NAME

rmalloc – allocate space from a resource map

SYNOPSIS

#include <sys/map.h>
#include <sys/ddi.h>

unsigned long rmalloc(struct map **mp*, **size_t** *size***);**

INTERFACE LEVEL

Architecture independent level 1 (DDI/DKI).

ARGUMENTS

mp Resource map from where the resource is drawn.

size Number of units of the resource.

DESCRIPTION

rmalloc( ) is used by a driver to allocate space from a previously defined and initialized resource map. The map itself is allocated by calling the function **rmallocmap**(9F). **rmalloc( )** is one of five functions used for resource map management. The other functions include:

rmalloc_wait(9F)	Allocate space from a resource map, wait if necessary.
rmfree(9F)	Return previously allocated space to a map.
rmallocmap(9F)	Allocate a resource map and initialize it.
rmfreemap(9F)	Deallocate a resource map.

rmalloc( ) allocates space from a resource map in terms of arbitrary units. The system maintains the resource map by size and index, computed in units appropriate for the resource. For example, units may be byte addresses, pages of memory, or blocks. The normal return value is an **unsigned long** set to the value of the index where sufficient free space in the resource was found.

RETURN VALUES

Under normal conditions, **rmalloc( )** returns the base index of the allocated space. Otherwise, **rmalloc( )** returns a **0** if all resource map entries are already allocated.

CONTEXT

rmalloc( ) can be called from user or interrupt context.

EXAMPLES

The following example is a simple memory map, but it illustrates the principles of map management. A driver allocates and initializes the map by calling both the **rmallocmap**(9F) and **rmfree**(9F) functions. **rmallocmap**(9F) is called to establish the number of slots or entries in the map, and **rmfree**(9F) to initialize the resource area the map is to manage. The following example is a fragment from a hypothetical **start** routine and illustrates the following procedures:

Panics the system if the required amount of memory can not be allocated (lines 11–15).

Uses **rmallocmap**(9F) to configure the total number of entries in the map, and **rmfree**(9F) to initialize the total resource area.

```
1   #define XX_MAPSIZE          12
2   #define XX_BUFSIZE  2560
3   static struct map *xx_mp;       /* Private buffer space map */
   . . .
4   xxstart()
5       /*
6        *  Allocate private buffer.  If insufficient memory,
7        *  display message and halt system.
8        */
9   {
10      register caddr_t bp;
   . . .
11      if ((bp = kmem_alloc(XX_BUFSIZE, KM_NOSLEEP) == 0)  {
12
13         cmn_err(CE_PANIC, "xxstart: kmem_alloc failed before %d buffer"
14               "allocation", XX_BUFSIZE);
15      }
16
17      /*
18       * Initialize the resource map with number
19       * of slots in map.
20       */
21      xx_mp = rmallocmap(XX_MAPSIZE);
22
24      /*
25       * Initialize space management map with total
26       * buffer area it is to manage.
27       */
28      rmfree(xx_mp, XX_BUFSIZE, bp);
      . . .
```

The **rmalloc( )** function is then used by the driver's **read** or **write** routine to allocate buffers for specific data transfers. The **uiomove**(9F) function is used to move the data between user space and local driver memory. The device then moves data between itself and local driver memory through DMA.

The next example illustrates the following procedures:

> The size of the I/O request is calculated and stored in the *size* variable (line 10).
>
> Buffers are allocated through the **rmalloc** function using the *size* value (line 15). If the allocation fails the system will panic.
>
> The **uiomove**(9F) function is used to move data to the allocated buffer (line 23).
>
> If the address passed to **uiomove**(9F) is invalid, **rmfree**(9F) is called to release the previously allocated buffer, and an **EFAULT** error is returned.

```
#define XX_BUFSIZE  2560
#define XX_MAXSIZE  (XX_BUFSIZE / 4)

static struct map *xx_mp;        /* Private buffer space map */
...
xxread(dev_t dev, uio_t *uiop, cred_t *credp)
{

register caddr_t addr;
register int     size;
    size = min(COUNT, XX_MAXSIZE);  /* Break large I/O request */
                                    /* into small ones */
    /*
     * Get buffer.
     */
    if ((addr = (caddr_t)rmalloc(xx_mp, size)) == 0)
        cmn_err(CE_PANIC, "read: rmalloc failed allocation of size %d",
                size);

    /*
     * Move data to buffer.  If invalid address is found,
     * return buffer to map and return error code.
     */
    if (uiomove(addr, size, UIO_READ, uiop) == -1)  {
        rmfree(xx_mp, size, addr);
        return(EFAULT);
    }
}
```

SEE ALSO

kmem_alloc(9F), **rmalloc_wait**(9F), **rmallocmap**(9F), **rmfree**(9F), **rmfreemap**(9F), **uiomove**(9F)

Writing Device Drivers