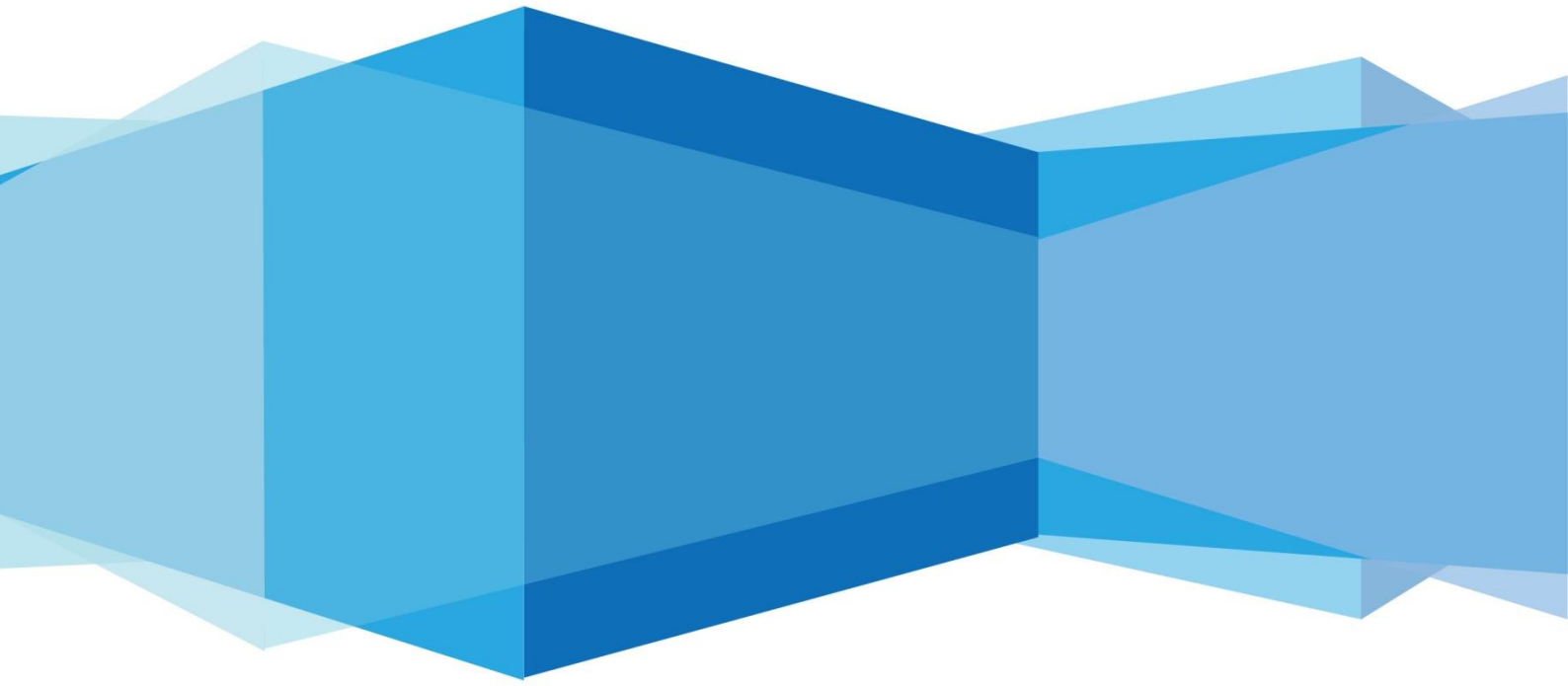
lierda 利尔达科技集团 | 证券代码 832149

Lierda NB86-G

OpenCPU_GPIO 应用笔记

版本：Rev1.0

日期：2018-12-15



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	2018-09-27	新建文档

适用模块型号

序号	模块型号	模块简介
1	NB86-G	全频段版本，20×16×2.2（mm）
2	NB86-G 宽压型	全频段版本，20×16×2.2（mm）

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_GPIO 的特性、软硬件参考设计、样例说明以及设计要点。

2. 特性概述

本设备上最多有 40 个 PIO，其中 24 个 PIO 可用于应用程序核心。对于每个 PIO，都有 IO 引脚功能的软件控制：方向，中断配置，驱动强度以及集成上拉和下拉电阻的使能。OpenCPU 方案留给客户可用的 PIO 为 17 个。

24 个可用的 PIO 中，每个 PIO 都可以分配给任何数字外设。在启动时，安全核和协议核心声明了有 40 个可用的 PIO，应用程序可以使用协议核和安全核未使用的 PIO。下表列出了可以从应用核映射到 PIO 的数字外设和接口，包括其相关的信号名称和所需的 PIO 数量。除了 UICC（SIM）接口外，下表中的所有外设都可供应用处理器使用。未分配给特定外设的 PIO 可用作通用 IO。

Peripheral Type	No. of PIOs required	Signal Names
SSP x2	4	SPI_SCLK
		SPI_MOSI
		SPI_MISO
		SPI_CSB
I2C x2	2	I2C_SCL
		I2C_SDA
UART	2 - 4	UART_TX
		UART_RX
		UART_CTS
		UART_RTS
UICC (Note 1)	3 or	USIM_RESET
		USIM_SIO
		USIM_SCLK
		USIM_PWR –
		VDD_IO_L2
		SIM_DET
		(optional)
PWM	1	PWM_OUT

Table 1 可以映射到 PIO 的数字外设（和相关信号）

这些 PIO 可用于所有操作模式。在活动和待机期间，数据采样与系统时钟同步，在这些模式中，中断是同步生成的。在深度睡眠期间，数据采样同步到 RTC 时钟，在此模式下，异

步生成中断。在所有模式中，中断可配置为在上升沿，下降沿，电平 HI 或电平 LO 上触发。PIO 可配置为高驱动强度或低驱动强度（默认），并具有可通过软件启用的集成上拉和下拉电阻。

PIO 分为四个电源域，每个域的电源由片内 LDO 或应用电路的外部输入提供。

3. 硬件参考设计

使用 I/O 口做应用时首先要注意电压匹配问题，如果 I/O 口电压和外设电压不匹配时建议增加电平转换电路使用，否则会造成设备不能稳定工作甚至损坏！

因为 I/O 的驱动能力有限，驱动电流不应超过 10mA。

R1 电源域下面的 I/O 由于在 PSM 状态下功能失效，所以不建议使用。

R2 电源域下面的 I/O 可以正常使用，如表 3-1。

L1 电源域下面的 I/O 可以正常使用，如表 3-1。

L2 电源域下面的 I/O 由于只在读取 SIM 卡时生效，所以不建议使用。

表 3-1

电源域	可用 PIO
VDD_IO_L1 电源域	PIO22、PIO23、PIO24、PIO25、PIO26、PIO27、PIO28、PIO29、PIO30、PIO31、PIO32、PIO33
VDD_IO_R2 电源域	PIO12、PIO14、PIO15、PIO20

4. 软件参考设计

```
#include "lierdaGPIO.h"
bool Vlaue;

lierdaGPIOInit();

Vlaue=lierdaGPIOClaim(PIN_25, GPIO_DIRECTION_OUTPUT);//声明 IO 输入
lierdaGPIOSet(PIN_25);//拉高电平
lierdaGPIOClear (PIN_25);//拉低电平
```

```
lierdaGPIORead(PIN_25);//读取当前电平状态
lierdaGPIOToggle(PIN_25);//翻转电平

/*下方代码实现案件中断*/
void Lierda_KEY_Init(PIN key_pin, GPIO_DIRECTION mode)
{
    lierdaGPIOClaim(key_pin, mode);   //GPIO 声明
    lierdaGPIORegisterCallback(key_pin, GPIO_INTERRUPT_LOW,sos_key_callback);
    //按键中断初始化
}
/*按键的中断回调函数，当按键触发时，发送消息队列*/
void sos_key_callback(PIN pin)
{
    if (0==lierdaGPIORead(pin))
    {
        osDelay(10);   //消抖
        if (lierdaGPIORead(pin)==0)   //GPIO 读取函数
        {
            while(lierdaGPIORead(pin)==0);
            osMessageQueuePut(mess_QueueId, &KEY_flag, 0,osNoWait);//队列发送消息
        }
    }
}
```

5. 测试样例

5.1. 翻转一次 IO 产生的最大电流

```
for(;;)
{
    if (osMessageQueueGet(mess_QueueId, &temp, NULL, 0xffffffff) == osOK)
    {//队列接收按键中断发送的消息
        for(uint8 count = 0;count < 3;count ++)
```

```
        {
            lierdaGPIOToggle(PIN_33);
            osDelay(1);
        }
    }
    osDelay(1);//用于任务切换
    lierdaGPIOToggle(LED_0);
}
```

如图所示，检测到在延时为 1ms 的情况下，翻转 3 次 PIO33，每次翻转产生的最大电流大致为 7.2mA。

图表 1 翻转 1 次 IO 电流

5.2. 连续翻转 IO100 次产生的平均电流

```
for(;;)
{
    if(osMessageQueueGet(mess_QueueId, &temp, NULL, 0xffffffff) == osOK)
    {//队列接收按键中断发送的消息
        for(uint8 count = 0;count < 100;count ++)
        {
            lierdaGPIOToggle(PIN_33);
```

```
        }
    }
    osDelay(1);//用于任务切换
    lierdaGPIOToggle(LED_0);
}
```

如图所示，连续无延时地翻转 100 次 PIO33 产生的平均电流大约为 3.1mA。

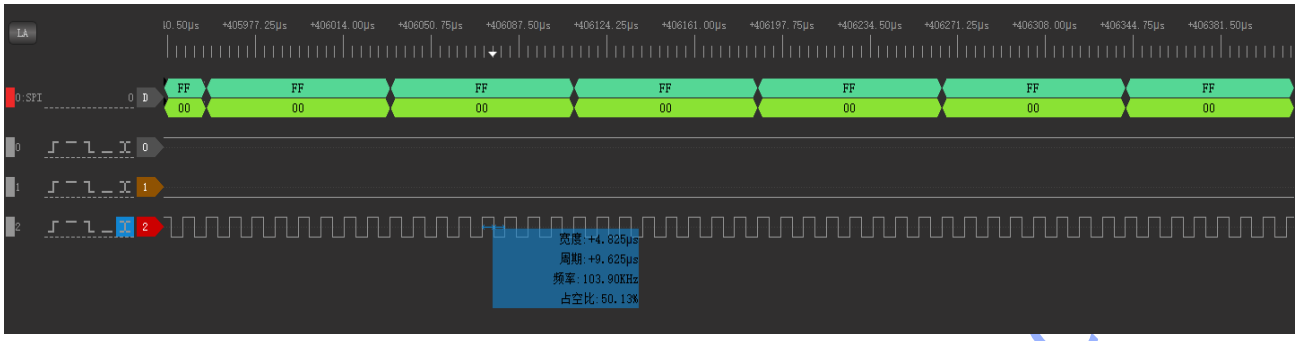
图表 2 连续翻转 100 次 IO 平均电流

5.3. 连续翻转 IO 的极限频率

```
for(;;)
{
    if (osMessageQueueGet(mess_QueueId, &temp, NULL, 0xffffffff) == osOK)
    {//队列接收按键中断发送的消息
        while(1)
        {
            lierdaGPIOToggle(PIN_33);
        }
    }
    osDelay(1);//用于任务切换
    lierdaGPIOToggle(LED_0);
```

```
}
```

如图所示，在任务中，无延时地翻转 PIO33，翻转一次的时间是 4.825us，频率为 103.90khz。



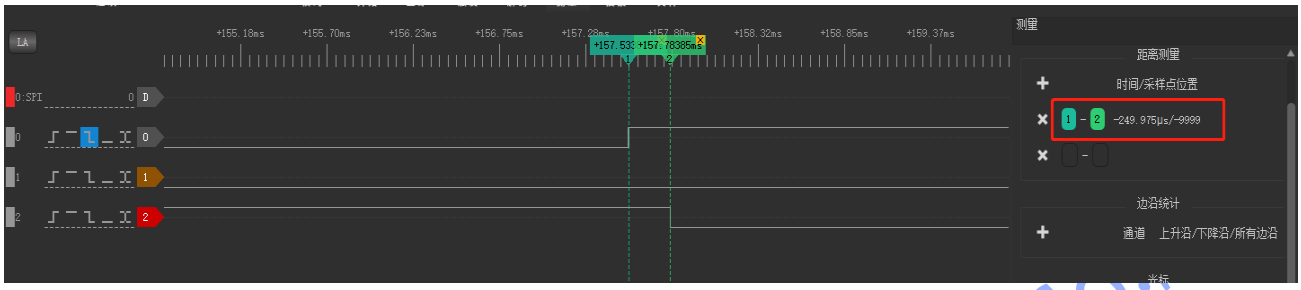
图表 3 连续翻转 IO 频率

5.4. 测试外部中断响应时间

```
//按键中断函数，用于发送队列
void sos_key_callback(PIN pin)
{
    if(0==lierdaGPIORead(pin))
    {
        osDelay(10);  //消抖
        if(lierdaGPIORead(pin)==0)  //GPIO读取函数
        {
            while(lierdaGPIORead(pin)==0);
            //队列发送消息
            osMessageQueuePut(mess_QueueId, &KEY_flag, 0,osNoWait);
        }
    }
}
//按键响应的任务
for(;;)
{
    if(osMessageQueueGet(mess_QueueId, &temp, NULL, 0xffffffff) == osOK)
    {//队列接收按键中断发送的消息
        lierdaGPIOToggle(PIN_33);
    }
    osDelay(1);//用于任务切换
    lierdaGPIOToggle(LED_0);
}
```

如图所示，当按键 IO 产生上升沿电平，将触发中断服务函数中的消息队列，随之 PIO33 将翻转，其间的时间间隔大约为 249.975us。

NOTE：经测试，将翻转动作置于中断函数中，与以上方式响应时间大致相同。



图表 4 IO 中断响应时间

6. 设计要点

1、 初始化函数，初始化所有的 GPIO，默认状态为浮空。

2、 声明函数，为具体的 IO 口分配方向，目前不支持芯片内对 IO 上拉。

3、 在对 IO 进行操作之前，需完成如下流程：

a) lierdaGPIOInit();

b) lierdaGPIOClaim();

7. 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 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