```
#include <linux/types.h
#include <linux/bitmap.h>
```

- define a bitmap。

```
#define DECLARE_BITMAP(name,bits) \
    unsigned long name[BITS_TO_LONGS(bits)]
```

```
#define MAX_PWMS 1024
......
static DECLARE_BITMAP(allocated_pwms, MAX_PWMS);
```

- find a contiguous aligned zero area

```
/*
 * bitmap_find_next_zero_area - find a contiguous aligned zero area
 * @map: The address to base the search on
 * @size: The bitmap size in bits
 * @start: The bitnumber to start searching at
 * @nr: The number of zeroed bits we're looking for
 * @align_mask: Alignment mask for zero area
 *
 * The @align_mask should be one less than a power of 2; the effect is that
 * the bit offset of all zero areas this function finds is multiples of that
 * power of 2. A @align_mask of 0 means no alignment is required.
 */
unsigned long bitmap_find_next_zero_area(unsigned long *map,
                     unsigned long size,
                     unsigned long start,
                     unsigned int nr,
                     unsigned long align_mask)
```

sample:

```c
static int alloc_pwms(int pwm, unsigned int count)
{
    unsigned int from = 0;
    unsigned int start;

    if (pwm >= MAX_PWMS)
        return -EINVAL;

    if (pwm >= 0)
        from = pwm;

    start = bitmap_find_next_zero_area(allocated_pwms, MAX_PWMS, from,
                       count, 0);

    if (pwm >= 0 && start != pwm)
        return -EEXIST;

    if (start + count > MAX_PWMS)
        return -ENOSPC;

    return start;
}
```

- 设置从start开始的len位

> void bitmap_set(unsigned long *map, unsigned int start, int len)

```c
bitmap_set(allocated_pwms, chip->base, chip->npwm);
```

- clear从start开始的len位

> void bitmap_clear(unsigned long *map, unsigned int start, int len)

```c
bitmap_clear(allocated_pwms, chip->base, chip->npwm);
```

- 打印bitmap(调试用)

```c
/**
 * bitmap_scnprintf - convert bitmap to an ASCII hex string.
 * @buf: byte buffer into which string is placed
 * @buflen: reserved size of @buf, in bytes
 * @maskp: pointer to bitmap to convert
 * @nmaskbits: size of bitmap, in bits
 *
 * Exactly @nmaskbits bits are displayed.  Hex digits are grouped into
 * comma-separated sets of eight digits per set.  Returns the number of
 * characters which were written to *buf, excluding the trailing \0.
 */
int bitmap_scnprintf(char *buf, unsigned int buflen,
    const unsigned long *maskp, int nmaskbits)
```

把maskp bitmap (size为nmaskbits)，打印到buf中。

- convert an ASCII hex string into a bitmap

```
/**
 * __bitmap_parse - convert an ASCII hex string into a bitmap.
 * @buf: pointer to buffer containing string.
 * @buflen: buffer size in bytes.  If string is smaller than this
 *    then it must be terminated with a \0.
 * @is_user: location of buffer, 0 indicates kernel space
 * @maskp: pointer to bitmap array that will contain result.
 * @nmaskbits: size of bitmap, in bits.
 *
 * Commas group hex digits into chunks.  Each chunk defines exactly 32
 * bits of the resultant bitmask.  No chunk may specify a value larger
 * than 32 bits (%-EOVERFLOW), and if a chunk specifies a smaller value
 * then leading 0-bits are prepended.  %-EINVAL is returned for illegal
 * characters and for grouping errors such as "1,,5", ",44", "," and "".
 * Leading and trailing whitespace accepted, but not embedded whitespace.
 */
int __bitmap_parse(const char *buf, unsigned int buflen,
        int is_user, unsigned long *maskp,
        int nmaskbits)
```

```
/**
 * bitmap_parse_user - convert an ASCII hex string in a user buffer into a bitmap
 *
 * @ubuf: pointer to user buffer containing string.
 * @ulen: buffer size in bytes.  If string is smaller than this
 *    then it must be terminated with a \0.
 * @maskp: pointer to bitmap array that will contain result.
 * @nmaskbits: size of bitmap, in bits.
 *
 * Wrapper for __bitmap_parse(), providing it with user buffer.
 *
 * We cannot have this as an inline function in bitmap.h because it needs
 * linux/uaccess.h to get the access_ok() declaration and this causes
 * cyclic dependencies.
 */
int bitmap_parse_user(const char __user *ubuf,
            unsigned int ulen, unsigned long *maskp,
            int nmaskbits)
{
    if (!access_ok(VERIFY_READ, ubuf, ulen))
        return -EFAULT;
    return __bitmap_parse((const char __force *)ubuf,
                ulen, 1, maskp, nmaskbits);

}
```

更多bitmap APIs，见bitmap.h file。