NAME

bus_alloc_resource, **bus_alloc_resource_any**, **bus_alloc_resource_anywhere** - allocate resources from a parent bus

SYNOPSIS

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

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

*struct resource **
bus_alloc_resource(*device_t dev, int type, int *rid, rman_res_t start, rman_res_t end, rman_res_t count, u_int flags*);

*struct resource **
bus_alloc_resource_any(*device_t dev, int type, int *rid, u_int flags*);

*struct resource **
bus_alloc_resource_anywhere(*device_t dev, int type, int *rid, rman_res_t count, u_int flags*);

DESCRIPTION

This is an easy interface to the resource-management functions. It hides the indirection through the parent's method table. This function generally should be called in attach, but (except in some rare cases) never earlier.

The **bus_alloc_resource_any**() and **bus_alloc_resource_anywhere**() functions are convenience wrappers for **bus_alloc_resource**(). **bus_alloc_resource_any**() sets *start*, *end*, and *count* to the default resource (see description of *start* below). **bus_alloc_resource_anywhere**() sets *start* and *end* to the default resource and uses the provided *count* argument.

The arguments are as follows:

dev is the device that requests ownership of the resource. Before allocation, the resource is owned by the parent bus.

type is the type of resource you want to allocate. It is one of:

PCI_RES_BUS for PCI bus numbers

SYS_RES_IRQ for IRQs

SYS_RES_DRQ for ISA DMA lines

SYS_RES_IOPORT for I/O ports

SYS_RES_MEMORY for I/O memory

rid points to a bus specific handle that identifies the resource being allocated. For ISA this is an index into an array of resources that have been setup for this device by either the PnP mechanism, or via the hints mechanism. For PCCARD, this is an index into the array of resources described by the PC Card's CIS entry. For PCI, the offset into PCI config space which has the BAR to use to access the resource. The bus methods are free to change the RIDs that they are given as a parameter. You must not depend on the value you gave it earlier.

start and *end* are the start/end addresses of the resource. If you specify values of 0ul for *start* and ~0ul for *end* and 1 for *count*, the default values for the bus are calculated.

count is the size of the resource. For example, the size of an I/O port is usually 1 byte (but some devices override this). If you specified the default values for *start* and *end*, then the default value of the bus is used if *count* is smaller than the default value and *count* is used, if it is bigger than the default value.

flags sets the flags for the resource. You can set one or more of these flags:

RF_ALLOCATED
: resource has been reserved. The resource still needs to be activated with bus_activate_resource(9).

RF_ACTIVE
: activate resource atomically.

RF_PREFETCHABLE
: resource is prefetchable.

RF_SHAREABLE
: resource permits contemporaneous sharing. It should always be set unless you know that the resource cannot be shared. It is the bus driver's task to filter out the flag if the bus does not support sharing. For example, pccard(4) cannot share IRQs while cardbus(4) can.

RF_UNMAPPED
: do not establish implicit mapping when activated via bus_activate_resource(9).

RETURN VALUES

A pointer to *struct resource* is returned on success, a null pointer otherwise.

EXAMPLES

This is some example code that allocates a 32 byte I/O port range and an IRQ. The values of *portid* and *irqid* should be saved in the softc of the device after these calls.

```
struct resource *portres, *irqres;
int portid, irqid;

portid = 0;
irqid = 0;
portres = bus_alloc_resource(dev, SYS_RES_IOPORT, &portid,
                             0ul, ~0ul, 32, RF_ACTIVE);
irqres = bus_alloc_resource_any(dev, SYS_RES_IRQ, &irqid,
                             RF_ACTIVE | RF_SHAREABLE);
```

SEE ALSO

bus_activate_resource(9), bus_adjust_resource(9), bus_map_resource(9), bus_release_resource(9), device(9), driver(9)

AUTHORS

This manual page was written by Alexander Langer <*alex@big.endian.de*> with parts by Warner Losh <*imp@FreeBSD.org*>.