ESP32 中的 NVS

淘宝店铺：

PC 端：

http://n-xytrt8gqu585po94mwj5atokcyd4.taobao.com/index.htm

手机端：

https://shop.m.taobao.com/shop/shop_index.htm?sellerId=755668508&shopId=104493595&inShopPageId=423890608&pathInfo=shop/index2

资料下载地址：

链接：https://pan.baidu.com/s/1kCjD8yktZECSGmHomx_veg?pwd=q8er
提取码：q8er

源码下载地址：

https://gitee.com/vi-iot/esp32-board.git

一、什么是 NVS？

NVS 即 Non-volatile storage，意思是非易失存储，也就是掉电后能依然能持久化保存数据。在我们应用 NVS 时，一般用于存储一些配置数据、状态数据等，一般不会用来存储存放大量的数据量。

在嵌入式系统中，NVS 主要是在 flash 进行键值对的存储。那么什么是键值对？这边举一个例子方便大家理解，假设我们要把东西存到 flash 中，按照底层的操作习惯，我们要先指定一个地址，然后对这个地址执行擦除操作，然后才能写入；读取的时候也需要根据这个地址，然后指定读取长度。如果我们要存的项比较多，又在代码中比较分散，我们对 flash 的地址就很难管理。因为我们很难知道要存的内容与其他地址有没冲突，会不会误擦除。存在诸多问题。那如果有一个机制，方便帮我们把这些检测判断活都干了，不需要我们指定地址操作，那岂不是很好。有的人又说了，文件系统不就是这样吗？是的，文件系统是这样，但 NVS 操作更加轻量级。

在 NVS 中，我们要存一个值，我们不需要指定地址，但需要指定一个“键”，我们称为 key，然后我们在这个“键”索引下存我们的值，也就是 value。假设我们要存 wifi 的 ssid 和密码，我们可以在 nvs 中这样定义

key = ssid,value = testwifi

key = password,value = 12345678

在键名 ssid 下，我们存的值是 testwifi，在键名 password 下，我们存的值是 12345678。因此键名 key 一般不会修改，经常修改的是 value。

在 ESP32 中对于 NVS 的操作，还需要指定一个命名空间，是因为还考虑了一种情况，在各个不同的功能模块中，键名是有可能取到一样的，比如对于 wifi 模块，存在一个 password 键名，对于管理员模块，可能也存在一个 password 键名，这样有可能就造成了重复，程序无法按我们的意思进行。如果我们增加了一个命名空间进行隔离，那么键名有重复也不怕，比如说在 wifi 模块中，我们指定一个命名空间”wifi”，在此命名空间下有 ssid 和 password 键名，在管理员模块，我们指定一个命名空间”manager”，在此命名空间下有 password 键名，这两组命名空间互不干扰。

二、ESP32 中 NVS 的使用

上一节介绍了 NVS 的基本概念，这一节我们看下 esp-idf 中给我们提供了什么接口。

```
//打开一个 NVS 存储操作
esp_err_t nvs_open(const char* namespace_name,    /*命名空间名称*/
                   nvs_open_mode_t open_mode,     /*  打开的模式*/
                                                    NVS_READONLY，只读
                                                    NVS_READWRITE  读写
                   nvs_handle_t *out_handle);     /*  返回的 nvs 操作句柄

//向 NVS 中读取字符串数据
esp_err_t nvs_get_str (nvs_handle_t handle,      /*nvs 操作句柄*/
                       const char* key,          /*键名*/
                       char* out_value,          /*读取到的值*/
                       size_t* length)           /*读取到的字符串长度*/

//向 NVS 中读取二进制数据
esp_err_t nvs_get_blob(nvs_handle_t handle,   /*nvs 操作句柄*/
```

```
                        const char* key,        /*键名*/
                        void* out_value,        /*返回的值
                        size_t* length);        /*返回值的长度*/

//向 NVS 中写入字符串数据(最大 4000 字节)
esp_err_t nvs_set_str (nvs_handle_t handle,     /*nvs 操作句柄*/
                        const char* key,        /*键名*/
                        const char* value);     /*值*/

//向 NVS 中写入二进制数据（最大值:NVS 分区大小*97.6%-4000）
esp_err_t nvs_set_blob(nvs_handle_t handle,     /*nvs 操作句柄
                        const char* key,        /*键名*/
                        const void* value,      /*值*/
                        size_t length);         /*大小*/

//擦除对应命名空间下所有的 nvs 值
esp_err_t nvs_erase_all(
                        nvs_handle_t handle); /*nvs 操作句柄*/


//在执行写操作后，为了确保写入到 NVS 区域中，需要调用此函数
esp_err_t nvs_commit(
                    nvs_handle_t handle);       /*nvs 操作句柄*/

//关闭 NVS 操作句柄，释放资源
void nvs_close(
                    nvs_handle_t handle);       /*nvs 操作句柄*/
```

以上操作有几点注意：

1) 操作 NVS 的第一步就是调用 nvs_open，获取到 nvs 句柄后才能执行其他操作
2) 命名空间长度、key 长度，默认是 15 字节
3) 在执行写操作后，需要 nvs_commit，才能确保写入到了 nvs 区域

在上一课程对分区表进行解析后，我们知道 nvs 默认的分区地址是 0x9000 开始，默认大小是 0x6000，flash 中这一段的区域是专门留给我们做 nvs 存储的。我们来看下具体例程部分源码，源码位于 esp32-board/nvs 中

```
#define NVS_BOB_NAMESPACE  "Bob"         //namespace 最长 15 字节
#define NVS_JOHN_NAMESPACE  "John"       //namespace 最长 15 字节

#define NVS_AGE_KEY     "age"         //年龄键名
#define NVS_SEX_KEY     "sex"         //性别键名

void app_main(void)
{
```

```
    //初始化 NVS
    esp_err_t ret = nvs_flash_init();
    if (ret == ESP_ERR_NVS_NO_FREE_PAGES || ret ==
ESP_ERR_NVS_NEW_VERSION_FOUND) {
        //NVS 出现错误，执行擦除
        ESP_ERROR_CHECK(nvs_flash_erase());
        //重新尝试初始化
        ESP_ERROR_CHECK(nvs_flash_init());
    }

    char read_buf[64];
    size_t len = 0;
    //以字节方式写入到 NVS 中
    write_nvs_str(NVS_BOB_NAMESPACE,NVS_SEX_KEY,"female");
    write_nvs_str(NVS_JOHN_NAMESPACE,NVS_SEX_KEY,"male");
    //读取 NVS_BOB_NAMESPACE 命名空间中的 SEX 键值
    len =read_nvs_str(NVS_BOB_NAMESPACE,NVS_SEX_KEY,read_buf,64);
    if(len)
        ESP_LOGI(TAG,"Read BOB SEX:%s",read_buf);
    else
        ESP_LOGI(TAG,"Read BOB SEX fail,please perform nvs_erase_key and try
again");
     //读取 NVS_JOHN_NAMESPACE 命名空间中的 SEX 键值
    len =read_nvs_str(NVS_JOHN_NAMESPACE,NVS_SEX_KEY,read_buf,64);
    if(len)
        ESP_LOGI(TAG,"Read JOHN SEX:%s",read_buf);
    else
        ESP_LOGI(TAG,"Read JOHN SEX fail,please perform nvs_erase_key and
try again");
    uint8_t blob_buf[32];
    blob_buf[0] = 19;
    //以字节方式写入
    write_nvs_blob(NVS_BOB_NAMESPACE,NVS_AGE_KEY,blob_buf,1);
    blob_buf[0] = 23;
    write_nvs_blob(NVS_JOHN_NAMESPACE,NVS_AGE_KEY,blob_buf,1);

    //以字节方式读取
    len = read_nvs_blob(NVS_BOB_NAMESPACE,NVS_AGE_KEY,blob_buf,32);
    if(len)
        ESP_LOGI(TAG,"Read BOB age:%d",blob_buf[0]);
    else
        ESP_LOGI(TAG,"Read BOB age fail,please perform nvs_erase_key and try
again");
    //以字节方式读取
```

```
    len = read_nvs_blob(NVS_JOHN_NAMESPACE,NVS_AGE_KEY,blob_buf,32);
    if(len)
        ESP_LOGI(TAG,"Read JOHN age:%d",blob_buf[0]);
    else
        ESP_LOGI(TAG,"Read JOHN age fail,please perform nvs_erase_key and
try again");

    while(1)
    {
        vTaskDelay(pdMS_TO_TICKS(1000));
    }
}
```

这个例程中，有两个命名空间”Bob”和”John”，这两个命名空间分别都有”age”和”sex”这两个键，这个例程可以看到，两组命名空间之间互不干扰，其他具体代码请看例程源码。