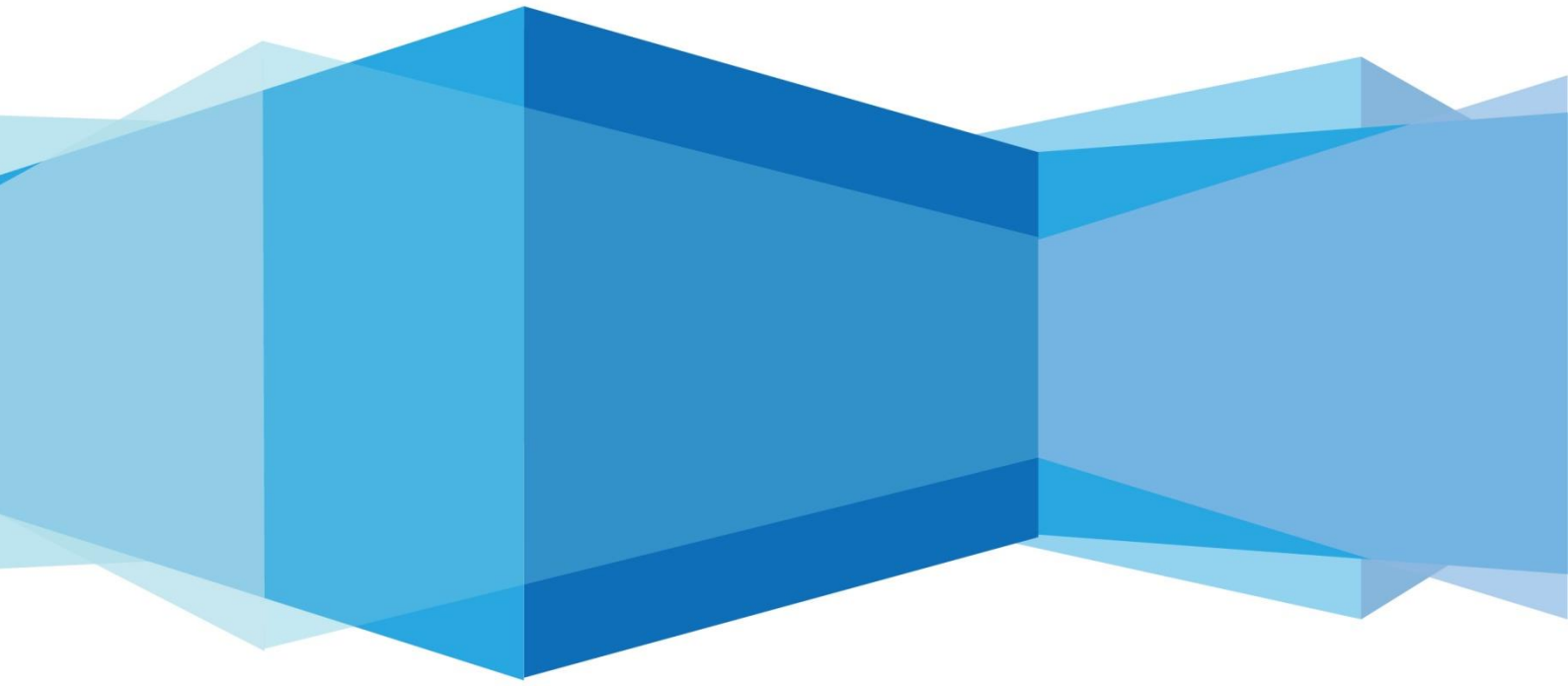
lierda 利尔达科技集团 | 证券代码 832149

Lierda NB86-G

OpenCPU_SPI 应用笔记

版本：Rev1.0

日期：2019-01-11



利尔达科技集团股份有限公司
LIERDA SCIENCE & TECHNOLOGY GROUP CO.,LTD

法律声明

若接收浙江利尔达物联网技术有限公司（以下称为“利尔达”）的此份文档，即表示您已经同意以下条款。若不同意以下条款，请停止使用本文档。

本产品符合有关环境保护和人身安全方面的设计要求，产品的存放、使用和弃置应遵照产品手册、相关合同或者相关法律、法规的要求进行。

本公司保留在不预先通知的情况下，对此手册中描述的产品进行修改和改进的权利；同时保留随时修订或收回本手册的权利。

文件修订历史

版本	修订日期	修订日志
1.0	2018-12-05	新建文档

安全须知

用户有责任遵循其他国家关于无线通信模块及设备的相关规定和具体的使用环境法规。通过遵循以下安全原则，可确保个人安全并有助于保护产品和工作环境免遭潜在损坏。我司不承担因客户未能遵循这些规定导致的相关损失。

	道路行驶安全第一！当您开车时，请勿使用手持移动终端设备，除非其有免提功能。请停车，再打电话！
	登机前请关闭移动终端设备。移动终端的无线功能在飞机上禁止开启用以防止对飞机通讯系统的干扰。忽略该提示项可能会导致飞行安全，甚至触犯法律。
	当在医院或健康看护场所，注意是否有移动终端设备使用限制。RF 干扰会导致医疗设备运行失常，因此可能需要关闭移动终端设备。
	移动终端设备并不保障任何情况下都能进行有效连接，例如在移动终端设备没有花费或 SIM 无效。当您在紧急情况下遇见以上情况，请记住使用紧急呼叫，同时保证您的设备开机并且处于信号强度足够的区域。
	您的移动终端设备在开机时会接收和发射射频信号，当靠近电视，收音机电脑或者其它电子设备时都会产生射频干扰。
	请将移动终端设备远离易燃气体。当您靠近加油站，油库，化工厂或爆炸作业场所，请关闭移动终端设备。在任何有潜在爆炸危险场所操作电子设备都有安全隐患。

目 录

1. 引言

本文旨在帮助基于使用 Lierda NB86-G 模组进行 OpenCPU 开发的用户，让其能快速使用模组本身的各种硬件资源（I2C、GPIO、UART）和 LiteOS 操作系统（创建、删除、挂起、恢复线程。创建、删除、启动、停止软件定时器），文章概述了 openCPU 硬件 SPI 和软件 SPI 特性、软硬件参考设计、样例说明以及设计要点。

2. NB86-G 引脚说明

VDD_IO_R1、 VDD_IO_R2、VDD_IO_L1、VDD_IO_L2 四个电源域的 DC 特性都相同，其中[PIO0~PIO9]从属 VDD_IO_R1 电源域，[PIO10~PIO21]从属 VDD_IO_R2 电源域，[PIO22~PIO33]从属 VDD_IO_L1 电源域., [PIO34~PIO39]从属 VDD_IO_L2 电源域，在选择 PIO 时注意电源域的选择，如图 2-1.

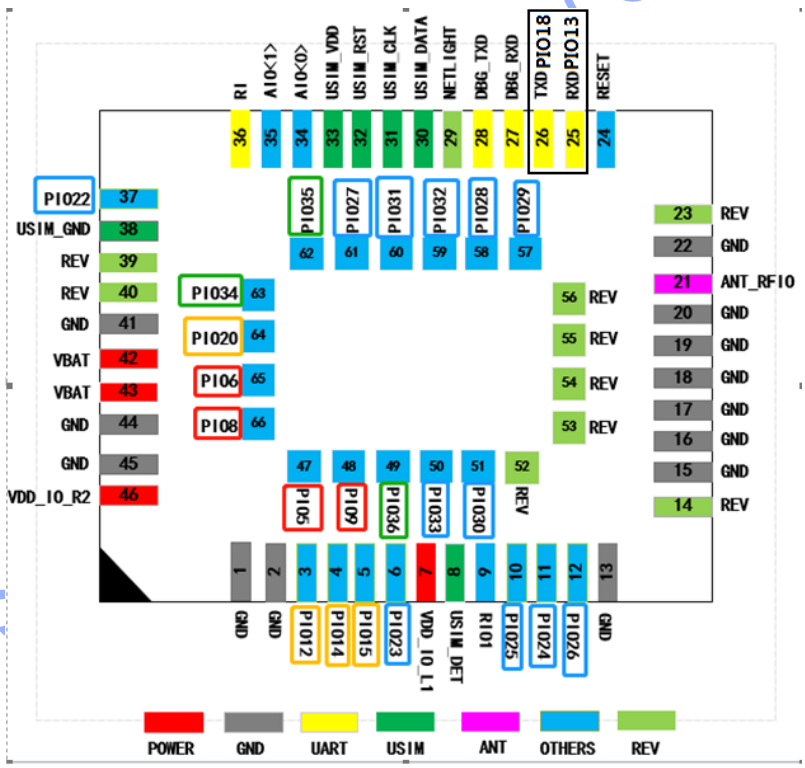


图 2-1 NB86-G 电源域图

说明：

红色框起来的 PIO 属于 VDD_IO_R1 电源域；黄色框起来的 PIO 属于 VDD_IO_R2 电源域；蓝色框起来的 PIO 属于 VDD_IO_L1 电源域；绿色框起来的 PIO 属于 VDD_IO_L2。

VDD_IO_R1 电源域：1.8V 电平；

VDD_IO_L2 电源域：由于 USIM 卡的 IO 是从属于 VDD_IO_L2 电源域的，所以该电源域的电平取决于外接 SIM 卡的电平。

VDD_IO_R2、VDD_IO_L1 电源域：默认值 3.0V。

NOTE:应定义 SPI 的 SCK、CS、MOSI、MISO 引脚在同一电源域，且注意 1.8V 电源和外部 sensor 电平的匹配问题。

3. 硬件 SPI

3.1. 硬件 SPI 特性说明

硬件 SPI 可在深度睡眠模式下运行，也可以在活动和待机模式下运行，发送/接收完数据后反初始化一下功耗会降至 PSM 正常功耗（3μA 左右），具体的参数配置如下表：

参数	取值范围
可用 SPI 数	1
速率	738-2M Hz（Master mode）
	738-2M Hz（Slave mode）
数据位	4—16
模式	Master mode、Slave mode
引脚定义	同一个电源域的通用 GPIO
是否影响 PSM 功耗	是，反初始化后 PSM 功耗恢复正常

3.2. 硬件参考电路

建议使用 R2 电源域所在的 IO 口，其它电源域所在 IO 不建议使用。

如果终端 SPI 电平不匹配时需要增加电平转换电路，可参考下图设计。SPI 引脚需要选择同一个电源域的通用 GPIO。Terminal_VCC 上拉电平需要和终端串口电平保持一致，硬件参考设计图如图 3-1.

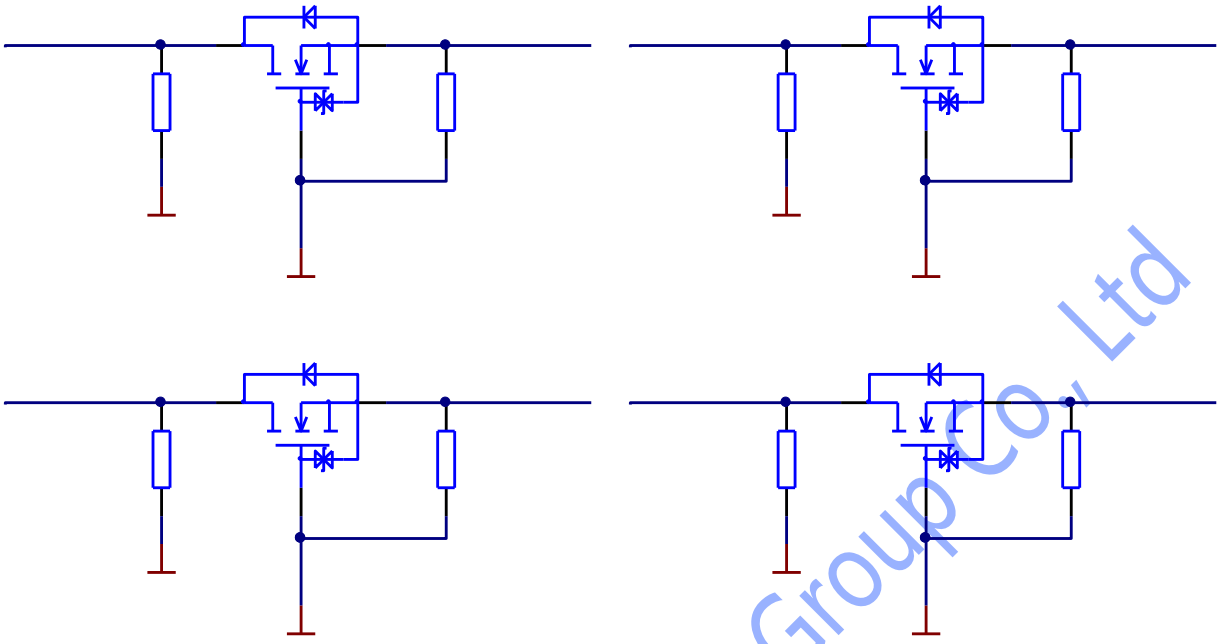


图 3-1 SPI 硬件参考设计

3.3. 软件参考设计

3.3.1. SPI 初始化

SPI_RET lierdaSPIInit(SPI_CONFIGURATION spi_config, SPI_PIN spi_pin);

参数解释：

```
typedef struct
{
    uint8          data_size;  //数据位
    uint8          clk_div;    //分频，调节SPI通信速率
    SPI_CLK_MODE   clk_mode;   //通信的时钟模式（0,1,2,3）
    }SPI_CONFIGURATION;
typedef struct
{
    SPI_INTERFACE interface;   //填写SPI_INTERFACE_SINGLE_UNIDIR即可
    PIN           clk_pin;  //SPI CLK引脚
    PIN           csb_pin;   //SPI CS引脚
    PIN           mosi_pin;  //SPI MOSI引脚
    PIN           miso_pin;  //SPI MISO引脚
    }SPI_PIN;
```

返回值：成功： SPI_RET_OK 失败：SPI_RET_ERROR

3.3.2. SPI 写数据

SPI_RET lierdaSPISendData(SPI_BUS bus, uint8* cmd_buff,uint16 cmd_len, uint8* data_buff, uint16 data_len, SPI_CALLBACK callback);

参数解释：

Bus：SPI 总线选择，一般选择 0 即可

cmd_buff：命令 buffer，对于某些不需要的设备使用时填 NULL，对应的"cmd_len"也填 0

cmd_len：命令长度

data_buff：写入的数据

data_len：写入的数据长度

callback：回调函数

返回值:若成功，返回 SPI_RET_OK 若失败，返回 SPI_RET_ERROR

3.3.3. SPI 读数据

SPI_RET lierdaSPISendData(SPI_BUS bus, uint8* cmd_buff,uint16 cmd_len, uint8* data_buff, uint16 data_len, SPI_CALLBACK callback);

参数解释：

bus:声明的 SPI 总线

cmd_buff：命令 buffer，对于某些不需要的设备使用时填 NULL，对应的"cmd_len"也填 0

cmd_len：命令长度

data_buff：写入的数据

data_len：写入的数据长度

callback：回调函数

ignore_rx_while_tx：当发送时，是否忽略接收。一般填 TRUE

返回值:若成功，返回 SPI_RET_OK 若失败，返回 SPI_RET_ERROR

3.3.4. SPI 反初始化

SPI_RET lierdaSPIDeinit(void);

参数:无

返回值:若成功，返回 SPI_RET_OK 若失败，返回 SPI_RET_ERROR

3.4. 测试样例

样例操作 FLASH 芯片（MX25L12835F），对芯片 ID 进行读取，进行读写数据。

3.4.1. SPI 初始化

```
SPI_CONFIGURATION lierdaSPIconfig;
SPI_PIN lierdaSPIpin;
lierdaSPIconfig.data_size = 8;
lierdaSPIconfig.clk_mode = SPI_CLK_MODE3;
lierdaSPIconfig.clk_div = 0x08;
lierdaSPIpin.interface = SPI_INTERFACE_SINGLE_UNIDIR;
lierdaSPIpin.clk_pin = SPI_SCK;
lierdaSPIpin.csb_pin = SPI_CS;
lierdaSPIpin.miso_pin = SPI_MISO;
lierdaSPIpin.mosi_pin = SPI_MOSI;
if (lierdaSPIInit(lierdaSPIconfig, lierdaSPIpin) == SPI_RET_OK)
    lierdaLog("SPI  Init  OK");
else
        lierdaLog("SPI  Init  ERROR");
```

NOTE: clk_div = 0x02 表示速率为 2M Hz

clk_div = 0x04 表示速率为 1M Hz

clk_div = 0x08 表示速率为 500k Hz ……

3.4.2. SPI 写数据

```
uint8 writaddr[5]={0};uint8 cmd_PageProgram=W25X_PageProgram;
writaddr[0]=cmd_PageProgram;
writaddr[1]=(WriteAddr & 0xFF0000)>>16;
writaddr[2]=(WriteAddr & 0xFF00)>>8;
writaddr[3]=WriteAddr & 0xFF;
SPI_FLASH_WriteEnable();
lierdaSPISendData(0, writaddr,4, pBuffer, NumByteToWrite, NULL);
SPI_FLASH_WaitForWriteEnd();
```

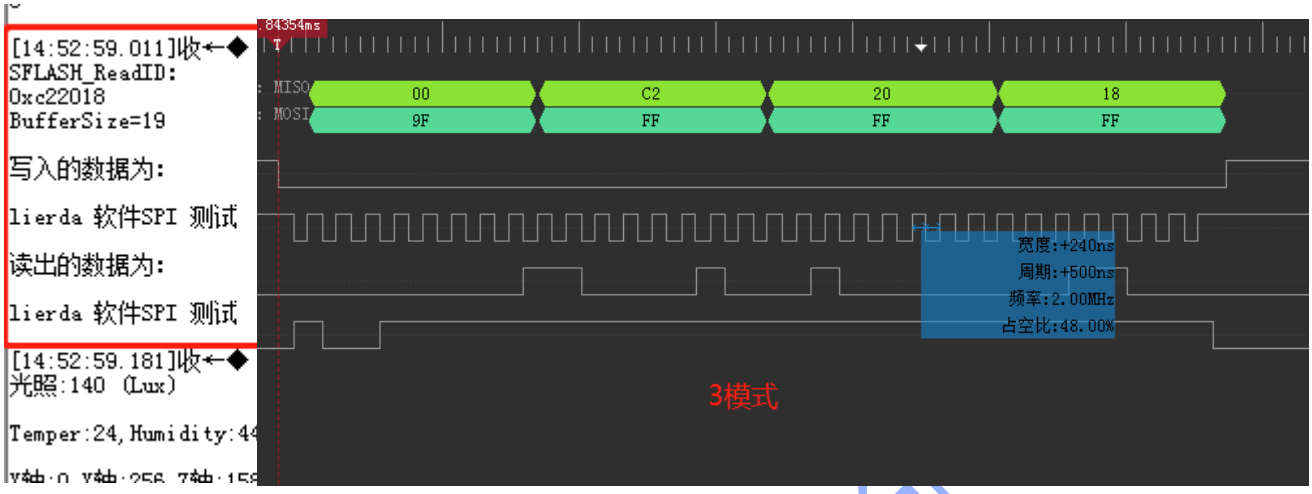
3.4.3. SPI 读数据

```
uint32 Temp = 0;
uint8 idbuff[4]={0};uint8 cmd_JedecDeviceID=W25X_JedecDeviceID;
```

lierdaSPIRecvData(0,&cmd_JedecDeviceID,1,idbuff, 3,NULL,true);

Temp = (idbuff[0] << 16) | (idbuff[1] << 8) | idbuff[2];

3.4.4. 样例测试结果



4. 软件 SPI

4.1. 软件 SPI 特性说明

软件模拟 SPI 可在深度睡眠模式下运行，也可以在活动和待机模式下运行，发送/接收完数据后反初始化一下功耗会降至 PSM 正常功耗（3μA 左右），具体的参数配置如下表：

表 4-1 SPI 参数配置

参数名	取值范围	备注
速率	62kHz	
数据位	8	
时钟模式	0,1,2,3	
引脚定义	同一电源域的通用 PIO	注意 1.8V 电源和外部传感器电平匹配问题

4.2. 硬件参考设计

参考 3.2 小节。

4.3. 软件参考设计

4.3.1. SPI 初始化

SPI 初始化函数：uint8 lierdaSPISoftInit(SPI_InitTypeDef *SPI_Init);

参数说明：SPI_InitTypeDef *SPI_Init：SPI 初始化配置结构体，用于配置 SPI 基本参数。

```
uint32 Mode;              /*指定 SPI 操作模式，四种模式：0,1,2,3 */
uint32 DataSize;          /*指定 SPI 数据位长度，目前只支持 8 位 */
PIN LierdaSPI_CS;         /*定义 SPI  CS 引脚 */
PIN LierdaSPI_SCK;        /*定义 SPI  SCK 引脚 */
PIN LierdaSPI_MISO;       /*定义 SPI  MISO 引脚 */
PIN LierdaSPI_MOSI;       /*定义 SPI  MOSI 引脚 */
```

返回值：成功： LierdaSPI_RET_OK 失败：LierdaSPI_RET_ERROR

4.3.2. SPI 写一个字节

SPI 读取一个字节函数：uint8 lierdaSPIWriteByte(SPI_InitTypeDef *lierda_spi,uint8 TxData);

参数说明：SPI_InitTypeDef *lierda_spi： SPI 初始化结构体地址；

uint8 TxData：发送数据。

返回值：成功： LierdaSPI_RET_OK 失败：SPI_RET_ERROR

4.3.3. SPI 读一个字节

SPI 读一个字节函数：uint8 lierdaSPIReadByte(SPI_InitTypeDef *lierda_spi);

参数说明：SPI_InitTypeDef *lierda_spi：SPI 初始化结构体地址；

返回值：读到的数据

4.3.4. SPI 反初始化函数

SPI 反初始化函数：void Lierda_SPIDeInit(SPI_InitTypeDef *SPI_Init);

参数说明：SPI_InitTypeDef *lierda_spi：SPI 初始化结构体地址；

返回值：无

4.4. 测试样例

样例操作 FLASH 芯片（MX25L12835F），对芯片 ID 进行读取，进行读写数据。

4.4.1.SPI 初始化

实例代码如下：

```
/*********************************************
函数名称 ： MX25L12835F_spiInit
功    能 ： SPI初始化
参    数 ： 无
返 回 值 ： 无
*********************************************/
void MX25L12835F_spiInit(void)
{
    SPI_MX25L12835F.DataSize=8;
    SPI_MX25L12835F.Mode=0;
    SPI_MX25L12835F.LierdaSPI_CS=SPI_CS;
    SPI_MX25L12835F.LierdaSPI_SCK=SPI_SCK;
    SPI_MX25L12835F.LierdaSPI_MISO=SPI_MISO;
    SPI_MX25L12835F.LierdaSPI_MOSI=SPI_MOSI;

    if (lierdaSPISoftInit(&SPI_MX25L12835F) == LierdaSPI_RET_OK)
        lierdaLog("SPI  Init  OK");
    else
        lierdaLog("SPI  Init  ERROR");
  }
```

4.4.2.SPI 读/写一个字节数据

实例代码如下：

```
/*********************************************
函数名称 ： SFLASH_ReadID
功    能 ： 读取芯片ID SFLASH的ID
参    数 ： 无
返 回 值 ： ID --- 32位ID号
*********************************************/
uint32 SFLASH_ReadID(void)
{
 uint32 Temp = 0, Temp0 = 0, Temp1 = 0, Temp2 = 0;
  SPI_CS_ENABLE();                              //使能器件

  lierdaSPIWriteByte(&SPI_MX25L12835F,W25X_JedecDeviceID);   //发送一个字节数据
《设备ID》指令

  Temp0= lierdaSPIReadByte(&SPI_MX25L12835F);//读一个字节数据
```

```
  Temp1= lierdaSPIReadByte(&SPI_MX25L12835F);//读一个字节数据

  Temp2= lierdaSPIReadByte(&SPI_MX25L12835F);//读一个字节数据

   SPI_CS_DISABLE();  //失能器件
  Temp = (Temp0 << 16) | (Temp1 << 8) | Temp2;

  return Temp;
  }
```

4.4.3.SPI 反初始化

实例代码如下：

```
/*******************************************************************************
* @函数名   SPI测试任务线程
* @参数param : 空参数
* @返回值   无
*******************************************************************************/
void lierda_App_task(void *param)
{
    UNUSED(param);
    osDelay(500); //等待模组初始化完成
    lierdaLog("DBG_INFO:Demo测试程序");  //通过AT指令串口打印Log测试
    for (;;)
    {
        MX25L12835F_spiInit();//MX25L12835F初始化（SPI测试）
        Lierda_SPITest();//SPI测试
        Lierda_SPIDeInit(&SPI_MX25L12835F);//反初始化SPI
        osDelay(5000);
    }
  }
```

4.4.4.样例测试结果

1. 测试 LOG 如图 5-1.

```
[13:59:56.813]收←◆
SPI  Init  OK

SFLASH_ReadID:
0xc22018
BufferSize=19

写入的数据为:

lierda 软件SPI 测试

读出的数据为:

lierda 软件SPI 测试

[14:00:01.932]收←◆
SPI  Init  OK

SFLASH_ReadID:
0xc22018
BufferSize=19

写入的数据为:

lierda 软件SPI 测试

读出的数据为:

lierda 软件SPI 测试
```

图 4-1 SPI 测试 LOG

2、逻辑分析仪波形分析如图 5-2.传感器时序如图 5-3.

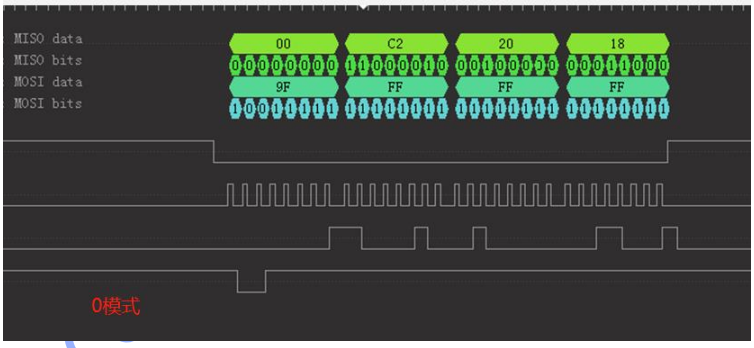


图 4-2 逻辑分析仪波形

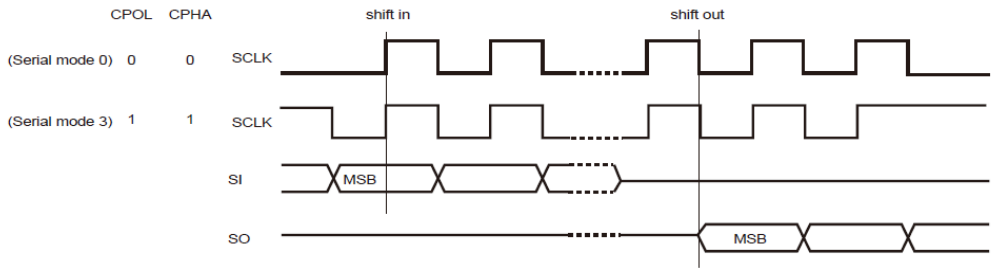


图 4-3 传感器时序

3、功耗测试

SPI 发送和接收数据的平均功耗：3.39mA,发送或接收完数据若反初始化 SPI，功耗会恢复至 PSM 正常功耗（3μA 附近）。如图 5-4.

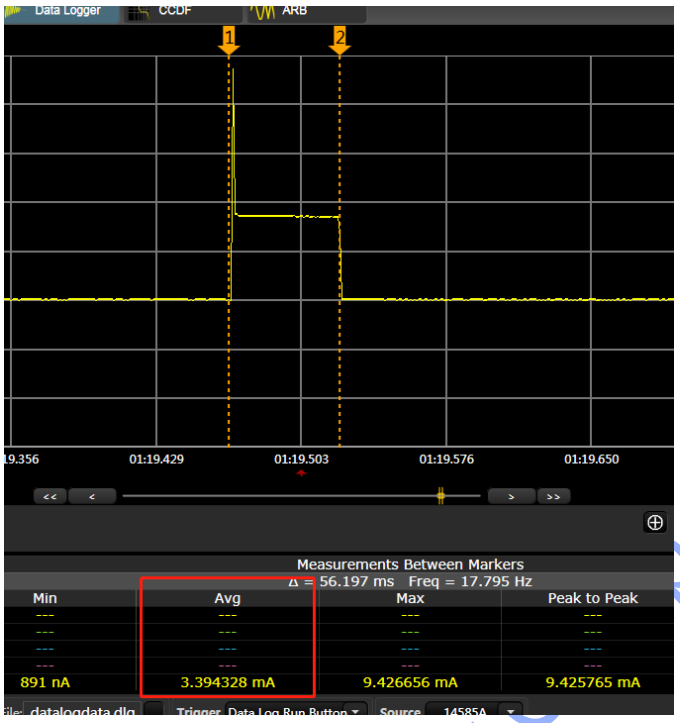


图 4-4 SPI 发送/接收数据功耗

5. 设计要点

- 定义的 SPI CS、SCK、MISO、MOSI 引脚在同一电源域
- 定义的 SPI 引脚电压和外部 sensor 引脚电平匹配

6. 相关文档及术语缩写

以下相关文档提供了文档的名称，版本请以最新发布的为准。

表格 6-1 相关文档

序号	文档名称	注释
[1]	NB86-G硬件应用手册	
[2]	Lierda NB Module V150_AT CommandSet_B300SP2	
[3]	Lierda NB86-EVK测试终端固件烧写教程	
[4]	Lierda NB-IoT模组API使用文档	
[5]	Lierda NB-IoT模组DEMO说明文档	
[6]	Lierda NB-IoT模组V150 OpenCPU开发环境搭建指南	

[7] Lierda NB86-EVK操作使用手册