SECTION 6

Paging and Disk Management

6.1 VIRTUAL PAGE MANAGEMENT

This section discusses techniques for dealing with the management of virtual pages and their associated disk storage. Virtual page management functions include the page replacement strategy, updating swap status indications, and mapping the paging-related secondary memory across different disk page bands, perhaps on different physical units.

The last portion of this section describes a number of virtual memory management primitives which can be used from Lisp.

6.2 PAGE MANAGEMENT OVERVIEW

The following is an overview of the paging and disk management algorithms for processing a page fault.

1. Given a faulted virtual address, find the disk device and the device block offset where the demanded page resides.

2. Issue a disk read operation into a previously cached memory page frame. If there is no available cached memory page frame, scan the PPD looking for a (preferably clean) page to evict. Issue the disk read into the page frame found.

3. While waiting for the demand page disk I/O to complete, find a new cached memory page frame for the next demanded page by scanning the PPD. Choose a page for eviction from physical memory. If the page is not dirty, the memory maps are cleared and the PHT entry deleted. The PPD entry is coded "FFFE" indicating a free page.

4. If the page to be evicted is dirty then it must be written to a swap partition. The PPD entry is coded Disk-Write-Pending. The PHT entry and the memory maps are cleared by the disk servicing routine later after the disk operation has completed.

5. When swapping out a dirty page, determine if there are contiguous dirty virtual memory pages that can be swapped out at the same time to adjacent disk locations. If so, prepare a multi-record write request for all such pages.

6. If any pages being swapped out have never been assigned read/write page device storage, find an available page device and assign space to the pages' clusters (group of 16 contiguous virtual pages). Initiate the disk write.

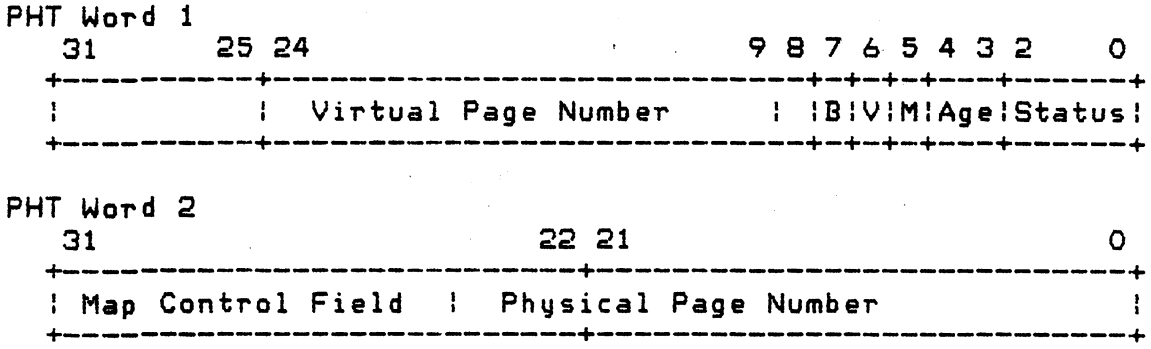
7. Create a PHT entry for the faulted page.

8. Wait for the original disk read to complete. If any disk writes were initiated they will continue in parallel with computation.

6.3 PAGE HASH TABLE

The Page Hash Table (PHT) contains physical memory mapping information for every virtual page currently in physical memory. Page exception handling may consult the PHT in order to find information to use in reloading the hardware maps. In addition, the page replacement algorithm (FINDCORE) uses information stored in the PHT. The format of a PHT entry is shown in Figure 6-1.

If there is a page exception and there is no entry in the Page Hash Table then it is a page fault, and the page needs to be read from disk. The disk address is calculated using a disk mapping scheme described later.



WORD 1:

STATUS	<0:2>	= Map status code. See table of status codes below.
AGE	<3:4>	= Number of times page has been aged.
M	<5>	= Modified bit. 1 = Modified (dirty).
V	<6>	= Valid bit. 1 = PHT entry valid.
B	<7>	= Background Write bit. 1 = Background write in progress.
VPN	<9:24>	= Virtual Page Number

WORD 2:

PHYS-PG	<0:21>	= Physical Page Number.
MAP-CONT	<22:31>	= Hardware map control data.

Figure 6-1 Page Hash Table Entry Format

6.3.1 Page Replacement Process.

The page replacement strategy is used to determine which pages should be swapped out so that a needed virtual page can use this physical memory page frame. It is also known as FINDCORE.

Page selection is accomplished on demand when a page is needed for reuse. The physical page data table is consulted starting with the top of the Least Recently Used (LRU) list formed by the entries in the PPD LINK FIELD of each entry. The Least Recently Referenced page index is held in an A-Register that defines starting entry of the list. The memory page indicated is the least recently referenced page. The status of the page is checked, and if it is available, it is designated for immediate use. If the page is not available for some reason, the scan proceeds to the page indicated by the contents of the PPD Link Field of the current entry and so forth until the oldest usable

page is found.

While scanning down the list, action is taken on individual pages as follows by consulting the page's PHT status code as described below. Table 6-1 provides a complete description of all PHT status codes.

1. If page status is normal then set status to age trap and the PPD link is removed from its current position and added to the end of the list as the new Most Recently Referenced page.

2. If page status is age trap, flushable or prepage, then remove the page from the list and designate it as the next page to be used for replacement. The scan stops at the first page found that satisfies this status.

3. If page status is anything else, or if the background writing bit is set, continue the scan with the next linked page leaving this page in the same place in the list.

Table 6-1 Page Hash Table: Swap Status Codes

Unused:	0 - Unused code.
Normal:	1 - An ordinary page is swapped in here.
Flushable:	2 - Means that there is a page here, but it has been marked as no longer needed and can be used to swap a new page into. This page may first have to be written out if the map status indicates that it has been modified since last written (map status code = 4)
Pre-page:	3 - Same as Flushable, but came in via Prepage.
Age trap set:	4 - This page was in normal status, but is now being considered for swap-out. The map may not be set up for this) page. If someone references the page, the swap status should be set back to "normal."
Wired down:	5 - The page swapping routines may not re-use the memory occupied by this page for some other page. This is used for the some other page. This is used for memory pages wired down by the SYS:WIRE-PAGE primitive and related routines.
Disk Read Pending:	6 - This page is being read in from disk. The virtual address assigned this page may not be used and the page swapping routines may not re-use the memory occupied by this page.
Disk Write Pending:	7 - This page is being written to disk in anticipation of freeing it for re-use by the page swapping routines. The virtual memory can be referenced and used, but the physical memory may not be re-used until the disk operation has completed.

6.3.2 Page Table Sizes.

The size of the page hash table is related to the size of physical memory. Since a hash technique is used to search the page hash table, two 2-word entries are usually allocated for every physical page in the system. See Figure 6-2 for sizes of both the PHT and the PPD for different memory sizes. The PHT and PPD together consume slightly less than 4% of the physical memory.

Virtual Memory Size = 128K Words

Physical Memory Size		PHT Size	PPD Size
2MB	(512K Words)	1024 pages	4096 words	1024 words
4MB	(1M Words)	2048 pages	8192 words	2048 words
8MB	(2M Words)	4096 pages	16393 words	4096 words
10MB	(2.5M Words)	5120 pages	20480 words	5120 words
12MB	(3M Words)	6144 pages	24576 words	6144 words
16MB	(4M Words)	8192 pages	32768 words	8192 words
32MB	(8M Words)	16393 pages	65536 words	16393 words
64MB	(16M Words)	32768 pages	131072 words	32768 words
128MB	(32M Words)	65536 pages	131072 words	65536 words

Figure 6-2 PHT and PPD Sizes

6.3.3 PHT Hashing Algorithm.

When the PHT is searched due to a page exception, a hash technique is used. A number of the most significant bits in the virtual page number are used as the hash key. The hash key is shifted left by one to produce a PHT-entry index into the table. The entry's PHT1 Virtual Page Number is then checked against the original virtual address (if the PHT1 Valid Bit is set).

Hash collisions are resolved by adding a linear rehash constant to the original hash value, and wrapping around the front of the table if necessary. The number of bits used in the hash is proportional to the PHT size. A physical memory size of 8MB, for example, uses the top 13 bits of the virtual address and has a maximum of 8 collisions per hash value. A 32MB system uses 15 bits and has a maximum of 2 collisions per hash value. At 64MB and larger, all 16 bits of the virtual address are used, and the PHT is effectively just straight-indexed.

No effort is made to keep collision chain-link information in the PHT entries because of the high overhead this would require.

Instead, a simple count of the longest collision chain encountered so far is maintained. This count, in the A-memory/Lisp counter SYS:%PHT-SEARCH-DEPTH, is used and incremented by the microcode. Each PHT hash must examine at least SYS:%PHT-SEARCH-DEPTH number of entries before declaring a hash failure. When a new virtual address is added to the hash table, a count of valid entries skipped over is kept until a free PHT entry is found. Then if that count is larger than the SYS:%PHT-SEARCH-DEPTH count the latter is incremented. The length of the longest collision chain seen can decrease when PHT entries are deleted. The microcode, however, is not authorized to decrement SYS:%PHT-SEARCH-DEPTH. Instead, a low-priority PAGE-BACKGROUND process periodically scans the table; computes the current longest collision chain; and updates SYS:%PHT-SEARCH-DEPTH accordingly.

Initially the Page Hash Table contains no entries and the Physical Page Data table entries are coded FFFE (page available for use). All cells in the Page Hash Table are zeroed. As each is allocated on demand, a PHT entry is created and entered into the table.

6.4 DISK PAGE MAPPING SCHEME

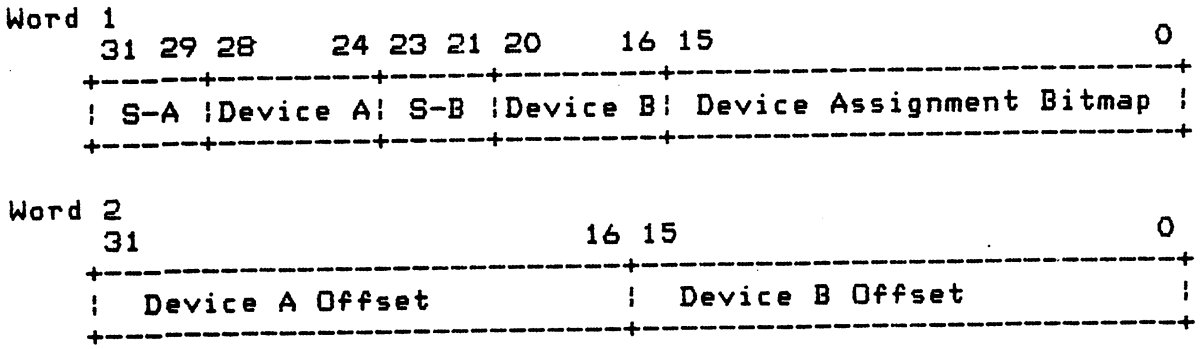
The following describes the mechanism by which a virtual page number is used to map to a disk address in one of the system's logical paging devices.

A logical page device in the Explorer system is a disk partition where disk storage is assigned to a number of virtual pages. The Lisp world LOD band is a read-only page device; that is, demand pages are read from this partition but if they are later dirtied they will be written out to a read/write page device (a PAGE band). New virtual pages created during program execution (when new Lisp objects are consed) will have read/write storage allocated to them when they are evicted from main memory.

Because of the large number of virtual pages (2**16) it is not practical use straight indexing (a one-to-one correspondence) between virtual pages and swap partition addresses. Instead, pages being swapped are be assigned the next available space in a swap partition and a disk page map is maintained for translating between virtual addresses and disk addresses. Each two-word entry in this Disk Page Map Table (DPMT) contains mapping information for a group of 16 contiguous virtual pages called a cluster; hence the DPMT is indexed by the top 12 bits of the virtual address, called the cluster number.

Through the DMPT, disk space is allocated in clusters of 32 blocks corresponding to 16 virtual memory pages (each disk block

is 1024 bytes or 256 words). The low 4 bits of the virtual page number select the page offset in the disk cluster. The format of a DPMT entry is shown in Figure 6-3.



Device Assignment Bitmap <0:15> = Indicates to which device (A or B) each of the 16 pages is assigned
0 = Device A (page band)
1 = Device B (load band)

Device B <16:20> = Logical page device number of device B

S-B <21:23> = Device B status

Device A <24:28> = Logical page device number of device A

S-A <29:31> = Device A status

Device B Offset <0:15> = 32-block offset from start of partition

Device A Offset <16:31> = 32-block offset from start of partition

Figure 6-3 Disk Page Mapping Table

*Status 0: No device assigned

*Status 1: Read only band (load band)

*Status 2: Read/Write band (page band)

*Status 3: Read/Write band assigned, however, a disk block has not yet been allocated

*Status 4: - 7: Unused

Figure 6-4 Device Status Codes

Each entry of the DPMT specifies two logical paging bands, Device A and Device B. By convention Device A describes a read/write device (swap band) while Device B describes a read-only device (the load band). A page represented by this cluster will have storage assigned on one of these two page devices; a bitmap in the entry specifies to which device each page in the cluster is mapped. The disk block corresponding to this virtual page on the page band not selected by the bit map is reserved but not used. If the entry in the bit map is switched this page would then be assigned to this other disk block.

A table is kept describing all the logical paging devices known to the virtual memory system. The Device A and Device B fields are conceptually an index into this table. Each PAGE partition on disk is represented as a separate logical page device In this way, there may be several paging bands on a single device or on several devices.

The fields S-A and S-B are the device assignment status fields for device A and device B respectively. They indicate whether the device is actually used by any pages in the cluster and if so, what kind of access is allowed. The values for these fields are described in Figure 6-4.

A virtual page operation would proceed as follows:

1. Using the most significant 12 bits of the virtual page number pick up the Disk Mapping Table Entry for the cluster.

2. Consulting the bit map decide whether this page is assigned to device A or device B of this cluster.

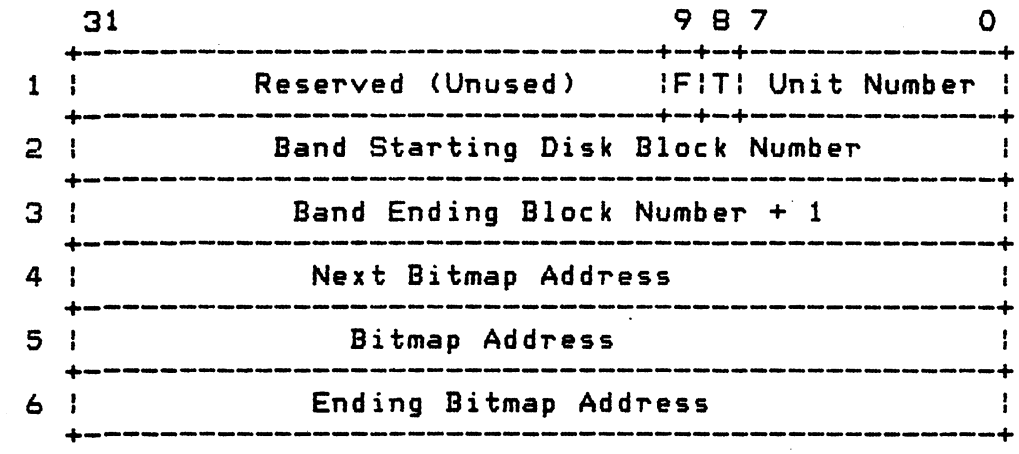
3. Using the device status field decide if a valid operation is being performed on this device. If no valid operation can be performed, call ILLOP (crash).

When the system is first booted, the DPMT indicates that all virtual pages reside in the read-only load band associated with Device B. As these pages are demanded they are read into memory from the Lisp load partition. They are not assigned swap space unless they are subsequently modified and swapped out. This avoids the necessity of copying the load band to the swap band before starting up Lisp.

Virtual pages that are as yet unused at boot time (unallocated virtual memory) are marked as unassigned in the DPMT. When a dirtied load band page or a freshly created virtual page is assigned swap space the logical paging device number of the swap partition is placed in the DPMT Device A field, the device A status is marked as read/write, and the appropriate Device A cluster offset is placed in word two of the DPMT entry. Unless this virtual page is later freed by garbage collection, it will be read from and written to this disk block from now on.

6.5 LOGICAL PAGING DEVICES

A logical paging device defines a set of contiguous blocks on a secondary storage device. The characteristics and allocation information for each paging device are maintained in a Logical Page Device Information Block structure (LPDIB). The LPDIB is described in Figure 6-5.



Word 1: Page device status information.
F <9> = Full bit
1 = all clusters allocated
T <8> = Device type
0 = read-only (load)
1 = read/write (page)
Unit <0:7> = Physical unit number of device
Word 2: Starting block number of the page band. This is an absolute block number on the device.
Word 3: The absolute block number of the first block that is outside of the band.
Word 4: Address of the next bitmap word that contains an unallocated cluster.
Word 5: Address of the first word of the cluster-allocation bitmap. Each bitmap bit represents one cluster (32 blocks) of storage.
Word 6: Address of last bitmap word.

Figure 6-5 Logical Page Device Information Block

The LPDIBs are allocated contiguously in the SYS:DEVICE-DESCRIPTOR-AREA beginning at the address stored in SYS:ADDRESS-OF-PAGE-DEVICE-TABLE. Enough space is allocated for a constant number of paging devices (the value of SYS:NUMBER-OF-PAGE-DEVICES). During boot the microcode initializes the LPDIB for the load band. Later, during Lisp boot (in SYS:LISP-REINITIALIZE), SYS:CONFIGURE-PAGE-BANDS is called to find all the Explorer PAGE bands in this configuration and allocate LPDIBs for each of them up to the system maximum.

Disk blocks are allocated to clusters by finding a free page cluster in the LPDIB bitmap. The bitmap has one bit for each page cluster. A 1 indicates the cluster is free and a 0

indicates it is in use. Word 4 of the page device information block points to the next bitmap word to check, and is incremented as all 32 clusters indicated by that word are allocated. When it gets to the last bitmap word, it starts again at the beginning. If there are no clusters left, then the Full Flag is set and the swap band is no longer checked until a cluster is returned. Garbage collection will return clusters as the virtual memory they represent is collected and freed for re-use.

6.6 VIRTUAL MEMORY SYSTEM SUBPRIMITIVES

The subprimitives and special variables described below affect the paging algorithms of the virtual memory system. As with all Explorer system subprimitives, some of these can be extremely dangerous (cause crashes or strange behavior) if misused.

Byte specifiers and constants described can be found in SYS:UCODE;LROY-QCOM. The Lisp-coded functions can be found in the MEMORY-MANAGMENT files listed below:

MEMORY-MANAGEMENT;	PAGE-DEFS
MEMORY-MANAGEMENT;	PAGE
MEMORY-MANAGEMENT;	PAGE-DEVICE
MEMORY-MANAGEMENT;	PAGING-PROCESS
MEMORY-MANAGEMENT;	PHYSICAL-MEMORY
MEMORY-MANAGEMENT;	VM-BOOT

The A-Memory counters (and some A-memory variables) used by the virtual memory system are documented in the section entitled Other Subprimitives, Variables, and Counters.

%disk-switches Variable

This variable contains bits that control various disk usage features. The byte specifiers listed below are stored in the DISK-SWITCHES-FIELDS list.

Bit 0 (%%Clean-Page-Search-Enable). Page replacement algorithm will scan through physical memory looking for a clean page to flush on a FINDCORE operation. Default is on.

Bit 1 (%%Time-Page-Faults-Enable). Enables %TOTAL-PAGE-FAULT-TIME in the counter block. Value of counter is microsecond time spent in the page fault microcode plus disk wait time, but excluding code that resolves page exceptions. Default is off.

Bit 2 (%%Multi-Page-Swapout-Enable). Enables the page

replacement algorithm to clean adjacent memory page images by writing them to disk in the same disk write for a page being flushed. Default is on. Turning it off will degrade paging performance.

Bit 7 (%%SB-During-Disk-Wait-Enable). Not used.

Bits <8:15> (%%Multi-Swapout-Page-Limit). Maximum number of pages that can be updated in a multi-swapout. Values must be between 0 and 255. Default is 128.

Bits <16:23> (%%Serial-Delay-Constant). Timing constant for microcode access to the serial chip registers. This must NOT be less than 12, which yields a delay of at least 2.641 microseconds on Explorer I. Don't change this unless you know what you're doing.

set-disk-switches
(&key clean-page-search time-page-faults multi-page-swapouts sequence-breaks-during-disk-wait multi-swapout-page-count-limit serial-delay-constant) SET-DISK-SWITCHES is a user interface to safely alter the dynamic paging variables using symbolic keyword definitions to specify the fields. The defaults for each switch are "safe" values. The value returned is the new value of %DISK-SWITCHES.

set-swapin-quantum-of-area (area &optional (swapin-quantum 3))
Specifies that pages of AREA (which should be an area-number) should be swapped in in groups of 2**SWAPIN-QUANTUM pages at a time. The default is 3, which means that prepaging (if active) will swap in up to 8 pages at a time.

The swapin quantum is used only if prepaging is enabled. Currently the Explorer II paging microcode uses the prepaging feature, but the Explorer I paging system does not.

set-all-swapin-quanta (&optional (swapin-quantum 3))
Specifies the SWAPIN-QUANTUMs of all non-fixed areas at once.

wire-page (address &optional (wire-p t))
If WIRE-P is T, the page containing ADDRESS is wired down; that is it cannot be paged-out. If WIRE-P is NIL, the page ceases to be wired down.

unwire-page (address)
(unwire-page address) is the same as (wire-page address nil).

page-in-structure (object)
Makes sure that the storage that represents OBJECT is in main memory. Any pages that have been swapped out to disk are read in. If OBJECT is large, this is useful in order to get all the paging required to bring OBJECT in over with at once, rather than having it occur a bit at a time as OBJECT is referenced.

The storage occupied by OBJECT is defined by the %FIND-STRUCTURE-LEADER and %STRUCTURE-TOTAL-SIZE subprimitives.

page-in-array (array &optional from to)
This is a version of PAGE-IN-STRUCTURE that can bring in a portion of an array. FROM and TO are lists of subscripts; if they are shorter than the dimensionality of ARRAY, the remaining subscripts are assumed to be zero.

page-in-pixel-array (array &optional from to)
Like PAGE-IN-ARRAY except that the lists FROM and TO, if present, are assumed to have their subscripts in the order horizontal, vertical, regardless of which of those two is actually the first axis of the ARRAY.

page-in-words (address n-words)
Any pages that have been swapped out to disk in the range of address space starting at ADDRESS and continuing for N-WORDS are read into main memory.

page-in-area (area-number)

page-in-region (region-number)
All swapped-out pages of the specified region or area are brought into main memory.

page-out-structure (object)

page-out-array (array &optional from to)

page-out-pixel-array (array &optional from to)

page-out-words (address n-words)

page-out-area (area-number

page-out-region (region-number)
These subprimitives exist only for compatibility with old code which may reference them. They currently do nothing and simply return NIL.

%page-status (virtual-address)
If the page containing virtual-address is swapped out, or if it is part of one of the low-numbered permanently-wired system areas, this returns NIL. Otherwise, it returns the entire first word of the Page Hash Table (PHT) entry for the page.

See the section on Paging and Disk Management for the format of a PHT entry. Byte specifiers for the PHT fields can be found in the PAGE-HASH-TABLE-FIELDS list.

%change-page-status
(virtual-address swap-status access-status-and-meta-bits)
The Page Hash Table (PHT) entry for the page containing VIRTUAL-ADDRESS is found and altered as specified. T is returned if it was found, NIL if it was not (presumably the page is swapped out). SWAP-STATUS and ACCESS-STATUS-AND-META-BITS can be NIL if those fields are not to be changed.

NOTE

This subprimitive is extremely dangerous since it does no error checking. The integrity of the virtual memory system can be irreparably damaged if this function is called improperly.

%compute-page-hash (virtual-address)

%compute-page-hash-lisp (va &optional max-byte-index max-byte-size)

%rehash (old-pht-index &optional max-index)
The first two return the hash value for VIRTUAL-ADDRESS (VA). The hash value is a byte offset into the Page Hash Table (located in physical memory) where the PHT entry for VIRTUAL-ADDRESS hashes to. However, this entry may already be in use. In that case, %REHASH may be used (given the old hash value) to find the next place to look.

%COMPUTE-PAGE-HASH-LISP and %REHASH are coded in Lisp. They take optional parameters which allow testing of the hash function on different sized hash tables. The defaults for the optional arguments are the values suitable for the whatever the running configuration is.

pages-of-physical-memory
Returns the total number of physical memory pages in the current memory configuration. Any number in the range [0 ..

(1- (PAGES-OF-PHYSICAL-MEMORY))] can be used as a valid page frame number (PFN) for functions which require them.

convert-physical-address-to-pfn (physical-address)

convert-pfn-to-physical-address (pfn)

convert-slot-offset-to-pfn (nubus-slot offset-into-slot)

convert-pfn-to-slot-offset (pfn)

convert-physical-page-to-pfn (phys-pg)

convert-pfn-to-physical-page (pfn)

These routines use the A-Memory Physical Memory Map to perform conversions between a logical page frame number (PFN) and a physical NuBus address. A PFN is simply a page number between zero and the number of physical pages available in the current configuration (which can be obtained by the PAGES-OF-PHYSICAL-MEMORY function). The physical address is a NuBus byte-oriented address which corresponds to PFN.

The manner in which the physical address is expressed depends on the function you use. CONVERT-PHYSICAL-ADDRESS-TO-PFN and CONVERT-PFN-TO-PHYSICAL-ADDRESS use 32-bit NuBus addresses. The next two use the slot/byte-offset into slot form of expressing a NuBus address (two values are returned from CONVERT-PFN-TO-SLOT-OFFSET). The ones with PHYSICAL-PAGE in their names use just the top 21 bits of the NuBus address.

%delete-physical-page (pfn)

This is used to delete pfn (which is a logical page number of physical memory) from the virtual memory pool. Any virtual page that is currently in pfn will be swapped out (if necessary) and the page will be marked so that the virtual memory system will not use it in the future for holding virtual pages. This is useful for reserving physical memory for use, say, as an I/O buffer for a device which does DMA I/O. It can also be used to force a virtual page to be swapped out of physical memory.

Returns T if page was deleted successfully. Returns NIL if page was already deleted.

%create-physical-page (pfn)

This reverses the action of %delete-physical page. That is, given a deleted pfn, %create-physical-page marks it so that it can be used in the by the virtual memory system to hold a virtual page.

set-memory-size (new-size-in-pages)
Specifies the size of physical memory available to the virtual memory system to NEW-SIZE-IN-PAGES. Can be used to decrease the number of physical pages available to virtual pages, or to increase it back to the system maximum if SET-MEMORY-SIZE has previously been used to lower it. This may be useful for measuring performance based on the amount of memory.

To determine the actual number of physical pages in the current configuration, use the PAGES-OF-PHYSICAL-MEMORY function. The variable %WORKING-MEMORY-SIZE contains the NEW-SIZE-IN-PAGES that you set using SET-MEMORY-SIZE.

SET-MEMORY-SIZE uses %DELETE-PHYSICAL-PAGE and %CREATE-PHYSICAL-PAGE.

get-contiguous-physical-pages (number-of-pages)

return-contiguous-physical-pages (number-of-pages slot offset)
GET-CONTIGUOUS-PHYSICAL-PAGES may be used to obtain a block of physically contiguous memory (and always on the same memory board). NUMBER-OF-PAGES is the amount desired. A physical page is 2048 bytes (the value of the variable PAGE-SIZE-IN-BYTES). The pages thus obtained will not be available to the virtual memory system until returned by the RETURN-CONTIGUOUS-PHYSICAL-PAGES function.

If NUMBER-OF-PAGES contiguous pages cannot currently be freed for use, NILs are returned. Otherwise, two values are returned: a NuBus slot number and the byte offset in the slot which together specify the 32-bit NuBus address of the start of the physical memory block obtained.

%physical-address (virtual-address)
Returns the NuBus physical address which VIRTUAL-ADDRESS currently occupies in main memory. This value is unpredictable if the virtual page is not swapped in; therefore, this function should be used only on wired pages, or you should do

```
(WITHOUT-INTERRUPTS
   (%P-POINTER virtual-address)    ;swap it in
   (%PHYSICAL-ADDRESS virtual-address))
```

Unless you assure that the page is wired, or use the physical address returned in a section of code that is guaranteed not to change the contents of physical memory, the value returned may not be meaningful for long.

%virtual-page-number (pfn)
Given a logical page frame number (PFN), returns the virtual

page number currently in the physical page, or NIL if there is none currently. The virtual page number returned can change unpredictably unless the page is wired down, or unless you assure that no interrupts (or consing of any kind, which can cause a swapout) occurs in the section of code using the value.

%page-frame-number (va)
Given a virtual address (VA), returns the logical page frame number (PFN) it currently resides in, or NIL if none. Use of this value is subject to the same restrictions as for the two preceding functions.

%add-page-device (real-unit start-block band-size)
This function is used by the CONFIGURE-PAGE-BANDS function to add a PAGE band to the virtual memory system. REAL-UNIT is the physical unit where the PAGE band resides. START-BLOCK is the PAGE band's first block number, and BAND-SIZE is its total size in blocks.

%findcore
Frees a page of physical memory (removes it from paging) and returns its logical page frame number (PFN).

%page-in (pfn vpn)
Creates a Page Hash Table (PHT) entry that indicates that virtual page number VPN is located in logical page frame number PFN. This had better be true or you'll be in trouble.

%page-trace
No longer implemented. Signals an illegal instruction error.

deallocate-swap-space (region)
Called by the garbage collector to free up any swap space on PAGE bands that is used by the virtual memory in REGION. REGION is an Oldspace region which the garbage collector has finished collecting, so that now its virtual memory (and any associated swap space) can be freed for later re-use.

%return-page-cluster (swap-device-number cluster-offset)
This is the routine called by DEALLOCATE-SWAP-SPACE to do the real work. SWAP-DEVICE-NUMBER is the logical page device number of a PAGE band and CLUSTER-OFFSET specifies the cluster number in that paging device to be freed.

%disk-address (va)
Used to find the disk address, if any associated with virtual address VA. This address may be on the LOD band or on any of the PAGE bands in the current configuration. Returns three values, if VA is a valid virtual memory address: the absolute disk block address of the page

containing VA, the physical disk unit, and the DPMT device status code for the page. See the constants in the %DPMTE-DEVICE-OFFSETS-FIELDS list for interpretation of this third value.