

**NAME**

**vm\_page\_bits**, **vm\_page\_set\_validclean**, **vm\_page\_clear\_dirty**, **vm\_page\_set\_invalid**,  
**vm\_page\_zero\_invalid**, **vm\_page\_is\_valid**, **vm\_page\_test\_dirty**, **vm\_page\_dirty**, **vm\_page\_undirty** -  
manage page clean and dirty bits

**SYNOPSIS**

```
#include <sys/param.h>
```

```
#include <vm/vm.h>
```

```
#include <vm/vm_page.h>
```

*int*

```
vm_page_bits(int base, int size);
```

*void*

```
vm_page_set_validclean(vm_page_t m, int base, int size);
```

*void*

```
vm_page_clear_dirty(vm_page_t m, int base, int size);
```

*void*

```
vm_page_set_invalid(vm_page_t m, int base, int size);
```

*void*

```
vm_page_zero_invalid(vm_page_t m, boolean_t setvalid);
```

*int*

```
vm_page_is_valid(vm_page_t m, int base, int size);
```

*void*

```
vm_page_test_dirty(vm_page_t m);
```

*void*

```
vm_page_dirty(vm_page_t m);
```

*void*

```
vm_page_undirty(vm_page_t m);
```

**DESCRIPTION**

**vm\_page\_bits()** calculates the bits representing the DEV\_BSIZE range of bytes between *base* and *size*. The byte range is expected to be within a single page, and if *size* is zero, no bits will be set.

**vm\_page\_set\_validclean()** flags the byte range between *base* and *size* as valid and clean. The range is expected to be DEV\_BSIZE aligned and no larger than PAGE\_SIZE. If it is not properly aligned, any unaligned chunks of the DEV\_BSIZE blocks at the beginning and end of the range will be zeroed.

If *base* is zero and *size* is one page, the modified bit in the page map is cleared; as well, the VPO\_NOSYNC flag is cleared.

**vm\_page\_clear\_dirty()** clears the dirty bits within a page in the range between *base* and *size*. The bits representing the range are calculated by calling **vm\_page\_bits()**.

**vm\_page\_set\_invalid()** clears the bits in both the valid and dirty flags representing the DEV\_BSIZE blocks between *base* and *size* in the page. The bits are calculated by calling **vm\_page\_bits()**. As well as clearing the bits within the page, the generation number within the object holding the page is incremented.

**vm\_page\_zero\_invalid()** zeroes all of the blocks within the page that are currently flagged as invalid. If *setvalid* is TRUE, all of the valid bits within the page are set.

In some cases, such as NFS, the valid bits cannot be set in order to maintain cache consistency.

**vm\_page\_is\_valid()** checks to determine if all of the DEV\_BSIZE blocks between *base* and *size* of the page are valid. If *size* is zero and the page is entirely invalid **vm\_page\_is\_valid()** will return TRUE, in all other cases a size of zero will return FALSE.

**vm\_page\_test\_dirty()** checks if a page has been modified via any of its physical maps, and if so, flags the entire page as dirty. **vm\_page\_dirty()** is called to modify the dirty bits.

**vm\_page\_dirty()** flags the entire page as dirty. It is expected that the page is not currently on the cache queue.

**vm\_page\_undirty()** clears all of the dirty bits in a page.

## NOTES

None of these functions are allowed to block.

## AUTHORS

This manual page was written by Chad David <davidc@acns.ab.ca>.