SECTION 5

The Virtual Memory System

5.1 VIRTUAL MEMORY

Virtual Memory is the simulation of a large fast primary memory by the use of a fast but smaller primary memory and a large but slow secondary memory. Blocks, called pages, are moved between primary and secondary memory according to a page management strategy.

A page management strategy that moves a page into primary memory when it is referenced but not present in primary memory (a page fault) is termed demand paging. Usually, a page being moved into primary memory displaces some other page. The choice of the page to remove is made by applying the page replacement policy. If the page chosen for replacement has been altered while in primary memory, it must be written to secondary memory before it can be replaced. A page in primary store that has been altered is called a dirty page.

Some pages are exempted from paging. These are termed wired pages. Wired pages are used for interrupt handler buffers because interrupts cannot take a page fault; for pages containing paging tables on which a page fault cannot be allowed; for pages involved in DMA transfers; and other pages containing critical data which must be accessed without a page fault or for which the performance penalty for taking a page fault is too great.

In the Explorer, semiconductor memory is used as primary memory and disk is used as secondary memory. Pages are moved into primary memory when referenced and not present -- demand paging. Every attempt is made to replace a page which is not dirty so that a write to secondary memory is not needed.

A page exception is said to occur when for some reason the virtual to physical address mapping could not be completed by the mapping hardware without microcode support. There are many reasons for this; only one of those reasons requires access to secondary storage. If a page is referenced and this reference cannot be completed without operations with secondary storage, then a page fault has occurred. This distinction is made so that page exception rates and page fault rates can be distinguished.

Virtual memory references are processed below everything except interrupts in the Explorer hierarchy (they are depended on by everything except interrupts). A virtual memory reference is considered an atomic operation on the Explorer system; that is, the time spent waiting for a virtual memory reference to complete (including time spent waiting for disk paging activity) is not available for use by any other part of the system except interrupt processing. This policy greatly simplifies the implementation of various low-level Explorer subsystems.

5.2 PHYSICAL ADDRESSES

Memory is accessed by use of a physical address, a system-wide name for some storage. The Explorer is based on the NuBus, a 32-bit, high-speed bus. All NuBus addresses are byte addresses with words aligned so that the low order 2 bits are zero.

Explorer physical memory, monitor screen memory and I/O peripheral control registers all reside within the same 32-bit address space. Not all bus addresses will be accessible directly mapped in the Lisp virtual address space. This is true because the 25-bit Lisp virtual address is smaller than the 32-bit NuBus address space. Data can be read and written into unmapped physical addresses using special physical I/O functions from Lisp.

5.3 VIRTUAL ADDRESSES

An address in the Explorer system is the size of a pointer field, which is 25 bits. The virtual address is divided into a virtual page number and a page offset. The virtual page number is the high order 16 bits of the virtual address, and page offset is the low order 9 bits of the virtual address. Thus, each virtual page contains 512 words (or 2048 bytes) of storage.

The Explorer I and Explorer II both have memory mapping hardware based on a map page size of 256 word (or 1024 bytes). Therefore, there are two hardware map pages for each virtual page. When referring to these maps, the map page number is based on the high order 17 bits of the virtual address and will be called the virtual page map number. The page offset is based on the low order 8 bits and will be called the page map offset.

The Explorer has a simple memory map. I/O is not, by default, part of virtual memory, but instead is accessed by special physical I/O operations. The A-memory (the processor memory for data accessed by the microcode) has a dedicated virtual address, which is at the very top of the virtual address space, but consumes none of the physical address space.

In order to translate from a virtual to a physical address, the virtual page map number is looked up in the hardware page maps to produces a 22-bit page frame number. The page frame number is concatenated with the 10-bit page map offset to make a 32-bit physical address. This is used to address the primary memory over the system (or other) bus.

The map also produces other outputs for use by the processor: 2 access bits, 2 status bits, 6 meta bits and 2 garbage collector volatility bits. These outputs are used by the microcode to help manage page aging, storage allocation attributes (on a per-region basis), garbage collection, and other functions.

The Explorer microprocessor has a standard NuBus interface. In addition, The Explorer I microprocessor has a special bus to "local memory". This local memory also exists in the NuBus address space but the special high-speed dedicated bus to this memory reduces the NuBus traffic.

5.4 PHYSICAL MEMORY USE

The physical memory present in the machine can be divided into two categories according to its usage. A portion of memory is set aside for use by the paging system itself and by microcode and Lisp device handlers that perform physical addressing. Data in this space is said to reside in physical memory which is not accessible in the virtual memory address space. Such physical memory is termed the permanently wired pages. The rest of the local physical memory is used as a transient page area, i.e., the virtual memory system assigns the pages of the virtual memory to physical locations as a part of its management functions.

5.5 VIRTUAL MEMORY PARTITIONING

To avoid the need for a very large mapping memory, or an associative memory, both the Explorer I and Explorer II use two-level memory maps for the virtual-to-physical address translation. Several sections following this one describe the hardware maps in detail. This section, however, is meant to provide an overview and motivation for the map functions.

The first level map (called the Level-1 map on the Explorer I and the Address Space Map on the Explorer II) effectively partitions the virtual address space into segments of 8K words (16 pages). They both have 4096 entries and are indexed by the top 12 bits of the virtual address. Note that there are two such entries per 32-page address space quantum. The 32-page address space quantum is the unit in which address space is allocated to regions; that is, a region must be at least 1 address space quantum long, and must have a length evenly divisible by the address space quantum.

On both Explorer I and Explorer II, a number of second level maps are allocated to the first level maps. The number of second level maps differs; but in both cases it is much less than the amount required to map in the entire 25-bit address space. If the address is not currently being mapped by the mapping hardware, the microcode consults the Page Hash Table (PHT, described later) which describes all pages in physical memory.

In general terms, the first level maps can be thought of as a way of segmenting the address space into blocks of contiguous virtual pages with identical attributes in order to aid storage allocation and garbage collection functions. The second level maps can be thought of as a cache describing the most recently accessed pages. Except for special system pages, the storage-management and garbage-collection-specific information in the memory maps is initially set up from the REGION BITS associated with a virtual address. They may later be altered by the garbage collector.

5.6 EXPLORER I MAPPING HARDWARE

The Explorer I Level-1 Map (Figure 5-1) is 4096 words long and is indexed by the top 12 bits of the virtual address. The first level map, if valid, produces a 7-bit index into the second level map along with several status bits.

```
                      15 14 13 12 11 10 9  7 6         0
+--------------------+--+--+--+--+--+--+----+-------------+
|    Unused          | M| F| W| A| V| O| GCV| LVL2 INDEX|
+--------------------+--+--+--+--+--+--+----+-------------+
```

M <15> = last access mapped (not physical)
F <14> = last access forced
W <13> = last access write fault
A <12> = last access access fault
V <11> = map entry valid
O <10> = oldspace meta bit; 0 = oldspace
GCV <9> = Garbage Collector volatility bits
LVL2 INDEX <6:0> = Level-2 block number

Figure 5-1 Explorer I Level-1 Map (LVL1)

The second level map (Figure 5-2) consists of 128 blocks of 32-registers each. Each set of two registers represents one map-page entry. After using the most significant 12 bits of the virtual page map number to index into LVL1, the remaining 5 bits of the virtual page map number select a register within the LVL2 block. If both the LVL1 and LVL2 entries are valid, and if all other status indications are positive, the selected LVL2 map register produces the 22-bit physical page frame address. Note that since the LVL2 map blocks are indexed by the least significant 5 bits of the 17-bit virtual page map address, each of the 128 blocks represents 16 contiguous virtual pages.

Memory Map Level 2 Control Bits

```
 31           16  15  14 13 12 11 10 9 8  6 5  4 3   0
+-------------+----+----+--+------+--+-+----+----+----+
|   Unused    |TM1 |TMO | L| GCV  | F|A|STAT| REP| SP |
+-------------+----+----+--+------+--+-+----+----+----+
```

TM1 <15> = last access TM1
TMO <14> = last access TMO
L <13> = last access locked
GCV <12:11> = Garbage Collector volatility bits
F <10> = force allowed
A <9> = access bit; 0 = no access
STAT <8:6> = status bits (see Figure 5-4)
REP <5:4> = representation type:
  0 = list
  1 = structure
  2,3 unused

Memory Map Level 2 Address Bits

```
 31              22 21                               0
+-----------------+----------------------------------+
|   Unused (0)    |       Physical Page Number       |
+-----------------+----------------------------------+
```

Figure 5-2 Explorer I Level-2 Maps (LVL2)

The second level map is read in as two separate functional sources since the data field is greater than 32 bits. The field is separated into Memory-Map-Level-2-Control-Bits and Memory-Map-Level-2-Address-Bits.

To handle a page exception a check must first be made to see if the LVL1 entry is valid. A bit in the Memory-Map-Level-1 register indicates this. If the first level map entry is not valid, a block of second level map must be allocated and initialized with "map not valid" entries. The first level map must be set up to point to it. From here the page exception is handled as a second level map miss (described later).

If the first level map entry is valid then the data in the second level memory map control register determines the action to be taken. The map status code is used to determine the processing case.

5.7 EXPLORER I REVERSE FIRST LEVEL MAP

For each block of 128 Second Level Map registers, there is an entry in the Reverse First Level Map which gives the number of the First Level Map entry which points to this block, or else indicates that this block is unused. It contains a value which, if placed in the Virtual Memory Address register (VMA), would address that first level map entry, or else it contains -1 to indicate that this block is not currently pointed to. The Reverse First Level Map is held in A-memory and is 128 words long. It is used when allocating map level-2 blocks.

5.8 EXPLORER I MEMORY MAP ALLOCATION

A simple clock scheme is used for allocation of second level memory map blocks. If a level-1 map fault occurs, the Reverse First Level Map is consulted to see if the level-2 map block is owned by this level-1 block. If so, the valid (V) bit is set and the memory reference is restarted. If not, a new level-2 map block must be allocated to this level-1 entry.

To find a level-2 map block to allocate, the Reverse First Level Map is scanned. If the value is -1 then this level-2 map block is free and can be allocated. If the value is not -1 then it contains the address of the map level-1 entry which owns this level-2 map block. If the level-1 valid (V) bit of this entry is not set then this block has not been used recently and is allocated to the new entry. If the valid (V) bit is set, then this entry is in use. The level one entry is aged by turning off the valid bit. The scan continues at the next Reverse First Level Map entry. The scan will wrap around if the end of the table is encountered. Since the aging is done during the scan, if the entire structure is scanned and no level-2 map block is found, then the scan merely continues and will choose the next entry since it was aged during the last scan.

The map level-2 block is allocated by setting the index in the map level-1 entry and updating the Reverse First Level Map to reflect the new allocation.

5.9 EXPLORER II MEMORY MAP HARDWARE

The Explorer II map hardware consists of the Address Space Map (ASM), the Virtual Memory Maps Status (VMM Status), and the Virtual Memory Maps (VMM). The Address Space Map and the VMM Status replace the level-1 maps on the Explorer I and the VMM replaces both level-2 maps.

The Address Space Map (see Table 5-1) contains garbage collection oldspace and volatility information. There are 4K entries in this table, one entry for every 16 pages or 8K words, which is equivalent to two entries for every address space quantum (16K words; the smallest region size). This table is addressed by the high order 12 bits of the 25 bit virtual page address.

Table 5-1 Address Space Map Bits Assignment

```
  7  6    5  4    3     0
+-----+-----+----+-------+
|GCVS| GCV | OS | Meta  |
+-----+-----+----+-------+
```

GCVS <7> : Garbage Collection volatility valid status, 0 = true
GCV <6:5> : Volatility level for this address space half quantum
OS <4> : Old Space bit, 0 = Oldspace
Meta <3:0> : Address space Map Meta bits

The Virtual Memory Map Status contains two entries for each virtual page and therefore contains 128K entries. Each table entry consists of two bits (see Table 5-2), which indicates whether the VMM table has a valid entry. It also indicates in which VMM bank, right or left, the valid entry can be found, if one exists.

Table 5-2 Virtual Memory Maps Status

```
  1   0
+---+---+
| B | V |
+---+---+
```

V <0> : Valid bit, 1 if map is set up
B <1> : Bank select, 1 if right bank, 0 if left bank

The Virtual Memory Map (VMM) consists of two banks. Each bank has 16K 32 bit maps. Each entry has the 22 bit physical frame address needed by the hardware to map from virtual memory to physical memory. The maps also contain two access bits that control the type of access to the maps. Bit 31 is the read/write access bit, and will cause a page exception if not set on any mapped cycle. Bit 30 is the write access bit, which is also the top bit of the 3 bits status field (see Table 5-3), will cause a page exception on mapped write cycles if not set.

When a mapped cycle is done, the mapping hardware will first check the the Virtual Memory Map Status Ram by indexing into it with bits 8 though 24 of the virtual address used. If the valid bit is not set, then a map miss page exception occurs. If the valid bit is set, then the hardware checks the bank select bit in the Virtual Memory Map Status to see which VMM bank it should use, the right or left. It then gets the entry from the correct bank by indexing into with bits 8 through 21 of the virtual address. If the read/write bit is set in the VMM and the cycle is a read or if both the read/write access and write access is set and the cycle is a write, then it can complete the cycle by using the physical address. If the access bits are not set correctly then a page exception occurs. The 32-bit physical address is constructed from the top 22-bit physical page number along with the 8-bit map page offset from bits 0 through 7 of the virtual page address, with byte offset, the last two digits, being 0.

Since the VMM banks are indexed by bits 8 through 21 of the virtual address, bits 22 through 24 indicate the 8 different pages that will index to the same VMM index. Since there are two banks, 2 of these 8 pages can be set up at the same time.

If the cache inhibit bit is not set, the hardware will use the cache if it can, and do cache fills as needed.

Table 5-3 Virtual Memory Map

```
31  30    28 27 26 25 24 23 22 21                                       0
+--+--------+-----+--+-----+--+-----------------------------------------+
!A !STATUS  ! REP !O ! GCV !CI!       PHYSICAL PAGE NUMBER              !
+--+--------+-----+--+-----+--+-----------------------------------------+
```

A	<31>	: Access bit, 0 is no access
Status	<30:28>	: Status (see Figure 5-4)
Rep	<27:26>	: Region representation, 0 is list space, 1 is structure space, 2 and 3 are unused
O	<25>	: Oldspace bit, 0 = oldspace
GCV	<24:23>	: Garbage Collection Volatility bits
CI	<22>	: Cache Inhibit, 1 = inhibit
Physical Page Number	<21:0>	: The top 22 bits of the 32-bit physical page address.

5.9.1 Explorer II Map Usage Table (MUT).

The Explorer II Map Usage Table (MUT) resides in physical memory and contains 8192 entries (see Table 5-4). Since there are two entries per 32 bit word, it occupies 4096 words of physical memory. Since one 2K page uses two maps, each entry has information for two left and two right VMM maps. For each pair of maps, it shows which of the 8 possible virtual memory pages is using the left bank and which is using the right bank. It also shows which bank to use next when one of the remaining 6 pages is accessed and therefore needs the maps set up.

Table 5-4 Map Usage Table (MUT)

```
 15 14 13    8 7     2  1  0
+-----+-------+-------+-----+
|     | left  | right | Next|
+-----+-------+-------+-----+
```

Next <0:1> : 00 --> wired page, always in left bank
01 --> use right bank next
10 --> use left bank next
11 --> unused

Right <2:7> : The bottom 3 bits of this field indicate the top 3 bits of the page that currently has the map in the right bank. If the field is -1, then it is currently unused.

Left <8:13> : The bottom 3 bits of this field indicate the top 3 bits of the page that currently has the map in the left bank. If the field is -1, then it is currently unused.

<15:14> : Unused.

5.10 EXPLORER II MEMORY MAP ALLOCATION

Explorer II map allocation is much easier then the Explorer I. When a map exception occurs, bits 9 through 21 are used to index into the MUT (actually, bits 10 through 21 are used to get the correct word and bit 9 is used to select the halfword). The NEXT field is accessed to see if the new map should be set up in the right or left VMM bank. If the left bank has a wired page, the right bank is always used. The appropriate RIGHT or LEFT field from the MUT entry is accessed to see who currently owns the map. If it is -1, then there is no current owner. The current user virtual address is calculated from the bottom 3 bits of the LEFT or RIGHT field from the MUT along with bits 9 through 21 of the current virtual address (i.e., the one that needs the maps set up). This is used to invalidate the VMM Status Table for the two maps that currently are using the bank needed. The MUT is changed to reflect the new owner, the NEXT field is updated to use the other bank next time (unless it has wired status) and the table entry is rewritten. The VMM data in the correct bank is set up, and the VMM Status is set to valid with the correct bank select. The memory cycle that caused the page exception is then restarted.

5.11 PAGE EXCEPTION HANDLING

A memory reference which causes the maps to be altered, but can then complete produces a <u>page exception</u>. This includes references which must go to the PHT to reset the maps. A reference which cannot complete without access to disk (no valid information even in the PHT) is said to cause a <u>page fault</u>. This section describes page exception handling.

A page exception can be generated by the memory mapping hardware for a number of different reasons during the 2-level map lookup. A level-1 miss can occur if the valid bit is not set in the level-1 map. A level-2 miss occurs when a non-normal map status is found in the second level map. Examples page exceptions include inappropriate read/write access privileges; page is in the processor's PDL buffer or A-memory; page is marked as "trap on any access" (the "MAR" break feature) and others. Depending on the circumstances, some of these conditions may eventually be signalled as traps by the microcode, or the reference may continue and complete normally. The specific exception handling is based on the Memory Access and Memory Status fields and is described in detail below.

5.11.1 Memory Access Codes.

Memory Access codes are listed in Figure 5-3. Access codes are determined by combining the access bit and the most significant bit of the status bits field. These codes specify what memory operations (read, read/write or none) are to be permitted to these pages. If the desired access is permitted the memory system performs the operation. If the desired access is inappropriate, the page fault condition is set and the page exception handler is invoked. The exception handler then uses the status code to determine what steps to take next.

Access Bit	MS Status Bit	Code	Access-Type Meaning
0	x	0, 1	No Access
1	0	2	Read Only
1	1	3	Read/Write

Figure 5-3 Memory Map Access Codes

5.11.2 Memory Map Status Codes.

In this section each memory map status code is examined in detail. The possible map status values and their interpretations location information. In other words, there is first-level map information available, but no physical mapping information. This type of map entry is created when a pointer to an object is used but the object itself is not referenced. The oldspace and region representation type bits in such a map entry are needed by the garbage collector. An attempt to access the storage associated with the object later will be treated like a map miss.

5.11.2.3 Status Code 2: Read Only.

An attempt was made to write to a page that is set to read-only will cause a page exception. A special case is made for a forced write operation. In this case, the write occurs and no access fault is declared. This is needed so that the garbage collector/compactor can move data structures that ordinarily need protection.

If the operation is a regular, non-garbage collection write and the special A-memory/Lisp variable SYS:%INHIBIT-READ-ONLY is NIL, then the operation is declared illegal and an error is signaled. If SYS:%INHIBIT-READ-ONLY is non-NIL, the access is allowed.

5.11.2.4 Status Code 3: Read/Write First.

A page exception occurs on an attempt to write. The processing for this exception consists of changing the status in the map and the page hash table to read/write, indicating the contents of the page have been modified. The reference is restarted. This facility implements the dirty page status.

If the page that is being set to dirty is currently assigned to a read-only page band, then it will be reassigned to a read/write page band (given swap space) later when the page needs to be swapped out of physical memory.

5.11.2.5 Status Code 4: Read/Write.

A normal, fully mapped page. No exception should occur on this type of page. If this status occurs, the hardware is faulty, and a crash sequence will be initiated.

5.11.2.6 Status Code 5: Page might be in PDL Buffer.

Certain areas which contain regular PDLs arrange to get the map set to this status for their pages (instead of read/write). The microcode has to decide, on every reference, whether the page is in the processor's PDL buffer registers or in main memory, and simulate the appropriate operation. It may be that only part, or none, of the page is in the PDL buffer on a particular reference. Thus the page exception handler must test the virtual address to see if it falls in the range which is really in the PDL buffer right now. If not, temporarily turn on read/write access, make the reference, and turn it off again. Pages may be swapped out without regard for whether or not they are in the PDL. This works because the normal course of swapping out invalidates the second-level map. If the page is then referenced as memory, it will be swapped in normally and its map status restored from the Region-Bits table, in the normal fashion. This will then restore the Maybe-PDL map status. Otherwise, the addressed word is in the PDL buffer. Translate the virtual address to a PDL buffer address and return the appropriate register contents.

5.11.2.7 Status Code 6: Possible MAR Trap.

The memory address register (MAR) facility allows any word or contiguous set of words to be monitored constantly, and cause a trap if the words are referenced in a specified manner. The name MAR is from the similar device on the ITS PDP-10's. The MAR trap status is set for all pages that are in the range of addresses being monitored. This range is indicated in the A-memory/Lisp variables SYS:%MAR-LOW and SYS:%MAR-HIGH. When the MAR trap occurs, the virtual address is checked to see if it falls in the range. If so, a trap may be called. It should be noted that traps, since they cause sequence breaks, are not allowed during stack group switches, so if a MAR-monitored address is referenced, a sequence break flag is set, and the break will occur at the next appropriate time.

If an address falls within the MAR'd range, then the action taken depends on the currently active MAR-mode (MAR-mode is actually a 2-bit field in each stack group's mode flags). The action taken for various flag word values is shown in Table 5-5.

Table 5-5 MAR Status Codes

Value	Memory Operation	MAR Mode	Action
0	Read	MAR disabled	No trap
1	Read	Read Trap	Trap
2	Read	Write Trap	No trap
3	Read	Read/Write Trap	Trap
4	Write	MAR disabled	No trap
5	Write	Read Trap	No trap
6	Write	Write Trap	Trap
7	Write	Read/Write Trap	Trap

5.11.3 GCV and Oldspace Exceptions.

In addition to access/status fault exception handling, exception handling is also performed for two garbage-collection- related conditions: Garbage Collection Volatility Fault and Oldspace. These two are described briefly below. More information on them is presented in Section 9 (Garbage Collection). These exceptions are different from access/status exceptions in that the mapping hardware does not signal the page fault condition for them. Rather, the reference produces GCV and Oldspace conditions that the microcode can then selectively dispatch upon.

5.11.3.1 Oldspace.

The Oldspace condition is indicated by the Oldspace Meta Bit in the first-level map being 0. It indicates that objects in these pages are currently being garbage collected. The Oldspace Meta Bit can be ORed selectively into a microcode dispatch. The TRANSPORT dispatches use this bit. The TRANSPORTER is invoked, which decides if this is an object that needs to be moved for garbage collection. For more information on the TRANSPORTER, see Section 9.

5.11.3.2 Garbage Collection Volatility Faults.

The Garbage Collection Volatility Fault condition is also a 1-bit field. It is signaled whenever a younger object (contained in MD, memory data register) is written into older memory (represented by VMA, the memory address register). The GCV output is determined by a comparator which examines the 3-bit GCV field in the first-level map and the 2-bit GCV field in the second-level map. The details of the fault inputs and output, along with their interpretations in the Temporal Garbage Collection implementation, are presented in Section 9.

On the Explorer I, the GCV fault is stored in an inaccessible 1-bit GCV register, and is preserved only until the next memory reference is performed. Thus, it can only be sensed by a dispatch using GCV done immediately after the memory cycle. On the Explorer II, the GCV register can be read and written as a functional source and destination, hence can be stored across other memory references.

5.12 VIRTUAL MEMORY SYSTEM TABLES

There are some additional tables associated with paging that are used by the microcode: the <u>Page Hash Table</u> (PHT) contains an entry for every virtual page that is currently memory resident. The <u>physical page data table</u> (PPD) contains an entry for every physical page of memory. These tables are described in further detail below.

5.12.1 Page Hash Table.

The page hash table (PHT) resides in physical memory and is not part of the virtual address space. It describes every virtual page that is currently resident in physical memory. When the microcode detects a map miss, the PHT is consulted to see if the virtual page is still in physical memory. If so, the Level 2 maps (Explorer I) or VMM hardware (Explorer II) can be set up from the information stored in the PHT, and the memory reference will be restarted. Since the mapping hardware now contains valid mapping information, the reference should complete normally. If there is no valid PHT entry for a virtual address, a page fault is said to have occurred, and the page must be read in from disk.

The page hash table is also used to store information used by the page aging and swap management functions. A full description of the role of the page hash table in these contexts is described in a later section.

5.12.2 Physical Page Data Table.

The Physical Page Data Table (PPD) resides in physical memory and is not part of virtual address space. When the system is booted, the system determines the size of primary memory and allocates a suitable portion of physical memory for this table.

An entry for a page in the Physical Page Data Table is shown in Table 5-6. There is one PPD entry for every 512-word page of physical memory. A PPD entry thus represents a logical page frame or page frame number (PFN) in the physical memory pool. Each entry is broken into two parts. The low order 16 bits of the word contain an index into the Page Hash Table or a page allocation status. The high order 16 bits of the word contain an index into the Physical Page Data Table linking this page to the next most recently used physical memory page.

All the physical pages that are allocated to hold the microcode management tables are marked #x+FFFFFFFF, to indicate that these pages are not to be used in the virtual memory page pool.

The Physical Page Data Table is used to determine which virtual page is contained in a given physical page. The microcode page aging and replacement algorithms are driven by a scan of the Physical Page Data Table.

Table 5-6 Physical Page Data Area Word Format

Value	Meaning
PHT INDEX FIELD:	
FFFF (hex)	System wired or permanently wired page. Page is not available in virtual memory pool.
FFFE	Free page. Page is available in the virtual memory pool but is not currently in use.
PHT Index	Normal page. Value contains the index of the page hash table entry for this page.
PPD LINK FIELD:	
FFFF (hex)	Not available. Page is not available in virtual memory pool.
FFFE	Write in progress. Page is in virtual memory pool, but is currently being written to disk.
FFFD	End of LRU list. Page is the most recently referenced page in virtual memory pool.
PPD Link	Normal page. Value is index of the physical page table entry of the next most recently referenced page in the virtual memory pool.

Given a page frame number, information about the virtual page associated with the physical page can be determined as follows. Use the PFN as a word index into the PPD. If the PPD index field is valid (greater than FFFE), then there is a virtual page in this page frame. Use the index field shifted left by 1 as a word index into the PHT.

5.13 A-MEMORY PAGING INFORMATION

In addition to the Explorer I Reverse-First-Level-Map, other virtual-memory-specific information is kept in portions of A-memory, as described below.

5.13.1 PDL Buffer Handling.

A-memory also contains the first virtual address which currently resides in the PDL buffer (in A-PDL-BUFFER-VIRTUAL-ADDRESS), and the PDL buffer index corresponding to that address (in A-PDL-BUFFER-HEAD). Note that the valid portion of the PDL buffer can wrap around as shown in Figure 5-5.

When a page exception is taken for a page that might be in the PDL Buffer, the microcode handler must check to see if the virtual memory address falls in the valid portion of the PDL Buffer. The further processing of this case is explained under the map level 2 status code for PDL Buffer (status code 5).

```
 0                    PDL Buffer                     1023
+-------------+----------------------+--------------+
!             !                      !              !
!  valid     top      invalid       base    valid   !
!             !                      !              !
+-------------+----------------------+--------------+
```

Figure 5-5 PDL Buffer Wrap-Around

5.13.2 Physical Memory Map.

The Physical Memory Map (PMM) is a set of A-memory locations which describe which logical page frame numbers (PFNs) reside at which physical memory (NuBus) board addresses. Each entry in the PMM describes one or more 2MB quantums of physical memory. The number of 2MB quantums is kept in the low 10 bits, while the top 22 bits give the (NuBus) physical page frame number where the first quantum starts (see Figure 5-6).

```
 31                                  10 9             0
+------------------------------------+---------------+
! physical page frame number         ! # of quanta   !
+------------------------------------+---------------+
```

Physical Memory Map Entry

Figure 5-6 Physical Memory Map Entry