读取hc1600芯片unique_id

适用于hcRTOS 和 hcLinux .

- make menuconfig中需要打开 `BR2_PACKAGE_PREBUILTS_LIBEFUSE`
- 源码中需要include `hcuapi/efuse.h`
- 读取 `unique_id` 的方式
  - 通过 `fd = open("/dev/efuse", O_RDWR)`
  - `ret = read(fd, &efuse_bits, sizeof(struct hc_efuse_bit_map));` , 通过read方式读取整个efuse区域
  - `unique_id` 就是在 `struct hc_efuse_bit_map` 里面
- `unique_id` 长度为 **64bits,**
  - 其中低32位为 `struct hc_efuse_bit_map .chip_vendor.unique_id0`
  - 其中高32位为 `struct hc_efuse_bit_map .chip_vendor.unique_id1`

```c
// from hcuapi/efuse.h

struct __chip_vendor {
        uint32_t unique_id0 : 32;           // unique id bit[31:0]
        uint32_t unique_id1 : 32;           // unique id bit[63:32]
        uint8_t hichip_reserve0 : 8;
        uint8_t hichip_reserve1 : 8;
        uint16_t hichip_reserve2 : 16;
} __attribute__((packed));

struct hc_efuse_bit_map {
        struct __chip_vendor chip_vendor;
        struct __customer customer;
        struct __write_protect wp;
} __attribute__((packed));
```

示例代码如下所示:

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <fcntl.h>
#include <sys/ioctl.h>
```

```c
#include <hcuapi/efuse.h>
#include <hcuapi/chipid.h>

int console_get_unique_id(int argc, char **argv)
{
    int fd, ret;
    struct hc_efuse_bit_map efuse_bits;
    uint64_t unique_id;

    fd = open("/dev/efuse", O_RDWR);
    if (fd < 0) {
        printf("[error] cannot open /dev/efuse, ret:%d\r\n", fd);
        return -1;
    }

    ret = read(fd, &efuse_bits, sizeof(struct hc_efuse_bit_map));
    if(ret != sizeof(struct hc_efuse_bit_map)){
        printf("[error] cannot read /dev/efuse correctly\n");
        close(fd);
        return -1;
    }

    unique_id = efuse_bits.chip_vendor.unique_id1;
    unique_id = unique_id << 32;
    unique_id |= (efuse_bits.chip_vendor.unique_id0 & 0xffffffff);

    printf("hichip unique id is 0x%llx\n", unique_id);

    close(fd);
    return 0;
}

```