NAME

bus_map_resource, **bus_unmap_resource**, **resource_init_map_request** - map or unmap an active resource

SYNOPSIS

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

#include <machine/bus.h>
#include <sys/rman.h>
#include <machine/resource.h>

int
bus_map_resource(*device_t dev*, *int type*, *struct resource *r*, *struct resource_map_request *args*, *struct resource_map *map*);

int
bus_unmap_resource(*device_t dev*, *int type*, *struct resource *r*, *struct resource_map *map*);

void
resource_init_map_request(*struct resource_map_request *args*);

DESCRIPTION

These functions create or destroy a mapping of a previously activated resource. Mappings permit CPU access to the resource via the bus_space(9) API.

The arguments are as follows:

dev The device that owns the resource.

type The type of resource to map. It is one of:

SYS_RES_IOPORT	for I/O ports
SYS_RES_MEMORY	for I/O memory

r A pointer to the *struct resource* returned by bus_alloc_resource(9).

args A set of optional properties to apply when creating a mapping. This argument can be set to NULL to request a mapping of the entire resource with the default properties.

map The resource mapping to create or destroy.

Resource Mappings

Resource mappings are described by a *struct resource_map* object. This structure contains a bus_space(9) tag and handle in the *r_bustag* and *r_bushandle* members that can be used for CPU access to the mapping. The structure also contains a *r_vaddr* member which contains the virtual address of the mapping if one exists.

The wrapper API for *struct resource* objects described in bus_activate_resource(9) can also be used with *struct resource_map*. For example, a pointer to a mapping object can be passed as the first argument to **bus_read_4**(). This wrapper API is preferred over using the *r_bustag* and *r_bushandle* members directly.

Optional Mapping Properties

The *struct resource_map_request* object passed in *args* can be used to specify optional properties of a mapping. The structure must be initialized by invoking **resource_init_map_request**(). Properties are then specified by setting one or more of these members:

offset, *length*

These two members specify a region of the resource to map. By default a mapping is created for the entire resource. The *offset* is relative to the start of the resource.

memattr

Specifies a memory attribute to use when mapping the resource. By default memory mappings use the VM_MEMATTR_UNCACHEABLE attribute.

RETURN VALUES

Zero is returned on success, otherwise an error is returned.

EXAMPLES

This maps a PCI memory BAR with the write-combining memory attribute and reads the first 32-bit word:

```
struct resource *r;
struct resource_map map;
struct resource_map_request req;
uint32_t val;
int rid;

rid = PCIR_BAR(0);
r = bus_alloc_resource_any(dev, SYS_RES_MEMORY, &rid, RF_ACTIVE |
    RF_UNMAPPED);
resource_init_map_request(&req);
```

```
req.memattr = VM_MEMATTR_WRITE_COMBINING;
bus_map_resource(dev, SYS_RES_MEMORY, r, &req, &map);
val = bus_read_4(&map, 0);
```

SEE ALSO

bus_activate_resource(9), bus_alloc_resource(9), bus_space(9), device(9), driver(9)

AUTHORS

This manual page was written by John Baldwin <*jhb@FreeBSD.org*>.