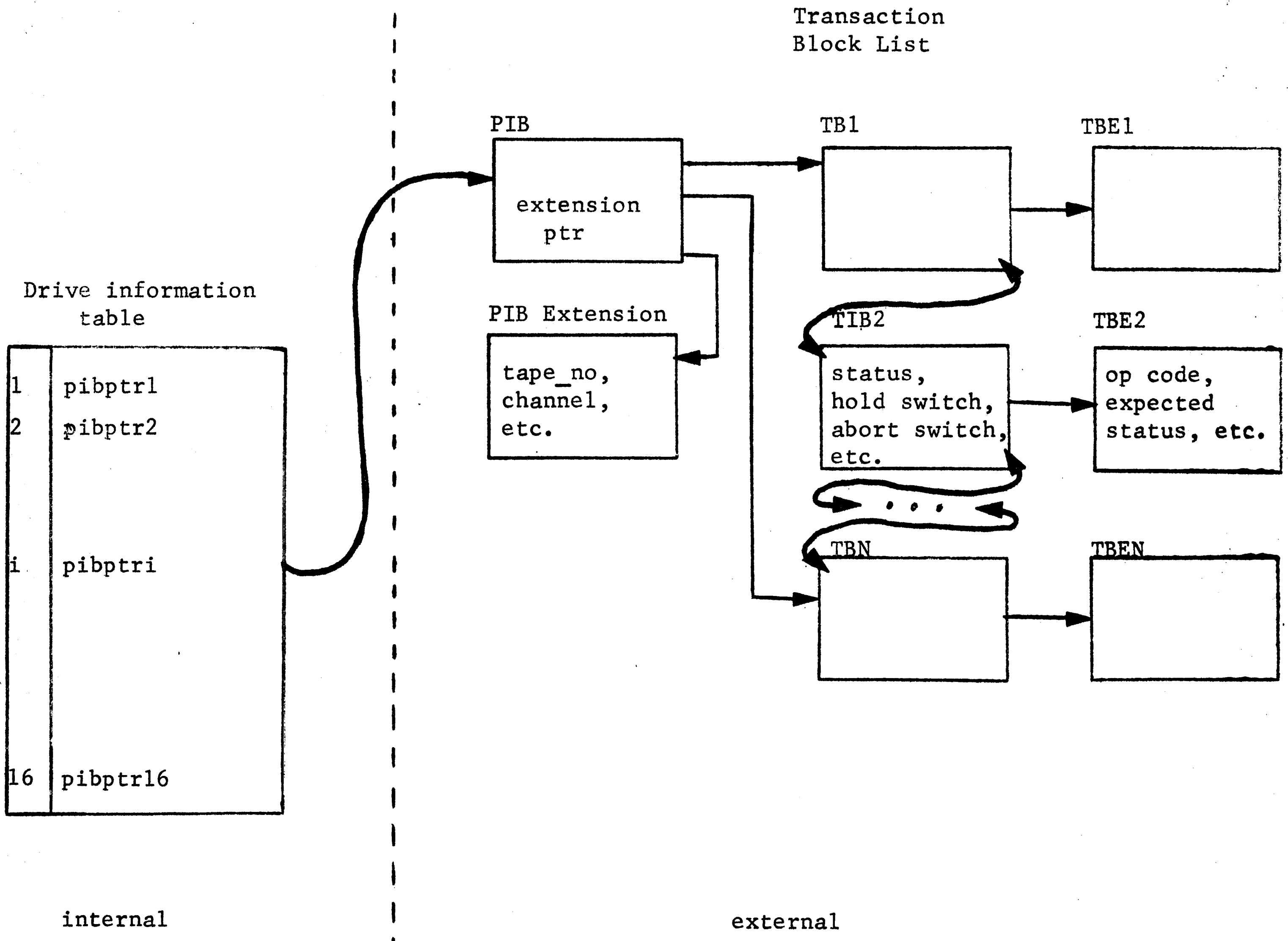
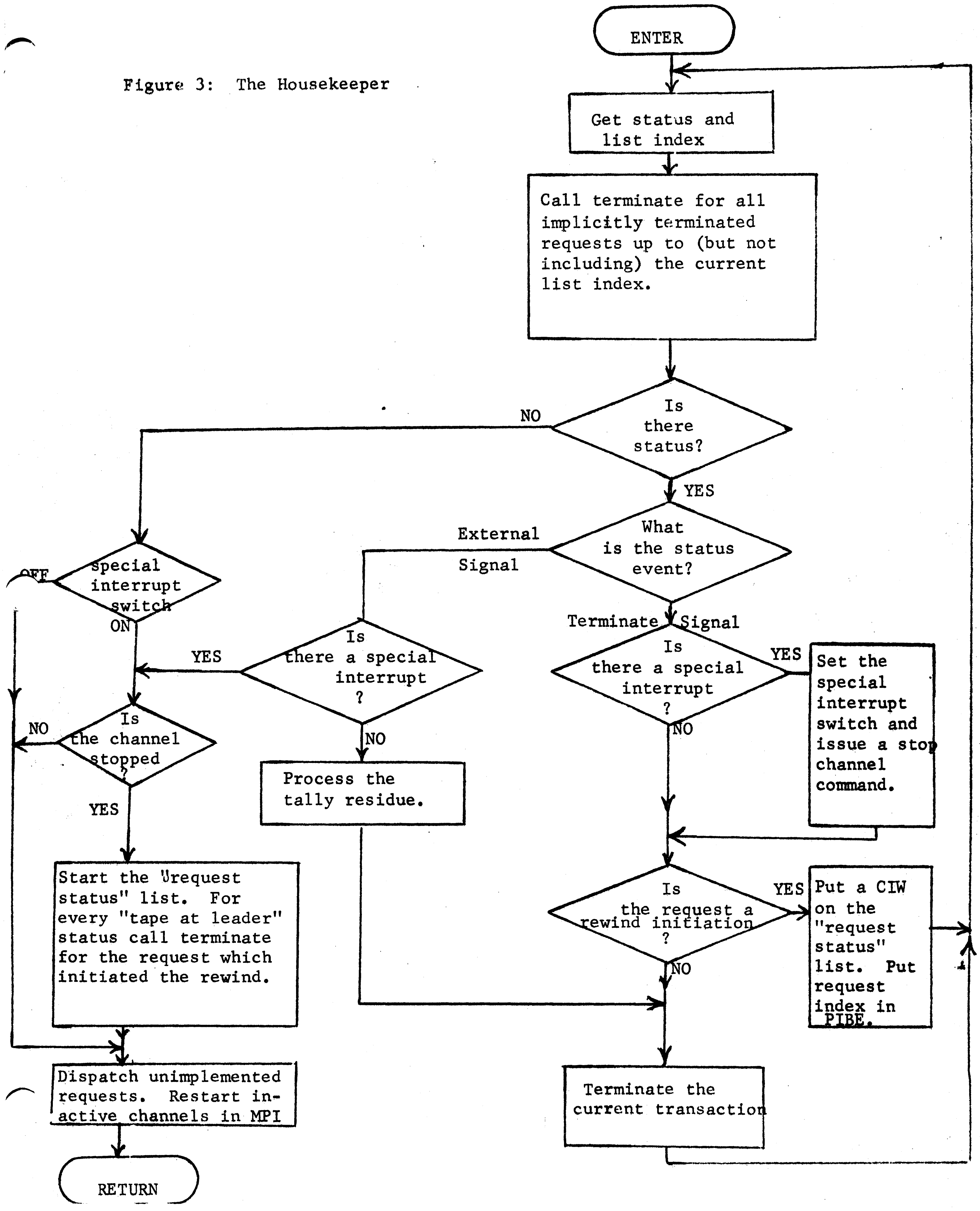


Figure 2: Tape DCM Data Bases

Figure 3: The Housekeeper


ENTER
Get status and list index
Call terminate for all implicitly terminated requests up to (but not including) the current list index.
Is there status?
NO
YES
What is the status event?
External Signal
Terminate Signal
special interrupt switch
OFF
ON
Is there a special interrupt ?
YES
NO
Is the channel stopped ?
NO
YES
Process the tally residue.
Is there a special interrupt ?
YES
NO
Set the special interrupt switch and issue a stop channel command.
Is the request a rewind initiation ?
YES
NO
Put a CIW on the "request status" list. Put request index in PIBE.
Start the "request status" list. For every "tape at leader" status call terminate for the request which initiated the rewind.
Terminate the current transaction
Dispatch unimplemented requests. Restart inactive channels in MPI
RETURN