



Espressif Systems

ESP8266 Interface I2C



ESP8266 I2C 使用说明

Version 0.2

Espressif Systems IOT Team

Copyright (c) 2015

免责声明和版权公告

本文中的信息，包括供参考的URL地址，如有变更，恕不另行通知。

文档“按现状”提供，不负任何担保责任，包括对适销性、适用于特定用途或非侵权性的任何担保，和任何提案、规格或样品在他处提到的任何担保。本文档不负任何责任，包括使用本文档内信息产生的侵犯任何专利权行为的责任。本文档在此未以禁止反言或其他方式授予任何知识产权使用许可，不管是明示许可还是暗示许可。

Wi-Fi联盟成员标志归Wi-Fi联盟所有。

文中提到的所有商标名称、商标和注册商标均属其各自所有者的财产，特此声明。

版权归© 2015 乐鑫信息科技（上海）有限公司所有。保留所有权利。



Table of Contents

1.	概述.....	4
2.	i2c master接口.....	4
2.1.	初始化	4
2.2.	i2c 起始	5
2.3.	i2c 停止	5
2.4.	i2c 主机回复ACK	6
2.5.	i2c 主机回复NACK.....	6
2.6.	检查 i2c 从机应答.....	7
2.7.	向 i2c 总线写数据.....	7
2.8.	从 i2c 总线读数据.....	7
3.	使用示例	8



1. 概述

ESP8266 目前提供作为 **I2C** 主设备的接口，可以对其他 I2C 从设备（例如大多数数字传感器）进行控制与读写。

每个 GPIO 管脚内部都可以配置为开漏模式（open-drain），从而可以灵活的将 GPIO 口用作 I2C data 或 clock 功能。

同时，芯片内部提供上拉电阻，以节省外部的上拉电阻。

ESP8266 作为 I2C 主机的 SDA 与 SCL 线波形由 GPIO 模拟产生，在 SCL 的上升沿之后 SDA 取数。SCL 高低电平各保持 5us，因此 I2C 时钟频率约为 100KHz。

2. i2c master接口

2.1. 初始化

i2c_master_gpio_init: GPIO硬件初始化

具体如下：

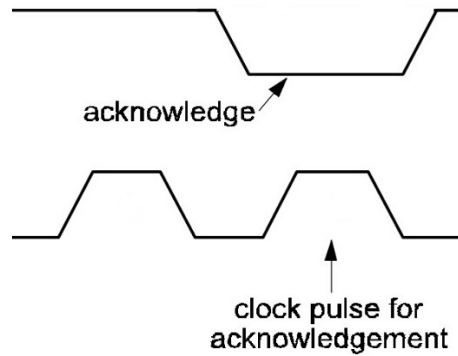
- (1) 选择pin脚功能，配置为GPIO
- (2) 配置GPIO为开漏模式
- (3) 初始化SDA与SCL为高电平
- (4) 使能GPIO中断并复位从机状态。

i2c_master_init(void): 复位从机状态



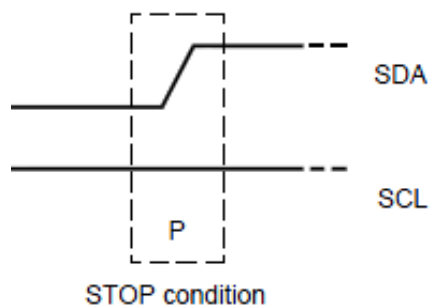
2.2. i2c 起始

i2c_master_start(void): 主机产生i2c起始条件
如下图:



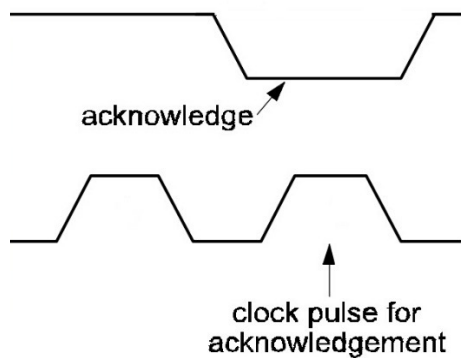
2.3. i2c 停止

i2c_master_stop(void): 主机产生i2c停止条件
如下图:



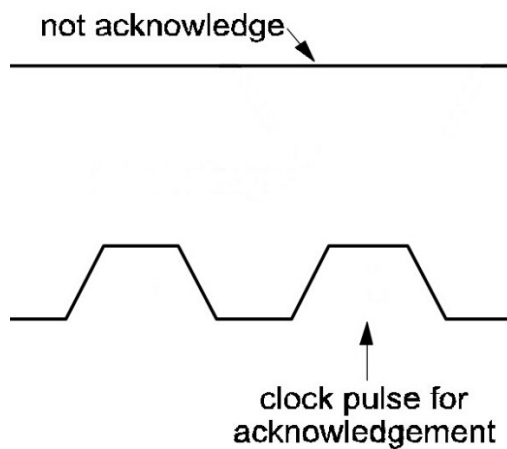
2.4. i2c 主机回复ACK

i2c_master_send_ack(void): 设置i2c主机应答ACK
如下图:



2.5. i2c 主机回复NACK

i2c_master_send_nack(void): 设置i2c主机回复 NACK
如下图:





2.6. 检查 i2c 从机应答

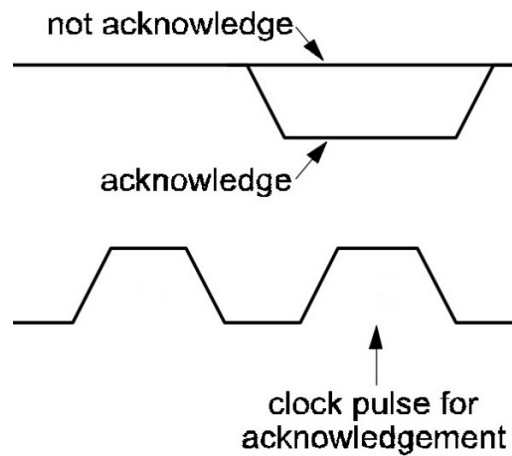
`bool i2c_master_checkAck(void)`: 检查从机应答状态

返回值:

TRUE: 查询到从机acknowledge

FALSE: 查询到从机not acknowledge

如下图:



2.7. 向 i2c 总线写数据

`i2c_master_writeByte(uint8 wrdata)`: 向i2c总线写数

参数:

1 Byte的数据

说明:

数据最高位先发送, 最低位最后发送。

可以发送从机地址, 或者发送数据。

2.8. 从 i2c 总线读数据

`i2c_master_readByte (void)`: 从SPI slave读取一个字节

返回值:

读取到的 1 Byte数据。



3. 使用示例

请参考 esp_iot_sdk 提供的 IOT_Demo 使用，例如：

```
void ICACHE_FLASH_ATTR
user_mvh3004_init(void)
{
    i2c_master_gpio_init();
}

LOCAL bool ICACHE_FLASH_ATTR
user_mvh3004_burst_read(uint8 addr, uint8 *pData, uint16 len)
{
    uint8 ack;
    uint16 i;

    i2c_master_start();
    i2c_master_writeByte(addr);
    ack = i2c_master_checkAck();

    if (!ack) {
        os_printf("addr not ack when tx write cmd \n");
        i2c_master_stop();
        return false;
    }

    i2c_master_stop();
    i2c_master_wait(40000);

    i2c_master_start();
    i2c_master_writeByte(addr + 1);
    ack = i2c_master_checkAck();

    if (!ack) {
        os_printf("addr not ack when tx write cmd \n");
        i2c_master_stop();
        return false;
    }

    for (i = 0; i < len; i++) {
        pData[i] = i2c_master_readByte();

        if (i == (len - 1))
            i2c_master_send_nack();
        else
            i2c_master_send_ack();
    }

    i2c_master_stop();

    return true;
} ? end user_mvh3004_burst_read ?
```