To: Distribution

From: John J. Bongiovanni

Date: 03/03/82

Subject: VTOCE Buffer Management

1. ABSTRACT

This document describes a redesign of the physical buffer manager for Volume Table of Contents Entries (VTOCEs). Under this redesign. the I/O activity to VTOCEs is reduced considerably. at the expense of wired buffer space. Existing interfaces are preserved, although some modules outside of the VTOCE physical buffer manager are modified to improve efficiency.

The current physical buffer manager is described in general terms. followed by an overview of the design. The internals of the redesigned physical buffer manager are presented in some detail. The performance trade-offs (I/O savings versus wired memory increases) are quantified.

Send comments on this MTB by one of the following means:

By Multics Mail, on MIT or System M:
Bongiovanni.Multics

By Telephone:
HVN 261-9316 or (617)-492-9316

2. CURRENT SYSTEM

Under the New Storage System (post MR 4.0). a segment resides entirely on one disk volume, and it is described by a Volume Table of Contents Entry (VTOCE). which also resides on that volume. The VTOCE, like all of Gaul, is divided into three parts: activation information. the file map. and permanent information. The activation information is a representation of the Active Segment Table Entry (ASTE). and it consists of information either needed when a segment is active or likely to be modified because a segment is active. The file map describes the disk addresses assigned to each page in the segment. The permanent information contains information which is never or rarely changed. A VTOCE is 192-words long, which is three 64-word sectors (a sector being the physical addressible unit of older disks).

The VTOCE is arranged so that the activation information and the first part of the file map are in the first sector (the first part of the file map reflects the first 96 pages of the segment). The second sector contains only the file map (pages 97 through 224). The third sector contains the remainder of the file map and the permanent information.

The physical buffer manager for VTOCEs is the module vtoc_man. vtoc_man does all I/Os as 64-word (sector) I/Os. That is, each I/O requested by vtoc_man is for a 64-word sector, or one-third of a VTOCE. vtoc_man contains the entry get_vtoce to read one or more sectors of a VTOCE. and the entry put vtoce to write one or more sectors of a VTOCE. The caller of each of these routines supplies a buffer and a bit mask (3 bits) indicating which of the three sectors in the VTOCE are to be read or written. The number of sectors read or written as a result of a call to vtoc_man can be 1, 2, or 3. depending on the setting of the bit mask. Correspondingly, vtoc_man issues 1. 2, or 3 I/Os to satisfy the request.

Typically, only part of a VTOCE is read or written. An example is segment activation. To activate a segment, the activation information in the VTOCE is required. along with as much of the file map as there are (non-null) records. The highest non-null record (the current segment length) is part of the activation information. So activation involves reading the first part of the VTOCE to get the current length. If this is 96 or less. no other VTOCE I/O is necessary, as the requisite portion of the file map has been read. If it is larger than 96, additional VTOCE I/Os are necessary. Similarly, deactivating a segment involves writing the activation information and file map back to the VTOCE. If the current segment length is 96 or smaller, only the first sector need be written.

vtoc_man uses an array of 64-word buffers in the unpaged segment vtoc buffer_seg. There may be up to 64 of these buffers (settable by the site). Each buffer can hold one VTOCE part. and the buffers are independent. When vtoc_man is called to read 3 VTOCE parts. for example. it acquires 3 buffers and then issues 3 I/Os. There is also an optimization wherein the buffers serve as a cache. so that I/Os are not initiated for a VTOCE part which is in some buffer already, left over from some previous operation.

Note that the "unit of issue", when dealing with vtoc_man. is the VTOCE part, and not the VTOCE. In a general sense, vtoc_man only barely realizes that the parts belong to the same logical entity (this is not entirely true. but it is a valid description of the buffering strategy).

vtoc_man was designed when main memory was a scarce system resource. It represents a very clean design aimed at conserving main memory (buffer space) without excessive I/O overhead. The careful design of the VTOCE itself has kept I/O activity to VTOCEs relatively light (typically, 10% of all disk I/O traffic is VTOCE I/O). However, as larger systems have evolved and the economics of components have changed. the assumptions behind the original design have become less valid. Main memory is much cheaper, and disk I/O is a common bottleneck on large systems. By redesigning vtoc_man, VTOCE I/O traffic can be reduced at the expense of large buffer space.

3. DESIGN OVERVIEW

The design is quite simple. and can be summarized concisely:

vtoc buffer_seg remains an unpaged segment. It contains a site-settable number of buffers. each of which is 192-words long (i.e., each can hold an entire VTOCE). There is no particular limit to the number of such buffers. other than the amount of abs-wirable space available. vtoc_buffer_seg is described in vtoc_buffer.incl.pl1, which is attached.

vtoc_man retains all existing entry points and interfaces.

All requests to vtoc_man$get vtoce result in reading the entire VTOCE (unless those parts requested are already in some buffer).

All requests to vtoc_man$put vtoce cause only those parts indicated by the bit mask to be written. Any optimization possible is done. so that as few I/Os as possible are issued. The alternative (viz., writing an entire VTOCE)

causes performance degradation at segment deactivation time, as it would be necessary to read the VTOCE before writing it back to disk (adding an extra I/O per deactivation).

The dctl and disk_control entries to read and write a single sector are changed to entries to read and write some (supplied) number of consecutive sectors. This requires additional information to be carried in the disk queue entry. The revised queue entry can be seen in dskdcl.incl.pl1, which is attached.

4. INTERNALS

The following is an outline the procedural flow of vtoc_man$get_vtoce and vtoc_man$put_vtoce. The other entries in vtoc_man are not changed substantially. The locking strategy is identical to that in the current system and is not discussed.

vtoc_man$get vtoce

Search for a buffer which holds some part of this VTOCE (identified by Physical Volume Table (PVT) index and VTOCE index). If found and out-of-service (I/O in progress). await the completion of the I/O.

If it is found and not out-of-service, check whether it contains the VTOCE parts requested by the caller. If so, return them.

Find a buffer which is not out-of-service and initiate an I/O request to read the VTOCE into that buffer.

Await completion of the I/O, and repeat the process (asynchronous things are going on, so it is not guaranteed to be the case that the buffer is still assigned to the requested VTOCE; if it is not, which is unlikely but possible. the process is repeated).

vtoc_man$put_vtoce

Search for a buffer which contains some part of this VTOCE. If it is out-of-service, await completion of the I/O. If no buffer is found, find a buffer which is not out-of-service.

Copy the VTOCE parts supplied into the buffer.

Initiate the I/O to write the VTOCE parts to disk. Note that

there is one case where this cannot be done in a single I/O (viz. parts 1 and 3). This will be done by initiating one I/O. awaiting its completion. and initiating the second I/O. There is currently no supervisor module which writes parts 1 and 3.

5. OTHER CHANGES

The following are other changes to vtoc_man. to be made at the same time, but not related to the primary objectives of the design. These changes will improve the processor efficiency of vtoc_man at a modest increase in memory.

The linear list of buffer descriptors is replaced by a doubly-threaded used list. As buffers are used, they are moved to the tail of this list. Finding a buffer involves following the used list until one is found which is not out-of-service. This replaces a linear search of buffer descriptors.

A hash table is used to determine whether a given VTOCE (as identified by a PVT index and a VTOCE index) has a buffer assigned. This replaces a linear search.

6. PERFORMANCE

In the words of the cashier at Colleen's Chinese Cuisine. "There's no such thing as a free lunch". And that's the case here.

Metering during peak activity on System M and MIT indicate that between 20% and 25% of all VTOCE I/Os can be eliminated with this design. This corresponds to between 1% and 2% of all disk I/Os.

The cost in wired memory is approximately 128 additional words per buffer. With the current default number of buffers (32). this is an additional 4KW of wired memory. With the current maximum number of buffers (64). this is an additional 8KW of wired memory.

7. SUMMARY OF CHANGES

All changes required to implement this design are indicated below by module.

activate
Call vtoc_man$get_vtoce for the entire VTOCE. instead of reading part 1 to determine how many parts to read.

dctl
Rework for new queue format.

device_meters
Recompile with new include files.

disk_control
Rework for new queue format.

disk_init
Recompile with new include files.

disk_meters
Recompile with new include files.

disk_queue
Rework for new queue format.

get_io_segs
Recompile with new include files.

ioi_assign_disk_channels
Recompile with new include files.

hc_dmpr_primitives
Call vtoc man$get_vtoce for the entire VTOCE. instead of reading part 1 to determine how many parts to read.

init vtoc_man
Change to initialize the new vtoc_buffer seg.

spg_fs_info_
Recompile with new include files.

truncate_vtoce
Call vtoc_man$get vtoce for the entire VTOCE. instead of reading part 1 to determine how many parts to read.

update vtoce
Call vtoc_man$get vtoce for the entire VTOCE. instead of reading part 1 to determine how many parts to read.

verify_lock
     Recompile for new include files.

vtoc_buffer_meters
     Rewrite to print new metering data.

vtoc_interrupt
     Rework for new sector I/O scheme.

vtoc_man
     Rewrite.

wired_shutdown
     Recompile with new include files.

```
"BEGIN INCLUDE FILE dskdcl.incl.alm

     "Created 02/04/82  1712.6 est Thu by convert_include_file,
     "   Version of 12/01/81  1540.3 est Tue.

     "Made from >user_dir_dir>Multics>Bonglovanni>htd>no_salvage_dir>dskdcl.incl.pl1,
     "   modified 02/04/82  1712.5 est Thu

"
"    Structure disk_data
"
     equ       disk_data_size,72


     equ       disk_data.subsystems,0

     equ       disk_data.free_offset,1        " UPPER

     equ       disk_data.status_mask,2
     equ       disk_data.array,8              " LEVEL 2

     equ       disk_data.offset,8             " UPPER

     equ       disk_data.name,9

"
"    Structure disktab
"
     equ       disktab.lock,0
     equ       disktab.nchan,1
     equ       disktab.ndrives,2
     equ       disktab.channels_online,3
     equ       disktab.dev_busy,4             " DOUBLE
     equ       disktab.dev_queued,6           " DOUBLE
     equ       disktab.wq,8                   " LEVEL 2

     equ       disktab.free_q,10              " LEVEL 2

     equ       disktab.abs_mem_addr,11
     equ       disktab.errors,13
     equ       disktab.ferrors,14
     equ       disktab.edac_errors,15
     equ       disktab.pg_io_count,16
     equ       disktab.vt_io_count,18
     equ       disktab.call_lock_meters,20    " LEVEL 2

     equ       disktab.int_lock_meters,24     " LEVEL 2

     equ       disktab.alloc_wait_meters,28   " LEVEL 2
```

```
equ       disktab.run_lock_meters,32     " LEVEL 2

equ       disktab.pg_wait,36             " DOUBLE
equ       disktab.vt_wait,40             " DOUBLE
equ       disktab.pg_io,44               " DOUBLE
equ       disktab.vt_io,48               " DOUBLE
equ       disktab.queue,52               " LEVEL 2

equ       disktab.chantab,308            " LEVEL 2

equ       disktab.devtab,500             " LEVEL 2


"
"         Structure quentry
"

equ       quentry_size,4

equ       quentry.next,0                 " UPPER
equ       quentry.write_sw_word,0
bool      quentry.write_sw,400000        " DL
equ       quentry.sect_sw_word,0
bool      quentry.sect_sw,200000         " DL
equ       quentry.testing_word,0
bool      quentry.testing,100000         " DL
equ       quentry.retry_word,0
bool      quentry.retry,040000           " DL
equ       quentry.used_word,0
bool      quentry.used,020000            " DL
equ       quentry.swap_word,0
bool      quentry.swap,010000            " DL
equ       quentry.cylinder_word,0
equ       quentry.cylinder_shift,0
bool      quentry.cylinder_mask,007777

equ       quentry.pdi_word,1
equ       quentry.pdi_shift,30
bool      quentry.pdi_mask,000077
equ       quentry.coreadd_word,1
equ       quentry.coreadd_shift,6
equ       quentry.dev_word,1
equ       quentry.dev_shift,0
bool      quentry.dev_mask,000077

equ       quentry.sector_word,2
equ       quentry.sector_shift,15
equ       quentry.n_sectors_word,2
equ       quentry.n_sectors_shift,9
bool      quentry.n_sectors_mask,000077

equ       quentry.time,3

"
"         Structure chantab
"

equ       chantab_size,24

equ       chantab.chx,0
equ       chantab.ioi_ctx,1
```

```
equ     chantab.statusp,2
equ     chantab.chanid,4                " DOUBLE

equ     chantab.in_use_word,6
bool    chantab.in_use,400000           " DL
equ     chantab.active_word,6
bool    chantab.active,200000           " DL
equ     chantab.rsr_word,6
bool    chantab.rsr,100000              " DL
equ     chantab.prior_word,6
bool    chantab.prior,040000            " DL
equ     chantab.ioi_use_word,6
bool    chantab.ioi_use,020000          " DL
equ     chantab.inop_word,6
bool    chantab.inop,010000             " DL
equ     chantab.broken_word,6
bool    chantab.broken,004000           " DL
equ     chantab.action_code_word,6
equ     chantab.action_code_shift,9
bool    chantab.action_code_mask,000003

equ     chantab.qrp,7                   " UPPER
equ     chantab.command_word,7
equ     chantab.command_shift,9
bool    chantab.command_mask,000077
equ     chantab.erct_word,7
equ     chantab.erct_shift,0
bool    chantab.erct_mask,000777

equ     chantab.select_data,8           " LEVEL 2

equ     chantab.limit_shift,24
bool    chantab.limit_mask,007777
equ     chantab.mbz_shift,21
bool    chantab.mbz_mask,000007
equ     chantab.sector_shift,0

equ     chantab.connect_time,10         " DOUBLE
equ     chantab.connects,12

equ     chantab.detailed_status_word,13
equ     chantab.detailed_status_shift,28
bool    chantab.detailed_status_mask,000377

equ     chantab.rstdcw,15
equ     chantab.scdcw,16
equ     chantab.sddcw,17
equ     chantab.dcdcw,18
equ     chantab.dddcw,19
equ     chantab.dscdcw,20
equ     chantab.dsddcw,21
equ     chantab.rssdcw,22
equ     chantab.status,23

"
"       Structure qht
"

equ     qht.head,0                      " UPPER
equ     qht.tail,0                      " LOWER
```

```
"
"       Structure devtab
"
        equ     devtab_size,8

        equ     devtab.pvtx_word,0
        equ     devtab.pvtx_shift,27
        bool    devtab.pvtx_mask,000777
        equ     devtab.inop_word,0
        bool    devtab.inop,000400              " DU
        equ     devtab.was_broken_word,0
        bool    devtab.was_broken,000200        " DU
        equ     devtab.broken_word,0
        bool    devtab.broken,000100            " DU
        equ     devtab.abandoned_word,0
        bool    devtab.abandoned,000040         " DU
        equ     devtab.buddy_word,0
        equ     devtab.buddy_shift,6
        bool    devtab.buddy_mask,000077
        equ     devtab.pdi_word,0
        equ     devtab.pdi_shift,0
        bool    devtab.pdi_mask,000077

        equ     devtab.queue_count,1
        equ     devtab.cylinder,2
        equ     devtab.seek_distance,3
        equ     devtab.read_count,4
        equ     devtab.write_count,5
        equ     devtab.time_inop,6              " DOUBLE

"
"       Structure pvtdi
"
        equ     pvtdi.sx_shift,24
        bool    pvtdi.sx_mask,007777
        equ     pvtdi.usable_sect_per_cyl_shift,12
        bool    pvtdi.usable_sect_per_cyl_mask,007777
        equ     pvtdi.unused_sect_per_cyl_shift,0
        bool    pvtdi.unused_sect_per_cyl_mask,007777

"
"       Structure disk_lock_meters
"
        equ     disk_lock_meters_size,4


        equ     disk_lock_meters.count,0
        equ     disk_lock_meters.waits,1
        equ     disk_lock_meters.wait_time,2    " DOUBLE

        equ     RST_LISTX,1                     " MANIFEST
        equ     SC_LISTX,2                      " MANIFEST
        equ     DSC_LISTX,6                     " MANIFEST
        equ     RSS_LISTX,8                     " MANIFEST

"END INCLUDE FILE dskdcl.incl.alm
```

```
/* Begin include file ..... dskdcl.incl.pl1     Last Modified February 1982 */
/* Structures used by the Disk DIM */

/* format: style4,delnl,insnl,tree,ifthenstmt,indnoniterend */
dcl  disk_seg$ ext;                                          /* disk data segment */

dcl  disksp ptr,                                             /* pointer to disk subsystem info */
     diskp ptr;                                              /* pointer to disk DIM info structure */

dcl  1 disk_data based (disksp) aligned,                     /* disk subsystem information */
       2 subsystems fixed bin,                               /* number of subsystems */
       2 free_offset bit (18),                               /* offset of first unused location in segment */
       2 status_mask bit (36),                               /* mask for checking for disk error */
       2 pad (5) fixed bin,                                  /* line up on 0 mod 8 boundary */
       2 array (32),                                         /* per subsystem info */
         (
         3 offset bit (18),                                  /* location of data for this subsystem */
         3 pad bit (18),
         3 name char (4)
         ) unal;                                             /* name of subsystem */

dcl  1 disktab based (diskp) aligned,                        /* control structure for DIM's */
       2 lock bit (36) unal,                                 /* data base lock */
       2 nchan fixed bin,                                    /* number of disk channels */
       2 ndrives fixed bin,                                  /* highest disk drive number */
       2 channels_online fixed bin,                          /* number of disk channels actually in use */
       2 dev_busy bit (64),                                  /* busy bit for each device */
       2 dev_queued bit (64),                                /* requests queued bit for each device */
       2 wq (0:1) like qht,                                  /* wait queue head/tail */
       2 free_q like qht,                                    /* free queue head/tail */
       2 abs_mem_addr fixed bin (26) unsigned,               /* absolute memory address of this structure */
       2 pad fixed bin,
       2 errors fixed bin,                                   /* error count */
       2 ferrors fixed bin,                                  /* fatal error count */
       2 edac_errors fixed bin,                              /* count of EDAC correctable errors */
       2 pg_io_count (0:1) fixed bin,                        /* count of page I/O operations */
       2 vt_io_count (0:1) fixed bin,                        /* count of VTOCE I/O operations */
       2 call_lock_meters like disk_lock_meters,             /* lock meters for call side of DIM */
       2 int_lock_meters like disk_lock_meters,              /* lock meters for interrupt side of DIM */
       2 alloc_wait_meters like disk_lock_meters,            /* meters for queue entry allocations */
       2 run_lock_meters like disk_lock_meters,              /* lock meters for run calls */
       2 pg_wait (0:1) fixed bin (52),                       /* total time spent waiting for page I/O */
       2 vt_wait (0:1) fixed bin (52),                       /* total time spent waiting for VTOCE I/O */
       2 pg_io (0:1) fixed bin (52),                         /* total time spent doing page I/O */
       2 vt_io (0:1) fixed bin (52),                         /* total time spent doing VTOCE I/O */
       2 queue (64) like quentry,   ;                        /* queue entries */
       2 chantab (8) like chantab,                           /* channel information table */
       2 devtab (0 refer (disktab.ndrives)) like devtab;     /* device information table */
%page;
dcl  qp ptr,                                                 /* pointer to queue entry */
     cp ptr;                                                 /* pointer to channel information table */
```

```
dcl  1 quentry based (qp) aligned,                  /* queue entry */
       (
       2 next bit (18),                             /* index to next queue entry */
       2 write_sw bit (1),                          /* non-zero for write operation */
       2 sect_sw bit (1),                           /* non-zero for single sector operation */
       2 testing bit (1),                           /* non-zero if quentry is for disk ready test */
       2 retry bit (1),                             /* non-zero if retry has been performed on broken device */
       2 used bit (1),                              /* non-zero if queue entry in use */
       2 swap bit (1),
       2 cylinder fixed bin (11),                   /* disk cylinder number */
       2 pdi unsigned fixed bin (6),                /* pdi of device */
       2 coreadd bit (24),                          /* memory address for data transfer */
       2 dev unsigned fixed bin (6),                /* disk device code */
       2 sector bit (21),                           /* disk sector address */
       2 n_sectors fixed bin (6) unsigned,          /* number of sectors for sector I/O */
       2 pad bit (9),
       2 time bit (36)                              /* low-order microsecond clock at queue */
       ) unal;                                      /* time entry was queued */

dcl  1 chantab based (cp) aligned,                  /* channel information table */
       2 chx fixed bin (35),                        /* io_manager channel index */
       2 ioi_ctx fixed bin (35),                    /* ioi channel table index */
       2 statusp ptr,                               /* pointer to hardware status word */
       2 chanid char (8),                           /* channel name */
       (
       2 pad0 bit (18),
       2 in_use bit (1),                            /* non-zero if channel being used */
       2 active bit (1),                            /* non-zero if channel active */
       2 rsr bit (1),                               /* non-zero if RSR in progress */
       2 prior bit (1),                             /* priority of current request */
       2 ioi_use bit (1),                           /* non-zero if channel usurped by IOI */
       2 inop bit (1),                              /* non-zero if channel inoperative */
       2 broken bit (1),                            /* non-zero if channel broken */
       2 action_code bit (2),                       /* saved from status */
       2 pad1 bit (9)
       ) unal,
       (
       2 qrp bit (18),                              /* rel ptr to queue entry */
       2 pad2 bit (3),
       2 command bit (6),                           /* peripheral command */
       2 erct fixed bin (8)                         /* error retry count */
       ) unal,
       2 select_data,                               /* data passed to IOM on select */
         (
         3 limit bit (12),                          /* limit on number of sectors */
         3 mbz bit (3),
         3 sector bit (21)                          /* sector address */
         ) unaligned,
       2 connect_time fixed bin (52),               /* time of last connect */
       2 connects fixed bin,                        /* count of connects performed */
       2 detailed_status (0:8) bit (8) unal,        /* detailed status bytes */
       2 rstdcw bit (36),                           /* restore command */
       2 scdcw bit (36),                            /* select command */
       2 sddcw bit (36),                            /* select data xfer */
       2 dcdcw bit (36),                            /* command to read or write */
       2 dddcw bit (36),                            /* data xfer DCW */
       2 dscdcw bit (36),                           /* RSR command */
       2 dsddcw bit (36),                           /* RSR data xfer */
       2 rssdcw bit (36),                           /* RSS command */
       2 status bit (36) aligned;                   /* saved status */
```

```
%page;
dcl  1 qht aligned based,                                  /* queue head/tail structure */
       2 (head, tail) bit (18) unal;

dcl   dp ptr,                                              /* pointer to device information table */
      pvtdip ptr;                                          /* pointer to dim_info in PVT entry */

dcl   1 devtab based (dp) aligned,                         /* device information table */
        (
        2 pvtx fixed bin (8),                              /* index of PVT entry for device */
        2 inop bit (1),                                    /* device inoperative */
        2 was_broken bit (1),                              /* device previously broken */
        2 broken bit (1),                                  /* device down */
        2 abandoned bit (1),                               /* device lost and gone forever */
        2 pad bit (11),
        2 buddy unsigned fixed bin (6),                    /* other device on this spindle or 0 */
        2 pdi unsigned fixed bin (6)
        ) unal,                                            /* primary device index */
        2 queue_count fixed bin (8),                       /* count of requests queued for device */
        2 cylinder fixed bin (11),                         /* current cylinder position */
        2 seek_distance fixed bin (35, 18),                /* average seek distance */
        2 read_count fixed bin,                            /* count of reads */
        2 write_count fixed bin,                           /* count of writes */
        2 time_inop fixed bin (52);                        /* time drive became inoperative */

dcl   1 pvtdi based (pvtdip) aligned,                      /* disk DIM info in PVT entry */
        (
        2 sx fixed bin (11),                               /* structure index */
        2 usable_sect_per_cyl fixed bin (11),              /* # of usable sectors on disk cylinder */
        2 unused_sect_per_cyl fixed bin (11)
        ) unal;                                            /* # of unused sectors at end of cylinder */

dcl   1 disk_lock_meters based aligned,                    /* lock meters for disk DIM */
        2 count fixed bin,                                 /* total number of attempts */
        2 waits fixed bin,                                 /* number of attempts which required waiting */
        2 wait_time fixed bin (52);                        /* total time spent waiting */

dcl   (
      RST_LISTX init (1),                                  /* listx for restore */
      SC_LISTX init (2),                                   /* listx for select */
      DSC_LISTX init (6),                                  /* listx for RSR */
      RSS_LISTX init (8)
      ) fixed bin (12) static options (constant);          /* listx for RSS */

/* End of include file ..... dskdcl.incl.pl1 */
```

```
"BEGIN INCLUDE FILE vtoc_buffer.incl.alm

     "Created 02/05/82  2036.8 est Fri by convert_include_file,
     "    Version of 12/01/81  1540.3 est Tue.

     "Made from >udd>Multics>Bongiovanni>hardcore_test_dir>no_salvage_dir>vtoc_buffer.incl.pl1,
     "    modified 02/05/82  2036.8 est Fri

"
"         Structure vtoc_buffer
"

          equ       vtoc_buffer.lock,0              " LEVEL 2

          equ       vtoc_buffer.processid,0
          equ       vtoc_buffer.wait_event,1

          equ       vtoc_buffer.notify_sw_word,2
          bool      vtoc_buffer.notify_sw,400000    " DU

          equ       vtoc_buffer.n_bufs,3
          equ       vtoc_buffer.n_hash_buckets,4
          equ       vtoc_buffer.hash_mask,5
          equ       vtoc_buffer.abs_addr,6
          equ       vtoc_buffer.wait_event_constant,8 " DOUBLE

          equ       vtoc_buffer.buf_desc_offset,10 " UPPER

          equ       vtoc_buffer.buf_offset,11       " UPPER

          equ       vtoc_buffer.hash_table_offset,12 " UPPER

          equ       vtoc_buffer.meters,13           " LEVEL 2


          equ       vtoc_buffer.hash_table,14       " UPPER

          equ       vtoc_buffer.buf_desc,0          " LEVEL 2

          equ       vtoc_buffer.buffer,0            " LEVEL 2


"
"         Structure vtoc_buf_desc
"

          equ       vtoc_buf_desc_size,4

          equ       vtoc_buf_desc.pvte_rel,0        " UPPER
          equ       vtoc_buf_desc.vtocx,0           " LOWER
```

```
        equ     vtoc_buf_desc.part_desc_rel,1 " UPPER
        equ     vtoc_buf_desc.parts_used_word,1
        equ     vtoc_buf_desc.parts_used_shift,15
        bool    vtoc_buf_desc.parts_used_mask,000007
        equ     vtoc_buf_desc.parts_os_word,1
        equ     vtoc_buf_desc.parts_os_shift,12
        bool    vtoc_buf_desc.parts_os_mask,000007
        equ     vtoc_buf_desc.parts_err_word,1
        equ     vtoc_buf_desc.parts_err_shift,9
        bool    vtoc_buf_desc.parts_err_mask,000007
        equ     vtoc_buf_desc.notify_sw_word,1
        bool    vtoc_buf_desc.notify_sw,000400 " DL
        equ     vtoc_buf_desc.write_sw_word,1
        bool    vtoc_buf_desc.write_sw,000200 " DL

        equ     vtoc_buf_desc.ht_thread,2       " UPPER

        equ     vtoc_buf_desc.used_thread,3     " LEVEL 2

        equ     vtoc_buf_desc.fp,3              " UPPER
        equ     vtoc_buf_desc.bp,3              " LOWER

"
"       Structure vtoce_buffer
"
        equ     vtoce_buffer_size,192


        equ     vtoce_buffer.words,0

        equ     N_PARTS_PER_VTOCE,3             " MANIFEST
        equ     VTOCE_BUFFER_SIZE,0192          " MANIFEST

"END INCLUDE FILE vtoc_buffer.incl.alm
```

```
/*  START OF:       vtoc_buffer.incl.pl1   Feruary 1982      * * * * * * * * * * * * * * * * * * * */

     dcl     vtoc_buffer_seg$        ext;

     dcl     vtoc_buffer_segp        ptr;
     dcl     vtoc_buf_descp          ptr;
     dcl     vtoc_bufp               ptr;


     dcl     1 vtoc_buffer           aligned based (vtoc_buffer_segp),

               2 lock,                                       /* Global lock for VTOC buffers */
                 3 processid         bit (36) aligned,       /* Owner */
                 3 wait_event        bit (36) aligned,       /* For lock */
                 3 notify_sw         bit (1) aligned,        /* ON => notify on unlock */

               2 n_bufs              fixed bin,              /* Number of full VTOCE buffers */
               2 n_hash_buckets      fixed bin,              /* Number of hash table buckets */
               2 hash_mask           bit (36) aligned,       /* Mask for hash algorithm */
               2 abs_addr            fixed bin (24),         /* Absolute address of vtoc_buffer_seg */
               2 wait_event_constant fixed bin (36) uns,     /* Constant to add to part index to form wait event */
               2 buf_desc_offset     bit (18),               /* Offset of buf_desc */
               2 buf_offset          bit (18),               /* Offset of buf */
               2 hash_table_offset   bit (18),               /* Offset of hash_table */
               2 meters,
                 3 pad               fixed bin,              /* For now */

               2 hash_table          (vtoc_buffer_seg.n_hash_buckets) bit (18) aligned,


               2 buf_desc            (vtoc_buffer_seg.n_bufs) aligned like vtoc_buf_desc,

               2 buffer              (vtoc_buffer_seg.n_bufs) aligned like vtoce_buffer;


     dcl     1 vtoc_buf_desc         aligned based (vtoc_buf_descp),
               2 pvte_rel            bit (18) unal,          /* Offset to PVTE within PVT */
               2 vtocx               fixed bin (17) unal,    /* VTOCE Index */
               2 part_desc_rel       bit (18) unal,          /* Offset to first part descriptor for this buffer */
               2 parts_used          bit (3) unal,           /* Mask of parts used or os */
               2 parts_os            bit (3) unal,           /* Mask of parts out-of-service */
               2 parts_err           bit (3) unal,           /* Mask of parts with I/O errors (hot) */
               2 notify_sw           bit (1) unal,           /* ON => notify requied on I/O completion */
               2 write_sw            bit (1) unal,           /* ON => write I/O */
               2 pad                 bit (7) unal,
               2 ht_thread           bit (18) unal,          /* Offset of next entry in hash table */
               2 pad1                bit (18) unal,
               2 used_thread         aligned,                /* Used list thread */
                 3 fp                bit (18) unal,          /* Forward pointer */
                 3 bp                bit (18) unal;          /* Backward pointer */
```

```
     dcl     1 vtoce_buffer          aligned based,
               2 words               (3 * 64) bit (36) aligned;


     dcl     N_PARTS_PER_VTOCE       fixed bin int static options (constant) init (3);
     dcl     VTOCE_BUFFER_SIZE       fixed bin int static options (constant) init (3 * 64);


/*  END OF:         vtoc_buffer.incl.pl1                  * * * * * * * * * * * * * * * * * * * */
```