NAME

bit_alloc, **bit_clear**, **bit_count**, **bit_decl**, **bit_ffc**, **bit_ffs**, **bit_ff_at**, **bit_ffc_at**, **bit_ffs_at**, **bit_ffc_area**, **bit_ffs_area**, **bit_ff_area_at**, **bit_ffc_area_at**, **bit_ffs_area_at**, **bit_nclear**, **bit_nset**, **bit_ntest**, **bit_set**, **bit_test**, **bitstr_size** - bit-string manipulation functions and macros

SYNOPSIS

#include <bitstring.h>

*bitstr_t **
bit_alloc(*size_t nbits*);

void
bit_decl(*bitstr_t *name*, *size_t nbits*);

void
bit_clear(*bitstr_t *name*, *size_t bit*);

void
bit_count(*bitstr_t *name*, *size_t count*, *size_t nbits*, *ssize_t *value*);

void
bit_ffc(*bitstr_t *name*, *size_t nbits*, *ssize_t *value*);

void
bit_ffs(*bitstr_t *name*, *size_t nbits*, *ssize_t *value*);

void
bit_ffc_at(*bitstr_t *name*, *size_t start*, *size_t nbits*, *ssize_t *value*);

void
bit_ffs_at(*bitstr_t *name*, *size_t start*, *size_t nbits*, *ssize_t *value*);

void
bit_ff_at(*bitstr_t *name*, *size_t start*, *size_t nbits*, *int match*, *ssize_t *value*);

void
bit_ffc_area(*bitstr_t *name*, *size_t nbits*, *size_t size*, *ssize_t *value*);

void
bit_ffs_area(*bitstr_t *name*, *size_t nbits*, *size_t size*, *ssize_t *value*);

void
bit_ffc_area_at(*bitstr_t *name, size_t start, size_t nbits, size_t size, ssize_t *value*);

void
bit_ffs_area_at(*bitstr_t *name, size_t start, size_t nbits, size_t size, ssize_t *value*);

void
bit_ff_area_at(*bitstr_t *name, size_t start, size_t nbits, size_t size, int match, ssize_t *value*);

bit_foreach(*bitstr_t *name, size_t nbits, size_t var*);

bit_foreach_at(*bitstr_t *name, size_t start, size_t nbits, size_t var*);

bit_foreach_unset(*bitstr_t *name, size_t nbits, size_t var*);

bit_foreach_unset_at(*bitstr_t *name, size_t start, size_t nbits, size_t var*);

void
bit_nclear(*bitstr_t *name, size_t start, size_t stop*);

void
bit_nset(*bitstr_t *name, size_t start, size_t stop*);

int
bit_ntest(*bitstr_t *name, size_t start, size_t stop, int match*);

void
bit_set(*bitstr_t *name, size_t bit*);

int
bitstr_size(*size_t nbits*);

int
bit_test(*bitstr_t *name, size_t bit*);

DESCRIPTION

These macros operate on strings of bits.

The function **bit_alloc**() returns a pointer of type "*bitstr_t **" to sufficient space to store *nbits* bits, or NULL if no space is available. If successful, the returned bit string is initialized with all bits cleared.

The macro **bit_decl**() declares a bit string with sufficient space to store *nbits* bits. **bit_decl**() may be used to include statically sized bit strings in structure definitions or to create bit strings on the stack. Users of this macro are responsible for initialization of the bit string, typically via a global initialization of the containing struct or use of the **bit_nset**() or **bin_nclear**() functions.

The macro **bitstr_size**() returns the number of bytes necessary to store *nbits* bits. This is useful for copying bit strings.

The functions **bit_clear**() and **bit_set**() clear or set the zero-based numbered bit *bit*, in the bit string *name*.

The **bit_nset**() and **bit_nclear**() functions set or clear the zero-based numbered bits from *start* through *stop* in the bit string *name*.

The **bit_test**() function evaluates to non-zero if the zero-based numbered bit *bit* of bit string *name* is set, and zero otherwise.

The **bit_ntest**() function evaluates to non-zero if the zero-based numbered bits from *start* through *stop* in the bit string *name* all have the value *match*.

The function **bit_ffc**() stores in the location referenced by *value* the zero-based number of the first bit not set in the array of *nbits* bits referenced by *name*. If all bits are set, the location referenced by *value* is set to -1.

The **bit_ffs**() function stores in the location referenced by *value* the zero-based number of the first bit set in the array of *nbits* bits referenced by *name*. If no bits are set, the location referenced by *value* is set to -1.

The function **bit_ffc_at**() stores in the location referenced by *value* the zero-based number of the first bit not set in the array of *nbits* bits referenced by *name*, at or after the zero-based bit index *start*. If all bits at or after *start* are set, the location referenced by *value* is set to -1.

The **bit_ffs_at**() function stores in the location referenced by *value* the zero-based number of the first bit set in the array of *nbits* bits referenced by *name*, at or after the zero-based bit index *start*. If no bits are set after *start*, the location referenced by *value* is set to -1.

The **bit_ff_at**() function stores in the location referenced by *value* the zero-based number of the first bit in the array of *nbits* bits referenced by *name*, at or after the zero-based bit index *start* that has value *match*. If no bits after *start* match that value, the location referenced by *value* is set to -1.

The **bit_ffc_area**() function stores in the location referenced by *value* the zero-based number of the first

bit beginning a sequence of unset bits of at least *size* unset bits in the array of *nbits* bits referenced by *name*. If no sequence of contiguous unset bits of the specified *size* can be found, the location referenced by *value* is set to -1.

The **bit_ffs_area**() function stores in the location referenced by *value* the zero-based number of the first bit beginning a sequence of set bits of at least *size* set bits in the array of *nbits* bits referenced by *name*. If no sequence of contiguous set bits of the specified *size* can be found, the location referenced by *value* is set to -1.

The **bit_ffc_area_at**() function stores in the location referenced by *value* the zero-based number of the first bit beginning a sequence of unset bits of at least *size* unset bits in the array of *nbits* bits referenced by *name*, at or after the zero-based bit index *start*. If no sequence of contiguous unset bits of the specified *size* can be found at or after *start*, the location referenced by *value* is set to -1.

The **bit_ffs_area_at**() function stores in the location referenced by *value* the zero-based number of the first bit beginning a sequence of set bits of at least *size* set bits in the array of *nbits* bits referenced by *name*, at or after the zero-based bit index *start*. If no sequence of contiguous set bits of the specified *size* can be found at or after *start*, the location referenced by *value* is set to -1.

The **bit_ff_area_at**() function stores in the location referenced by *value* the zero-based number of the first bit beginning a sequence of bits of at least *size* bits in the array of *nbits* bits referenced by *name*, at or after the zero-based bit index *start* in which all bits have the value *match*. If no sequence of contiguous such bits of the specified *size* can be found at or after *start*, the location referenced by *value* is set to -1.

The **bit_count**() function stores in the location referenced by *value* the number of bits set in the array of *nbits* bits referenced by *name*, at or after the zero-based bit index *start*.

The macro **bit_foreach**() traverses all set bits in the array of *nbits* referenced by *name* in the forward direction, assigning each location in turn to *var*.

The macro **bit_foreach_at**() traverses all set bits in the array of *nbits* referenced by *name* in the forward direction at or after the zero-based bit index *start*, assigning each location in turn to *var*.

The macro **bit_foreach_unset**() traverses all unset bits in the array of *nbits* referenced by *name* in the forward direction, assigning each location in turn to *var*.

The macro **bit_foreach_unset_at**() traverses all unset bits in the array of *nbits* referenced by *name* in the forward direction at or after the zero-based bit index *start*, assigning each location in turn to *var*.

The arguments in bit string macros are evaluated only once and may safely have side effects.

EXAMPLES

```
#include <limits.h>
#include <bitstring.h>

...
#define   LPR_BUSY_BIT          0
#define   LPR_FORMAT_BIT            1
#define   LPR_DOWNLOAD_BIT      2
...
#define   LPR_AVAILABLE_BIT     9
#define   LPR_MAX_BITS          10

make_lpr_available()
{
        bitstr_t bit_decl(bitlist, LPR_MAX_BITS);
        ...
        bit_nclear(bitlist, 0, LPR_MAX_BITS - 1);
        ...
        if (!bit_test(bitlist, LPR_BUSY_BIT)) {
                bit_clear(bitlist, LPR_FORMAT_BIT);
                bit_clear(bitlist, LPR_DOWNLOAD_BIT);
                bit_set(bitlist, LPR_AVAILABLE_BIT);
        }
}
```

SEE ALSO

malloc(3), bitset(9)

HISTORY

The **bitstring** functions first appeared in 4.4BSD.