T6 AILINK 广播体脂秤应用手册

版本：V0.2

更新日期：2023 年 11 月 16 日

本产品的规格书如有变更，恕不另行通知。

深圳市易连物联网有限公司保留在不另行通知的情况下，对其中所包含的规格书和材料进行更改的权利，同时由于信任所引用的材料所造成的损害（包括结果性损害），包括但不限于印刷上的错误和其他与此出版物相关的错误，易连物联网将不承担责任。

深圳市易连物联网有限公司
电话：（86）0755-81773367　　邮箱：hw@elinkthings.com
地址：深圳市宝安区西乡街道银田工业区侨鸿盛文化创意园写字楼 A 栋五层 502 室　　邮编：518000

修改记录

文档版本	作者	审核	发布日期	修改说明
V0.1	LYX	Lxl	2023/8/28	初版
V0.2	LYX	Lxl	2023/11/16	修改部分描述

深圳市易连物联网有限公司
电话：（86）0755-81773367　　邮箱：hw@elinkthings.com
地址：深圳市宝安区西乡街道银田工业区侨鸿盛文化创意园写字楼 A 栋五层 502 室　　邮编：518000

目录

修改记录 ..... - 2 -
目录 ..... - 3 -
1 概述 ..... - 4 -
2 说明 ..... - 4 -
3 模块版本 ..... - 4 -
4 硬件参考设计 ..... - 5 -
4.1 SPI 通信(三线通信) ..... - 5 -
4.2 SPI 时序 ..... - 5 -
5 通信协议 ..... - 6 -
5.1.1 通信流程 ..... - 6 -
6 生产测试指导 ..... - 12 -
7 联系我们 ..... - 12 -
8 附件 ..... - 13 -

1 概述

1.1 本文档适用于深圳市易连物联网 T6 系列蓝牙模块 接入 ailink APP。

1.2 本文档适用于体脂秤的 MCU 端开发工程师使用。

1.3 本文档讲详细介绍硬件对接、固件对接。

1.4 文档会保持更新，以官网链接为最新版本。

2 说明

2.1 我们提供标准化的连接模块、app、云平台帮助客户的体脂秤快速实现智能化，并提供 sdk、云平台配置、增值服务和技术支持帮忙客户差异化、个性化。

2.2 我们提供的蓝牙模块具有功耗低、认证齐全、APP 功能强大体验好等特点。扫描下面二维码下载 APP。

（AIlink APP）

2.3 支持 MCU 配置模块（VID、PID）实现 APP 连接产品时型号自定义、图标自定义等个性化设计(后台获取 VID、PID 说明:http://doc.elinkthings.com/web/#/40?page_id=144)。

3 模块版本

T6 芯片

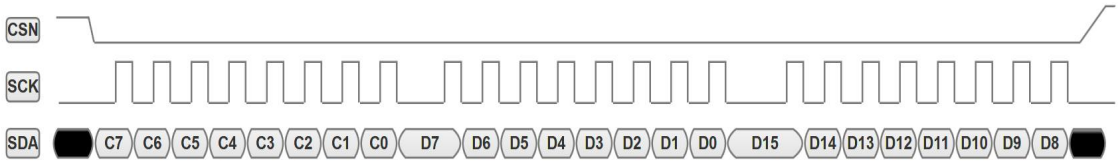
4 硬件参考设计

4.1 SPI 通信(三线通信)

CSN	SPI chip select, low active
SCK	SPI clock
SDA	SPI data

4.2 SPI 时序

1. SPI 读写时序



读写时序

备注(通信协议里面使用):
(1) SPI_write_1byte(地址, 内容);
//spi 写一个字节的命令,例如 SPI_write_1byte(0x3D , 0x20); 往 0x3D 地址写入 0x20;

(2) sta = SPI_read_1byte(地址) ;
//spi 读一个字节的命令,sta 为读到的 8bit 值 , 例如 sta = SPI_read_1byte(0x06) ;读 0x06 地址里面的值,读出来的值赋给 sta .

(3) SPI_write_nbyte(地址,长度);
//spi 写 n 个字节的命令,例如 char data[3] ; SPI_write_nbyte(0x23 ,0x03); 往 0x23 地址写入 3 个 byte 的内容,具体的内容是 data 里面的值;

(4) SPI_write_op(地址);
//spi 写一个字节的地址,例如 SPI_write_op(0xE1),直接写地址即可,不带内容数据,写完这个地址即可写操作.

注意:无论是发多个 byte 还是 1byte,都需要遵循 spi 时序图.

5 通信协议

5.1.1 通信流程

1. 模块上电
   设置模式进入工作模式(模块休眠后,再起来工作时,需要进行设置)

```
char sta;
SPI_write_1byte(0x20, 0x83);
delay_us(1000);
sta=SPI_read_1byte( 0x03 ) ;
sta =    sta &0xfc;
SPI_write_1byte( 0x23, sta);
```

2. 初始化
   模块断电起来时,需要配置.

```
char sta;//读寄存器时回来的数据
char data[6];//写多字节数据用到的数据
SPI_write_1byte( 0x3D , 0x20 );
SPI_write_1byte( 0x3C , 0x00 );
SPI_write_op(0xD6);
SPI_write_1byte( 0x20 , 0x8B );
delay_ms(3);                          //延时 3 毫秒
SPI_write_1byte( 0x23 , 0x8C );
delay_ms(2);                          //延时 2 毫秒
SPI_write_1byte( 0x25 , 0x28 );

sta=SPI_read_1byte( 0x07 ) ;
if( !(sta &0x80) )          //读回来的值的 Bit7 为 0 时,则需要写 1byte 数据(切 bank1)
{
    SPI_write_1byte( 0x50 , 0x53 );
}
SPI_write_1byte( 0x3F , 0x20 );
SPI_write_1byte( 0x33 , 0x01 );
SPI_write_1byte( 0x2C , 0x25 );

sta=SPI_read_1byte( 0x07 ) ;
if( sta &0x80)          //读回来的值的 Bit7 为 1 时,则需要写 1byte 数据(切 bank0)
{
    SPI_write_1byte( 0x50 , 0x53 );
}
```

```
SPI_write_1byte( 0x3D , 0x10 );
SPI_write_op(0xD5);
delay_us(300);                          //延时 300us
SPI_write_op(0xD6);

while(1)//死循环
{
    sta=SPI_read_1byte( 0x06 ) ;
    if( (sta&0x20) != 0x00)   //读回来的值的 Bit5 不为 0 时
    {
        break;//则退出死循环
    }
}

sta=SPI_read_1byte( 0x06 ) ;
sta = sta & 0xEF;
SPI_write_1byte( 0x26, sta);

SPI_write_1byte( 0x3D, 0x48);
SPI_write_1byte( 0x3F, 0x67);

sta=SPI_read_1byte( 0x07 ) ;
if( !(sta &0x80) )                  //读回来的值的 Bit7 为 0 时,则需要写 1byte 数据(切 bank1)
{
    SPI_write_1byte( 0x50 , 0x53 );
}

data[0] = 0x20;
data[1] = 0x98;
data[2] = 0x75;
SPI_write_nbyte( 0x23, 0x03); //写三个数据到

SPI_write_1byte( 0x3E, 0x17);

sta=SPI_read_1byte( 0x07 ) ;
if( sta &0x80) //读回来的值的 Bit7 为 1 时,则需要写 1byte 数据(切 bank0)
{
    SPI_write_1byte( 0x50 , 0x53 );
}
SPI_write_1byte( 0x3D, 0x40);
SPI_write_1byte( 0x3F, 0x76);
SPI_write_1byte( 0x21, 0x00);
SPI_write_1byte( 0x38, 0x1D);
```

```
SPI_write_1byte( 0x33, 0xF0);
```

//设置设备 mac 地址(这个 mac 地址一定要设置,且需要每台设备都不一样,是设备的唯一码. 下面举例设置 mac 地址为 01:02:03:07:08:09)

```
data[0] = 0x01;
data[1] = 0x02;
data[2] = 0x03;
data[3] = 0x07;
data[4] = 0x08;
data[5] = 0x09;
SPI_write_nbyte( 0x30, 0x06);

data[0] = 0x55;
data[1] = 0x55;
data[2] = 0x55;
SPI_write_nbyte( 0x32, 0x03);

data[0]=0xd6;
data[1]=0xbe;
data[2]=0x89;
data[3]=0x8e;
SPI_write_nbyte( 0x2A,0x04);

SPI_write_1byte( 0x2C, 0x03);

SPI_write_1byte( 0x20, 0x8a);// power up
/*初始化完成*/
```

3. 更新待发送的数据(体脂秤功能)

char sData[31];//蓝牙数据数据共 31 个字节

sData[0]	sData[1]	sData[2]	sData[3]	sData[4]	sData[5]	sData[6]	sData[7]
0x03(固定)	0x03(固定)	0xA0(固定)	0xF0(固定)	0x04(固定)	0x09(固定)	0x45(固定)	0x6c(固定)
sData[8]	sData[9]	sData[10]	sData[11]	sData[12]	sData[13]	sData[14]	sData[15]
0x69(固定)	0x15(固定)	0xFF(固定)	0x09(固定)	VID(申请)	PID(申请)	Mac[5]	Mac[4]
sData[16]	sData[17]	sData[18]	sData[19]	sData[20]	sData[21]	sData[22]	sData[23]
Mac[3]	Mac[2]	Mac[1]	Mac[0]	校验和	流水号	测量标识	数据属性
sData[24]	sData[25]	sData[26]	sData[27]	sData[28]	sData[29]	sData[30]	
体重高字节	体重低字节	阻抗高字节	阻抗低字节	算法 ID	温度高字节	温度低字节	

备注:

(1) TEA 加密:

① sData[21]到 sData[28]的内容格式如上表所述.

② sData[21]到 sData[28]的共 8byte 数据需要进行 TEA 加密,加密后会得到新的 8byte 数据,再重新放回到 sData[21]到 sData[28]里面.

③ TEA 加密函数在文档最后的附件里面,TEA 秘钥在我司申请获取.

④ TEA 加密出错,APP 无法正确解析数据.

(2) 固定的值的内容不能变动

(3) VID,PID 值需要在我司官网申请获取.

(4) Mac 值:和第一步初始化 mac 地址值是同一个值,例如前面初始化值为 mac 地址为 01:02:03:07:08:09，则这里的 mac[5]=0x09,mac[4]=0x08,mac[3]=0x07,mac[2]=0x03,mac[1]=0x02,mac[0]=0x01;

(5) 校验和: sData[21]到 sData[30]的累加和(注意:sData[21]到 sData[28]用加密后的数据进行累加,即先加密再累加)

(6) 流水号 ： 当数据有变化时,流水号的值需要加 1. 数据没更新时,流水号值不需变化 ,默认从 0 开始.

(7) 测量标识(sData[22])

0x00 ： 开始测试

0x00 ： 正在测量体重 （此时阻抗数值为 0）

0x01 ： 正在测量阻抗（此时阻抗数值为 0）

0x02 ： 阻抗测量成功

0x03 ： 阻抗测量失败（此时阻抗数值为 0xFFFF）

0xFF ： 测试结束

(8) 数据属性:

Bit7 ： 温度单位(0=℃ ; 1=℉)

Bit6-3：体重单位(0=kg; 1=斤; 4=st:lb; 6=lb)

Bit2-1：体重小数点(0 =无小数点;1=1 个小数点;2=2 个小数点;3 =3 个小数点)

Bit0 ： 重量类型 (0=实时重量;1=稳定重量)

(9) 体重:

最高位 =0 ： 正重量

最高位 = 1 ： 负重量

例如：

重量值为 -100,则 sData[24]=0x80，sData[25]=0x64

重量值为 100,则 sData[24]=0x00，sData[25]=0x64

备注：对于 ST：LB 单位，需要把数值转为 LB，但是单位需要声明为 ST:LB .

(10) 阻抗:

分辨率:1 Ω

(11) 算法 ID:

默认 0x00

(12) 温度

分辨率 0.1

最高位 = 0 ： 正温度

最高位 = 1 ： 负单位

无温度测量，则该值为 0xFFFF

4. 数据发送

MCU 需要定时 50ms 进行数据发送(就算内容不变,也需要定时发送).

```
char sta;
char data[6];
SPI_write_1byte( 0x20, 0x8a);// power up
SPI_write_op( 0xE1);
SPI_write_1byte( 0x27, 0x70);
for(int i = 0 ; i <3 ;i++)
{
if(i==0)SPI_write_1byte( 0x25, 0x02);//(通道 1)
else if(i==1)SPI_write_1byte( 0x25, 0x1A);////(通道 2)
else if(i==2)SPI_write_1byte( 0x25, 0x50);//(通道 3)

data[0] = 0x02;
data[1] = 0x25;
SPI_write_nbyte( 0x31, 0x02);

//char sData[31];
SPI_write_nbyte( 0xA0, 31); //spi 写入的 31 个字节内容为第 3 步的内容

SPI_write_op(0xD5);
delay_us(40);//延时 40us
SPI_write_op(0xD6);
while(1)//死循环
{
    sta=SPI_read_1byte( 0x07 ) ;
    if( sta&0x20)   //读回来的值的 Bit5 为 1 时
    {
        SPI_write_1byte( 0x27, sta);
        break;          //则退出死循环
    }
    if( sta&0x10)   //读回来的值的 Bit4 为 1 时
    {
        SPI_write_1byte( 0x27, sta);
        break;          //则退出死循环
    }
}
}

/*发送完毕*/
```

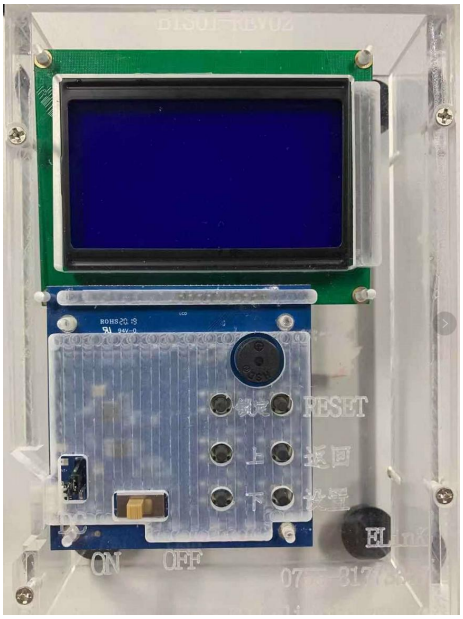
5. 关机

当设备工作完毕,调用以下配置让模块进入休眠模式,若想要模块进入工作模式,需要调用第一步的设置.

```
char sta;
sta=SPI_read_1byte( 0x03 ) ;
sta = sta & 0xfc;
sta = sta | 0x01;
SPI_write_1byte( 0x23, sta);
delay_us(1000);
SPI_write_1byte(0x20, 0x80);
```

6 生产测试指导

我们有生产使用的测试盒（BTS02），能够高效、快速、批量辅助生产测试。批量时，联系我司购买即可。



7 联系我们

深圳市易连物联网有限公司

地址：深圳市宝安区西乡街道银田工业区侨鸿盛文化创意园写字楼 A 栋五层 502 室

Tel: +（86）0755-81773367

Email:hw@elinkthings.com

Web: www.elinkthings.com

8 附件

标准 TEA 加密算法(C 语言)

uint8_t* p 为需要加密的数据的指针

uint32_t* tea_key 为秘钥

举例:

data[8] = {0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08};

const uint32_t tea_key[4] = {0x01020304, 0x02030405, 0x03040506, 0x06070809};

encrypt_8byte(data,tea_key);

调用 encrypt_8byte 加密函数后,data 的数据就变成加密后的数据了,加密后的数据为

data[1]到 data[7]分别为: 0xD0,0xB9,0x11,0x4E,0xBB,0xBC,0xC9,0x21

源码:

```
void encrypt_8byte(uint8_t* p , uint32_t* tea_key) // 8byte 数据加密
{
	union u8_to_u32
	{
		uint8_t	u8_data[8];
		uint32_t u32_data[2];
	}v;
	memcpy((uint8_t *)&v.u8_data, p, sizeof(v));
	encrypt_tea(&v.u32_data[0],(uint32_t *)tea_key1);
	memcpy(p, (uint8_t *)&v.u8_data, sizeof(v));
}
void encrypt_tea(uint32_t* v, uint32_t* k)
{
    uint32_t v0 = v[0], v1 = v[1], sum = 0, i;                  /* set up */
    uint32_t delta = 0x9e3779b9;                                /* a key schedule constant */
    uint32_t k0 = k[0], k1 = k[1], k2 = k[2], k3 = k[3];   /* cache key */
    for (i=0; i < 32; i++) {                                    /* basic cycle start */
        sum += delta;
        v0 += ((v1 << 4) + k0) ^ (v1 + sum) ^ ((v1 >> 5) + k1);
        v1 += ((v0 << 4) + k2) ^ (v0 + sum) ^ ((v0 >> 5) + k3);
    }                                                           /* end cycle */
    v[0] = v0; v[1] = v1;
}
```