NAME

pmap_quick_enter_page, **pmap_quick_remove_page -** manage fast, single-page kernel address space mappings

SYNOPSIS

#include <sys/param.h>
#include <vm/vm.h>
#include <vm/pmap.h>

vm_offset_t
pmap_quick_enter_page(*vm_page_t m*);

void
pmap_quick_remove_page(*vm_offset_t kva*);

DESCRIPTION

The **pmap_quick_enter_page**() function accepts a single page *m*, and enters this page into a preallocated address in kernel virtual address (KVA) space. This function is intended for temporary mappings that will only be used for a very short period, for example a copy operation on the page contents.

The **pmap_quick_remove_page**() function removes a mapping previously created by **pmap_quick_enter_page**() at *kva*, making the KVA frame used by **pmap_quick_enter_page**() available for reuse.

On many architectures, **pmap_quick_enter_page**() uses a per-CPU pageframe. In those cases, it must disable preemption on the local CPU. The corresponding call to **pmap_quick_remove_page**() then re-enables preemption. It is therefore not safe for machine-independent code to sleep or perform locking operations while holding these mappings. Current implementations only guarantee the availability of a single page for the calling thread, so calls to **pmap_quick_enter_page**() must not be nested.

pmap_quick_enter_page() and **pmap_quick_remove_page**() do not sleep, and **pmap_quick_enter_page**() always returns a valid address. It is safe to use these functions under all types of locks except spin mutexes. It is also safe to use them in all thread contexts except primary interrupt context.

The page *must* not be swapped or otherwise reused while the mapping is active. It must be either wired or held, or it must belong to an unmanaged region such as I/O device memory.

RETURN VALUES

The **pmap_quick_enter_page**() function returns the kernel virtual address that is mapped to the page *m*.

SEE ALSO

pmap(9)

AUTHORS

This manual page was written by Jason A Harmening <*jah@FreeBSD.org*>.