ZLBUS 通讯协议用户指令手册

目录

手册修订历史

版本号	说明	日期
1.01	初版	2024-02-20

一、通信协议简介

1、基本通信格式

	帧头	指令 ID	数据长度	数据区	校验
字节数	1	1	2	N	1
	0xAA	T	L	D	C
		校验区域			

简写说明:

- ✧ T：指令 ID
- ✧ L：数据区数据长度
- ✧ D：数据区
- ✧ N：数据区长度
- ✧ C：Check-Xor 校验，校验区域包含指令 ID、数据长度、数据区

通信协议数据编码方式

- ✧ 二进制格式 –
- ✧ 浮点数采用 **单精度** 4 字节 IEEE-754
- ✧ 数据模式：小端模式 Little-endian

1.1 数据区基本格式

1.1.1 格式一(指令发送)

	子指令 ID	RF_ID、DOT_ID	数据段
字节数	1	2	N

- ✧ 子指令 ID: 0x00 ~ 0x7F
- ✧ RF_ID :缺省配置 0x3F 或 0x21；专用协议中，RF_ID 由于硬件自动设置；
- ✧ DOT_ID：缺省配置 0x00, (广播 ID=0xFF)
- ✧ 数据段：有效数据区
- ✧ N 范围： 0 ~ 239 字节

1.1.2 格式二(数据上传)

	子指令 ID	RF_ID、DOT_ID	流水号	数据段
字节数	1	2	1 (或 2)	N

- ✧ 子指令 ID: 0x00 ~ 0x7F
- ✧ RF_ID、DOT_ID: 见 格式一
- ✧ 流水号：可通过指令配置选择
  - 1 字节：0x00 ~ 0xFF 循环
  - 2 字节：0x0000 ~ 0xFFFF 循环
- ✧ 数据段：有效数据区
- ✧ N：0 ~ 238 字节

2、应答 ID 结构

	错误识别位	ID 位 6	ID 位 5	ID 位 4	ID 位 3	ID 位 2	ID 位 1	ID 位 0
错误 ID	1	x	x	x	x	x	x	x
子指令 ID	0	x	x	x	x	x	x	x

3、Check-Xor 程序编码

3.1 C 语言

```c
uint8_t checkXor8_compute(uint8_t const * p_data, uint32_t size) {
    uint8_t checkXor = 0xFF;
    for (uint32_t i = 0; i < size; ++i) {
        checkXor ^= p_data[i];
    }
    return checkXor;
}
```

3.2 python

```python
def CheckXor8_compute(value: typing.Union[bytes, bytearray]) -> int:
    courts = len(value)
    check_xor = 0xFF
    for i in range(courts):
        check_xor ^= value[i]
    return check_xor & 0xFF
```

二、数据上传格式

1、IMU 数据上报

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	流水号	数据段	校验码
0xAA	0x10	N	子指令 ID	RF_ID、DOT_ID	流水号	IMU 数据	Check-Xor

✧ 类型简介

名称类型	数据类型	占用字节	说明
数据区长度	int16_t	2	长度大小 = 数据段的字节数 + 4（或 5）
子指令 ID	uint8_t	1	Bit 0: 表示此时姿态数据融合轴数
RF_ID	uint8_t	1	蓝牙通信忽略该项
DOT_ID	uint8_t	1	蓝牙通信忽略该项
流水号	uint8_t （或 uint16_t）	1 （或 2）	0x00-0xFF 循环变化 （或 0x0000-0xFFFF 循环变化）， 详情见 读取上传流水号格式 部分
校验码	uint8_t	1	指令 ID 到 数据段部分 进行异或运算

✧ 子指令 ID

子指令 ID	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Value	0	0	0	0	0	0	姿态轴	

- 姿态轴：
  - 0x00：IC 姿态由加速度计、陀螺仪、磁力计融合组成
  - 0x01：IC 姿态由陀螺仪、磁力计融合组成
  - 0x02：IC 姿态由加速度计、陀螺仪融合组成
  - 0x03：IC 姿态由陀螺仪融合组成

✧ IMU 数据:

- IMU 中数据字段由上传数据格式中决定，未配置的数据项，则该数据项不上报
- 单个数据格式：小端格式
- 排列顺序: 时间戳、四元数、欧拉角、加速度、陀螺仪、磁力计、线性加速度、IMU 温度

✧ 支持的 IMU 数据字段

IMU 数据字段	数据类型	字节数	顺序	单位
时间戳	float	4		毫秒(ms)
四元数	float	16	w x y z	
欧拉角	float	12	roll pitch yaw	度(°)
加速度	float	12	x y z	g
陀螺仪	float	12	x y z	度/秒(°/s)
磁力计	float	12	x y z	微特(μT)
线性加速度	float	12	x y z	g
IMU 温度	float	4		摄氏度(℃)

2、IC 状态上报

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	流水号	数据段	校验码
0xAA	0x11	8	0x00	RF_ID 、DOT_ID	流水号	IC 状态码	Check-Xor

✧ 类型简介 见 **IMU 数据上报** 部分

✧ 数据段：IC 状态码 (uint32_t)

名称	Bit 位	说明
加速度 X 轴	0	0：正常， 1：故障
加速度 Y 轴	1	0：正常， 1：故障
加速度 Z 轴	2	0：正常， 1：故障
陀螺仪 X 轴	3	0：正常， 1：故障
陀螺仪 Y 轴	4	0：正常， 1：故障
陀螺仪 Z 轴	5	0：正常， 1：故障
磁力计 X 轴	6	0：正常， 1：故障
磁力计 Y 轴	7	0：正常， 1：故障
磁力计 Z 轴	8	0：正常， 1：故障
静态校准	27	0：校准，1：未校准
动态校准	28	0：校准，1：未校准
IMU 6 轴初始化	29	0：正常， 1：故障
IMU 磁力计初始化	30	0：正常， 1：故障
磁力计数据（异常报警）	31	0：正常， 1：故障

3、电池电量上报

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	流水号	数据段	校验码
0xAA	0x14	N	0x00	RF_ID、DOT_ID	流水号	电池状态值	Check-Xor

✧ 类型简介 见 **IMU 数据上报**部分

✧ 子指令 ID:

子指令 ID	数据段内容	数据区长度 N	备注
0x00	电量 + 电压	7	流水号格式（0x00~0xFF）
0x01	电压	6	流水号格式（0x00~0xFF）
0x02	电量	5	流水号格式（0x00~0xFF）
0x00	电量 + 电压	8	流水号格式（0x0000~0xFFFF）
0x01	电压	7	流水号格式（0x0000~0xFFFF）
0x02	电量	6	流水号格式（0x0000~0xFFFF）

✧ 数据段：电池状态值

名称类型	数据类型	占用字节	说明
电池电量	uint8_t	1	百分比，范围 0x00 - 0x64 （0% - 100%）
电池电压	int16_t	2	电压值，单位 mv

4、ANT(ADC)数据

帧头	指令 ID		数据区长度	子指令 ID	MT_ID	流水号	数据段	校验码
0xAA	0x15		N	子指令 ID	RF_ID、DOT_ID	流水号	ANT 数据字段	Check-Xor

✧ 类型简介 见 **IMU 数据上报**部分

✧ -子指令 ID

子指令 ID	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Value	保留为 0				ANT 数据个数			

✧ ANT 数据字段：

- 单个 ANT 数据格式 uint16_t, 2 字节
- 数据排序：时间戳、ANT 0、ANT 1、ANT 2、ANT 3、ANT 4、ANT 5
  - 时间戳、ANT 数据：由**上传数据格式**决定是否上传
  - ANT 数据个数：由高级指令中**配置 ANT** 指令有关

三、基础指令格式

1、配置 数据上传格式

✧ 此指令：不识别 DOT_ID, DOT_ID 设置成 0xFF 即可

✧ RF_ID: 设置成 0x3F 即可（专用协议时，RF_ID 需设置成指定 ID）

1.1 修改数据上传格式

➢ API:

- C: ul_modifyDataFormat
- Python: ul_modifyDataFormat

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	数据段	校验码
0xAA	0xD5	0x0007	0x00	RF_ID、DOT_ID	数据上传 MAP	Check-Xor

数据上传 MAP ： 数据类型 uint32_t， 4 字节

数据上传 MAP	Bit 31	Bit 30	Bit 29	Bit 28	Bit 27	Bit 26	Bit 25	Bit 24
Value	时间戳	0	0	0	0	0	0	0
数据上传 MAP	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16
Value	0	0	0	0	0	0:	0	ANT
数据上传 MAP	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Value	0	温度	0	0	0	0	0	0
数据上传 MAP	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Value	0	0	线性加速度	磁力计	陀螺仪	加速度	欧拉角	四元数

返回帧:

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD5	0x0003	0x00	RF_ID、DOT_ID		Check-Xor
操作错误	0xAA	0xD5	0x0004	0x80	RF_ID、DOT_ID	错误码	Check-Xor

✧ 错误码：单个字节长度，详情见错误码表

1.2 读取数据上传格式

➢ API:

- C: ul_getDataFormat
- Python: ul_getDataFormat

上位机写操作指令帧，如下表所示:

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	校验码
0xAA	0xD5	0x0003	0x01	RF_ID、DOT_ID	Check-Xor

返回帧:

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD5	0x0007	0x01	RF_ID、DOT_ID	数据上传 MAP	Check-Xor

操作错误	0xAA	0xD5	0x0004	0x81	RF_ID、DOT_ID	错误码	Check-Xor

✧ 数据上传 MAP: 详情见设置数据上传 MAP

✧ 错误码：单个字节长度，详情见错误码表

2、配置采样频率

✧ 此指令：不识别 DOT_ID, DOT_ID 设置成 0xFF 即可

✧ RF_ID: 设置成 0x3F 即可（专用协议时，RF_ID 需设置成指定 ID）

2.1 修改采样频率

➢ API:

- C: ul_modifySampleHz
- Python: ul_modifySampleHz

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	数据段	校验码
0xAA	0xD5	0x0004	0x02	RF_ID、DOT_ID	采样频率	Check-Xor

采样频率：uint16_t，2 字节

采样频率	数据段编码
200Hz	0xC8
240Hz	0xF0
250Hz	0xFA

✧ **注**: 设置采样频率后，上报频率恢复到默认设置

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD5	0x0003	0x02	RF_ID、DOT_ID		Check-Xor
操作错误	0xAA	0xD5	0x0004	0x82	RF_ID、DOT_ID	错误码	Check-Xor

✧ 错误码：单个字节长度，详情见错误码表

2.2 读取采样频率

➢ API:

- C: ul_getSampleHz
- Python: ul_getSampleHz

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	校验码
0xAA	0xD5	0x0003	0x03	RF_ID、DOT_ID	Check-Xor

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD5	0x0004	0x03	RF_ID、DOT_ID	采样频率	Check-Xor
操作错误	0xAA	0xD5	0x0004	0x83	RF_ID、DOT_ID	错误码	Check-Xor

- ✧ 采样频率: 详情见采样频率配置
- ✧ 错误码：单个字节长度，详情见错误码表

3、配置上报频率

- ✧ 此指令：不识别 DOT_ID, DOT_ID 设置成 0xFF 即可
- ✧ RF_ID: 设置成 0x3F 即可（专用协议时，RF_ID 需设置成指定 ID）

3.1 修改上报频率

- ➢ API:
  - ■ C: ul_modifyUploadHz
  - ■ Python: ul_modifyUploadHz

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	数据段	校验码
0xAA	0xD5	0x0004	0x04	RF_ID、DOT_ID	分频编码	Check-Xor

分频编码: 数据类型 uint16_t, 2 字节

上报频率	200Hz 采样下分频编码	240Hz 采样下分频编码	250Hz 采样下分频编码
1Hz	0x00C8	0x00F0	0x00FA
5Hz	0x0028	0x0030	0x0032
10Hz	0x0014	0x0018	0x0019
20Hz	0x000A	0x000C	/
25Hz	0x0008	0x0009	0x000A
30Hz	/	0x0008	/
50Hz	0x0004 (默认设置)	/	0x0005 (默认设置)
60Hz	/	0x0004 (默认设置)	/
100Hz	0x0002	/	/
120Hz	/	0x0002	/
200Hz	0x0001		/
240Hz		0x0001	/
250Hz			0x0001

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD5	0x03	0x04	RF_ID、DOT_ID		Check-Xor
操作错误	0xAA	0xD5	0x04	0x84	RF_ID、DOT_ID	错误码	Check-Xor

- ✧ 错误码：单个字节长度，详情见错误码表

3.2 读取上报频率

- API:
  - C: ul_getUploadHz
  - Python: ul_getUploadHz

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	校验码
0xAA	0xD5	0x0003	0x05	RF_ID、DOT_ID	Check-Xor

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD5	0x0005	0x05	RF_ID、DOT_ID	分频编码	Check-Xor
操作错误	0xAA	0xD5	0x0004	0x85	RF_ID、DOT_ID	错误码	Check-Xor

- 分频编码: 详情见上报频率配置
- 错误码：单个字节长度，详情见错误码表

4、启动磁力计校准

- 此指令：不识别 DOT_ID, DOT_ID 设置成 0xFF 即可
- RF_ID: 设置成 0x3F 即可 （专用协议时，RF_ID 需设置成指定 ID）
- 建议配合 RGB 灯，使用耗时约 2 分钟
- API:
  - C: ul_startMagnetometerCalibration
  - Python: ul_startMagnetometerCalibration

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	校验码
0xAA	0xD5	0x0003	0x06	RF_ID、DOT_ID	Check-Xor

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD5	0x0003	0x06	RF_ID、DOT_ID		Check-Xor
操作错误	0xAA	0xD5	0x0004	0x86	RF_ID、DOT_ID	错误码	Check-Xor

- 错误码：单个字节长度，详情见错误码表

5、配置滤波参数

- 此指令：不识别 DOT_ID, DOT_ID 设置成 0xFF 即可
- RF_ID: 设置成 0x3F 即可（专用协议时，RF_ID 需设置成指定 ID）

滤波参数: 数据类型 uint16_t, 2 字节 ^80d7e7

滤波参数	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Value	0	0	0	0	0	0	0	绝对静止
滤波参数	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Value	零点滤波	0	0	抗磁干扰	灵活磁场变换	0	6 轴模式	静态滤波

- ✧ 9 轴模式：加速度计、陀螺仪、磁力计融合计算 IC 姿态
- ✧ 6 轴模式：加速度计、陀螺仪融合计算 IC 姿态
- ✧ 零点滤波：IC 静态放置时，实时计算零点偏移（使用该设备前，静止放置 1 分钟，使用效果更佳）
- ✧ 抗磁干扰：
  - 仅 9 轴模式下生效
  - 环境磁场变化时，IC 自动切换到 6 轴模式计算当前 IC 姿态
- ✧ 灵活磁场变换：
  - 仅 9 轴模式下生效
  - 环境磁场发生变化时，从之前磁场环境切换到新磁场环境需要 5 秒钟
  - 在切换过程中，使用 6 轴模式计算当前 IC 姿态
  - 在切换过程中，新磁场环境一直变动不稳定，IC 一直保持切换状态工作在 6 轴模式下
  - 在切换过程中，当 IC 重新回到之前磁场环境或新磁场环境稳定，IC 重新切换到 9 轴模式；
- ✧ 静态滤波：启动该设置后，IC 姿态更加稳定

5.1 设置滤波参数

- ➢ API:
  - C: ul_configDataFilter
  - Python: ul_configDataFilter

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	数据段	校验码
0xAA	0xD5	0x0005	0x08	RF_ID、DOT_ID	滤波参数	Check-Xor

- ✧ 滤波参数值：
  - 详情见滤波参数
  - 对应 Bit 位配置为 1 时生效，Bit 配置为 0 无效
  - 生效位打开该位功能

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD5	0x0003	0x08	RF_ID、DOT_ID		Check-Xor
操作错误	0xAA	0xD5	0x0004	0x88	RF_ID、DOT_ID	错误码	Check-Xor

- ✧ 错误码：单个字节长度，详情见错误码表

5.2 清除滤波参数

- ➢ API:
  - C: ul_clearDataFilter
  - Python: ul_clearDataFilter

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	数据段	校验码
0xAA	0xD5	0x0004	0x0A	RF_ID、DOT_ID	滤波参数值	Check-Xor

- ✧ 滤波参数值：
  - 详情见滤波参数
  - 对应 Bit 位配置为 1 时生效，Bit 配置为 0 无效
  - 生效位关闭该位功能

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD5	0x0003	0x0A	RF_ID、DOT_ID		Check-Xor
操作错误	0xAA	0xD5	0x0004	0x8A	RF_ID、DOT_ID	错误码	Check-Xor

✧ 错误码：单个字节长度，详情见错误码表

5.3 读取滤波参数

➢ API:

- C: ul_getDataFilter
- Python: ul_getDataFilter

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	校验码
0xAA	0xD5	0x0003	0x0B	RF_ID、DOT_ID	Check-Xor

返回帧：

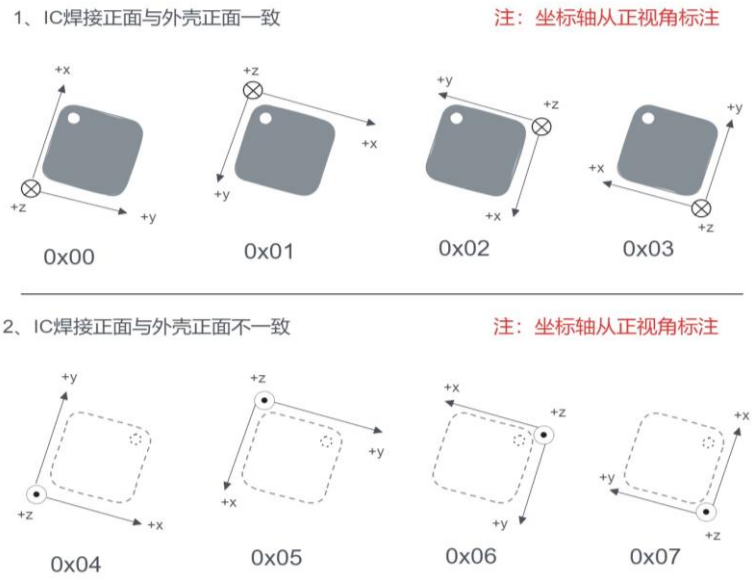
	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD5	0x0005	0x0B	RF_ID、DOT_ID	滤波参数	Check-Xor
操作错误	0xAA	0xD5	0x0004	0x8B	RF_ID、DOT_ID	错误码	Check-Xor

✧ 滤波参数: 详情见滤波参数

✧ 错误码：单个字节长度，详情见错误码表

6、配置 IC 安装方向

- ✧ 此指令：不识别 DOT_ID, DOT_ID 设置成 0xFF 即可
- ✧ RF_ID: 设置成 0x3F 即可（专用协议时，RF_ID 需设置成指定 ID）
- ✧ 安装方向：数据类型 uint8_t, 1 字节



6.1 修改 IC 安装方向

- ➢ API:
  - ■ C: ul_modifyIcConvention
  - ■ Python: ul_modifyIcConvention

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	数据段	校验码
0xAA	0xD5	0x0004	0x0C	RF_ID、DOT_ID	安装方向	Check-Xor

安装方向：编码如下

右手坐标系数据段编码
0x00 (默认值，与 IC 方向一致)
0x01
0x02
0x04
0x03
0x05
0x06
0x07

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD5	0x0003	0x0C	RF_ID、DOT_ID		Check-Xor
操作错误	0xAA	0xD5	0x0004	0x8C	RF_ID、DOT_ID	错误码	Check-Xor

✧ 错误码：单个字节长度，详情见错误码表

6.2 读取 IC 安装方向

➢ API:
- C: ul_getIcConvention
- Python: ul_getIcConvention

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	校验码
0xAA	0xD5	0x0003	0x0D	RF_ID、DOT_ID	Check-Xor

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD5	0x0004	0x0D	RF_ID、DOT_ID	安装方向	Check-Xor
操作错误	0xAA	0xD5	0x0004	0x8D	RF_ID、DOT_ID	错误码	Check-Xor

✧ 安装方向： 详情见 IC 安装方向设置

✧ 错误码：单个字节长度，详情见错误码表

7、配置 RF 设备名称

✧ 此指令：不识别 DOT_ID, DOT_ID 设置成 0xFF 即可

✧ RF_ID: 设置成 0x3F 即可（专用协议时，RF_ID 需设置成指定 ID）

7.1 修改 RF 设备名称

（芯片重启后生效）

➢ API:
- C: ul_modifyIcAdvName
- Python: ul_modifyIcAdvName

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	数据段	校验码
0xAA	0xD5	N	0x0E	RF_ID、DOT_ID	用户编码‘-’传感器编号	Check-Xor

✧ 用户编号：
- 格式：字符串
- 字符串长度: 4~8 字节 （最小 4 字节，最大 8 字节）
- 注：不支持中文

✧ '-':分隔符

✧ 传感器编号：
- 格式：字符串
- 字符串长度: 4 字节
- 注：不支持中文

- ✧ 数据区长度:
  - 用户编号长度 + 8 字节
  - 单位: 字节
- ✧ 示例：将设备广播名称设置为"ZL23-WangHu-Dot0"
  - 用户编号："WangHu"
  - 传感器编号："Dot0"
  - 完整数据段："WangHu-"Dot0"

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD5	0x0003	0x0E	RF_ID、DOT_ID		Check-Xor
操作错误	0xAA	0xD5	0x0004	0x8E	RF_ID、DOT_ID	错误码	Check-Xor

- ✧ 错误码：单个字节长度，详情见错误码表

7.2 读取 RF 设备广播名称

- ➢ API:
  - C: ul_getIcAdvName
  - Python: ul_getIcAdvName

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	校验码
0xAA	0xD5	0x0003	0x0F	RF_ID、DOT_ID	Check-Xor

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD5	N	0x0F	RF_ID、DOT_ID	RF 设备广播名称	Check-Xor
操作错误	0xAA	0xD5	0x0004	0x8F	RF_ID、DOT_ID	错误码	Check-Xor

- ✧ RF 设备广播名称:
  - 格式：字符串
  - 字符串长度: 4~8 字节 （最小 4 字节，最大 8 字节）
  - 示例："ZL23-WangHu-Dot0"
    - "ZL23": 厂家与出厂年份
    - "WangHu" : 用户编号
      - "Dot0":传感器编号
- ✧ 数据区长度:
  - 用户编号长度 + 13 字节
  - 单位: 字节
- ✧ 错误码：单个字节长度，详情见错误码

8、配置 RF 功率

✧ 此指令：不识别 DOT_ID, DOT_ID 设置成 0xFF 即可

✧ RF_ID: 设置成 0x3F 即可（专用协议时，RF_ID 需设置成指定 ID）

8.1 修改 RF 功率

➢ API:

- C: ul_modifyIcRfPower
- Python: ul_modifyIcRfPower

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	数据段	校验码
0xAA	0xD5	0x0004	0x10	RF_ID、DOT_ID	RF 发射功率	Check-Xor

RF 发射功率: 数据类型 int8_t, 1 字节

RF 发射功率	数据段编码	备注
-8dBm	0xF8	
-4dBm	0xF4	
0dBm	0x00	
3dBm	0x03	部分设置无效
4dBm	0x04	
8dBm	0x08	部分设置无效
10dBm	0x0A	部分设置无效

✧ 部分设置无效：设置无效时，IC 自动匹配最近功率值

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD5	0x0003	0x10	RF_ID、DOT_ID		Check-Xor
操作错误	0xAA	0xD5	0x0004	0x90	RF_ID、DOT_ID	错误码	Check-Xor

✧ 错误码：单个字节长度，详情见错误码表

8.2 读取 RF 功率

➢ API:

- C: ul_getIcRfPower
- Python: ul_getIcRfPower

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	校验码
0xAA	0xD5	0x0003	0x11	RF_ID、DOT_ID	Check-Xor

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD5	0x0004	0x11	RF_ID、DOT_ID	RF 发射功率	Check-Xor
操作错误	0xAA	0xD5	0x0004	0x91	RF_ID、DOT_ID	错误码	Check-Xor

✧ RF 发射功率: 详情见 RF 发射功率

✧ 错误码：单个字节长度，详情见错误码表

9、断开 RF 连接

- ✧ 此指令：不识别 DOT_ID, DOT_ID 设置成 0xFF 即可
- ✧ RF_ID: 设置成 0x3F 即可 （专用协议时，RF_ID 需设置成指定 ID）
- ✧ API:
  - C: ul_disconnectIcRf
  - Python: ul_disconnectIcRf

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	校验码
0xAA	0xD5	0x0003	0x12	RF_ID、DOT_ID	Check-Xor

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD5	0x0003	0x12	RF_ID、DOT_ID		Check-Xor
操作错误	0xAA	0xD5	0x0004	0x92	RF_ID、DOT_ID	错误码	Check-Xor

- ✧ 操作成功：
  - 从 RF 接口执行该指令，指令执行成功，收不到返回值
- ✧ 错误码：单个字节长度，详情见错误码表

10、数据输出

- ✧ 此指令：不识别 DOT_ID, DOT_ID 设置成 0xFF 即可
- ✧ RF_ID: 设置成 0x3F 即可（专用协议时，RF_ID 需设置成指定 ID）

10.1 使能输出

- ➢ API:
  - C: ul_enableDataOutPut
  - Python: ul_enableDataOutPut

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	校验码
0xAA	0xD5	0x0003	0x14	RF_ID、DOT_ID	Check-Xor

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD5	0x0003	0x14	RF_ID、DOT_ID		Check-Xor
操作错误	0xAA	0xD5	0x0004	0x94	RF_ID、DOT_ID	错误码	Check-Xor

- ✧ 错误码：单个字节长度，详情见错误码表

10.2 禁止输出

- ➢ API:
  - C: ul_disEnableDataOutPut
  - Python: ul_disEnableDataOutPut

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	校验码
0xAA	0xD5	0x0003	0x15	RF_ID、DOT_ID	Check-Xor

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD5	0x0003	0x15	RF_ID、DOT_ID		Check-Xor
操作错误	0xAA	0xD5	0x0004	0x95	RF_ID、DOT_ID	错误码	Check-Xor

✧ 错误码：单个字节长度，详情见错误码表

11、LED 交互

✧ 此指令：不识别 DOT_ID, DOT_ID 设置成 0xFF 即可

✧ RF_ID: 设置成 0x3F 即可（专用协议时，RF_ID 需设置成指定 ID）

11.1 进入 LED 模式

➢ API:

- C: ul_enterLedInteraction
- Python: ul_enterLedInteraction

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	校验码
0xAA	0xD5	0x0003	0x60	RF_ID、DOT_ID	Check-Xor

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD5	0x0003	0x60	RF_ID、DOT_ID		Check-Xor
操作错误	0xAA	0xD5	0x0004	0xE0	RF_ID、DOT_ID	错误码	Check-Xor

✧ 错误码：单个字节长度，详情见错误码表

11.2 退出 LED 模式

➢ API:

- C: ul_exitLedInteraction
- Python: ul_exitLedInteraction

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	校验码
0xAA	0xD5	0x0003	0x61	RF_ID、DOT_ID	Check-Xor

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD5	0x0003	0x61	RF_ID、DOT_ID		Check-Xor
操作错误	0xAA	0xD5	0x0004	0xE1	RF_ID、DOT_ID	错误码	Check-Xor

✧ 错误码：单个字节长度，详情见错误码表

11.3 设置 LED 颜色

- API:
  - C: ul_modifyLedInteractionColor
  - Python: ul_modifyLedInteractionColor

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	数据段	校验码
0xAA	0xD5	0x0005	0x62	RF_ID、DOT_ID	LED 交互参数	Check-Xor

- LED 交互参数:
  - 2 字节（颜色 + 交互方式）
  - 数据段排列顺序：颜色、交互方式

LED 交互参数	数据类型	Value
颜色	uint8_t	0x01：红色
		0x02：绿色
		0x03：黄色
		0x04：蓝色
		0x05：紫色
		0x06：青色
		0x07：白色
交互方式	uint8_t	0x00: 长亮
		0x01: 呼吸灯
		0x02: 中频闪
		0x03: 高频闪

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD5	0x0003	0x62	RF_ID、DOT_ID		Check-Xor
操作错误	0xAA	0xD5	0x0004	0xE2	RF_ID、DOT_ID	错误码	Check-Xor

- 错误码：单个字节长度，详情见错误码表

11.4 读取 LED 颜色

- API:
  - C: ul_getLedInteractionColor
  - Python: ul_getLedInteractionColor

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	校验码
0xAA	0xD5	0x0003	0x63	RF_ID、DOT_ID	Check-Xor

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD5	0x0005	0x63	RF_ID、DOT_ID	LED 交互参数	Check-Xor
操作错误	0xAA	0xD5	0x0004	0xE3	RF_ID、DOT_ID	错误码	Check-Xor

- LED 交互参数：详情见 LED 颜色与交互方式配置

✧ 错误码：单个字节长度，详情见错误码表

12、配置串口波特率

✧ 此指令：不识别 DOT_ID, DOT_ID 设置成 0xFF 即可

✧ RF_ID: 设置成 0x3F 即可（专用协议时，RF_ID 需设置成指定 ID）

✧ 串口上电默认波特率：115200

12.1 修改串口波特率

➢ API:

- C: ul_modifyUartBaudRate
- Python: ul_modifyUartBaudRate

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	数据段	校验码
0xAA	0xD5	0x0007	0x64	RF_ID、DOT_ID	串口波特率	Check-Xor

串口波特率：数据类型 uint32_t, 4 字节，异步时钟通信方式，速率越高误码率越高

串口波特率	数据段编码	备注
115200	0x1C200	上电缺省值
128000	0x1F400	
256000	0x3E800	
460800	0x70800	
512000	0x7D000	
750000	0xB71B0	
921600	0xE1000	

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD5	0x0003	0x64	RF_ID、DOT_ID		Check-Xor
操作错误	0xAA	0xD5	0x0004	0xE4	RF_ID、DOT_ID	错误码	Check-Xor

✧ 错误码：单个字节长度，详情见错误码表

12.2 读取串口波特率

➢ API:

- C: ul_getUartBaudRate
- Python: ul_getUartBaudRate

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	校验码
0xAA	0xD5	0x0003	0x65	RF_ID、DOT_ID	Check-Xor

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD5	0x0007	0x65	RF_ID、DOT_ID	串口波特率	Check-Xor
操作错误	0xAA	0xD5	0x0004	0xE5	RF_ID、DOT_ID	错误码	Check-Xor

- ✧ 串口波特率： 详情见串口波特率配置
- ✧ 错误码：单个字节长度，详情见错误码表

13、六面静态校准

- ✧ 此指令：不识别 DOT_ID, DOT_ID 设置成 0xFF 即可
- ✧ RF_ID: 设置成 0x3F 即可（专用协议时，RF_ID 需设置成指定 ID）
- ✧ 建议配合 RGB 灯使用，每面约耗时 12S(秒)
- ✧ API:
  - C:
    - ul_imuStaticCalibrationInit
    - ul_imuStaticCalibration
    - ul_imuStaticCalibrationExit
    - ul_clearStaticCalibrationParam
  - Python:
    - ul_imuStaticCalibrationInit
    - ul_imuStaticCalibration
    - ul_imuStaticCalibrationExit
    - ul_clearStaticCalibrationParam

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	数据段	校验码
0xAA	0xD5	0x0005	0x6E	RF_ID、DOT_ID	校准指令	Check-Xor

校准指令：1 字节

数据类型		Value
校准指令	uint8_t	0xFF: 六面参数初始化
		0x01: 六面校准
		0x00: 六面校准结束

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD5	0x0003	0x6E	RF_ID、DOT_ID		Check-Xor
操作错误	0xAA	0xD5	0x0004	0xEE	RF_ID、DOT_ID	错误码	Check-Xor

- ✧ 错误码：单个字节长度，详情见错误码表

14、获取 MAC 地址

✧ 此指令：不识别 DOT_ID, DOT_ID 设置成 0xFF 即可

✧ RF_ID: 设置成 0x3F 即可 （专用协议时，RF_ID 需设置成指定 ID）

✧ API:

- C: ul_getMacAddressStr
- Python: ul_getMacAddressStr

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	校验码
0xAA	0xD5	0x0003	0x77	RF_ID、DOT_ID	Check-Xor

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD5	0x0014	0x77	RF_ID、DOT_ID	MAC 地址	Check-Xor
操作错误	0xAA	0xD5	0x0004	0xF7	RF_ID、DOT_ID	错误码	Check-Xor

✧ MAC 地址：

- 字符串类型
- 格式 “XX:XX:XX:XX:XX:XX”

✧ 错误码：单个字节长度，**详情见错误码表**

15、获取设备完整序列号

✧ 此指令：不识别 DOT_ID, DOT_ID 设置成 0xFF 即可

✧ RF_ID: 设置成 0x3F 即可 （专用协议时，RF_ID 需设置成指定 ID）

✧ API:

- C: ul_getDeviceSnStr
- Python: ul_getDeviceSnStr

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	校验码
0xAA	0xD5	0x0003	0x79	RF_ID、DOT_ID	Check-Xor

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD5	0x001A	0x79	RF_ID、DOT_ID	设备序号	Check-Xor
操作错误	0xAA	0xD5	0x0004	0xF9	RF_ID、DOT_ID	错误码	Check-Xor

✧ 设备序号：

- 字符串类型
- 格式“ZLXX-XXXX-XXXX-XXXXXXXX”

✧ 错误码：单个字节长度，详情见错误码表

16、获取硬件版本号

- ✧ 此指令：不识别 DOT_ID, DOT_ID 设置成 0xFF 即可
- ✧ RF_ID: 设置成 0x3F 即可 （专用协议时，RF_ID 需设置成指定 ID）
- ✧ API:
  - ■ C: ul_getBoardVesionStr
  - ■ Python: ul_getBoardVesionStr

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	校验码
0xAA	0xD5	0x0003	0x7B	RF_ID、DOT_ID	Check-Xor

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD5	N	0x7B	RF_ID、DOT_ID	硬件版本号	Check-Xor
操作错误	0xAA	0xD5	0x0004	0xFB	RF_ID、DOT_ID	错误码	Check-Xor

- ✧ 数据区长度：硬件版本号字符串字节长度 + 3
- ✧ 硬件版本号：字符串类型
- ✧ 错误码：单个字节长度，详情见错误码表

17、获取固件版本号

- ✧ 此指令：不识别 DOT_ID, DOT_ID 设置成 0xFF 即可
- ✧ RF_ID: 设置成 0x3F 即可 （专用协议时，RF_ID 需设置成指定 ID）
- ✧ API:
  - ■ C: ul_getFirmwareVesionStr
  - ■ Python: ul_getFirmwareVesionStr

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	校验码
0xAA	0xD5	0x0003	0x7D	RF_ID、DOT_ID	Check-Xor

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD5	N	0x7D	RF_ID、DOT_ID	固件版本号	Check-Xor
操作错误	0xAA	0xD5	0x0004	0xFD	RF_ID、DOT_ID	错误码	Check-Xor

- ✧ 数据区长度：固件版本号字符串字节长度 + 3
- ✧ 软件版本号：字符串类型
- ✧ 错误码：单个字节长度，详情见错误码表

18、设备关机（或重启）

- ✧ 此指令：不识别 DOT_ID, DOT_ID 设置成 0xFF 即可
- ✧ RF_ID: 设置成 0x3F 即可（专用协议时，RF_ID 需设置成指定 ID）
- ✧ API:
  - C: ul_deviceShutdown
  - Python: ul_deviceShutdown

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	校验码
0xAA	0xD5	0x0003	0x7E	RF_ID、DOT_ID	Check-Xor

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD5	0x0003	0x7E	RF_ID、DOT_ID		Check-Xor
操作错误	0xAA	0xD5	0x0004	0xFE	RF_ID、DOT_ID	错误码	Check-Xor

- ✧ 指令执行成功，部分时刻收不到返回值
- ✧ 错误码：单个字节长度，详情见错误码表

19、恢复出厂参数

- ✧ 此指令：不识别 DOT_ID, DOT_ID 设置成 0xFF 即可
- ✧ RF_ID: 设置成 0x3F 即可（专用协议时，RF_ID 需设置成指定 ID）
- ✧ 指令设置成功后，设备自动重启(或关机)
- ✧ API:
  - C: ul_restoreFactorySettings
  - Python: ul_restoreFactorySettings

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	校验码
0xAA	0xD5	0x0003	0x7F	RF_ID、DOT_ID	Check-Xor

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD5	0x0003	0x7F	RF_ID、DOT_ID		Check-Xor
操作错误	0xAA	0xD5	0x0004	0xFF	RF_ID、DOT_ID	错误码	Check-Xor

- ✧ 指令执行成功，部分时刻收不到返回值
- ✧ 错误码：单个字节长度，详情见错误码表

四、高级指令格式

1、配置 RF Conn Interval （Ble 协议）

✧ 此指令：不识别 DOT_ID, DOT_ID 设置成 0xFF 即可

✧ RF_ID: 设置成 0x3F 即可（专用协议时，RF_ID 需设置成指定 ID）

1.1 修改 RF Conn Interval

✧ API:

- C: hl_modifyRfConnInterval
- Python: hl_modifyRfConnInterval

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	数据段	校验码
0xAA	0xD6	0x0007	0x06	RF_ID、DOT_ID	Conn interval	Check-Xor

✧ Conn interval:

- 数据类型 float, 4 字节
- 设置范围 7.5ms~100.0ms, 步长 1.25ms

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD6	0x0003	0x06	RF_ID、DOT_ID		Check-Xor
操作错误	0xAA	0xD6	0x0004	0x86	RF_ID、DOT_ID	错误码	Check-Xor

✧ 错误码：单个字节长度，详情见错误码表

1.2 读取 RF Conn Interval

✧ API:

- C: hl_getRfConnInterval
- Python: hl_getRfConnInterval

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	校验码
0xAA	0xD6	0x0003	0x07	RF_ID、DOT_ID	Check-Xor

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD6	0x0007	0x07	RF_ID、DOT_ID	Conn interval	Check-Xor
操作错误	0xAA	0xD6	0x0004	0x87	RF_ID、DOT_ID	错误码	Check-Xor

✧ Conn interval: 详情见设置蓝牙 Conn interval

✧ 错误码：单个字节长度，详情见错误码表

2、配置加速度计量程

✧ 此指令：不识别 DOT_ID, DOT_ID 设置成 0xFF 即可

✧ RF_ID: 设置成 0x3F 即可（专用协议时，RF_ID 需设置成指定 ID）

2.1 修改加速度计量程

（芯片重启后生效）

✧ API:

- C: hl_modifyAccRange
- Python: hl_modifyAccRange

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	数据段	校验码
0xAA	0xD6	0x0004	0x10	RF_ID、DOT_ID	加速度计量程	Check-Xor

加速度计量程编码

加速度计量程	数据段编码
±2G	0x00
±4G	0x01
±8G	0x02
±16G	0x03

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD6	0x0003	0x10	RF_ID、DOT_ID		Check-Xor
操作错误	0xAA	0xD6	0x0004	0x90	RF_ID、DOT_ID	错误码	Check-Xor

✧ 错误码：单个字节长度，详情见错误码表

2.2 读取加速度计量程

✧ API:

- C: hl_getAccRange
- Python: hl_getAccRange

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	校验码
0xAA	0xD6	0x0003	0x11	RF_ID、DOT_ID	Check-Xor

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD6	0x0004	0x11	RF_ID、DOT_ID	加速度计量程	Check-Xor
操作错误	0xAA	0xD6	0x0004	0x91	RF_ID、DOT_ID	错误码	Check-Xor

✧ 加速度计量程： 详情见加速度量程编码

✧ 错误码：单个字节长度，详情见错误码表

3、配置陀螺仪量程

✧ 此指令：不识别 DOT_ID, DOT_ID 设置成 0xFF 即可

✧ RF_ID: 设置成 0x3F 即可（专用协议时，RF_ID 需设置成指定 ID）

3.1 修改陀螺仪量程

（芯片重启后生效）

✧ API:

- C: hl_modifyGyroRange
- Python: hl_modifyGyroRange

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	数据段	校验码
0xAA	0xD6	0x0004	0x12	RF_ID、DOT_ID	陀螺仪量程	Check-Xor

陀螺仪量程编码

陀螺仪量程	数据段编码	类别
±250dps	0x00	
±500dps	0x01	
±1000dps	0x02	
±2000dps	0x03	
±4000dps	0x04	特定 IC 支持

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD6	0x0003	0x12	RF_ID、DOT_ID		Check-Xor
操作错误	0xAA	0xD6	0x0004	0x92	RF_ID、DOT_ID	错误码	Check-Xor

✧ 错误码：单个字节长度，详情见错误码表

3.2 读取陀螺仪量程

✧ API:

- C: hl_getGyroRange
- Python: hl_getGyroRange

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	校验码
0xAA	0xD6	0x0003	0x13	RF_ID、DOT_ID	Check-Xor

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD5	0x0004	0x13	RF_ID、DOT_ID	陀螺仪量程	Check-Xor
操作错误	0xAA	0xD5	0x0004	0x93	RF_ID、DOT_ID	错误码	Check-Xor

✧ 陀螺仪量程： 详情见陀螺仪量程编码

✧ 错误码：单个字节长度，详情见错误码表

4、磁力计椭球拟合参数（用户设置区）

✧ 此指令：识别 DOT_ID, 对于非专用协议 IC，DOT_ID 设置成 0xFF 即可

✧ RF_ID: 设置成 0x3F 即可（专用协议时，RF_ID 需设置成指定 ID）

4.1 修改磁力计椭球拟合参数

✧ API:

- C: hl_modifyMagCalParam_Ex
- Python: hl_modifyMagCalParam_Ex

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	数据段	校验码
0xAA	0xD6	0x001B	0x1A	RF_ID、DOT_ID	磁力计椭球拟合参数	Check-Xor

磁力计椭球拟合参数：

✧ kX kY kZ： 磁力计 XYZ 放大系数

- kX =（Rx + Ry + Rz）/ (3.0 * Rx)
- kY =（Rx + Ry + Rz）/ (3.0 * Ry)
- kZ =（Rx + Ry + Rz）/ (3.0 * Rz)

✧ Ox Oy Oz： 磁力计 XYZ 零点 Offset（椭球拟合的圆心）

✧ 排列顺序：kX kY kZ Ox Oy Oz

- 数据类型：float， 小端存储格式

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD6	0x0003	0x1A	RF_ID、DOT_ID		Check-Xor
操作错误	0xAA	0xD6	0x0004	0x9A	RF_ID、DOT_ID	错误码	Check-Xor

✧ 错误码：单个字节长度，详情见错误码表

4.2 读取磁力计椭球拟合参数

✧ API:

- C: hl_getMagCalParam_Ex
- Python: hl_getMagCalParam_Ex

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	校验码
0xAA	0xD6	0x0003	0x1B	RF_ID、DOT_ID	Check-Xor

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD6	0x001B	0x1B	RF_ID、DOT_ID	磁力计椭球拟合参数	Check-Xor
操作错误	0xAA	0xD6	0x0004	0x9B	RF_ID、DOT_ID	错误码	Check-Xor

✧ 磁力计椭球拟合参数：详情见磁力计椭球拟合参数

✧ 错误码：单个字节长度，详情见错误码表

5、配置流水号格式

✧ 此指令：不识别 DOT_ID, DOT_ID 设置成 0xFF 即可

✧ RF_ID: 设置成 0x3F 即可（专用协议时，RF_ID 需设置成指定 ID）

5.1 修改流水号格式

✧ API:

- C: hl_configFlowFormat
- Python: hl_configFlowFormat

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	数据段	校验码
0xAA	0xD6	0x0004	0x20	RF_ID、DOT_ID	流水号格式	Check-Xor

流水号格式编码：数据类型：uint8_t

流水号格式	编码	说明
8 位流水号	0x00	流水号 范围 0x00 - 0xFF
16 位流水号	0x01	流水号 范围 0x0000 - 0xFFFF

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD6	0x0003	0x20	RF_ID、DOT_ID		Check-Xor
操作错误	0xAA	0xD6	0x0004	0xA0	RF_ID、DOT_ID	错误码	Check-Xor

✧ 错误码：单个字节长度，详情见错误码表

5.3 读取流水号格式

✧ API:

- C: hl_getFlowFormat
- Python: hl_getFlowFormat

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	校验码
0xAA	0xD6	0x0003	0x21	RF_ID、DOT_ID	Check-Xor

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD6	0x0004	0x21	RF_ID、DOT_ID	流水号格式	Check-Xor
操作错误	0xAA	0xD6	0x0004	0xA1	RF_ID、DOT_ID	错误码	Check-Xor

✧ 流水号格式：详情见流水号格式编码

✧ 错误码：单个字节长度，详情见错误码表

6、重置流水号

- ✧ 此指令：不识别 DOT_ID, DOT_ID 设置成 0xFF 即可
- ✧ RF_ID: 设置成 0x3F 即可（专用协议时，RF_ID 需设置成指定 ID）
- ✧ API:
  - ■ C: hl_resetFlowNums
  - ■ Python: hl_resetFlowNums

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	校验码
0xAA	0xD6	0x0003	0x22	RF_ID、DOT_ID	Check-Xor

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD6	0x0003	0x22	RF_ID、DOT_ID		Check-Xor
操作错误	0xAA	0xD6	0x0004	0xA2	RF_ID、DOT_ID	错误码	Check-Xor

- ✧ 错误码：单个字节长度，详情见错误码表

7、配置数据输出接口

- ✧ 此指令：不识别 DOT_ID, DOT_ID 设置成 0xFF 即可
- ✧ RF_ID: 设置成 0x3F 即可（专用协议时，RF_ID 需设置成指定 ID）

7.1 修改数据输出端口

- ✧ API:
  - ■ C: hl_configOutDataPort
  - ■ Python: hl_configOutDataPort

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	数据段	校验码
0xAA	0xD6	0x0005	0x30	RF_ID、DOT_ID	输出端口编码	Check-Xor

输出端口编码：

- ✧ 数据格式：uint16_t
- ✧ **注**：对应的输出端口，需要开启才能启动

输出端口	编码	说明
None	0x0000	禁止数据输出
RF	0x0001	数据从 RF 端口输出
UART	0x0002	数据从 UART 端口输出
SPIM	0x0008	数据从 SPIM 端口输出

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD6	0x0005	0x31	RF_ID、DOT_ID	输出端口	Check-Xor
操作错误	0xAA	0xD6	0x0004	0xB1	RF_ID、DOT_ID	错误码	Check-Xor

- 输出端口：详情见输出端口编码
- 错误码：单个字节长度，详情见错误码表

7.2 读取数据输出端口

- API:
  - C: hl_getOutDataPort
  - Python: hl_getOutDataPort

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	校验码
0xAA	0xD6	0x0003	0x31	RF_ID、DOT_ID	Check-Xor

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD6	0x0005	0x31	RF_ID、DOT_ID	输出端口	Check-Xor
操作错误	0xAA	0xD6	0x0004	0xB1	RF_ID、DOT_ID	错误码	Check-Xor

- 输出端口：见数据输出端口查询项部分
- 错误码：单个字节长度，详情见错误码表

7.3 数据输出端口检查

- API:
  - C: hl_checkOutDataPort
  - Python: hl_checkOutDataPort

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	校验码
0xAA	0xD6	0x0003	0x33	RF_ID、DOT_ID	Check-Xor

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD6	0x0005	0x33	RF_ID、DOT_ID	输出端口 MAP	Check-Xor
操作错误	0xAA	0xD6	0x0004	0xB3	RF_ID、DOT_ID	错误码	Check-Xor

- 错误码：单个字节长度，详情见错误码表

✧ 输出端口 MAP：

- 数据格式：uint16_t，2 个字节

输出端口 Map	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
value	0	0	0	0	0	0	0	0
	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
value	0	0	0	0	SPIM	0	UART	RF

8、配置 UART

✧ 此指令：不识别 DOT_ID，DOT_ID 设置成 0xFF 即可

✧ RF_ID: 设置成 0x3F 即可（专用协议时，RF_ID 需设置成指定 ID）

8.1 交换 UART TxPin、RxPin

✧ API:

- C: hl_swapUserUartTrxPin
- Python: hl_swapUserUartTrxPin

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	校验码
0xAA	0xD6	0x0003	0x60	RF_ID、DOT_ID	Check-Xor

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD6	0x0003	0x60	RF_ID、DOT_ID		Check-Xor
操作错误	0xAA	0xD6	0x0004	0xE0	RF_ID、DOT_ID	错误码	Check-Xor

- 错误码：单个字节长度，详情见错误码表

9、配置 SPIM

✧ 此指令：不识别 DOT_ID，DOT_ID 设置成 0xFF 即可

✧ RF_ID: 设置成 0x3F 即可（专用协议时，RF_ID 需设置成指定 ID）

✧ 该指令，掉电失效！！！

9.1 开启、关闭 SPIM

（重启后失效）

✧ API:

- C:
  - hl_enableUserSpim
  - hl_disEnableUserSpim
- Python:
  - hl_enableUserSpim
  - hl_disEnableUserSpim

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	数据段	校验码
0xAA	0xD6	0x0004	0x64	RF_ID、DOT_ID	开启/关闭	Check-Xor

✧ 开启/关闭:

- 数据类型：uint8_t
- 开启：0x01
- 关闭：0x00

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD6	0x0003	0x64	RF_ID、DOT_ID		Check-Xor
操作错误	0xAA	0xD6	0x0004	0xE4	RF_ID、DOT_ID	错误码	Check-Xor

✧ 错误码：单个字节长度，详情见错误码表

9.2 读取 SPIM IO

✧ API:

- C: hl_getUserSpimIO
- Python: hl_getUserSpimIO

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	校验码
0xAA	0xD6	0x0003	0x65	RF_ID、DOT_ID	Check-Xor

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD6	0x000C	0x65	RF_ID、DOT_ID	SPIM 结构	Check-Xor
操作错误	0xAA	0xD6	0x0004	0xE5	RF_ID、DOT_ID	错误码	Check-Xor

- ✧ SPIM 结构：
  - ■ 使能：数据类型 uint8_t
    - ◆ 开启：0x01
    - ◆ 关闭：0x00
  - ■ 模式：数据类型 uint8_t
  - ■ bitOrder：数据类型 uint8_t
  - ■ SPI 速率：数据类型 uint8_t
  - ■ SPI 传输 Block 大小：
    - ◆ 数据类型：uint8_t
  - ■ sckPin：
    - ◆ 数据类型：uint8_t
  - ■ misoPin:
    - ◆ 数据类型：uint8_t
  - ■ mosiPin:
    - ◆ 数据类型：uint8_t
  - ■ csnPin：
    - ◆ 数据类型：uint8_t
- ✧ 错误码：单个字节长度，详情见错误码表

10、配置 ANT（ADC）

- ✧ 此指令：不识别 DOT_ID, DOT_ID 设置成 0xFF 即可
- ✧ RF_ID: 设置成 0x3F 即可（专用协议时，RF_ID 需设置成指定 ID）

10.1 开启、关闭 ANT(ADC)

- ✧ API:
  - ■ C: hl_configUserAnt
  - ■ Python: hl_configUserAnt

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	数据段	校验码
0xAA	0xD6	0x0004	0x6A	RF_ID、DOT_ID	Ant Enable	Check-Xor

- ✧ Ant Enable：
  - ● 数据类型：uint8_t

BIT	ANT	开启/关闭	备注
7	/	0	
6	/	0	
5	Ant5	1/ 0	1: 开启，0：关闭
4	Ant4	1/ 0	1: 开启，0：关闭
3	Ant3	1/ 0	1: 开启，0：关闭
2	Ant2	1/ 0	1: 开启，0：关闭
1	Ant1	1/ 0	1: 开启，0：关闭
0	Ant0	1/ 0	1: 开启，0：关闭

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD6	0x0003	0x6C	RF_ID、DOT_ID		Check-Xor
操作错误	0xAA	0xD6	0x0004	0xEC	RF_ID、DOT_ID	错误码	Check-Xor

✧ 错误码：单个字节长度，详情见错误码表

10.2 读取 ANT IO

✧ API:

- C: hl_getUserAntIO
- Python: hl_getUserAntIO

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	校验码
0xAA	0xD6	0x0003	0x6B	RF_ID、DOT_ID	Check-Xor

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD6	0x000A	0x6B	RF_ID、DOT_ID	ANT 结构	Check-Xor
操作错误	0xAA	0xD6	0x0004	0xEB	RF_ID、DOT_ID	错误码	Check-Xor

✧ ANT 结构：

- Ant Enable：数据类型：uint8_t
- Ant0 Pin：数据类型：uint8_t
- Ant1 Pin：数据类型：uint8_t
- Ant2 Pin：数据类型：uint8_t
- Ant3 Pin：数据类型：uint8_t
- Ant4 Pin：数据类型：uint8_t
- Ant5 Pin：数据类型：uint8_t

✧ 错误码：单个字节长度，详情见错误码表

11、配置 电量检测

✧ 此指令：不识别 DOT_ID, DOT_ID 设置成 0xFF 即可

✧ RF_ID: 设置成 0x3F 即可（专用协议时，RF_ID 需设置成指定 ID）

11.1 开启、关闭 电池检测

✧ API:

- C:
  - hl_enableUserBattery
  - hl_disEnableUserBattery
- Python:
  - hl_enableUserBattery
  - hl_disEnableUserBattery

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	数据段	校验码
0xAA	0xD6	0x0004	0x6C	RF_ID、DOT_ID	开启/关闭	Check-Xor

✧ 开启/关闭:
- 数据类型：uint8_t
- 开启：0x01
- 关闭：0x00

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD6	0x0003	0x6C	RF_ID、DOT_ID		Check-Xor
操作错误	0xAA	0xD6	0x0004	0xEC	RF_ID、DOT_ID	错误码	Check-Xor

✧ 错误码：单个字节长度，详情见错误码表

11.2 读取 电量检测 IO

✧ API:
- C: hl_getUserBatteryIO
- Python: hl_getUserBatteryIO

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	校验码
0xAA	0xD6	0x0003	0x6D	RF_ID、DOT_ID	Check-Xor

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD6	0x0005	0x6D	RF_ID、DOT_ID	电量检测 结构	Check-Xor
操作错误	0xAA	0xD6	0x0004	0xED	RF_ID、DOT_ID	错误码	Check-Xor

✧ 电量检测 结构：
- 使能：
  - 数据类型：uint8_t
  - 开启：0x01
  - 关闭：0x00
- 模式:
  - 数据类型：uint8_t
- ANT：
  - 数据类型：uint8_t

✧ 错误码：单个字节长度，详情见错误码表

11.3 读取电池电量

✧ API:
- C: hl_getUserBatteryLevel
- Python: hl_getUserBatteryLevel

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	校验码
0xAA	0xD6	0x0003	0x6F	RF_ID、DOT_ID	Check-Xor

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD6	0x0004	0x6F	RF_ID、DOT_ID	电量	Check-Xor
操作错误	0xAA	0xD6	0x0004	0xEF	RF_ID、DOT_ID	错误码	Check-Xor

✧ 电量：
- 数据类型： uint8_t
- 表示百分比：范围 0x00 ~ 0x64 （0% ~ 100%）

✧ 错误码：单个字节长度，详情见错误码表

12、配置 RGB LED

✧ 此指令：不识别 DOT_ID, DOT_ID 设置成 0xFF 即可

✧ RF_ID: 设置成 0x3F 即可（专用协议时，RF_ID 需设置成指定 ID）

12.1 开启、关闭RGB

✧ API:
- C:
  - hl_enableUserRgbLed
  - hl_disEnableUserRgbLed
- Python:
  - hl_enableUserRgbLed
  - hl_disEnableUserRgbLed

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	数据段	校验码
0xAA	0xD6	0x0004	0x70	RF_ID、DOT_ID	开启/关闭	Check-Xor

✧ 开启/关闭:
- 数据类型：uint8_t
- 开启：0x01
- 关闭：0x00

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD6	0x0003	0x70	RF_ID、DOT_ID		Check-Xor
操作错误	0xAA	0xD6	0x0004	0xF0	RF_ID、DOT_ID	错误码	Check-Xor

✧ 错误码：单个字节长度，详情见错误码表

12.2 读取 RGB IO

✧ API:
- C: hl_getUserRgbLedIO
- Python: hl_getUserRgbLedIO

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	校验码
0xAA	0xD6	0x0003	0x71	RF_ID、DOT_ID	Check-Xor

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD6	0x000A	0x71	RF_ID、DOT_ID	RGB 结构	Check-Xor
操作错误	0xAA	0xD6	0x0004	0xF1	RF_ID、DOT_ID	错误码	Check-Xor

✧ RGB 结构：
- 使能：
  - 数据类型：uint8_t
  - 开启：0x01
  - 关闭：0x00
- redPin：
  - IO 电平：uint8_t
  - IO 端口：uint8_t
- greenPin：
  - IO 电平：uint8_t
  - IO 端口：uint8_t
- bluePin：
  - IO 电平：uint8_t
  - IO 端口：uint8_t

✧ 错误码：单个字节长度，详情见错误码表

13、配置 Btn 按钮

✧ 此指令：不识别 DOT_ID，DOT_ID 设置成 0xFF 即可

✧ RF_ID: 设置成 0x3F 即可（专用协议时，RF_ID 需设置成指定 ID）

13.1 开启、关闭 Btn

✧ API:
- C:
  - hl_enableUserBtn
  - hl_disEnableUserBtn
- Python:
  - hl_enableUserBtn
  - hl_disEnableUserBtn

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	数据段	校验码
0xAA	0xD6	0x0004	0x72	RF_ID、DOT_ID	开启/关闭	Check-Xor

✧ 开启/关闭:
- 数据类型：uint8_t
- 开启：0x01
- 关闭：0x00

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD6	0x0003	0x72	RF_ID、DOT_ID		Check-Xor
操作错误	0xAA	0xD6	0x0004	0xF2	RF_ID、DOT_ID	错误码	Check-Xor

✧ 错误码：单个字节长度，详情见错误码表

13.2 读取 Btn IO

✧ API:

- C: hl_getUserBtnIO
- Python: hl_getUserBtnIO

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	校验码
0xAA	0xD6	0x0003	0x73	RF_ID、DOT_ID	Check-Xor

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD6	0x0006	0x73	RF_ID、DOT_ID	Btn 结构	Check-Xor
操作错误	0xAA	0xD6	0x0004	0xF3	RF_ID、DOT_ID	错误码	Check-Xor

✧ Btn 结构：

- 使能：
  - 数据类型：uint8_t
  - 开启：0x01
  - 关闭：0x00
- btnPin：
  - IO 电平：uint8_t
  - IO 端口：uint8_t

✧ 错误码：单个字节长度，详情见错误码表

14、配置 电源管理 IO

✧ 此指令：不识别 DOT_ID, DOT_ID 设置成 0xFF 即可

✧ RF_ID: 设置成 0x3F 即可（专用协议时，RF_ID 需设置成指定 ID）

14.1 开启、关闭 电源管理

✧ API:

- C:
  - hl_enableUserPowerEn
  - hl_disEnableUserPowerEn
- Python:
  - hl_enableUserPowerEn
  - hl_disEnableUserPowerEn

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	数据段	校验码
0xAA	0xD6	0x0004	0x74	RF_ID、DOT_ID	开启/关闭	Check-Xor

✧ 开启/关闭:

- 数据类型：uint8_t
- 开启：0x01
- 关闭：0x00

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD6	0x0003	0x74	RF_ID、DOT_ID		Check-Xor
操作错误	0xAA	0xD6	0x0004	0xF4	RF_ID、DOT_ID	错误码	Check-Xor

✧ 错误码：单个字节长度，详情见错误码表

14.2 读取 电源管理 IO

✧ API:

- C: hl_getUserPowerEnIO
- Python: hl_getUserPowerEnIO

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	校验码
0xAA	0xD6	0x0003	0x75	RF_ID、DOT_ID	Check-Xor

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD6	0x0006	0x75	RF_ID、DOT_ID	电源管理 结构	Check-Xor
操作错误	0xAA	0xD6	0x0004	0xF5	RF_ID、DOT_ID	错误码	Check-Xor

✧ 电源管理 结构：

- 使能：
  - 数据类型：uint8_t
  - 开启：0x01
  - 关闭：0x00
- powerPin：
  - IO 电平：uint8_t
  - IO 端口：uint8_t

✧ 错误码：单个字节长度，详情见错误码表

15、配置 RF

✧ 此指令：不识别 DOT_ID, DOT_ID 设置成 0xFF 即可

✧ RF_ID: 设置成 0x3F 即可（专用协议时，RF_ID 需设置成指定 ID）

15.1 开启 RF

仅支持开启不支持关闭，默认开启由硬件配置启动 （重启后失效）

✧ API:

- C: hl_enableUserRf
- Python: hl_enableUserRf

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	数据段	校验码
0xAA	0xD6	0x0004	0x76	RF_ID、DOT_ID	开启/关闭	Check-Xor

✧ 开启/关闭:

- 数据类型：uint8_t
- 开启：0x01
- 关闭：0x00

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD6	0x0003	0x76	RF_ID、DOT_ID		Check-Xor
操作错误	0xAA	0xD6	0x0004	0xF6	RF_ID、DOT_ID	错误码	Check-Xor

✧ 错误码：单个字节长度，详情见错误码表

15.2 开启 / 关闭 RF PA

配置完成后，设备重启后有效，该配置掉电保存

✧ API:

- C:
  - hl_enableUserRfPa
  - hl_disEnableUserRfPa
- Python:
  - hl_enableUserRfPa
  - hl_disEnableUserRfPa

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	数据段	校验码
0xAA	0xD6	0x0004	0x78	RF_ID、DOT_ID	开启/关闭	Check-Xor

✧ 开启/关闭:

- 数据类型：uint8_t
- 开启：0x01
- 关闭：0x00

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD6	0x0003	0x78	RF_ID、DOT_ID		Check-Xor
操作错误	0xAA	0xD6	0x0004	0xF8	RF_ID、DOT_ID	错误码	Check-Xor

✧ 错误码：单个字节长度，详情见错误码表

15.3 读取 RF PA IO

✧ API:

- C: hl_getUserRfPaIO
- Python: hl_getUserRfPaIO

上位机写操作指令帧，如下表所示：

帧头	指令 ID	数据区长度	子指令 ID	MT_ID	校验码
0xAA	0xD6	0x0003	0x79	RF_ID、DOT_ID	Check-Xor

返回帧：

	帧头	指令 ID	数据区长度	应答 ID	MT_ID	数据段	校验码
操作成功	0xAA	0xD6	0x0008	0x79	RF_ID、DOT_ID	PA 结构	Check-Xor
操作错误	0xAA	0xD6	0x0004	0xF9	RF_ID、DOT_ID	错误码	Check-Xor

✧ PA 结构：

- 使能：
  - 数据类型：uint8_t
  - 开启：0x01
  - 关闭：0x00
- txEnPin：
  - IO 电平：uint8_t
  - IO 端口：uint8_t
- rxEnPin：
  - IO 电平：uint8_t
  - IO 端口：uint8_t

✧ 错误码：单个字节长度，详情见错误码表

16、高级指令中，参数编码

16.1 GPIO 相关编码

GPIO 模式	GPIO 状态	编码	备注
IO 上下拉方式	无	0x00	不进行任何上下拉处理
	IO 下拉	0x01	下拉到 GND
	IO 上拉	0x03	上拉到 VDD
IO 电平	高电平	0x01	
	低电平	0x00	
IO 触发模式	上升沿	0x01	
	下降沿	0x02	
	电平翻转	0x03	

16.2 SPI 相关编码

	SPI	编码	备注
SPI 速率	125kHz	0x01	
	250kHz	0x02	
	500kHz	0z03	
	1MHz	0x0A	
	2MHz	0x0B	
	4MHz	0x0C	
	8MHz	0x0D	
SPI 模式	模式 0	0x00	
	模式 1	0x01	
	模式 2	0x02	
	模式	0x03	
SPI Bit 输出模式	MSB First	0x00	
	LSB First	0x01	

16.3 IIC 速率编码

IIC 速率	编码
100kHz	0x0064
250kHz	0x00FA
400kHz	0x0190

16.4 电量模式编码

电量模式	编码
百分比 + 电压(mv)	0x00
电压(mv)	0x01
百分比	0x02

16.5 ANT 端口 编码

ANT 端口	编码
ANT 0	0x01
ANT 1	0x02
ANT 2	0x03
ANT 3	0x04
ANT 4	0x05
ANT 5	0x06
ANT VDD	0x09

16.6 IC IO 编码

ZL9Nxx1	IO	编码	备注
A5	IO_0	0x00	数字 IO
A6	IO_1	0x01	数字 IO
A7	IO_2	0z02	数字 IO
A8	IO_3	0x03	数字 IO
B4	IO_4	0x04	数字 IO
B5	IO_5	0x05	数字 IO
B6	IO_6	0x06	数字 IO
H1	IO_7	0x07	数字 IO
J1	IO_8	0x08	数字 IO
L1	IO_9	0x09	数字 IO
L2	IO_10	0x0A	数字 IO
C1	ANT_0	0x01	模拟 IO
C2	ANT_1	0x02	模拟 IO
E2	ANT_2	0x03	模拟 IO
F1	ANT_3	0x04	模拟 IO
H2	ANT_4	0x05	模拟 IO
J2	ANT_5	0x06	模拟 IO

五、错误指令表

编号	错误标记	错误编号
1	0x00	无错误
2	0x01	包长度错误
3	0x02	未知指令类型
4	0x03	未知包格式
5	0x04	校验错误
-		
6	0x05	寄存器 ID 错误
7	0x06	Dot ID 不匹配
8	0x07	数据格式错误(或数据错误)
-		
9	0x0A	RF ID 不匹配
10	0x0B	RF 未连接
-		
11	0x0D	RF MAC 格式错误
12	0x0E	I/O 错误
13	0x10	功能未初始化
14	0x11	功能未配置
15	0x12	功能未开启

六、ZLBUS API

1.1 Python 库

1.2.1 安装

pip install -i https://pypi.org/simple/ pyZlBus

1.2.2 升级

pip install --upgrade -i https://pypi.org/simple/ pyZlBus

1.2.3 demo 测试运行：

```
# pyZlBus 库使用

## 1、pyZlBus 安装:  pip install -i https://pypi.org/simple/ pyZlBus

## 2、pyZlBus 升级:  pip install --upgrade -i https://pypi.org/simple/ pyZlBus


## 3、Demo 运行:

import pyZlBus.pyZlBus as zlb
import pyZlBus.test as ts

ts.bleDemo()
```

1.2 C 语言 dll 库

获取方式：AE