Disk I/O Request Service

(PDP-11 - PDP-1Ø Disk I/O Request Interface)

Enoch Wun
July 16, 1974

Technical Memo

TM.74-28

1. Introduction

Part of the standard TENEX's processing time is spent in the supervision of disk input and output operations. The CPU time of the PDP-1∅ is very valuable; therefore, it is wasteful to use the PDP-1∅ to handle disk operations when the same job can be done by a minicomputer. Hence, the PDP-1∅ of I4 TENEX employs a PDP-11 to handle the disk operations; the PDP-11 acts like a mediator between the disk controller and the PDP-1∅. This memo illustrates the disk I/O request interface between the PDP-11 and the PDP-1∅. The use of the PDP-11 is peculiar to the I4 TENEX; it is not standard TENEX.

2. General Theory of Operation and Overall Data Organization

Figure 1 shows the general theory of the TENEX disk I/O request service. The memory management and utility I/O services place disk I/O requests on the swapper I/O request queue and utility I/O request queue respectively. The memory management code calls DSKIO to place a request on the Swapper I/O request queue while the utility service calls UDSKIO to place a request on the utility I/O request queue. The processor clock causes PIAPR to be invoked every millisecond; PIAPR calls DSKSV, the disk interrupt service routine, to see if a disk interrupt needs to be stimulated. See Section 3 - "Entry into Disk Interrupt Service Routine (DSKSV)" - for further details. DSKSV does three major functions to complete one disk interrupt service. These three functions are:

1) Clean up for the last data transfer done.
2) Check all drives for waiting requests, and call DSKRCK to dequeue an I/O request for each drive and send seek requests to the PDP-11.
3) Call DFTGO to send a read or write request to the PDP-11.

Note: The disk I/O requests are seek, read, and write requests.

All the disk I/O requests are buffered in a preassigned area; the PDP-11 picks it up from there and then sends it to the disk controller. When the disk has completed seek operations or data transfers, the disk controller tells the PDP-11 to set some flags (these flags are also in this preassigned buffer area) to signal completion. The dotted line in Figure 1 informs us that the disk interrupt service can be triggered by DSKIO and UDSKIO to start the periodic programmed disk interruption, see Section 3 again for a further detailed explanation.

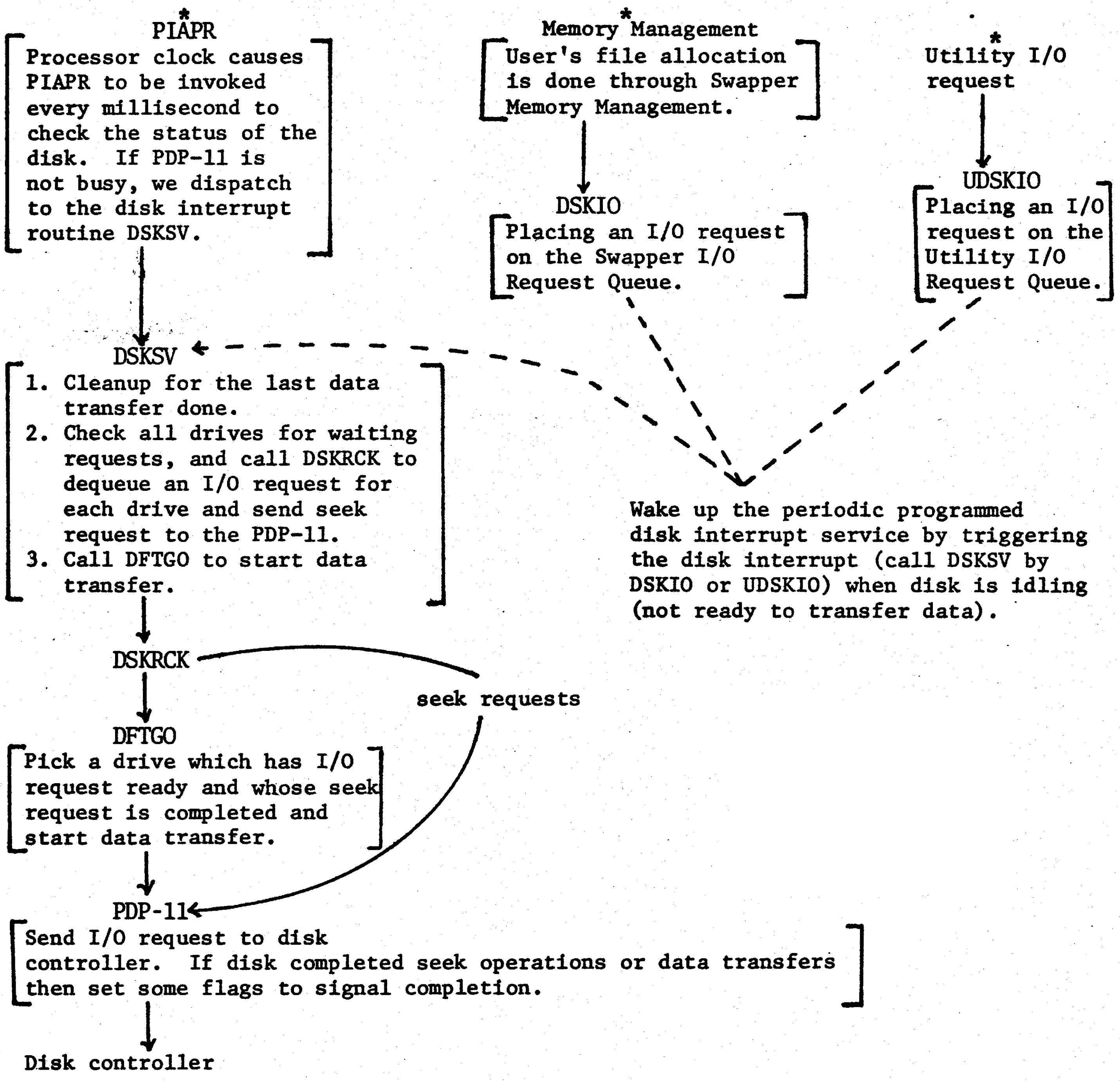

PIAPR*
Processor clock causes PIAPR to be invoked every millisecond to check the status of the disk. If PDP-11 is not busy, we dispatch to the disk interrupt routine DSKSV.
Memory*Management
User's file allocation is done through Swapper Memory Management.
Utility* I/O request
DSKIO
Placing an I/O request on the Swapper I/O Request Queue.
UDSKIO
Placing an I/O request on the Utility I/O Request Queue.
DSKSV
1. Cleanup for the last data transfer done.
2. Check all drives for waiting requests, and call DSKRCK to dequeue an I/O request for each drive and send seek request to the PDP-11.
3. Call DFTGO to start data transfer.
Wake up the periodic programmed disk interrupt service by triggering the disk interrupt (call DSKSV by DSKIO or UDSKIO) when disk is idling (not ready to transfer data).
DSKRCK
seek requests
DFTGO
Pick a drive which has I/O request ready and whose seek request is completed and start data transfer.
PDP-11
Send I/O request to disk controller. If disk completed seek operations or data transfers then set some flags to signal completion.
Disk controller

Figure 1: General Theory of TENEX Disk I/O Operation.

Comments: A) The I/O requests are the seek, read, and write requests.

B) Utility disk I/O requests (UDSKIO) have higher priority than the memory management disk I/O requests (DSKIO); UDSKIO is used only by code which is in some way special or critical, such as the "disk operate" JSYS [.DSKOP], the periodic disk update code [DDMP], the "get swappable monitor" code [GETSWM], etc.

C) When a utility I/O request is completed, bit Ø of the second word of the corresponding command word pair (Section 4.2) is cleared and the page transfer flag (PSKED) is set to be nonzero to indicate that the transfer is done.

When a memory management I/O request is completed, SWPDON is called to set up for unblocking (wake up) of the process and notifying the memory manager that the page transfer has been completed.

<u>Note</u>: PSKED is a full word flag in the scheduler

= Ø if transfer in progress

≠ Ø when no transfer in progress

D) DSKRCK: This routine dequeues an I/O request and loads it into the drive I/O request tables (see Fig. 2), and the seek request queue (see Fig. 1 and 2).

E) DFTGO: This routine starts the data transfer for a drive by sending an I/O request to the PDP-11. The PDP-1Ø puts the I/O request in a particular area in page Ø and page 1 (they both are logical core) in resident monitor space, and the PDP-11 takes I/O requests from there (i.e., we use this area for the interface between the PDP-1∅ and the PDP-11).

Figure 2 depicts the path through which the I/O requests are sent to the PDP-11 and through which the status of disk controller DSK11S is sent to the PDP-1Ø after the completion of an I/O request. First of all the I/O request is placed in the Swapper I/O Request Queue or Utility I/O Request Queue using DSKIO or UDSKIO respectively; using DSKRCK we retrieve an I/O request from either one of the two queues mentioned above. The read or write request part of the retrieved I/O request is placed in the 3 Drive I/O Request Tables, and the seek request part of the retrieved I/O request is placed in Seek Queue; we use DFTGO to get a read/write request and put it in DSK11D, DSK11S, and DSKCCM. (These 3 words are part of the preassigned area mentioned in Section 2 for the communication with the PDP-11.) When the seek operation is done, the PDP-11 sets the seek done flag DSK1ØF and seek hung flag SEEKH (if (SEEKH) = -1, then no seek hung, else contains a drive number of a drive on which seek has been timed out); hence, the PDP-1Ø can be informed about the seek operations. Similarly, when data transfer is done the PDP-11 will put the disk controller status in DSK11S and the PDP-1Ø will come and check the error bits and the done bit of DSK11S. In Section 4 - "Data Structures", we will depict each queue, each table, and each variable/flag in Figure 2 in detail.

Note: Just by looking at the code I suspect the disk controller can simultaneously handle several seek operations, but it only can transfer data for one drive at one time. We have only one controller for the TENEX disk.

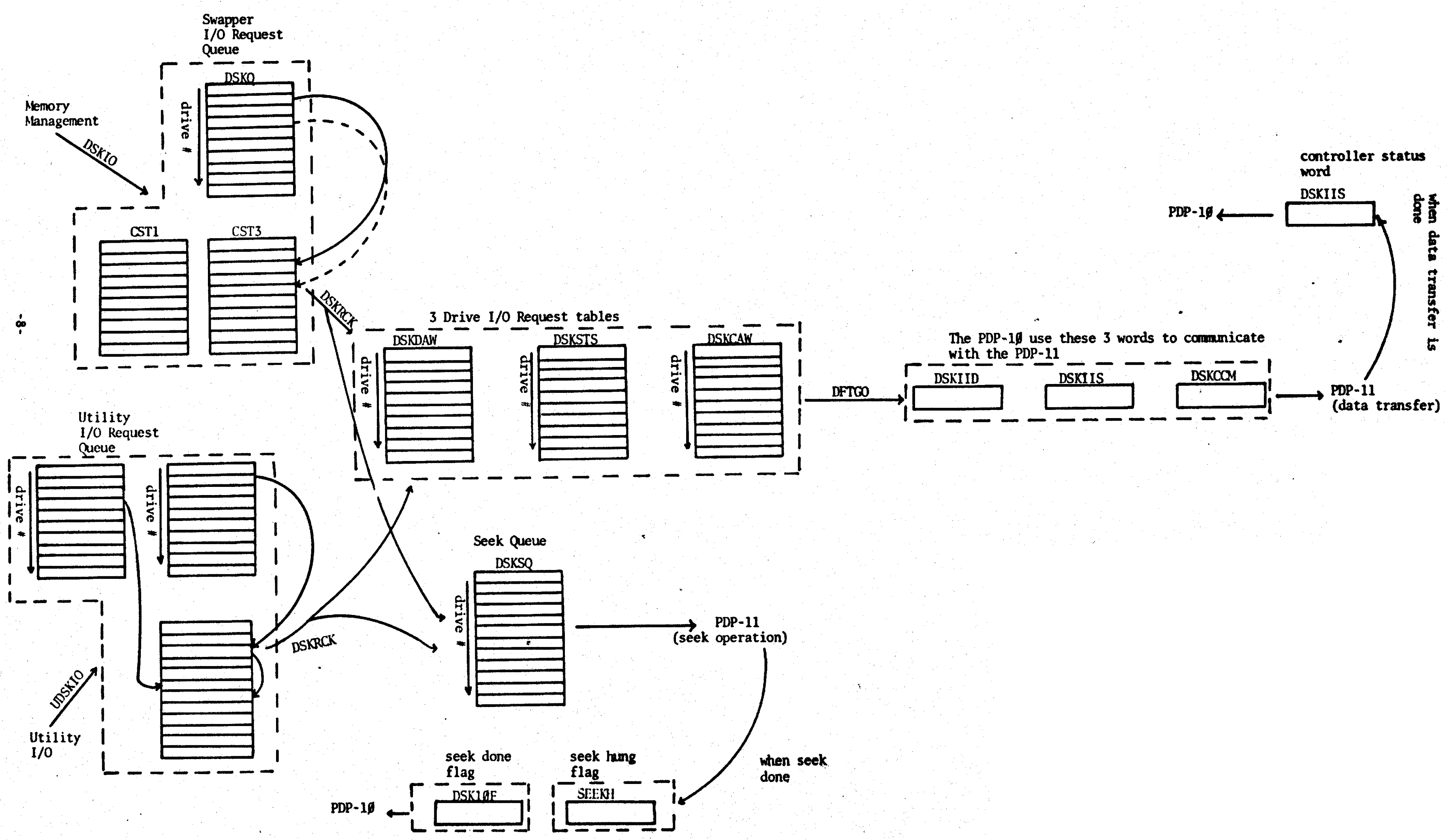


Figure 2: The Overall Data Organization of TENEX Disk I/O Request Service

3. Entry into Disk Interrupt Service Routine (DSKSV)

There are various hardware channels assigned among different devices. Channel 3 receives arithmetic processor hardware interrupts, channel 4 receives drum hardware interrupts, channel 5 receives interrupts from the disk, IMP, DEC TAPE, etc. Upon receiving a given interrupt, the corresponding interrupt routine is invoked.

Because interrupting the processor for various I/O reasons creates costly overhead, the trend is to have smaller peripheral processors (PDP-11's) field the interrupts and process them. The PDP-1Ø and PDP-11 then communicate by flag setting, and passing of requests. PDP-11's are used to service interrupts for the IMP, terminal, disk, and others.

Specifically for the disk, the following scheme is used. A processor clock regularly interrupts the processor via channel 3 every millisecond. From this basic clock derive various statistical and measuring clocks, as well as the time-of-day clock. Periodic routines are invoked at given times. The interrupt routine for channel 3 is called PIAPR. One of the sections of this routine (which is invoked every millisecond) checks the status of the disk. Certain flags signal whether the disk is up, whether the PDP-11 wants another disk request to process, or whether it has its hands full, etc. If the PDP-11 is busy, then no attempt is made to give it more work. If, however, it is not busy, we dispatch to the disk interrupt routine by simulating the occurrence of a hardware interrupt. This simulation is affected by a software instruction which is called a programmed interrupt. The routine DSKSV then takes the appropriate action, making disk transfer requests, etc.

Some of the flags used to communicate between the PDP-1Ø and the PDP-11 are described below. DSKSV is discussed in further detail in a following section.

INTFLG -- This flag is just an interlock on the DSKSV routine. It is set when DSKSV is in progress and is cleared when DSKSV finishes its processing.
-1 means a disk interrupt service routine DSKSV is not allowed.
Ø means the call may be allowed.

DSK1ØF -- It is a seek done flag set by the PDP-11 cleared by PDP-1Ø. (See Section 4.4 (A) for details.)
-1 means seek done.
Ø means seek operation may be in progress.

DSKLUN -- This flag contains the drive # of the last unit that was transferring data, or equal to -1 if disk is idling (not transferring data).

Note: Only when we do not have any I/O requests or all the I/O requests were completed, the disk is idling; DSKLUN is initialized to -1.

When we do not have any I/O request or all the I/O requests were completed, we do not want to waste CPU time to execute the disk interrupt service routine

[DSKSV] every millisecond, since it would have nothing to do. DSKLUN is the variable introduced to serve this purpose. DSKSV will not be executed unless DSKLUN is greater than or equal to zero (drive #). DSKLUN is set to -1 only when the disk interrupt routine checks that all the I/O requests were completed. Then there is a question, "When will you set DSKLUN to greater than or equal to zero to resume the periodical disk interrupt service?" Do not forget the dotted line in Figure 1, the disk interrupt service routine [DSKSV] can be called by DSKIO or UDSKIO. After the periodic disk interrupt was stopped and we have now an I/O request placed on the queue by DSKIO or UDSKIO, DSKSV will put the drive # of this I/O request in DSKLUN to resume periodic disk interrupt servicing and to indicate for which drive we were last transferring data. DSKSV will be subsequently triggered by periodic disk interrupts until all the I/O requests have again been completed.

The channel dispatch routine [PIAPR] is called every millisecond; PIAPR calls DSKSV if the interrupt flag [INTFLG] is cleared, the PDP-11 "seek done" flag [DSK1ØF] is set to be -1 by the PDP-11, and disk is not idling [(DSKLUN) $\geq$ Ø]. Hence, every millisecond we will perform disk interrupt servicing to complete one outstanding I/O request, if an interrupt is allowed.

The way to call DSKSV is:

1) Set INTFLG to -1 (-1 means a disk interrupt is not allowed to be processed), because we do not allow another call of DSKSV when

one is already in progress.

2) Trigger disk interrupt [ISB DSKCHN]

4. Data Structures

The data structures of all the queues, tables, and words shown in Figure 2 will be discussed in some detail in the following subsections.

4.1 The Swapper I/O Request Queue

The data structure of the Swapper I/O Request "queue" is shown in Figure 3a. This queue consists of 3 tables -- DSKQ, CST1, and CST3. CST1 and CST3 are two of the four core status tables (CSTØ to CST3). CSTØ contains the disk addresses of core pages, while CST3 is used to store the chainpointer and the actual (read/write) request. We store/retrieve the request by indexing into the core status tables (using the core page number as an index). Thus, we see that the core status tables (CSTØ and CST3) which are normally used to keep track of real core pages, serve a second role as part of the Swapper I/O Request Queue. DSKQ table has the size of 8 words (the maximum # of drives is 8); each nonzero (page #) entry of DSKQ contains a pointer to the head of a linked list of I/O requests in CST3.

When we are placing an I/O request on this queue, first of all, we use the drive # of this I/O request index into DSKQ to get the head pointer of the list of the I/O requests for the same drive, then we put the new I/O request on the head of this list, so that the I/O requests in each list are in random order.

Although this queue consists of 3 tables, only DSKQ needs to be initialized to zero. If an entry of DSKQ is equal to zero, then it means we do not have a swapper I/O request for the respective drive. Otherwise, we do have at least one swapper I/O request listed in CST3 for this drive.

One should be aware of the fact that the retrieval of requests from this queue is neither first in first out (FIFO) nor last in first out (LIFO). For each drive, we have a list of I/O requests waiting to be sent to the PDP-11, and the track # of the track at which the disk arm is currently positioned. We only retrieve the I/O request for which the required track is the current track or closest to the current track in one fixed direction to minimize arm movement.

DSKRCK is the routine which performs this dequeueing process. As depicted in Figure 2, three drive I/O request tables (DSKDAW, DSKCAW, and DSKSTS) are indexed by drive # (like DSKQ), and all the dequeued I/O requests will be stored in these three tables (indexed by the corresponding index of the DSKQ).

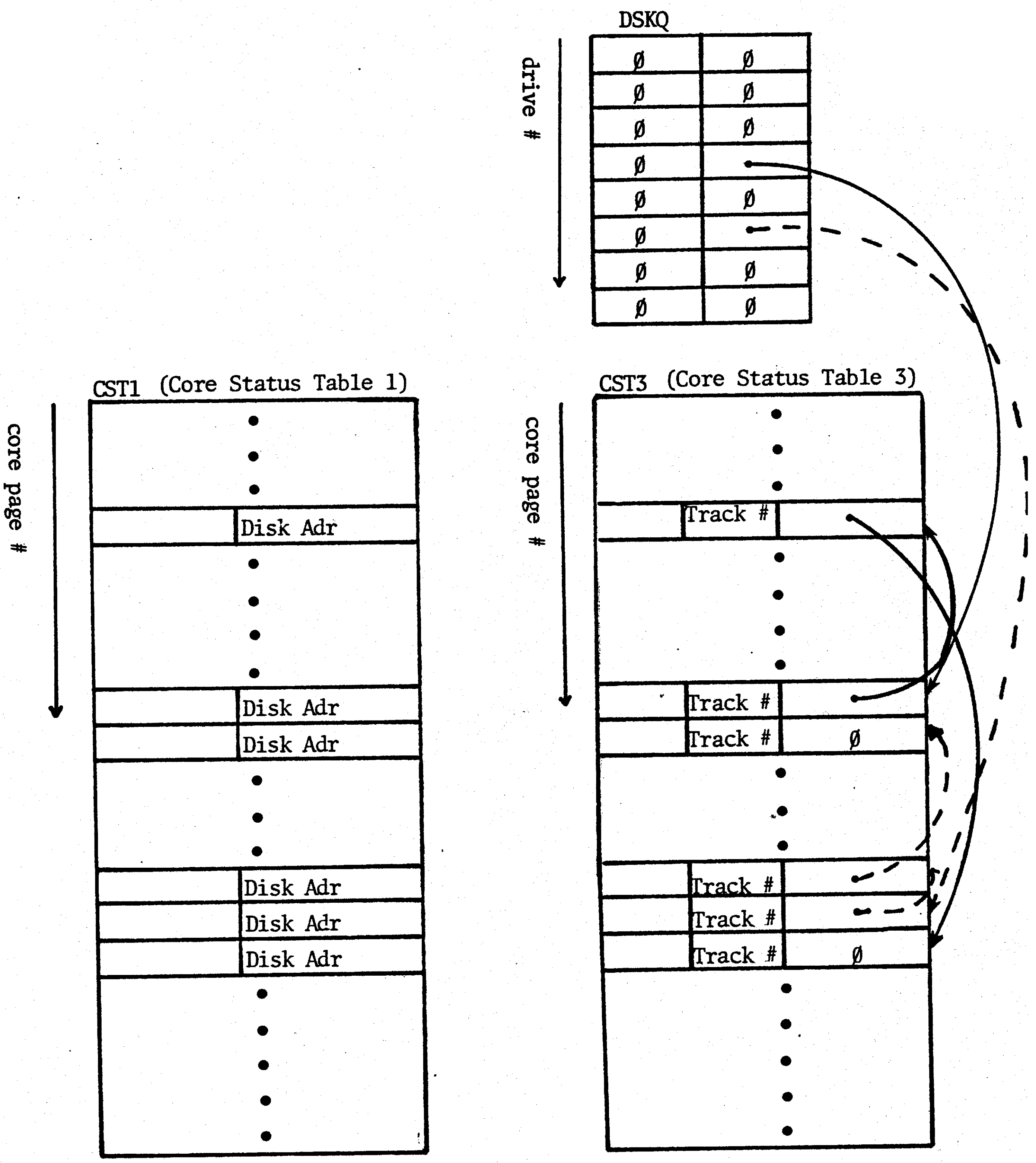
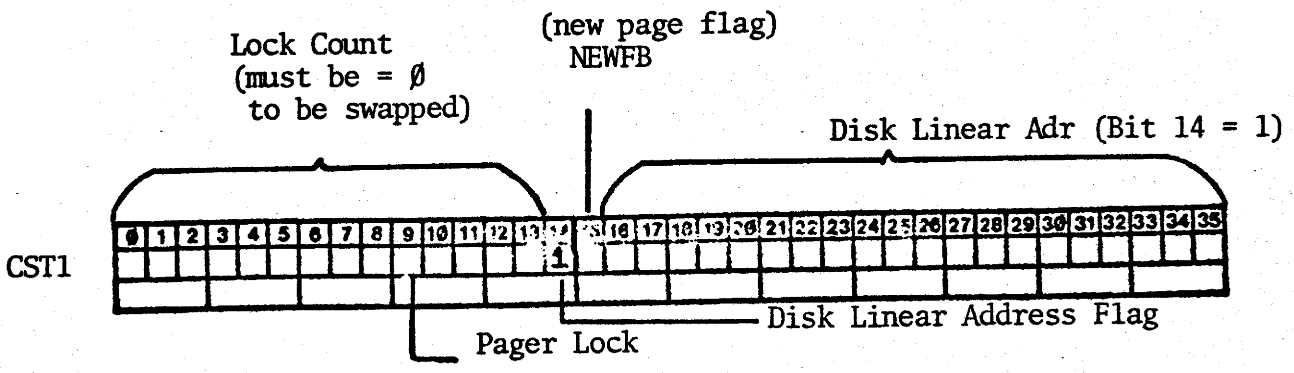


Figure 3a: The Data Structure of Swapper I/O Request Queue.



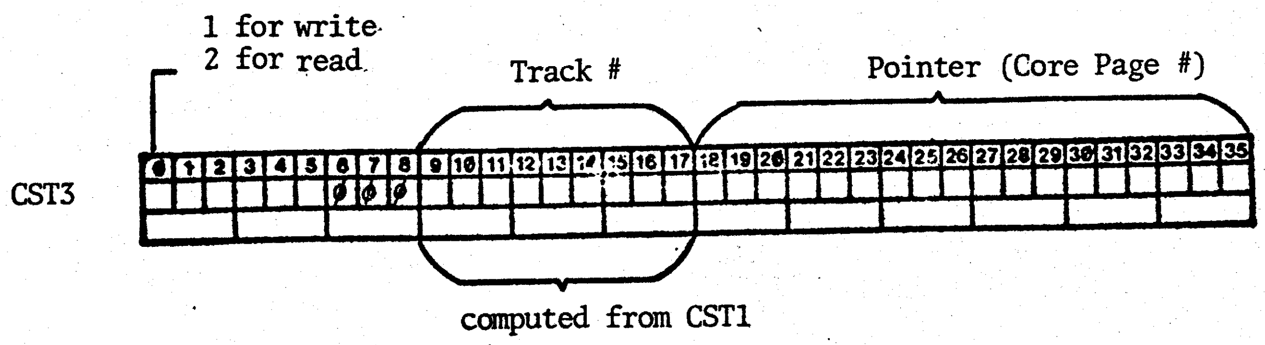
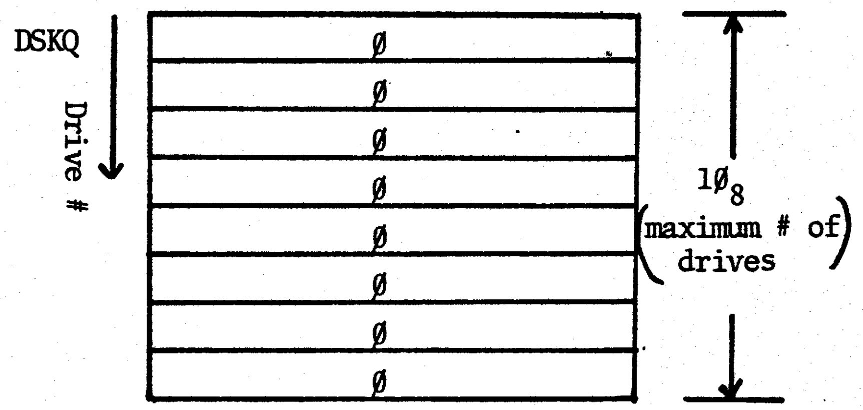


Figure 3b: An Entry in the Swapper I/O Request Queue

The DSKQ is
initialized as:



4.2 Utility Disk I/O Request Queue ---- DSKUI and DSKUO:

The initialization of DSKUI, DSKUO, and DSKCL is described below:

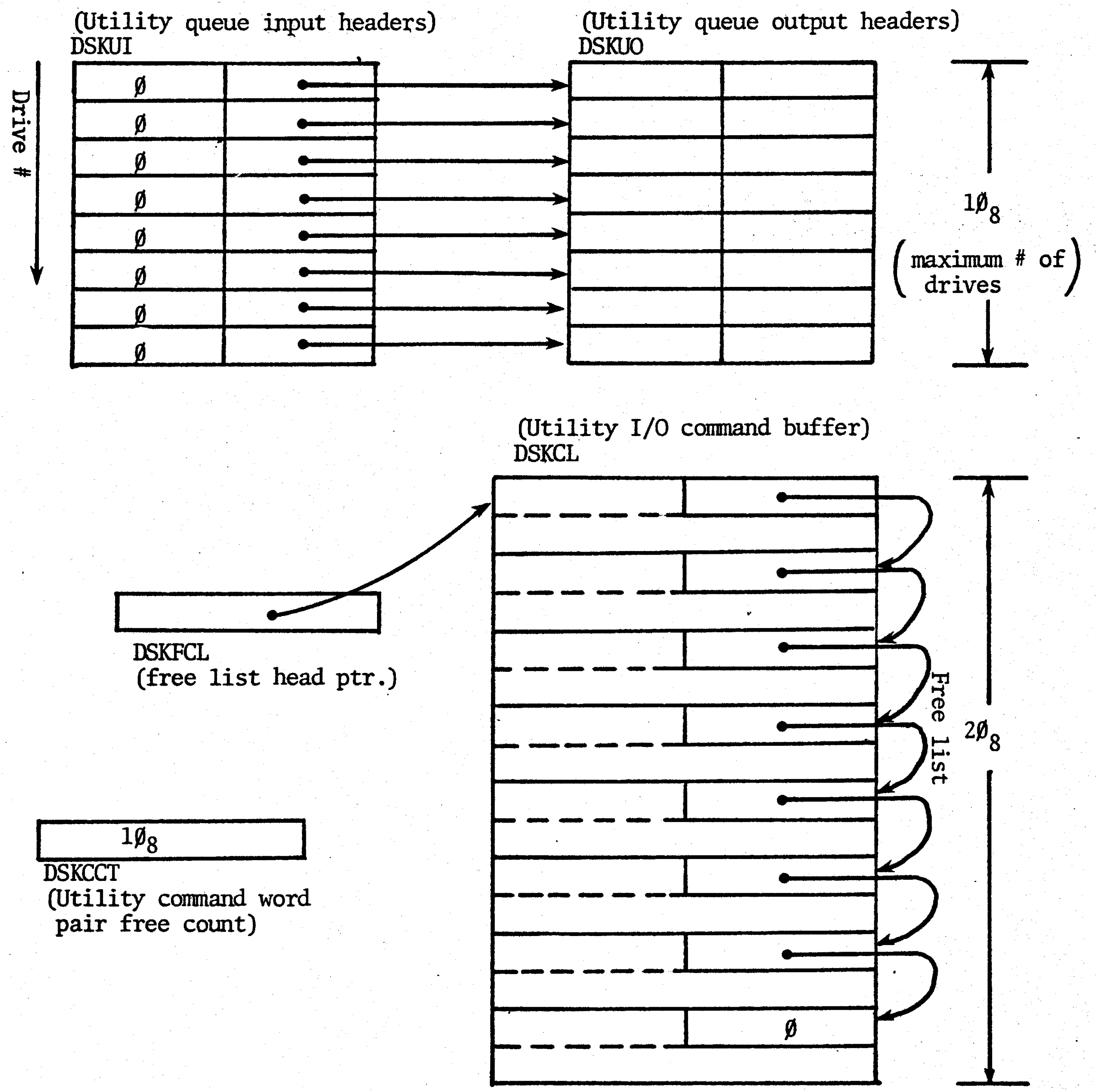

(Utility queue input headers)
DSKUI
(Utility queue output headers)
DSKUO
Drive #
Ø
Ø
Ø
Ø
Ø
Ø
Ø
Ø
1Ø8
(maximum # of drives)
(Utility I/O command buffer)
DSKCL
DSKFCL
(free list head ptr.)
1Ø8
DSKCCT
(Utility command word pair free count)
Free list
2Ø8
Ø

Figure 4a: The Initialization of Utility Disk I/O Request Queue

Note: Possible TENEX bug

DSKUO entries should be initialized to zero because the TENEX code checks whether we have at least one utility I/O request for a drive by checking whether the corresponding DSKUO entry is equal to zero. This entry is indexed by the drive #. If DSKUO is not initialized to zero, then the system will think that some utility I/O request for this drive is waiting to be sent to the PDP-11, when in fact we do not have an I/O request.

The picture on the next page illustrates the structure of the utility I/O request queue. Again, in the utility request queue, for each drive we have a list of I/O requests. DSKUI contains the pointers to the tails of a linked list, and DSKUO contains pointers to the heads. We put the utility I/O request of a drive at the tail of the list and retrieve it from the head, on a first in first out (FIFO) basis.

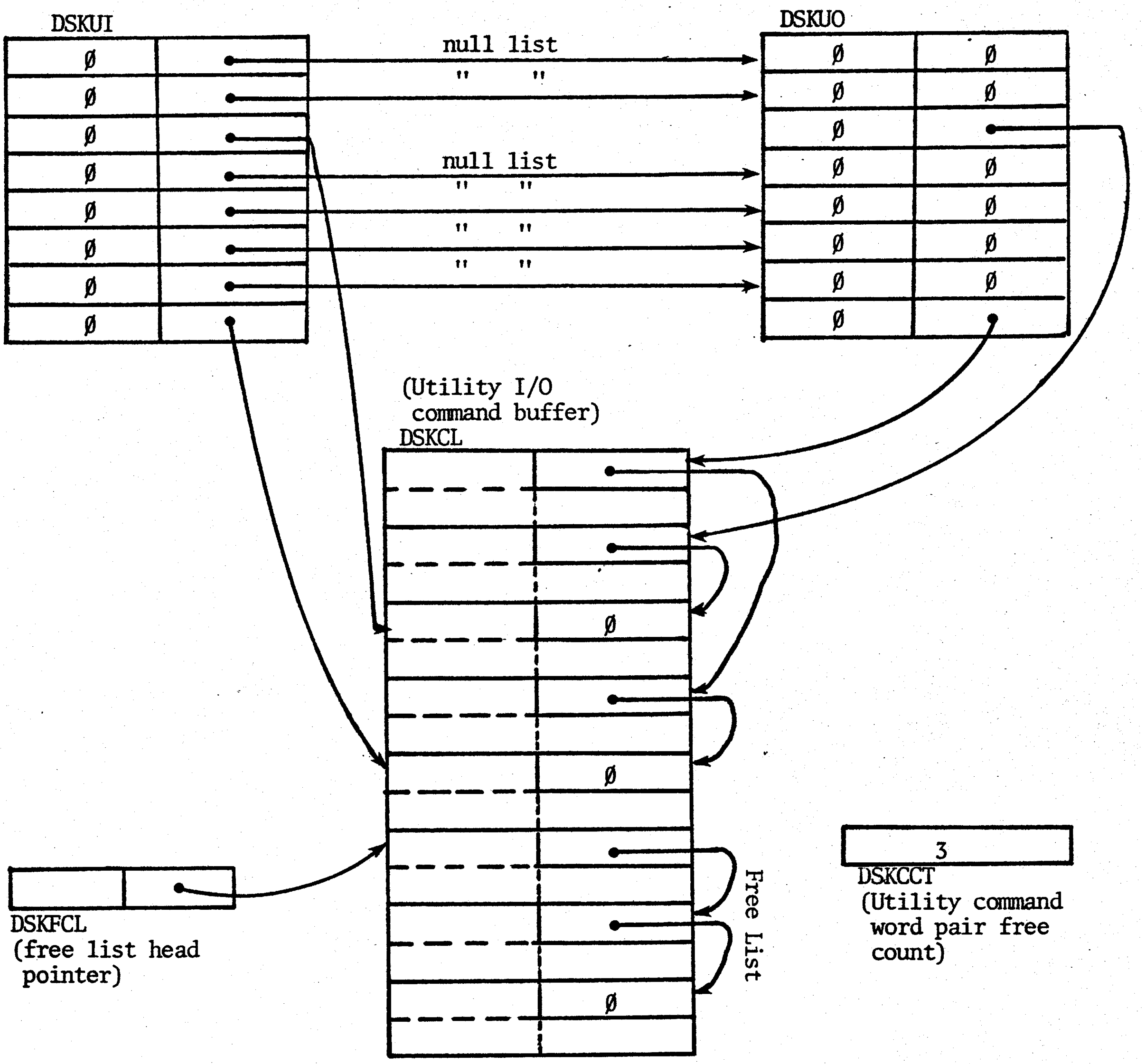
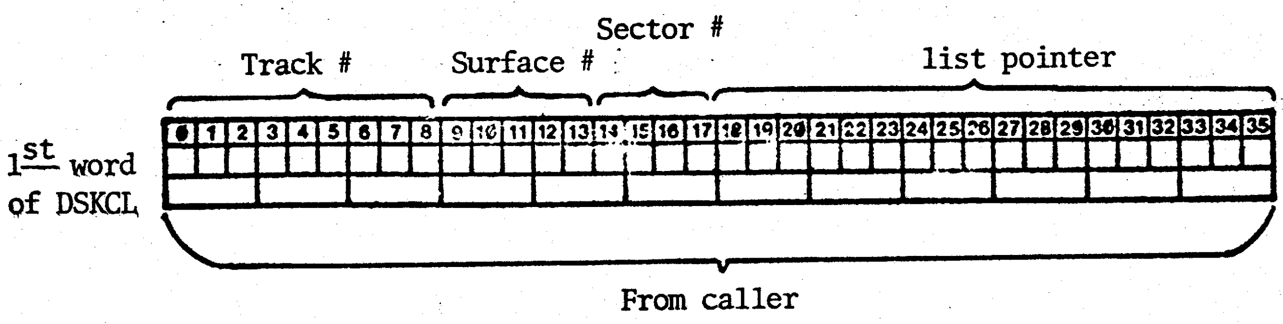


Figure 4b: The Data Structure of Utility I/O Request Queue



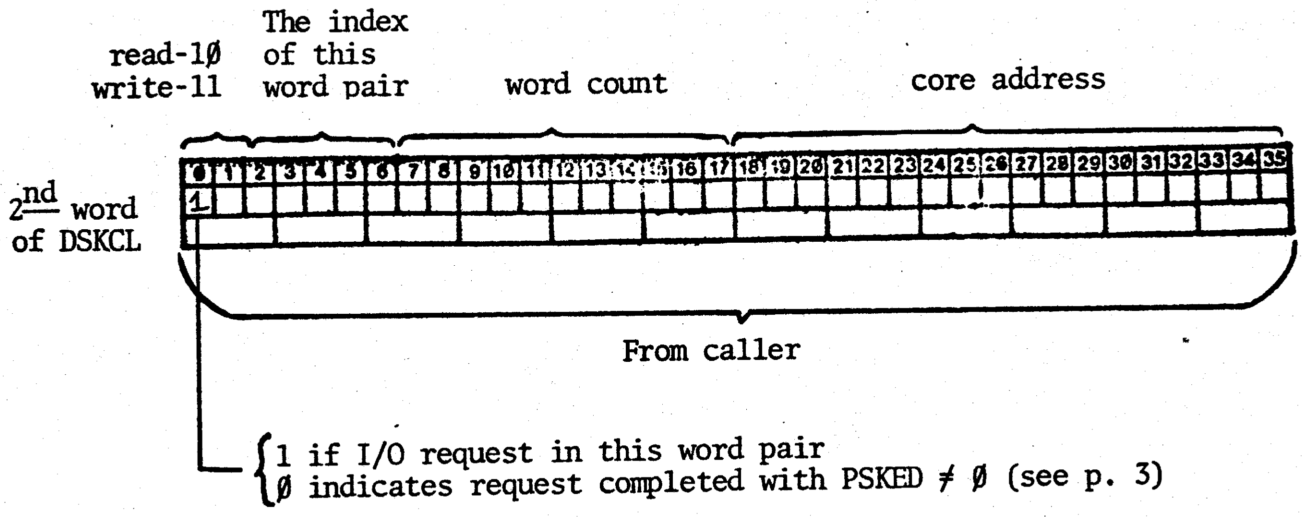

read-10
write-11
The index of this word pair
word count
core address
2nd word of DSKCL
From caller
1 if I/O request in this word pair
0 indicates request completed with PSKED ≠ 0 (see p. 3)

Figure 4c: The Structure of an Utility I/O Request Queue Entry

DSKRCK is the routine which performs this dequeueing process (retrieve an I/O request from this queue). As depicted in Figure 2, three drive I/O request tables (DSKDAW, DSKCAW, and DSKSTS) are indexed by drive # (like DSKUO), and all the dequeued I/O requests will be stored in these 3 tables (indexed by the corresponding index of the DSKQ).

When a utility I/O request is completed, bit Ø of the second word of the command word pair (Fig. 4c) is cleared to indicate done.

4.3 Drive I/O Request Tables

We have four tables to store the current I/O request of each drive. I have designated them as drive I/O request tables (this code does not have an official name for this group of tables). These four parallel tables are indexed into by drive #. The I/O request is redistributed into the first three tables by DSKRCK, so that the I/O request is ready after seek done.

A) DSKDAW - disk address for current operation.

B) DSKCAW - core address and word count for current operation.

C) DSKSTS - status of drive and current track # (the track where the arm is positioned).

D) DSKLSV - time at which last seek or data transfer was started, or -1 if inactive. (For time-out logic.) For instance, DSKLSV is used by time-out logic to check whether seek time has run out for each drive in order to set the seek hung flag.

Note: I suspect the code of this time-out logic is in another package, but I do not know where.

DSKDAW and DSKCAW need not be initialized. DSKSTS and DSKLSV are initialized as below:

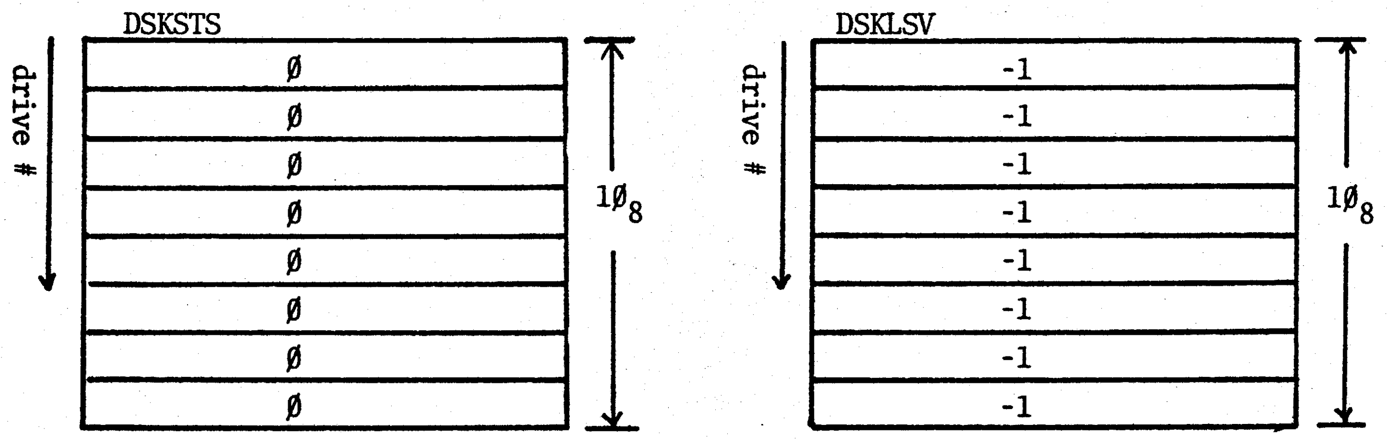
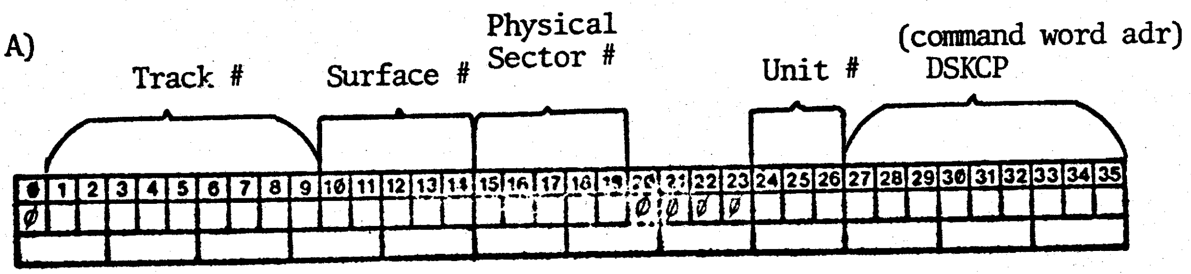


Figure 5: The Initialization of Drive Status [DSKSTS] table and DSKLSV table

The data formats of these four tables are shown below:

A)

DSKDAW

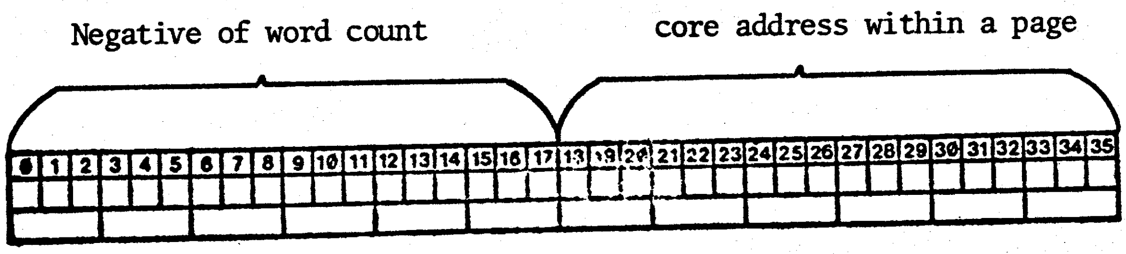


(This is the disk adr in DC1Ø hardware form)

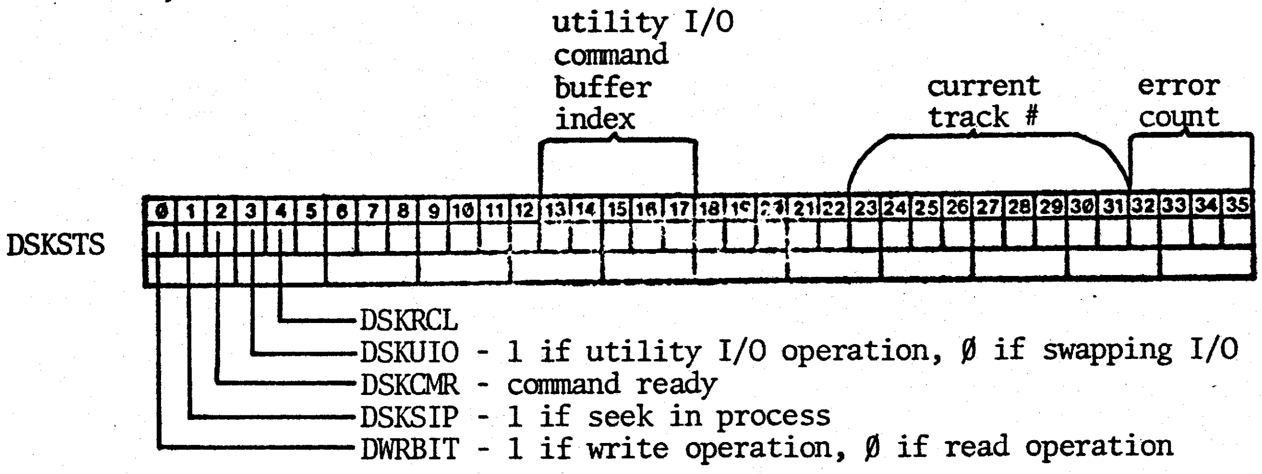
Note: At this moment please do not worry about DSKCP. I will explain it later in detail.

B)

DSKCAW



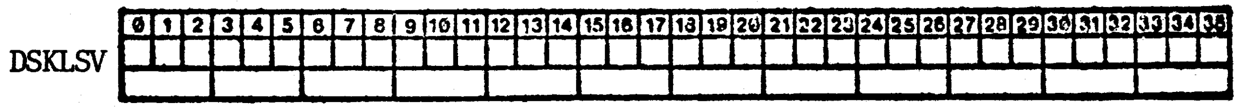
C)


utility I/O command buffer index
current track #
error count
DSKSTS
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
DSKRCL
DSKUIO - 1 if utility I/O operation, ∅ if swapping I/O
DSKCMR - command ready
DSKSIP - 1 if seek in process
DWRBIT - 1 if write operation, ∅ if read operation

The state of a drive is indicated by DSKSIP and DSKCMR flags.

DSKSIP	DSKCMR	states
∅	∅	that Drive is free
1	∅	Drive seeking in progress
∅	1	Drive positioned, transfer ready to be started
1	1	Transfer in progress, or I/O request is being unqueued

D)



(time of day in milliseconds, (same as TODCLK format), when seek or data transfer was begun)

Figure 6: The Data Structure of the 4 Drive I/O Request Table Entries

4.4 Words and Flags for Communication with the PDP-11

Note: Found in the "JOHN"* area and logical core Page ∅ (both in resident monitor space).

A) Disk Seek Queue ---- DSKSQ:

We initialize DSKSQ as below:

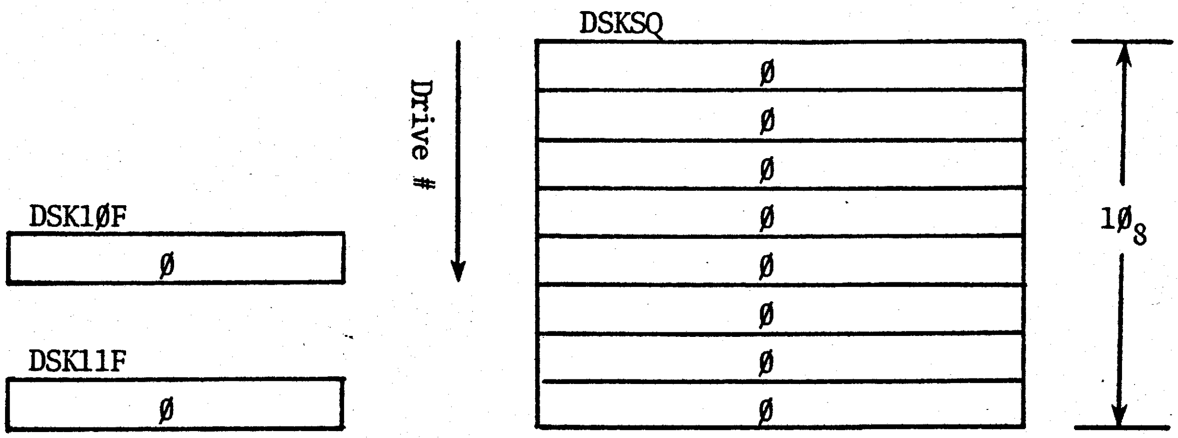


Figure 7: The Initialization of the Disk Seek Queue

* This region in monitor space is known as the "JOHN" area for unknown reasons.

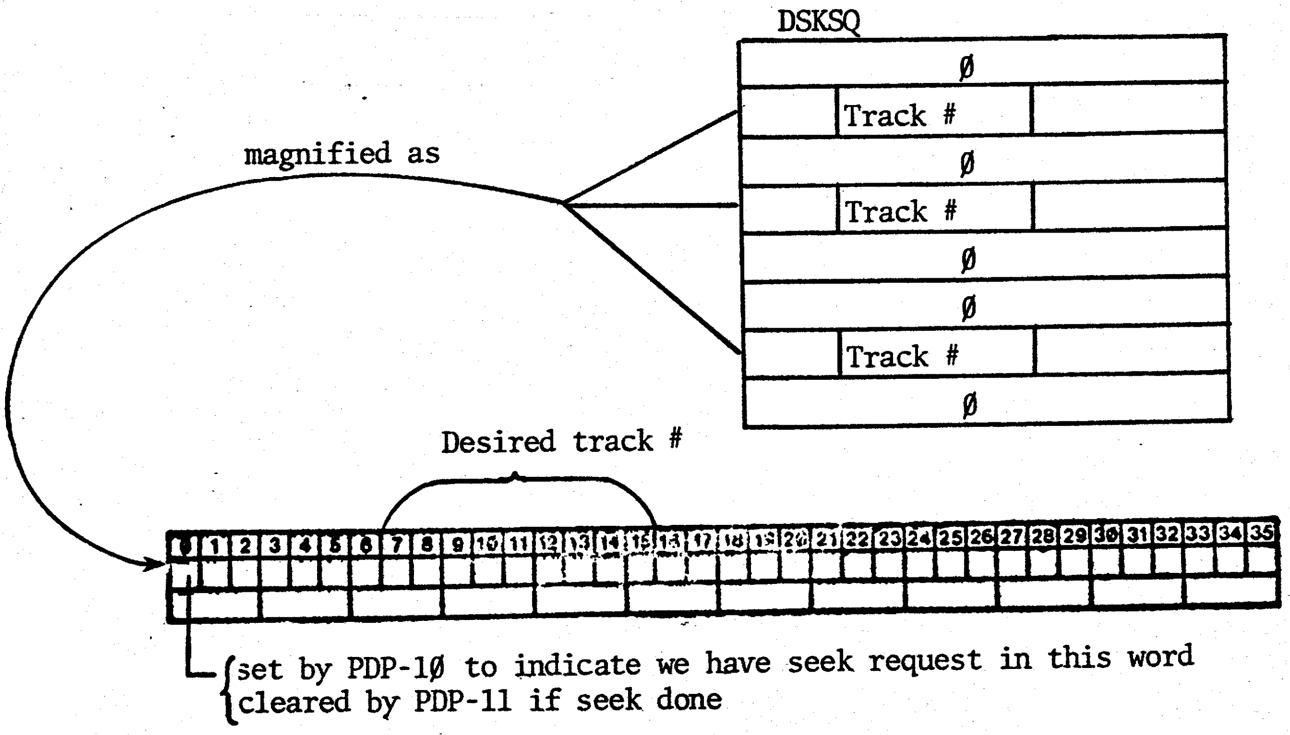


Figure 8: The Data Structure of the Disk Seek Queue (DSKSQ)

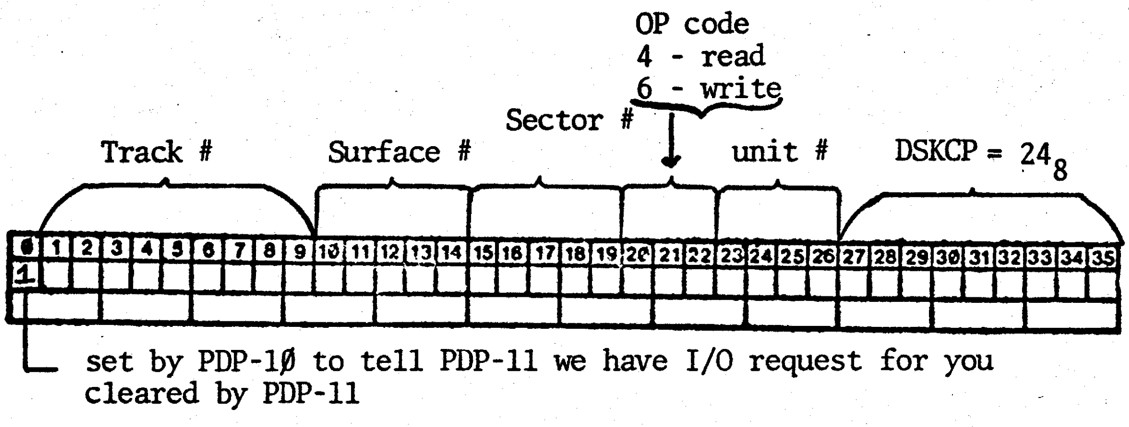
There are two flags, DSK11F and DSK1ØF, which the PDP-1Ø uses to communicate with the PDP-11 to perform seek operations.

DSK11F: is set to be -1 by the PDP-1Ø to tell the PDP-11 to complete all the seek requests in the seek queue DSKSQ, and is cleared by the PDP-11 when the seek requests are finished.

DSK1ØF: is set to be -1 by the PDP-11 to tell the PDP-1Ø all the seek requests in the seek queue DSKSQ are finished, and is cleared by the PDP-1Ø when DSK11F is set to be -1.

B) The next three words described are used by the PDP-1Ø to give an I/O request to the PDP-11 and to get the controller status back from the PDP-11. Up to this point (refer back to Figure 2), we have I/O requests in the three drive I/O request tables [DSKDAW, DSKCAW, and DSKSTS] and the seek operation is completed. The DFTGO routine picks up one drive I/O request, for which the drive is positioned [DSKSIP flag is Ø and DSKCMR flag is 1], and places the request into the following three words [DSK11D, DSK11S, and DSKCCM].

a) DSK11D is the DATAO word in the "JOHN" area:



Note: DSKCP (location 24_8 of logical core page ∅), contains the

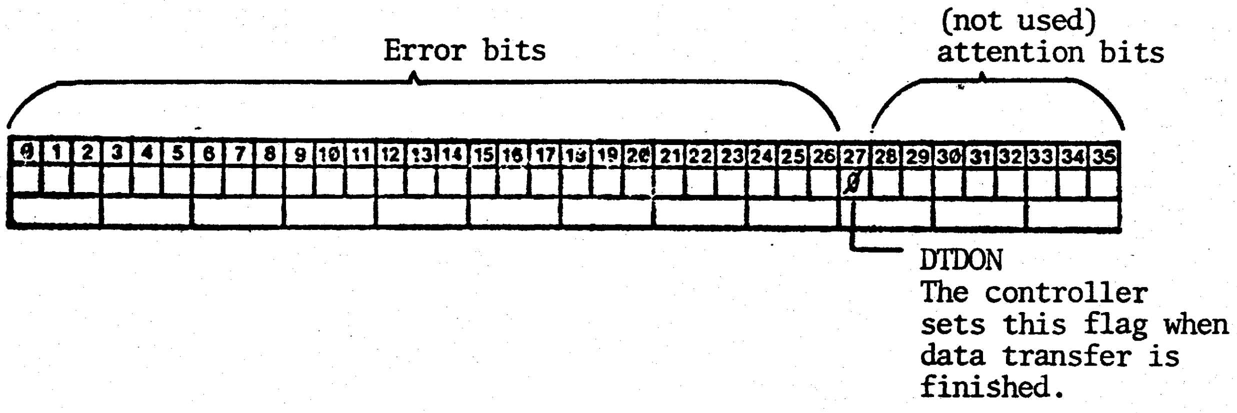
address of the disk command word [DSKCCM + DRELOC].

DRELOC is a relocation factor for the system

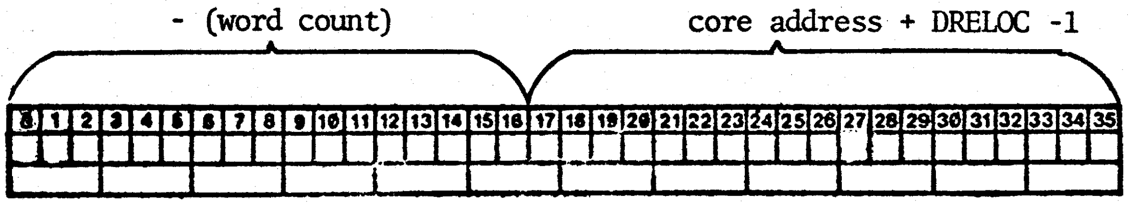
[DRELOC: KIMAIN]. KIMAIN is memory displacement for disk.

b) DSK11S is the controller status word in the JOHN area,

which is cleared by the PDP-1∅, assigned by the PDP-11.



c) DSKCCM is disk command word in page ∅:



5) Words in the PDP-1∅ tell us about the current status of the disk system.

A) DRMUSE ---- count of drum operations was done to disk.

B) DSKLUN ---- contains the drive # of last unit that was transferring data, or equal to -1 if disk is idling (not ready to transfer data).

C) DF1∅LK ---- ∅ if disk owns DF1∅; Job number of other owner if DF1∅ owned by other owner. (DF1∅ is channel connecting the disk and the PDP-1∅.)

D) SEEKHF ---- seek hung flag, -1 if no seek hung, drive no. of drive on which seek has been timed out if seek hung.

5. Disk Handler Packages

1) DSKIO - 1. puts the swapper disk I/O request into the swapper I/O request queue DSKQ (see Fig. 4.1). The request is appended to the head of a list.

2. trigger the disk interrupt if disk is inactive (DSKLUN = -1).

3. return + 1.

2) UDSKIO - 1. puts the utility I/O request into the utility I/O request queue. (see Section 4.2)

2. trigger the interrupt if the disk is inactive (DSKLUN = -1).

3. return + 1.

3) DSKRCK - 1. first of all, we check whether utility I/O request is waiting, because it has highest priority.

Yes, dequeue the utility I/O request queue, and put the request into Drive I/O request tables.

No, is memory management I/O request waiting?

Yes, dequeue the swapper I/O request queue DSKQ, and put a request into Drive I/O request tables.

No, return + 1.

2. In drive status table DSKSTS set DSKSIP and clear DSKCMR (for this drive) to indicate seek in process. (A request was retrieved from a disk I/O queue and is ready to be sent to the PDP-11.)

3. send the seek request of this drive to the PDP-11 through seek queue DSKSQ.

4. return + 1.

Note: The PDP-11 clears bit ∅ of all the entries of the seek queue after the seek is done; therefore, a drive with DSKCMR = ∅, DSKSIP = 1, and DSKSQ $\geq$ ∅ means the seek request of this drive is finished in the last seek done.

4) DSKSV - 1. save DSK11S and clear DSK11S

2. if the disk system is active ((DSKLUN) $\geq$ ∅), then do the following, or else go to next step.

Mark the last unit data transfer finished by clearing DSKSIP and DSKCMR flags of that unit (drive). And tell the scheduler the data transfer is completed

3. if the PDP-11 seek is done (DSK1∅F = -1), then check all drives; then clear DSKSIP and set DSKCMR of a drive if the seek request of that drive is completed (DSKCMR = ∅, DSKSIP = 1, and DSKSQ $\geq$ ∅).

4. check all drives for waiting requests. Send the waiting requests to drive I/O request tables and send seek requests to the PDP-11 by calling DSKRCK.

5. if seek hung, we only can seek one more time.

6. call DFTGO to start the data transfer.

7. return + 1.

5) DFTGO - 1. starting from the last drive which was doing data transfer, pick a

drive which has I/O request ready and no seek in process (DSKCMR = 1 and DSKSIP = ∅).

2. set DSKSIP and DSKCMR of that drive to indicate transfer in process.

3. send the I/O request of that drive to the place where we communicate with the PDP-11 (see Section 4.2).

4. put the current drive # into DSKLUN to remember which drive was last doing a data transfer.

5. return + 1.