FANCONN™ 开源 NBIOT 开发平台	文档编号	产品版本	密级
	V1.0	V1.0	公开
	文档名称： LiteOneBC28 IIC 指南		共 10 页

FANCONN 开源 NBIOT 开发平台

LiteOneBC28 IIC 指南

广州准捷电子科技有限公司

版本：V1.0

地址：广州市天河区软件路 17 号亿牛大厦 5 楼
电话：+86 18078868880 电子邮件：yellow@fanconn.com

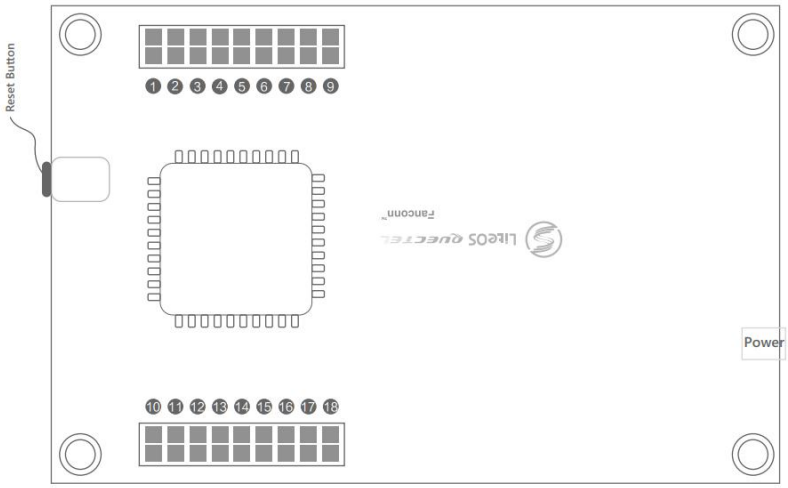
版权声明

版本记录

版本号	作者	版本说明	日期
V1.0	小曾	初始版本	2018-11-14

目录

1. GPIO 引脚对应图



丝印的对应引脚名：		软件 SDK 对应引脚名：	
1	NC	NC	
2	SWD-C	SWD-D	NULL	NULL
3	RTS	VDDEXT	PINNAME_RST	NULL
4	GPIO9	CTS	PINNAME_USIM_DETECT	PINNAME_CTS
5	RI	DTR	NULL	PINNAME_DTR
6	RX1	TX1	NULL	NULL
7	REST	NETLIGHT	NULL	NULL（*特殊配置）
8	GND	GND		
9	5V	5V		
10	SPI2-SI	SPI2-CKL	PINNAME_SPI_SI	PINNAME_SPI_CLK
11	SPI2-SO	SPI2-CS	PINNAME_SPI_SO	PINNAME_SPI_CS
12	TX29	RX28	PINNAME_UART_TX3	PINNAME_UART_RX3
13	RXDB	TXDB	PINNAME_DBG_RXD	NULL(*特殊配置）
14	ADC	NC	NULL	NULL
15	SPI1-CS	SPI1-CL	PINNAME_SPI_CS2	PINNAME_SPI_CLK2
16	SPI1-SI	SPI1-SO	PINNAME_SPI_SI2	PINNAME_SPI_SO2
17	GND	GND		
18	3.3	3.3		

可以参考开箱说明书。

根据移远提供的资料不是很完善导致引脚丝印一些地方出现小瑕疵，具体正确引脚图请参考上面的开箱说明书为准

程序配置参考对应例程：

配套程序读取 MAX44009。

```
    if(MAX44009_iic_init() == IIC_RET_OK)
    {
        APP_DEBUG("IIC is ok\n");
    }


    max44009_init();


    for (;;)
    {
        light = max44009_get();
        APP_DEBUG("\r\n<-- light:%d-->\r\n",light);
        osDelay(2000);
        (void)osThreadYield();
    }
}
```

2. IIC 初始化函数

IIC 初始化函数如下：

```
void i2c_init(void)
```

初始化内部 IIC。

3. IIC 映射引脚设置

Iic 映射引脚函数如下：

```
I2C_RET i2c_claim(I2C_BUS* bus, I2C_PIN i2c_pin)
```

bus：IIC 的对应接口

i2c_pin：IIC 对应的引脚

```
PIN     scl;/* Serial Clock Line */
PIN     sda;/* Serial Data Line */
```

返回：配置状态

参考配置：

```
if (i2c_claim(&I2C_bus, i2c_pin_name) != I2C_RET_OK)
{
   QDEBUG_NORMAL("<---I2C claim failed ! --->");
   return I2C_RET_ERROR;
}
```

其中 IIC 引脚：

只要在 GPIO 引脚对应图中的 GPIO 都可以映射成为 SDA 和 SCK，映射方式如下：

```
#define SCL_I2C_PINNAME      pin_map[PINNAME_SPI_CLK]//
#define SDA_I2C_PINNAME      pin_map[PINNAME_SPI_SO] //
```

最后采用 ISC_PIN 定义一个相同的类型，再将 i2c_pin_name 作为实参传递给函数。

```
I2C_PIN i2c_pin_name = { SCL_I2C_PINNAME, SDA_I2C_PINNAME};  //sck sda
```

4. IIC 使能函数

GPIO 设置引脚函数如下：

```
I2C_RET i2c_activate(I2C_BUS bus, I2C_CONFIGURATION config)
```

bus： IIC 的对应接口

config：IIC 对应的参数配置

返回：配置状态

参考配置

```
if (i2c_activate(&I2C_bus, i2c_config) != I2C_RET_OK)
   {
      QDEBUG_NORMAL("<---I2C activate failed ! --->");
      return I2C_RET_ERROR;
   }
```

其中关于 IIC 配置如下，自己通过 typedef 定义三组 IIC 的通用配置，这些都需要用户自己编写。

```
typedef enum{
    i2c_addr_type_7bit,                    地址位
    i2c_addr_type_10bit,
}I2C_ADDR_TYPE;

typedef enum{
    I2C_Half_Time_100kbit = 256,
    I2C_Half_Time_400kbit = 64,            速度
}I2C_HALf_TIME;


typedef enum{
    I2C_MODE_SLAVE,
    I2C_MODE_MASTER,                       主从机
}I2C_MODE;
```

最后通过 I2C_CONFIGURATION 定义一组 i2c_config，按照这个顺序进行选择配置，最后作为实参传递给配置函数

```
    I2C_CONFIGURATION i2c_config = {
                                        I2C_MODE_MASTER,
                                        i2c_addr_type_7bit,
                                        I2C_Half_Time_100kbit
                                    };
```

5. 用户封装 IIC 初始化函数

由于上面配置起来有些零散，对于移植起来可能会有一些稍微繁琐，所以基于上面几个函数的基础上，作为用户再次封装一层，具体代码可以参考配套程序读取 MAX44009。下面是 IIC 初始化封装，以下代码仅供参考。

```
static I2C_RET iic_init(I2C_CONFIG* iic_struct)
{
    i2c_init();
    if (i2c_claim(&(iic_struct->i2c_bus), iic_struct->i2c_pin) != I2C_RET_OK)
    {
        QDEBUG_NORMAL("<---I2C claim failed ! --->");
        return I2C_RET_ERROR;
    }
    else
    {
        QDEBUG_NORMAL("<---I2C claim succeed ! --->");
    }

    if (i2c_activate(iic_struct->i2c_bus, iic_struct->i2c_conf) != I2C_RET_OK)
    {
        QDEBUG_NORMAL("<---I2C activate failed ! --->");
        return I2C_RET_ERROR;
    }
    else
    {
        QDEBUG_NORMAL("<---I2C activate succeed ! --->");
    }

    return I2C_RET_OK;
}
```

下面代码是根据不同的传感器，用户可以根据自己的配置设置好，最后通过结构体实参传递给初始化函数即可，以下代码仅供参考。

```
static I2C_status MAX44009_iic_init(void)
{
    I2C_CONFIG iic_struct;
    I2C_PIN i2c_pin_name = { SCL_I2C_PINNAME, SDA_I2C_PINNAME};  //sck sda
    I2C_CONFIGURATION i2c_config = {
                                        I2C_MODE_MASTER,
                                        i2c_addr_type_7bit,
                                        I2C_Half_Time_100kbit
                                    };

    iic_struct.i2c_bus  = I2C_bus;
    iic_struct.i2c_pin  = i2c_pin_name;
    iic_struct.i2c_conf = i2c_config;


    if(iic_init(&iic_struct) == I2C_RET_OK)
        return IIC_RET_OK;
    else
        return IIC_RET_ERR;
}
```

6. IIC 写函数

IIC 写函数如下：

```
I2C_RET i2c_master_send_data(I2C_BUS  bus, uint16 addr, const uint8 *pdata, uint8
                             num_of_bytes);
```

bus: IIC 的对应接口
addr: 从机地址
pdata: 发送的数据地址
第一位数据为从机的寄存器地址
num_of_bytes: 数据的长度，包括第一位的寄存器地址
返回：配置状态

参考配置

```
if (i2c_master_send_data(bus, slave_addr,buffer, 9) != I2C_RET_OK)
   {
     return IIC_RET_ERR;
    }
```

发送 1 寄存器地址+8 个数据，所以一共 9 个数据。

7. IIC 读函数

IIC 读函数如下：

```
I2C_RET i2c_master_receive_data(I2C_BUS  bus, uint16 addr, const uint8 *pdata,
                             uint8 num_of_bytes);
```

bus: IIC 的对应接口
addr: 从机地址
pdata: 读取的数据
num_of_bytes: 数据的长度
返回：配置状态

参考配置

```
if (i2c_master_send_data(bus,slave_addr,&reg_addr,1) != I2C_RET_OK)
{
   return IIC_RET_ERR;
}

if (i2c_master_receive_data(bus, slave_addr,pData, lenght) != I2C_RET_OK)
{
   return IIC_RET_ERR;
}
```

在使用 IIC 读函数之前，必须通过写函数告诉从机要从那个寄存器地址开始读起，所以必须先发送地址再开始读数据。

8. 在使用 IIC 读写的时候需要注意的一些细节

由于海思芯片的一些问题，根据移远那边工程师的反馈，超过 16 个字节读写，IIC 容易卡死，所以建议在读写多个字节的时候八个字节来读写。

以下代码就是封装了一下，八个字节读写，代码仅供参考。

```
I2C_status iic_write_date(I2C_BUS bus,uint8 slave_addr,uint8 reg_addr, uint8 *pData, uint16 lenght)
{
    uint8 num = 0,num1 = 0,i;
    uint8 buffer[9];
    if((NULL == pData)||(lenght <= 0))
        return IIC_RET_ERR;

    num1 = lenght/8;
    for(num = 0;num < num1;num ++)
    {
        buffer[0]=reg_addr;
        for(i=0;i<8;i++)
           buffer[1+i] = pData[i];

        if (i2c_master_send_data(bus, slave_addr,buffer, 9) != I2C_RET_OK)
        {
            return IIC_RET_ERR;
        }
        reg_addr += 8;
        pData    += 8;
        lenght   -= 8;
        osDelay(100);
    }
    if(lenght)
    {
        buffer[0]=reg_addr;
        for(i=0;i<lenght;i++)
           buffer[1+i] = pData[i];

        if (i2c_master_send_data(bus, slave_addr,buffer, lenght+1) != I2C_RET_OK)
        {
            return IIC_RET_ERR;
        }
    }
    return IIC_RET_OK;
}
```

```
I2C_status iic_read_date(I2C_BUS bus,uint8 slave_addr,uint8 reg_addr, uint8 *pData, uint16 lenght)
{
    uint8 num = 0,num1 = 0;
    if((NULL == pData)||(lenght <= 0))
        return IIC_RET_ERR;
    num1 = lenght/8;
    for(num = 0;num < num1;num ++)
    {
        if (i2c_master_send_data(bus,slave_addr,&reg_addr,1) != I2C_RET_OK)//先发送要读写的寄存器地址
        {
            return IIC_RET_ERR;
        }

        if (i2c_master_receive_data(bus, slave_addr,pData, 8) != I2C_RET_OK)
        {
            return IIC_RET_ERR;
        }
        reg_addr += 8;
        pData    += 8;
        lenght   -= 8;
        osDelay(100);
    }
    if(lenght)
    {
        if (i2c_master_send_data(bus,slave_addr,&reg_addr,1) != I2C_RET_OK)//先发送要读写的寄存器地址
        {
            return IIC_RET_ERR;
        }

        if (i2c_master_receive_data(bus, slave_addr,pData, lenght) != I2C_RET_OK)
        {
            return IIC_RET_ERR;
        }
    }
    return IIC_RET_OK;
}
```

具体代码可以参考配套程序读取 MAX44009。