数据通信共有2种模式

1. 被动监听
2. APP 发送“设置时间并获取待监听的数据”命令（详见系统设置1号命令）
3. 设备收到命令后上传运动、睡眠、心率、血氧、PPG、ECG、设备信息、MAC信息、完成标志，共9组数据（详见被动监听数据说明）
4. 系统设置
5. APP向设备发送带参数的系统设置命令
6. 每次蓝牙连接成功会发送多条系统设置，包括（详见系统设置指令说明）
7. 0x01：时间设置,length\_L=0x09

2、0x04:个人资料设置，length\_L=0x08,data[12]=4

1. 被动监听数据解析（发送0x01号设置时间指令后设备返回多条监听数据）

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11-end |
| 0xAA | 0 | Length\_H | Length\_L | crc | 数据间隔标志 | 0 | 0 | 0x03 | 0 | 数据类型标志 | 数据包 |

图一 蓝牙帧格式

Byte[0]:0xAA 帧头

Byte[1]:0x00 预留

Byte[2]: Length\_H 数据长度高位

Byte[3]: Length\_L 数据长度低位，帧长度= Length\_H<<8+ Length\_L+8

Byte[4]:crc校验

data\_crc=get\_crc(sport\_data\_today,13,((\*(sport\_data\_today+3))-5));

uint8\_t get\_crc(uint8\_t \*data,uint8\_t start,uint8\_t lenght)

{

uint8\_t crc;

uint8\_t i;

crc = 0;

for (i=start; i<start+lenght; i++)

{

crc = crc16\_ccitt(\*(data+i), crc);

}

return crc;

}

uint8\_t crc16\_ccitt(uint8\_t data, uint8\_t crc)

{

uint8\_t polynomial = 0x97;

uint8\_t i;

crc ^= data;

for (i=0; i<8; i++)

{

if (crc & 0x80)

{

crc <<= 1;

crc ^= polynomial;

}

else

{

crc <<= 1;

}

}

return crc;

}

Byte[5]:数据间隔标志低位，对于部分定时自动采集的数据，该位表示数据采集间隔时间，单位为分，非定时自动采集数据，该位为0

Byte[6]: 数据间隔标志高位，对于部分定时自动采集的数据，该位表示数据采集间隔时间，单位为分，非定时自动采集数据，该位为0，数据采集间隔=byte[5]+byte[6]\*256

Byte[7]:0预留

Byte[8]:0x03

Byte[9]:0预留

Byte[10]:数据类型标志，根据不同数据类型解析数据包

1、0x02运动数据

单日数据为sport\_data\_today[]，最多可存储7日数据

当有历史数据时，n天的数据分n次发送（n<=7）

卡路里和距离通过用户信息和总步数计算：

卡路里（Kcal）=用户体重（kg）\*1.036\*用户身高(cm)\*当天总步数\*0.41\*0.00001

距离：

单位为公制km时：

距离=用户身高\*0.41\*当天总步数\*0.00001

单位为英制mile时：

距离=用户身高\*0.41\*当天总步数\*0.00001\*0.62

sport\_data\_today[10]=0x02;//数据标志，0x02为运动数据

sport\_data\_today[11]=0x00;

sport\_data\_today[12]=mSportHead.length=0x34;

sport\_data\_today[13]=mSportHead.Date.data>>8;//年月日高位

sport\_data\_today[14]=mSportHead.Date.data&0xff;//年月日低位

Time=sport\_data\_today[13]<<8|sport\_data\_today[14];

Bit15:reserved预留；

Bit14-9:year

Bit8-5:month

Bit4-0:day

typedef struct

{

uint16\_t day :

5;

uint16\_t month :

4;

uint16\_t year :

6;

uint16\_t reserved :

1;

}Date\_bit\_field\_type\_t;

sport\_data\_today[15]=mSportTimeData.Date.data>>8;

sport\_data\_today[16]=mSportTimeData.Date.data&0xff;

mSportTimeData=sport\_data\_today[15]<<8|sport\_data\_today[16];

Bit15-5:sport\_time=0

Bit4-0: sport\_count=24

typedef struct

{

uint16\_t sport\_count :

5;

uint16\_t sport\_time :

11;

}SportsTimeData\_bit\_field\_type\_t;

sport\_data\_today[17]=0:00-0:15步数高位

sport\_data\_today[18]=0:00-0:15步数低位；//默认每15分钟记录一次，每个数据占用2字节

sport\_data\_today[19]=0:15-0:30步数高位

sport\_data\_today[20]=0:15-0:30步数低位

sport\_data\_today[21]=0:30-0:45步数高位

sport\_data\_today[22]=0:30-0:45步数低位

sport\_data\_today[23]=0:45-1:00步数高位

sport\_data\_today[24]=0:45-1:00步数低位

sport\_data\_today[25]=1:00-1:15步数高位

sport\_data\_today[26]=1:00-1:15步数低位

...

sport\_data\_today[57]=5:00-5:15步数高位

sport\_data\_today[58]=5:00-5:15步数低位

sport\_data\_today[59]=5:15-5:30步数高位

sport\_data\_today[60]=5:15-5:30步数低位

sport\_data\_today[61]=5:30-5:45步数高位

sport\_data\_today[62]=5:30-5:45步数低位

sport\_data\_today[63]=5:45-6:00步数高位

sport\_data\_today[64]=5:45-6:00步数低位

...

sport\_data\_today[205]=23:30-23:45步数高位

sport\_data\_today[206]=23:30-23:45步数低位

sport\_data\_today[207]=23:45-24:00步数高位

sport\_data\_today[208]=23:45-24:00步数低位

sport\_data\_today[209]=全天总步数高位

sport\_data\_today[210]=全天总步数低位

2、0x03 睡眠数据

单日数据为sleep\_data[68]，最多可存储7日数据

当有历史数据时，n天的数据分n次发送（n<=7）

sleep\_data[10]=0x03;//数据标志，0x03为睡眠数据

sleep\_data[11]=0

sleep\_data[12]=3+睡眠时长\*2；

sleep\_data[13]=年月日高位；

sleep\_data[14]=年月日低位；

Time=sleep\_data[13]<<8|sleep\_data[14];

Bit15:reserved预留；

Bit14-9:year

Bit8-5:month

Bit4-0:day

sleep\_data[15]=睡眠时长；

sleep\_data[16]=睡眠数据高位；

sleep\_data[17]=睡眠数据低位；

...

sleep\_data[26]=睡眠数据高位；

sleep\_data[27]=睡眠数据低位；

...

Para=睡眠数据高位<<8|睡眠数据低位

Bit15-11:起始时间-小时 uint16

Bit10-5:起始时间-分钟 uint16

Bit4:预留

Bit3-0:睡眠模式，00 = 熬夜，01 = 进入睡眠，02 = 浅睡，03 = 熟睡，04 = 睡醒，05 = 退出睡眠

睡眠数据说明：

1. 输出格式示例：

[SleepData{sleep\_type='0', startTime='23:00'}, SleepData{sleep\_type='1', startTime='05:00'},

SleepData{sleep\_type='2', startTime='05:01'}, SleepData{sleep\_type='3', startTime='05:30'},

SleepData{sleep\_type='5', startTime='08:00'},

1. SleepData{sleep\_type='0', startTime='23:00'},这一段数据不会在数组中输出，需要自行补上
2. 有6个参数需要自行计算后从SDK输出：

深睡时长getSleepDeepTime=每段((03模式结束进入下一个模式的起始时间）-（03模式的起始时间）)之和

浅睡时长getSleepLightTime=每段((02模式结束进入下一个模式的起始时间）-（02模式的起始时间）)之和

清醒时长getSleepStayupTime=每段((00模式结束进入下一个模式的起始时间）-（00模式的起始时间）)之和

总睡眠时长getSleepTotalTime=01,02,03模式时长总和

醒来次数getSleepWakingNumber=出现04模式的次数

总时长getTotalTime=睡眠结束时间-开始记录睡眠的时间

3、0x44午睡数据：

uint8 data[21]

data[10]=0x44;//数据标志，0x44为午睡数据

data[11]=0x00;

data[12]=0x14;//午睡开始时间，年20

data[13]=0x16;//午睡开始时间，低位年22，2022

data[14]=0x09;//午睡开始时间，月09

data[15]=0x07;//午睡开始时间，日07

data[16]=0x0D;//午睡开始时间，时13

data[17]=0x01;//午睡开始时间，分01

data[18]=0x0D;//午睡结束时间，时13

data[19]=0x1E;//午睡结束时间，分30

Data[20]=0x3C;//午睡总时间，时长60分钟

4、0x05心率数据

单日数据为auto\_heart\_data[]，最多可存储7日数据

当有历史数据时，n天的数据分n次发送（n<=7）

uint8\_t auto\_heart\_data[]：

auto\_heart\_data[10]=0x05;//数据标志，0x05为心率数据

auto\_heart\_data[11]=0x00;

auto\_heart\_data[12]= mHeartHead.length=0x7C;

auto\_heart\_data[13]= mHeartHead.Date.data>>8;//年月日高位

auto\_heart\_data[14]= mHeartHead.Date.data&0xff;//年月日低位

Time=auto\_heart\_data[13]<<8|auto\_heart\_data[14];

Bit15:reserved预留；

Bit14-9:year

Bit8-5:month

Bit4-0:day

auto\_heart\_data[15]=0x00;

auto\_heart\_data[16]=0x30;

auto\_heart\_data[17]=0:00心率

auto\_heart\_data[18]=0:01心率；//每1分钟记录一次，每个数据占用1字节

auto\_heart\_data[19]=0:02心率

auto\_heart\_data[20]=0:03心率

auto\_heart\_data[21]=0:04心率

auto\_heart\_data[22]=0:05心率

auto\_heart\_data[23]=0:06心率

auto\_heart\_data[24]=0:07心率

auto\_heart\_data[25]=0:08心率

...

auto\_heart\_data[1453]=23:56心率

auto\_heart\_data[1454]=23:57心率

auto\_heart\_data[1455]=23:58心率

auto\_heart\_data[1456]=23:59心率

5、0x10压力情绪数据

uint8\_t bracelet\_pressure\_data[65]：

bracelet\_pressure\_data[0]=0xAB;

bracelet\_pressure\_data[1]=0;

bracelet\_pressure\_data[2]=0;

bracelet\_pressure\_data[3]=0x21; //数组长度=0x21+8

bracelet\_pressure\_data[4]= data\_crc；//参照前面的crc校验算法

bracelet\_pressure\_data[5]=0x3C;//间隔时间60分钟

bracelet\_pressure\_data[8]=0x03;

bracelet\_pressure\_data[9]=0;

bracelet\_pressure\_data[10]=0x10; //数据标志，0x10为压力数据

bracelet\_pressure\_data[12]=0;

bracelet\_pressure\_data[13]= TimeData.data>>8；//年月日高位

bracelet\_pressure\_data[14]= TimeData.data&0xff;//年月日低位

Time= data[13]<<8| \_data[14];

Bit15:reserved预留；

Bit14-9:year

Bit8-5:month

Bit4-0:day

bracelet\_pressure\_data[15]=0;

bracelet\_pressure\_data[16]=0;

bracelet\_pressure\_data[17]=0;//0:00-1:00压力

bracelet\_pressure\_data[18]=0;//0:00-1:00情绪

bracelet\_pressure\_data[19]=0; //1:00-2:00压力

bracelet\_pressure\_data[20]=0; //1:00-2:00情绪

bracelet\_pressure\_data[21]=0; //2:00-3:00压力

bracelet\_pressure\_data[22]=0; //2:00-3:00情绪

…

bracelet\_pressure\_data[63]=0; //23:00-24:00压力

bracelet\_pressure\_data[64]=0; //23:00-24:00情绪

6、0x11疲劳数据

uint8\_t bracelet\_fatg\_data[41]：

bracelet\_fatg\_data[0]=0xAB;

bracelet\_ fatg \_data[1]=0;

bracelet\_ fatg \_data[2]=0;

bracelet\_ fatg \_data[3]=0x21; //数组长度=0x21+8

bracelet\_ fatg \_data[4]= data\_crc；//参照前面的crc校验算法

bracelet\_ fatg \_data[5]=0x3C;//间隔时间60分钟

bracelet\_ fatg \_data[8]=0x03;

bracelet\_ fatg \_data[9]=0;

bracelet\_ fatg \_data[10]=0x11; //数据标志，0x11为疲劳数据

bracelet\_ fatg \_data[12]=0;

bracelet\_ fatg \_data[13]= TimeData.data>>8；//年月日高位

bracelet\_ fatg \_data[14]= TimeData.data&0xff;//年月日低位

Time= data[13]<<8| \_data[14];

Bit15:reserved预留；

Bit14-9:year

Bit8-5:month

Bit4-0:day

bracelet\_ fatg \_data[15]=0;

bracelet\_ fatg \_data[16]=0;

bracelet\_ fatg \_data[17]=0;//0:00-1:00疲劳

bracelet\_ fatg \_data[18]=0; //1:00-2:00疲劳

bracelet\_ fatg \_data[19]=0; //2:00-3:00疲劳

bracelet\_fatg\_data[20]=0; //3:00-4:00疲劳

…

bracelet\_fatg\_data[40]=0; //23:00-24:00疲劳

7、0x14血氧数据

uint8\_t bracelet\_spo2\_data[]：

bracelet\_spo2\_data[10]=0x14；//数据标志，0x14为血氧数据

bracelet\_spo2\_data[11]=0x00；

bracelet\_spo2\_data[12]=mSPO2Head.length=已记录的血氧值个数\*6+3;

bracelet\_spo2\_data[13]=0x00；

bracelet\_spo2\_data[14]=0x00；

bracelet\_spo2\_data[15]=mSPO2Head.length=已记录的血氧值个数;//最多记录20个

bracelet\_spo2\_data[16]=(spo2\_measure\_time.data>>24)&0xff;

bracelet\_spo2\_data[17]=(spo2\_measure\_time.data>>16)&0xff;

bracelet\_spo2\_data[18]=(spo2\_measure\_time.data>>8)&0xff;

bracelet\_spo2\_data[19]=spo2\_measure\_time.data&0xff;

Time=(bracelet\_spo2\_data[16]<<24)|(bracelet\_spo2\_data[17]<<16)(bracelet\_spo2\_data[18]<<8)|(bracelet\_spo2\_data[19]);

Bit31-26:year uint32

Bit25-22:month uint32

Bit21-17:day uint32

Bit16-12:hours uint32

Bit11-6:minute uint32

Bit5-0:seconds uint32

bracelet\_spo2\_data[20]=血氧值;

bracelet\_spo2\_data[21]=心率值;

bracelet\_spo2\_data[22]=(spo2\_measure\_time.data>>24)&0xff;

bracelet\_spo2\_data[23]=(spo2\_measure\_time.data>>16)&0xff;

bracelet\_spo2\_data[24]=(spo2\_measure\_time.data>>8)&0xff;

bracelet\_spo2\_data[25]=spo2\_measure\_time.data&0xff;

bracelet\_spo2\_data[26]=血氧值;

bracelet\_spo2\_data[27]=心率值;

...

8、0x16呼吸数据

uint8\_t bracelet\_rr\_data[117]：

bracelet\_rr\_data[0]=0xAB;

bracelet\_ rr \_data[1]=0;

bracelet\_ rr \_data[2]=0;

bracelet\_ rr \_data[3]=0x6D; //数组长度=0x69+8

bracelet\_ rr \_data[4]= data\_crc；//参照前面的crc校验算法

bracelet\_ rr \_data[5]=0x0F;//间隔时间15分钟

bracelet\_ rr \_data[8]=0x03;

bracelet\_ rr \_data[9]=0;

bracelet\_ rr \_data[10]=0x16; //数据标志，0x16为呼吸数据

bracelet\_ rr \_data[12]=0;

bracelet\_ rr \_data[13]= TimeData.data>>8；//年月日高位

bracelet\_ rr \_data[14]= TimeData.data&0xff;//年月日低位

Time= data[13]<<8| \_data[14];

Bit15:reserved预留；

Bit14-9:year

Bit8-5:month

Bit4-0:day

bracelet\_ rr \_data[15]=0;

bracelet\_ rr \_data[16]=0;

bracelet\_ rr \_data[17]=0;//0:00-0:15呼吸

bracelet\_ rr \_data[18]=0;//0:15-0:30呼吸

bracelet\_ rr \_data[19]=0;//0:30-0:45呼吸

bracelet\_ rr \_data[20]=0;//0:45-1:00呼吸

bracelet\_ rr \_data[21]=0; //1:00-1:15呼吸

bracelet\_ rr \_data[22]=0; //1:15-1:30呼吸

bracelet\_rr\_data[23]=0; //1:30-1:45呼吸

…

bracelet\_rr\_data[112]=0; //23:45-24:00呼吸

bracelet\_rr\_data[113]=低通气指数;

bracelet\_rr\_data[114]=累计阻塞时长;

bracelet\_rr\_data[115]=呼吸紊乱指数;

bracelet\_rr\_data[116]=呼吸暂停次数;

9、0x08设备信息

uint8\_t send\_device\_inf\_data[20]：

SEND\_DEVCIE\_INF[0]=0xAB

SEND\_DEVCIE\_INF[1]=0

SEND\_DEVCIE\_INF[2]=0

SEND\_DEVCIE\_INF[3]=0x0C//数组长度=0x0c+8

SEND\_DEVCIE\_INF[4]=0

SEND\_DEVCIE\_INF[5]=0

SEND\_DEVCIE\_INF[6]=0

SEND\_DEVCIE\_INF[7]=0x01

SEND\_DEVCIE\_INF[8]=0x03

SEND\_DEVCIE\_INF[9]=0

SEND\_DEVCIE\_INF[10]=0x08//数据标志，0x08为设备信息数据

SEND\_DEVCIE\_INF[11]=0

SEND\_DEVCIE\_INF[12]=0x7

SEND\_DEVCIE\_INF[13]=0x64=电池电量\*25

SEND\_DEVCIE\_INF[14]=0xE2

SEND\_DEVCIE\_INF[15]=0x21//功能版本号

SEND\_DEVCIE\_INF[16]=0x23 //主版本号：重大功能调整导致项目全局变化时，主版本号加1。

SEND\_DEVCIE\_INF[17]=0x10 //次版本号：局部功能调整，和以前版本比功能变化较大则加1。

SEND\_DEVCIE\_INF[18]=0x40//修订版本号：修复日出BUG缺陷之后，修订版本号加1.

SEND\_DEVCIE\_INF[19]=0x00//血压校准标记，0为未校准，1为已校准

10、0x09 MAC信息

uint8\_t SEND\_MAC[19]：

SEND\_MAC[1]=0

SEND\_MAC[2]=0

SEND\_MAC[3]=0x0B

SEND\_MAC[4]=0

SEND\_MAC[5]=0

SEND\_MAC[6]=0

SEND\_MAC[7]=0

SEND\_MAC[8]=0x03

SEND\_MAC[9]=0

SEND\_MAC[10]=0x09//数据标志，0x09为MAC信息数据

SEND\_MAC[11]=0

SEND\_MAC[12]=0x06//MAC地址

SEND\_MAC[13]=0xD3//MAC地址

SEND\_MAC[14]=0xF1//MAC地址

SEND\_MAC[15]=0x61//MAC地址

SEND\_MAC[16]=0xA0//MAC地址

SEND\_MAC[17]=0x26//MAC地址

SEND\_MAC[18]=0xF0//MAC地址

11、0x04完成标志

uint8\_t SEND\_COMPELETE[13]:

SEND\_COMPELETE[0]=0xAB

SEND\_COMPELETE[1]=0

SEND\_COMPELETE[2]=0

SEND\_COMPELETE[3]=0x05

SEND\_COMPELETE[4]=0

SEND\_COMPELETE[5]=0

SEND\_COMPELETE[6]=0

SEND\_COMPELETE[7]=0

SEND\_COMPELETE[8]=0x03

SEND\_COMPELETE[9]=0

SEND\_COMPELETE[10]=0x04//数据标志，0x04为完成信息数据

SEND\_COMPELETE[11]=0

SEND\_COMPELETE[12]=0

1. 系统设置

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | … |
| 0xAB | 0 | length\_H | length\_L | 0 | 0 | 0 | 0 | ID | 0 | 数据标志 | 0 | length\_L-5 | 数据包 |

data[0]=0xAA帧头

data[1]=0预留

data[2]=length\_H=0//数据长度高位

data[3]=length\_L//数据长度低位,数据长度=data[3]+8

data[4]=0预留

data[5]=0预留

data[6]=0预留

data[7]=0预留

data[8]= id: SET\_CONFIG\_COMMAND\_ID = 0x01,

NOTIFY\_COMMAND\_ID = 0x02,

GET\_DEVICE\_DATA\_COMMAND\_ID=0x03,

SEND\_ECG\_TEMPLATE\_COMMAND\_ID=0x04,

data[9]=0预留

data[10]=数据标志

data[10]参数说明：

0x01:设置时间并获取待监听的数据,length\_H=0,length\_L=0x09

Para=(data[13]<<24)|(data[14]<<16)|(data[15]<<8)|(data[16])

para示例：

0101 0010 1010 0110 1110 1100 1100 0001

0101 00, 10 10, 10 011,0 1110, 1100 11,00 0001

Year month day hour min second

0x04:个人资料，length\_L=0x08

Para=(data[13]<<24)|(data[14]<<16)|(data[15]<<8)|(data[16])

Para示例：

Bit31: 性别/\*\*0: female, 1: male\*/

Bit30-24: 年龄/\*\*age 0~127\*/

Bit23-16：身高/\*\* hight accuracy : 0.5 m \*/

Bit15-8：体重/\*\* accuracy: 0.5 kg, \*/

Bit7-0：预留

0x06:重启设备，data[13]={0xAB 0 0 5 0 0 0 0 1 0 0x06 0 1 1}

0x18:寻找设备，length\_L=0x05，data[13]={0xAB 0 0 5 0 0 0 0 1 0 0x18 0 0 }

0x1a:设置心率血压值，length\_L=0x08，data[13]=心率，data[14]=高压值，data[15]=低压值

0x1c:获取设备电量信息，length\_L=0x05，length\_L=0x05，data[13]={0xAB 0 0 5 0 0 0 0 1 0 0x1c 0 0 }

0x21:获取MAC地址，length\_L=0x05，data[13]={0xAB 0 0 5 0 0 0 0 1 0 0x21 0 0 }

表开始测量心率，手表返回实时心率send[14]={ 0xAB 0 0 6 0 0 0 0 3 0 0x0B 0 1 心率}，25秒后自动停止测量

0x5A:开始血氧测量，APP向手表发送获取指令data[14]={0xAB 0 0 6 0 0 0 0 1 0 0x5A 0 1 0}，手表开始测量血氧，手表返回实时血氧send[14]={ 0xAB 0 0 6 0 0 0 0 3 0 0x5A 0 1 血氧}，25秒后自动停止测量

0x5C:开始测量疲劳压力，APP向手表发送获取指令data[14]={0xAB 0 0 6 0 0 0 0 1 0 0x5C 0 1 0}，手表开始测量疲劳压力，手表返回实时值send[15]={ 0xAB 0 0 7 0 0 0 0 3 0 0x5C 0 2 疲劳 压力}，

0x5D:工厂测量模式，APP向手表发送获取指令data[14]={0xAA 0 0 6 0 0 0 0 1 0 0x5D 0 1 0}，设备进入工厂测量模式，测完设备回复data[14]={0xAA 0 0 6 0 0 0 0 3 0 0x5D 0 1 测试结果}，测试结果0表示测试成功，bit0：三轴结果，0成功，1失败 bit1:心率结果，0成功，1失败

Bit2血氧结果，0成功，1失败

0x5E:设置采集时间间隔，data[16]={0xAA,0,0, 8, 0 ,0 ,0, 0 ,1 ,0 ,0x5E, 0, 3, 类型，间隔时间H，间隔时间L};

类型：1心率，2血氧，3血压，4疲劳，5情绪压力，FF全部

0x5F:关闭设备蓝牙和定时采集，进入低功耗模式

0x60:上传下发SN码及圈号，

APP向设备发送读SN指令data[14]={0xAA 0 0 6 0 0 0 0 1 0 0x60 0 1 0}，

设备返回SN，send[21]={ 0xAA 0 0 0x0D 0 0 0 0 3 0 0x60 0 8 SN[7] SN[6] SN[5] SN[4] SN[3] SN[2] SN[1] SN[0]}

APP向设备发送写SN指令data[22]={0xAA 0 0 0x0E 0 0 0 0 1 0 0x60 0 9 1 SN[7] SN[6] SN[5] SN[4] SN[3] SN[2] SN[1] SN[0]}，

例：SN[8]={0x08,0x07,0x18,0x0A,0x00,0x00,0x01,0x01}

SN[0]=0x08,戒指圈号为SN[0]-1=7号圈（圈号最小为6号（0x07），最大13号(0x0E)）

SN[1]=0x07,转为十进制，表示7月

SN[2]=0x18，转为十进制，表示24年

SN[3]=0x0A,不用转，直接获取A

SN[4] SN[5] SN[6] SN[7]为序列号(一共8位十进制数字)，高位在前，低位在后

序列号=(SN[4]<<24)|(SN[5]<<16)|(SN[6]<<8)|SN[7]=00000257

固定前缀：BIOSRI1

解析出完整SN码为BIOSRI10724A00000257，圈号7号

0x61:下发HASH码

APP向设备发送写HASH指令data[17]={0xAA 0 0 0x09 0 0 0 0 1 0 0x61 0 0x04 h[3] h[2] h[1] h[0]}，

0x80:关闭设备,data[13]={0xAA 0 0 5 0 0 0 0 1 0 0x80 0 0 }