智阳软件 GPS 接入规范

V2.04.15

版本	修改记录	时间	作者
1.0	定义基本数据结构	2017/4/5	肖小波
1.1	增加了存储方式	2017/4/6	肖小波
1.2	增加了加密库接口	2017/4/7	肖小波
1.3	为了方便开发，修改了接口	2017/5/12	肖小波
1.4	增加了原始数据块说明	2017/8/12	肖小波
1.5	增加第三方加密经纬度支持	2018/5/22	
1.6	杰理平台增加文件末尾字节（附录三）	2018/10/22	肖小波
1.7	增加 FAQ	2018/12/07	胡小力
1.8	增加了速度不一致和播放进度不致的处理	2019/08/01	胡小力
2.0	增加了角度与高度	2019/11/01	肖小波
2.1	修复有些方案商集成时经纬度超长导致的数据超长问题	2019/11/25	肖小波
2.3	增加了经纬度解密函数接口的说明	2020/03/09	胡小力
2.5	增加了新 GPS 模块的描述	2020/7/22	胡小力
2.6	增加了说明	2020/08/20	胡小力
2.8	增加实时 GPS 数据显示功能	2021/05/10	胡小力
3.1	增加了参考代码	2024/04/18	肖小波
3.2	增加了区分加密的指导方法	2024/05/30	肖小波
3.3	多路文件命名参考	2024/08/27	肖小波

一．数据格式

数据头	数据块 0	.....	数据块 n
20 字节	N 字节	.....	N 字节

1 数据头

数据标识	加密算法	数据块个数
15 字节: **chIdentification**	1 字节：**nType**	4 字节:nCount

说明：

1) **chIdentification**：默认为字符串：“LIGOGPSINFO”；
2) **nType**:可以 type 5 或者 6,11，调试的时候需要跟我们确认，后面主要支持 11；
3) nCount:数据块的总数，每 1 秒生成一个数据块,，默认为 0xffff。

2 数据块

非加密数据块：

数据块恒定大小为：132 字节；

数据块	
数据快头	数据块内容
4 字节	128 字节
nFrame	

数据快头：

1) **nFrame** ：4 字节---帧数是不连续的。基本上，每 1 秒生成一个形式数据块，表示当 i 前这个数据快是第几秒。因此对 30fps 的 Movie 而言，每 30 帧生成一个行驶数据；对 60fps 的 Movie 而言，每 60 帧生成一个行驶数据，图片里面恒为 0，gpstxt 文件里面为索引每次自加 1；
2) **chInfomation**:固定 128 个字节，不足的地方填’\0’；

chInfomation 数据块说明：

Type 11 模块：

内容 ASII 码如下：

日期，时间，纬度，经度，海拔，速度，角度，ID，gsensorx，gsensory，gsensorz；

各个部分以逗号隔开；

如下例：

"100424,042324,23.943216,112.648182,0.000,63.672,239.41,36373939318C3A5A4330FFFFFFFFFFF,0.1,0.2,0.1"

GPS 原始数据如下：

$GXBDT11,A,220921,102308,31.228115,121.032504,0.000,40.934,0.00,000A220811156015*44

注意内容：

1 如果当前没有定位成功，经纬度全部为 0，G sensor 数据也为 0，每秒种采集一次，有定位数据就用真实数据，没有就用 0， N:22.587164 E:113.871490，解释如下：

a) N 为北半球的意思，如果在南半球则为 S，E 为东经，如果是西经则为 W;

b) 纬度 22.587164 为以度为单位的 gps 数据，比如 gps 模块获取到的数据为 1122.34566，需要转化为以度为单位的数据，转换公式为：

Float Value = 1122.34566 /100 (获取到度的整数) + 22.34566 /60.0f（把分转换为度，60 分为一度）；

经度 113.871490 同纬度一样需要同上公式进行转换，比如如果经度为 1122.23456，转换后为：

Float Value = 1122.23456/100 (获取到度的整数) + 22.23456/60.0f（把分转换为度，60 分为一度）；

2 速度单位（参考$GPRMC），速度为原始的 gps 里面的数据，不需要要进行转换为公里。单位为海里，请千万注意此处的 KM/H 只是一个识别符，而不是速度单位。

3 航向数据（参考$GPRMC），A:21.1,A 表示是航向数据，后面接角度 0 到 360，数据直接来源与 GPRMC，不需要转换;

4 高度（参考$GPGGA），H 表示是高度数据，后面的数据内容直接来源于$GPGGA，不要做任何转换，

5 Gsensor 只保存两位小数，每个方向偏角最多只保存两位小数，如 x:+0.00 y:-0.31 z:+0.93，单位是 G；

6 经纬度只允许小数点后保留 6 位，速度只保留一位小数，航向与高度只保留以为小数；

参考代码如下：

sprintf(chInfomation, ,

"100424,042324,23.943216,112.648182,0.000,63.672,239.41,36373939318C3A5A4330FFFFFFFFFFFF,0.1,0.2,0.1");

100424 ----24 年 4 月 10 号；

042324 --- 4 点 23 分 24 秒；

23.943216,112.648182-----纬度，经度

非 type11 模块（type 5 与 type6）：

内容 ASII 码如下：

“2014/04/24 10:20:34 N:22.587164 E:113.871490 29.47 km/h x:+0.00 y:-0.31 z:+0.98 A:12.1 H:123.5

注意内容：

1 如果当前没有定位成功，经纬度全部为 0，G sensor 数据也为 0，每秒种采集一次，有定位数据就用真实数据，没有就用 0， N:22.587164 E:113.871490，解释如下：

a) N 为北半球的意思，如果在南半球则为 S，E 为东经，如果是西经则为 W;

b) 纬度 22.587164 为以度为单位的 gps 数据，比如 gps 模块获取到的数据为 1122.34566，需要转化为以度为单位的数据，转换公式为：

Float Value = 1122.34566 /100 (获取到度的整数) + 22.34566 /60.0f（把分转换为度，60 分为一度）；

经度 113.871490 同纬度一样需要同上公式进行转换，比如如果经度为 1122.23456，转换后为：

Float Value = 1122.23456/100 (获取到度的整数) + 22.23456/60.0f（把分转换为度，60 分为一度）；

2 速度单位（参考$GPRMC），速度为原始的 gps 里面的数据，不需要要进行转换为公里。单位为海里，请千万注意此处的 KM/H 只是一个识别符，而不是速度单位。

3 航向数据（参考$GPRMC），A:21.1,A 表示是航向数据，后面接角度 0 到 360，数据直接来源与 GPRMC，不需要转换;

4 高度（参考$GPGGA），H 表示是高度数据，后面的数据内容直接来源于$GPGGA，不要做任何转换，

5 Gsensor 只保存两位小数，每个方向偏角最多只保存两位小数，如 x:+0.00 y:-0.31 z:+0.93，单位是 G;

6 经纬度只允许小数点后保留 6 位，速度只保留一位小数，航向与高度只保留以为小数；

参考代码如下：

```
sprintf(info->info, "%04d/%02d/%02d %02d:%02d:%02d %c:%.06f %c:%.06f %.01f km/h x:%.02f y:%.02f z:%.02f A:%.01f H:%.01f", rmcinfo.Year, rmcinfo.Month, rmcinfo.Day, rmcinfo.Hour, rmcinfo.Minute, rmcinfo.Second, rmcinfo.NSInd, rmcinfo.Latitude, rmcinfo.EWInd, rmcinfo.Longitude, rmcinfo.Speed, rmcinfo.fX,rmcinfo.fY,
```

加密数据块：

数据块大小不恒定

数据块		
数据快标识	数据块有效数据长度	数据内容
4 字节	4 字节	124 字节
chInfoTag	nInfoLength	**ch**InfoBy**E**ncryption

说明：

1） chInfoTag:”####” ，判断一个数据块是否

2）

3） 完整：1 找到 tag，然后找到长度，判断长度结束时候是否为下一个 tag，或者为 NULL（最后一个数据块）；

2) nInfoLength:**ch**InfoBy**E**ncryption 中有效数据的长度，非有效数据用 0 填充；

3) **ch**InfoBy**E**ncryption:加密数据，分两部分，一个是有效数据一个是填充数据，有效数据解密后数据格式如下，跟非加密数据块的格式一样，无效数据直接丢弃：

chInfoBy**E**ncryption	
数据快头	数据块内容
4 字节	120 字节
nFrame	**chInfomation**

跟非加密数据格式一致；

3 GPS 模块与数据（需要重点注意）

目前 GPS 模块支持两类，一种是 type11，一种是非 type11；

判断是否是 type11 的 gps 模块，只需要验证有没有以“$GXBDT”开头的语句；

后面将重点支持 type11.

注意：GPS 模块出来的 gps 数据保存在两个字符串里面：

1 以 GNGGA，里面的 GPS 数据是没有加密，可以用与水印上面，还有海拔数据，海拔的计算方式是，水平海拔减去椭圆形水平海拔；

2 以$GXBDT11,开头的字符串，只能保存到视频文件里面给播放器使用，不能用来做水印；

3 以$GXBDT5,开头的字符串，只能保存到视频文件里面给播放器使用，不能用来做水印；

注意：$GXBDT11 与$GXBDT5 开头的语句里面的内容完全一致，只需要判断“$GXBDT”即可；

Gps 模块输出的字段里面有一个新加的$GXBDT11,开头的字符串，这个里面有我们需要的所有数据：格式如下，$GXBDT11,是否定位，日期，时间，纬度，经度，速度，角度，高度，芯片 id，*校验码；如下参考数据：

$GXBDT11,A,220921,102308,31.228115,121.032504,0.000,40.934,0.00,000A220811156015*44

$GXBDT5,A,220921,102308,31.228115,121.032504,0.000,40.934,0.00,000A220811156015*44

只需要把其中的：日期，时间，纬度，经度，速度，角度，高度，芯片 id，加上 gsensor 的数据加入即可；

RMC 字段的 gps 数据只做水印，不能保存到视频里面；

Gsensor 数据加载后面，完整的数据包是,加入 gsensor 的 x:0.1 y:0.02 z:0.03 那么例子如下：

220921,102308,31.228115,121.032504,0.000,40.934,0.00,000A220811156015,0.01,0.02,0.0

二 加密库 Sdk 接口

1 设置加密类型

函数：int SunSetEncType(int type)

参数：type，5，6，或者 11，后面主要支持 11；
使用： 设置当前加密算法的类型与是否加密；

2 获取加密类型

函数：int SunGetEncType()
说明： 返回当前加密算法的类型；
使用： 用于给数据头的最有一个 byte；

3 生成数据头

函数： int MakeHeader(char *output,int count);
参数： output：长度为 20 个字节，count 为数据块个数；
使用： 生成数据头
返回值：成功返回 0，失败返回-1；
说明：
按照协议生成数据头的大小为 20 个字节，其中前面 15 个字节内容固定为“LIGOGPSINFO”，不够部分为 0，第十六个字节根据 SunSetEncType 设置而定，默认为 5；最后四个字节保存（int） count 的数据小端序保存。
例子：

4 生成加密数据块

函数： int MakeEncryptDataBlock(int nframe,char *chInfomation ,int chInfolength,unsigned char *chOutput);
参数： nframe：nframe 为协议中数据块中的 **nFrame** ，chInfomation 为后面的日期，时间，gps，gsensor 等字符串数据，chInfolength 为字符串数据长度，chOutput 输出加密后的数据块，长度恒为 132 字节；
返回值：成功返回 0，失败返回-1；
使用： 生成加密数据
例子（附件里有完整参考代码）参考附录三 type11 参考代码：
：

```
    unsigned char header[20];
//type 11 模块的加密数据实例如下
    char chInfomation[128] =
“220921,102308,31.228115,121.032504,0.000,40.934,0.00,000A220811156015,0.01,0.02,0.0”;
//非 type 11 模块的未加密数据实例如下
char chInfomation[128] = “2014/04/24 10:20:34 N:22.587164 E:113.871490 29.47 km/h x:+0.00 y:-0.31 z:+0.98 A:12.1 H:123.5”;
    unsigned char chOutput[132];
    int chInfolength = 100;
    //生成数据头
    memset(header,0,20);
```

```
SunSetEncType(11); //设置类型
MakeHeader(header,10);
//生成一条数据块
MakeEncryptDataBlock(10,chInfomation ,strlen(chInfomation),chOutput);
```

。。。。。。。。。。。。。

将 header（固定 20 字节）与 chOutput（固定 132 字节）写入文件相应位置即可。

获取正确的经纬度放在水印

1） Type11 模块：

从 gps 的语句$GNGGA，里面的 GPS 数据是没有加密，可以用与水印上面，还有海拔数据，海拔的计算方式是，水平海拔减去椭圆形水平海拔； GGA 代码解释参考附录伍；

2） 非 type11 模块：

这个里面的 gps 数据从$GPRMC 字段里面的 GPS 解码出来；

先优先使用函数：

```
void sunGpsGXDecode(int codectype,float *outflon, float *outfLat,float fLon, float fLat);
// 如果平台不支持 float,则使用如下函数：
int SunCoordTransformation(int codectype,
int *OutLong, //经度，出参
int *OutLat, //纬度,出参
int InLong,
//入参，经度
int InLat
//入参，纬度
);
```

需要注意以下几点：

Codectype:标示采用哪一种解码方式，目前只支持输入 5，与 6；

1. 出参和入参单位都是为度，国兴模块出来的单位是度分，所以需要先转换成度，再传参进去。千万不要传其它单位数据进去。

2. 因为很多平台不支持浮点数据，所以出参和入参请除以或者乘以 100000 得到整形数再使用。

三 GPS 数据存储

1 视频文件

保存在视频文件末尾，每秒钟有一个 gps 数据块，格式如下：

总体文件格式：

描述信息		数据内容
4 字节长度	4 字节描述信息	N 个字节
8+N	SKIP （或者 skip）	数据头+n 个数据块

描述信息：

8 个字节，前 4 个字节保存数据内容总长度+8；保存方式：高位在前（保存在低位储存区），大端序保存。

后 4 个字节是描述符: asii 码 'skip'或者'hgps'；优先建议使用 skip；

数据内容：

数据头	数据块 0	.....	数据块 n
20 字节	N 字节	.....	N 字节

2 图片文件

保存在视频文件末尾，一张图片只有一个 gps 数据，数据格式如下：

总体文件格式：

描述信息		数据内容	结束信息	
4 字节长度	4 字节描述信息	N 个字节	4 个字节描述符	四个字节长度
8+N+8	SKIP （或者 skip）	数据头+n 个数据块	“&&&&”	8+N+8

描述信息：

8 个字节，前 4 个字节保存数据内容总长度+8；保存方式：高位在前（保存在低位储存区），大端序保存。

后 4 个字节是描述符: asii 码 'skip'或者'hgps'；优先建议使用 skip；

数据内容：

数据头	数据块 0
20 字节	N 字节

其中：

chInfoBy**E**ncryption	
数据快头	数据块内容
4 字节	128 字节

nFrame	**chInfomation**

注意：nFrame 这个字段有所不同，在图片文件存储中，nFrame 字段无效，建议配置为 0。

结束信息：

1 前四个字节”&&&&”;

2 后四个字节保存为总长度（8+N+8），大端序方式存储；

3 txt 数据文件

1. 命名规则

日期+时间(时分秒,24 小时制)+gps.txt

比如：2017/04/07 01:01:02

创建的文件名字为：2017-04-07-01-01-02.gps.txt

当天数据只保存在同一个文件中。

2. 保存路径

保存在 sdcard/gps 目录下面。

3. 文件格式：

文件为 Asii 码字符文件，文件内容结构如下：

总体文件格式：

描述信息		数据内容
4 字节长度	4 字节描述信息	N 个字节
8+N	SKIP（或者 skip）	数据头+n 个数据块

描述信息：

文件长度（总长度）保存在 4 个字节里面，高位在前（保存在低位储存区），大端序保存。

数据内容：

数据头	数据块 0	.....	数据块 n
20 字节	N 字节	.....	N 字节

其中：

chInfoBy**E**ncryption	
数据快头	数据块内容
4 字节	128 字节
nFrame	**chInfomation**

注意：nFrame 这个字段有所不同，在 txt 文件存储中，nFrame 保存为 gps 的当前序号，比如当前文件的第一 gps 数据，nFrame 为 0，第二条为 1，一次累计第那条为 n；每个新文件的第一条 gps 数据的 nFrame 均为 0；

4. 文件操作：

App(应用软件) 使用 wifi,通过指令或者 http 文件协议等方式能读与删除 gps 数据文件。Gps 定位成功后，即开始保存 gps 数据，一直到关闭 gps 或者关机。当天数据只保存在同一个文件中。每个文件保存一天记录，一次最多保存一年。

四 实时 GPS 数据

1. 数据格式

   char chInfomation[128] = “2014/04/24 10:20:34 N:22.587164 E:113.871490 29.47 km/h x:+0.00 y:-0.31 z:+0.98 A:12.1 H:123.5”;

   注：该格式与写入视频中的数据格式一样，不需要经过加密处理

2. 数据传输

   采用 SOCKET 方式实时透传过来,CMD 为：8566

   <String>格式定义

字段	示例数据	释义
ligo	Zy01	标识数据来源，固定数据由智阳指定
type	加密类型	指示加密模块类型 现有 5,6,7,11 三种
gps	2014 / 04 / 24 10: 20: 34 N: 22.587164 E: 113.871490 29.47 km / h x: +0.00 y: -0.31 z: +0.98 A: 12.1 H: 123.5	GPS 存储到视频的中报文同样格式

3.数据返回格式：

<?xml version="1.0" encoding="UTF-8" ?>

- <Function>

<Cmd>8566</Cmd>

<Status>

{

"ligo": "zy01",

"type": "5",

"gps": "2014 / 04 / 24 10: 20: 34 N: 22.587164 E: 113.871490 29.47 km / h x: +0.00 y: -0.31 z: +0.98 A: 12.1 H: 123.5"

}

</Status>

</Function>

五 模块类型区分办法

type 11，类型有两种：

1）带有语句 GXBDT5：GGA 里面数据不加密，RMC 里面数据采用 type5 加密， GXBDT5 里面采用 type11 加密，主要是方便兼容老的 type5 模块；

2）带有语句 GXBDT511：GGA 里面数据不加密，RMC 里面数据采用 type6 加密，GXBDT511 里面采用 type11 加密，主要是方便兼容老的 type6 模块；

type 6

带有新增语句 “GXBD,”主要要与 type11 区分开，优先判断是否是 type11，在判断是否是 type 6；

type 5

以上语句都没有的时候，如果是加密模块那就是 type5；只要跟供应商区分是否是加密与非加密模块；gga 与 rmc 都加密了，要解密需要采用解密函数；

附录一 基础协议(目前不支持)

兼容---昂行科技 行车记录仪

昂行科技行车记录仪 GPS 数据格式说明

每一个视频段的行驶数据存储在 Skip Atom 数据块中。

Skip Atom 数据块的格式是：16Bytes Header(GPS INFORMATION) + 4Bytes Number(N, Number of
Samples) + 132Bytes x N

16Bytes Header 为信息头，其内容是 GPS INFORMATION 共计 15 个有效字符（含空格），再加一个
NULL 结尾符。

4Bytes 为行驶数据块的个数 N。一般情况下，每 1 秒生成一个形式数据块。举例来说，

对 1080P
30fps 时长为 1 分钟的 Movie File 而言，N = 60(DEC) = 0x3C(HEX). 上图中的示例为 N = 0x3D = 61.

每个行驶数据块的大小为 132 个字节，数据格式如下：

o 4Bytes(当前数据所属帧数, 数据范围 1-3600) ----每一个视频最多 2min x 60s x 30fps = 3600frames。注：帧数是不连续的。基本上，每 1 秒生成一个形式数据块。因此对 30fps 的 Movie 而言，每 30 帧生成一个行驶数据；对 60fps 的 Movie 而言，每 60 帧生成一个行驶数
据；

o 128Bytes 的行驶数据, “日期 时间 经度 纬度 速度 X 轴加速度 Y 轴加速度 Z 轴加速度”。数据
之间以空格分开： 2014/04/24 10:20:34 N:22.587164 E:113.871490 29.47 km/h x:+0.000 y:-0.031 z:+0.938。

注 意：

根绝昂行科技的 gps 数据规范，我们规范中的非加密模式默认兼容，只需要判断数据头的标识，如果是”GPS INFORMATION”,即可使用非加密模式去解码。

附录二 数据存储到文件末尾

添加 gps 数据到 mp4 文件末尾的方法

(mstar ,联永，安霸，凌通等平台)

MP4 每个段叫一个容器，基本结构是：8 个字节（描述信息）+ N 字节（数据内容）描述信息里面：4 个字节是指长度，后 4 个字节是段名（asii）我们一般命名为 skip；

4 个字节长度，存储方式 ：

按照高位在前低位在后的方法排列，比如长度为 0x12345678,此处存储方式是 0x12 0x34 0x56 0x78.

4 个字节长度值的由来 数据内容的长度 + 8。

图例如下：

描述信息		数据内容
4 字节长度	4 字节描述信息	N 个字节
8+N+8	SKIP（或者 skip）	数据头+n 个数据块

杰理平台----添加 gps 数据到 mp4 文件末尾的方法

基本结构是：8 个字节（描述信息）+ N 字节（数据内容）+ 8 字节（结束标识）
描述信息里面：4 个字节是指长度，后 4 个字节是段名（asii）我们一般命名为 skip；
4 个字节长度，存储方式 ：

按照高位在前低位在后的方法排列，比如长度为 0x12345678,此处存储方式是 0x12 0x34 0x56 0x78.

4 个字节长度值的由来 数据内容的长度 + 8。
结束信息：4 个字节标识（杰理平台“&&&&”）（mstar 平台用 “####”），后面 4 个字节是长度跟描述信息里面的 4 字节长度完全一致；
图例如下：

描述信息		数据内容	结束信息	
4 字节长度	4 字节描述信息	N 个字节	4 个字节描述符	四个字节长度
8+N+8	SKIP （或者 skip）	数据头+n 个数据块	“&&&&”	8+N+8

通用添加 gps 数据到 AVI，TS，MP4，MOV 格式文件末尾

基本结构是：8 个字节（描述信息）+ N 字节（数据内容）+ 8 字节（结束标识）
描述信息里面：4 个字节是指长度，后 4 个字节是段名（asii）我们一般命名为 skip；
4 个字节长度，存储方式 ：

按照高位在前低位在后的方法排列，比如长度为 0x12345678,此处存储方式是 0x12 0x34 0x56 0x78.

4 个字节长度值的由来 数据内容的长度 + 8。
结束信息：4 个字节标识“&&&&”，后面 4 个字节是长度跟描述信息里面的 4 字节长度完全一致；
图例如下：

描述信息		数据内容	结束信息	
4 字节长度	4 字节描述信息	N 个字节	4 个字节描述符	四个字节长度
8+N+8	SKIP （或者 skip）	数据头+n 个数据块	“&&&&”	8+N+8

参考代码看附录六；

附录三 type11 参考代码(目前重点推荐)

```
int gps_data_tp11(char *rawGps,unsigned char *chOutput,int second,float gx,float gy,float gz)
```

```
{
    char *pDataCheck = NULL;
    memset(chInfomation,0,128);
    if (strstr(rawGps, "GXBDT"))
    {
        pDataCheck = strstr(rawGps, "GXBDT");
        pDataCheck = strstr(pDataCheck, ",");
        pDataCheck ++;
        // 是否定位成功
        if (pDataCheck && strstr(rawGps, ",A,") && (strlen(pDataCheck)) <
120)
        {
            //成功
            pDataCheck = strstr(pDataCheck, ",");
            pDataCheck ++ ;
            char *pEnd = strstr(pDataCheck, "*");
            if (pEnd) {
                memset(chInfomation,0,sizeof(chInfomation));
                memcpy(chInfomation, pDataCheck, pEnd - pDataCheck);
                sprintf(&chInfomation[strlen(chInfomation)],",%.02f,%.02
f,%.02f",gx, gy, gz);
                MakeEncryptDataBlock(second, (char*)chInfomation,
strlen(chInfomation),chOutput);
            }
        }
    }
    return 0;
}
// 判断语句是否完整有效
static bool CheckGpsValid(char* rawGps)
{
    bool bCheck = FALSE;
    unsigned char s_verify;
    unsigned char tempValid;
    char* pstr = NULL;
    char* s_start = strstr(rawGps, "$");
    char* s_end = strstr(rawGps, "*");
    s_verify = *(s_end + 1);

    if (s_verify >= 'A') s_verify = s_verify - 'A' + 10;
    else s_verify = s_verify = s_verify - '0';
    tempValid = *(s_end + 2);
    if (tempValid >= 'A') tempValid = tempValid - 'A' + 10;
    else tempValid = tempValid = tempValid - '0';
```

```
    s_verify *= 16;
    s_verify += tempValid;
    unsigned char xor_verify = *(s_start + 1);
    for (pstr = s_start + 2; pstr < s_end; pstr++) {
        //xor the key information
        xor_verify ^= *pstr;
    }
    if(xor_verify == s_verify) {
       bCheck = TRUE;
    }
    return bCheck;
}
/***/
void PaserGps()
{
    char *rawGps[2] =
{ "$GXBDT11,A,120124,032729,23.943572,112.648701,0.000,111.746,0.00,3637
3939318603C94330FFFFFFFFFFFF*72",
    "$GXBDT15,A,120124,032729,23.943572,112.648701,0.000,111.746,0.00,36
373939318603C94330FFFFFFFFFFFF*72",
    };
    static int second =0; //这个要注意是指 当前视频的第几秒,
    Unsigned char g_ligoHeader[20];
    Unsigned char g_ligochoutput[132];
    Unsigned char g_gpsOutPutToFile[152];

    SunSetEncType(11);
    MakeHeader(g_ligoHeader,1);
    // 判断 gps 数据是否完整有效
    if(CheckGpsValid(rawGps)){
        //将 type11 的模块添加到
        gps_data_tp11(rawGps,s_headerBuf,second,0.0,0.0,0.0);
        memcpy(g_gpsOutPutToFile,g_ligoHeader,20);
        memcpy(&(g_gpsOutPutToFile[20]),g_ligochoutput, 132);
    }
}
```

附录四 type5，type6 ,type11 参考代码

```
static float gpsTranslate(float gpsData)
{
    int a = (int)gpsData / 100;
```

```
    float fRet = 0.0f;
    float fmin = (gpsData / 100 - (float)a) * 100.0f;
    fmin /= 60.0f;
    fRet = (float)a + fmin;
    return fRet;
}

// $GNRMC, 072800.016, A, 2126.98202, N, 11056.91337, E, 0.86, 62.07,
040221, , , A, V * 32
int  gps_data_rmc(char *chRmcInfo, int length, double *flat, double *flon,
    int *year, int *mon,int *mday, int *hour, int *min, int *sec,float
*speed,float *angle)
{
    int i = 0;
    char * pTemp = chRmcInfo, *pTab= NULL;
    char  chTemp[16];
    int    m_nLocation = 0;
    double fRmcLat,fRmcLon;
    if (!chRmcInfo) return -1;
    pTemp = strstr(chRmcInfo, "RMC");
    if (!pTemp)  return -1;
    i = 0;
    do {
        pTemp = strchr(pTemp, ',');
        if (pTemp == NULL) break;

        pTemp++;
        // 纬度
        if (i == 0)
        {
            //timer
            *hour = pTemp[0] - '0';
            *hour *= 10;
            *hour += pTemp[1] - '0';

            *min = pTemp[2] - '0';
            *min *= 10;
            *min += pTemp[3] - '0';
            *sec = pTemp[4] - '0';
            *sec *= 10;
            *sec += pTemp[5] - '0';


        }
```

```
        else if(i == 1)
        {
            if(*pTemp == 'A' || *pTemp == 'a')
                m_nLocation = 1;
            else {
                m_nLocation = 0;
                break;
            }
        }
        else if (i == 2)
        {
            pTab = strchr(pTemp, ',');
            memset(chTemp, 0, sizeof(chTemp));
            memcpy(chTemp, pTemp, pTab - pTemp);
            fRmcLat =(float)atof(chTemp);
            fRmcLat = gpsTranslate(fRmcLat);
        }
        else if (i == 3)
        {
            if (*pTemp != 'N' && *pTemp != 'n') {
                if (fRmcLat > 0)
                    fRmcLat = 0 - fRmcLat;
            }
            *flat = fRmcLat;
        }
        // 经度
        else if (i == 4)
        {
            pTab = strchr(pTemp, ',');
            memset(chTemp, 0, sizeof(chTemp));
            memcpy(chTemp, pTemp, pTab - pTemp);
            fRmcLon = (float)atof(chTemp);
            fRmcLon = gpsTranslate(fRmcLon);
        }
        else if (i == 5)
        {
            if (*pTemp != 'E' && *pTemp != 'e') {
                if (fRmcLon > 0)
                    fRmcLon = 0 - fRmcLon;
            }
            *flon = fRmcLon;
        }
        else if (i == 6) //速度
        {
```

```
            int j = 0;
            memset(chTemp, 0, sizeof(chTemp));
            while (pTemp[j] != ',') {
                chTemp[j] = pTemp[j];
                j++;
            }
            *speed =(float)atof(chTemp);
        }
        else if (i == 7) //角度
        {
            int j = 0;
            memset(chTemp, 0, sizeof(chTemp));
            while (pTemp[j] != ',') {
                chTemp[j] = pTemp[j];
                j++;
            }
             *angle =(float)atof(chTemp);
        //  break;
        }
        else if (i == 8)
        {
            *mday = pTemp[0] - '0';
            *mday *= 10;
            *mday += pTemp[1] - '
            *mon = pTemp[2] - '0';
            *mon *= 10;
            *mon += pTemp[3] - '0'
            *year = pTemp[4] - '0';
            *year *= 10;
            *year += pTemp[5] - '0';
            *year += 2000;
        }
        i++;
    } while (i <= 9);
    return m_nLocation;
}
int  gps_data_gga(char *chGGaInfo, float *high,char *clat,char *clon,float
*flat,float  *flon)
{
    int i = 0;
    char * pTemp = chGGaInfo, *pTab= NULL;
    char  chTemp[16];
    float fRawLat,fRawLon;
    float fseaHigh=0.0,floadHigh=0.0;
```

```
if (!chGGaInfo) return -1;
pTemp = strstr(chGGaInfo, "GGA");
if (!pTemp)  return -1;
i = 0;
do {
    pTemp = strchr(pTemp, ',');
    if (pTemp == NULL) break;
    pTemp++;
    // 纬度
    if (i == 2)
    {
        pTab = strchr(pTemp, ',');
        memset(chTemp, 0, sizeof(chTemp));
        memcpy(chTemp, pTemp, pTab - pTemp);
        fRawLat =(float)atof(chTemp);
        fRawLat = gpsTranslate(fRawLat);
        *flat = fRawLat;
    }
    else if (i == 3)
    {
        if (*pTemp != 'N' && *pTemp != 'n') {
            *clat = 'S';
        }
        else{
            *clat = *pTemp;
        }
    }
    // 经度
    else if (i == 4)
    {
        pTab = strchr(pTemp, ',');
        memset(chTemp, 0, sizeof(chTemp));
        memcpy(chTemp, pTemp, pTab - pTemp);
        fRawLon = (float)atof(chTemp);
        fRawLon = gpsTranslate(fRawLon);
        *flon = fRawLon;
    }
    else if (i == 5)
    {
        if (*pTemp != 'E' && *pTemp != 'e') {
            *clon = 'W';
        }
        else{
            *clon = *pTemp;
```

```
            }
        }
        else if (i == 6) //速度
        {
            if (*pTemp == '0' ) {
               return -1;
            }
        }
        else if (i == 9) //角度
        {
            pTab = strchr(pTemp, ',');
            memset(chTemp, 0, sizeof(chTemp));
            memcpy(chTemp, pTemp, pTab - pTemp);
            fseaHigh =(float)atof(chTemp);
        }
        else if (i == 10)
        {
            pTab = strchr(pTemp, ',');
            memset(chTemp, 0, sizeof(chTemp));
            memcpy(chTemp, pTemp, pTab - pTemp);
            floadHigh =(float)atof(chTemp);
            *high = fseaHigh - floadHigh;
        }
        i++;
    } while (i <= 9);
    return 1;
}

static int gxbd_flag =5;
static int second = 0;
unsigned char ligoGpsOutPutToFile[152];
static float rawFlat,rawFlon,rawHigh;
static char rawclat,rawclon,
void addGps()
{
    char *rawGps =
{"$GNRMC,062952.00,A,2239.76198,N,11147.27389,E,0.000,102.04,280224,,,A
*7D"};
 //   char *rawGps =
{"$GXBDT5,A,280224,070808,23.943433,112.648159,0.000,89.676,0.00,363739
393101D9404330FFF
//FFFFFFFFF*75""};
```

```
  //  char *rawGps =
{"$GXBDT11,A,280224,070808,23.943433,112.648159,0.000,89.676,0.00,36373
9393101D9404330FFF
//FFFFFFFFF*75""};

    int   rawGps,max_index = 10;
    char  g_ligoHeader[20];
    char  g_ligochoutput[132];
    char chInfomation[132];
    double flat, lon;
    int year, mon,mday, hour, min, sec;
    float speed,angle;
    float gsensorx,gsensory,gsensorz;
    //判断是否 gps 数据完整与合法
    if(CheckGpsValid(rawGps[i]) == 0)
    {
    //如果非法
        continue ;
    }
    //兼容多种类型，避免生产问题
    if(strstr(rawGps,"GXBDT5")){
        gxbd_flag=11;
        printf("====GXBDT5====");
        SunSetEncType(11);
        //读取协议数据
        MakeHeader(g_ligoHeader,0xffff);
        gps_data_tp11(rawGps,g_ligochoutput,second,gsensorx,
gsensory,gsensorz);
        memcpy(ligoGpsOutPutToFile,g_ligoHeader,20);
        memcpy(&(ligoGpsOutPutToFile[20]),g_ligochoutput, 132);
    }
    else if(strstr(rawGps,"GXBDT11")){
        gxbd_flag=11;
        printf("====GXBDT11====");
        SunSetEncType(11);
        //读取协议数据
        MakeHeader(g_ligoHeader,0xffff);
        gps_data_tp11(rawGps,g_ligochoutput,second,gsensorx,
gsensory,gsensorz);
        memcpy(ligoGpsOutPutToFile,g_ligoHeader,20);
        memcpy(&(ligoGpsOutPutToFile[20]),g_ligochoutput, 132);
    }
    else if(strstr(rawGps,"GXBD,")){
        gxbd_flag=6;
```

```
        printf("====GXBD 6====");
    }
    else if(strstr(rawGps,"RMC,"))
    {
        if(gxbd_flag == 5 || gxbd_flag == 6)
        {
            SunSetEncType(6);
            MakeHeader(g_ligoHeader,0xffff);
            memset(chInfomation,0,sizeof(chInfomation));
            if(gps_data_rmc(rawGps,strlen(rawGps),&flat, &flon,&year,
&mon, &mday, &hour, &min, &sec,&speed,&angle) == 1){
                sprintf(chInfomation,"%04d/%02d/%02d %02d:%02d:%02d %c:%.
06f %c:%.06f %.01f km/h x:%.02f
                y:%.02f z:%.02f A:%.01f H:%.01f",year,mon,mday, hour, min,
sec,(flat>0)?'N':'S',flat,(flon>0)?'E':'W',
                flon,speed,gsensorx, gsensory,gsensorz,angle,rawHigh);
                MakeEncryptDataBlock(second, (char*)chInfomation,
strlen(chInfomation),g_ligochoutput);
                memcpy(ligoGpsOutPutToFile,g_ligoHeader,20);
                memcpy(&(ligoGpsOutPutToFile[20]),g_ligochoutput, 132);
            }
        }
    }
    //  获取水印与海拔
    else if(strstr(rawGps,"GGA,"))
    {
        //  GGA
        if(gps_data_gga(rawGps,&rawHigh,&rawclat,&rawclon,&rawFlat,&rawF
lon)  != 1)
        {
            rawHigh = 0.0;
            rawclat = 'N';
            rawclon = 'E';
            rawFlat = 0.0;
            rawFlon= 0.0;
        }
    }
}
```

附录伍 GGA 解释参考代码

```
int  gps_data_gga(char *chGGaInfo, float *high,char *clat,char *clon,float
*flat,float  *flon)
```

```
{
    int i = 0;
    char * pTemp = chGGaInfo, *pTab= NULL;
    char  chTemp[16];
    float fRawLat,fRawLon;
    float fseaHigh=0.0,floadHigh=0.0;
    if (!chGGaInfo) return -1;
    pTemp = strstr(chGGaInfo, "GGA");
    if (!pTemp)  return -1;
    i = 0;
    do {
        pTemp = strchr(pTemp, ',');
        if (pTemp == NULL) break;
        pTemp++;
        // 纬度
        if (i == 2)
        {
            pTab = strchr(pTemp, ',');
            memset(chTemp, 0, sizeof(chTemp));
            memcpy(chTemp, pTemp, pTab - pTemp);

            fRawLat =(float)atof(chTemp);
            fRawLat = gpsTranslate(fRawLat);
            *flat = fRawLat;
        }
        else if (i == 3)
        {
            if (*pTemp != 'N' && *pTemp != 'n') {
                *clat = 'S';
            }
            else{
                *clat = *pTemp;
            }
        }
        // 经度
        else if (i == 4)
        {
            pTab = strchr(pTemp, ',');
            memset(chTemp, 0, sizeof(chTemp));
            memcpy(chTemp, pTemp, pTab - pTemp);
            fRawLon = (float)atof(chTemp);
            fRawLon = gpsTranslate(fRawLon);
            *flon = fRawLon;
        }
```

```
        else if (i == 5)
        {
            if (*pTemp != 'E' && *pTemp != 'e') {
                *clon = 'W';
            }
            else{
                *clon = *pTemp;
            }
        }
        else if (i == 6) //速度
        {
            if (*pTemp == '0' ) {
               return -1;
            }
        }
        else if (i == 9) //角度
        {
            pTab = strchr(pTemp, ',');
            memset(chTemp, 0, sizeof(chTemp));
            memcpy(chTemp, pTemp, pTab - pTemp);
            fseaHigh =(float)atof(chTemp);
        }
        else if (i == 10)
        {
            pTab = strchr(pTemp, ',');
            memset(chTemp, 0, sizeof(chTemp));
            memcpy(chTemp, pTemp, pTab - pTemp);
            floadHigh =(float)atof(chTemp);
            *high = fseaHigh - floadHigh;
        }
        i++;
    } while (i <= 9);
    return 1;
}
```

附录六 添加数据到文件末尾参考代码

```
void WriteDataToFile()
{
    float gSensorX, gSensorY, gSensorZ;
    //1.0 单独一条写入文件中
    int second = 0; //当前 gps 是视频中的第几秒
    int type = 11;
    unsigned char gpsHeader[20];
```

```
    SunSetEncType(11);
    //读取协议数据
    MakeHeader(gpsHeader,0xffff);
    //2.0 整体写入文件末尾
    fp = fopen("d:\\media.mp5", "a+");
    unsigned char* pGpsOutData = (unsigned char*)malloc(132 * 3);
    //判断模块类型
    //  pGpsOutData --- 保存有三条 gps 数据;
    ..............
    //增加 skip 头
    int datalength = 0;
    unsigned char skipHeader[8];
    memset(skipHeader, 0, 8);
    datalength = 8 + 20 + 132 * 3 + 8;//skip(8 字节) + gpsheader(20 字
节)+gpsdata(3*132)+tail(8 字节)
    skipHeader[0] = (datalength >>24) & 0xff;
    skipHeader[1] = (datalength >>16) & 0xff;
    skipHeader[2] = (datalength >> 8) & 0xff;
    skipHeader[3] = datalength & 0xff;
    skipHeader[4] = 'S';
    skipHeader[5] = 'K';
    skipHeader[6] = 'I';
    skipHeader[7] = 'P';
    fwrite(skipHeader, 1, 8, fp);
    //LIGO GPS 信息头写入文件中
    fwrite(gpsHeader, 1, 20, fp);
    //LIGO GPS 数据写入文件中
    fwrite(pGpsOutData, 1, 132 * 3, fp);
    //增加末尾
    unsigned char skipTail[8];
    memset(skipTail, 0, 8);
    skipTail[0] = '&';
    skipTail[1] = '&';
    skipTail[2] = '&';
    skipTail[3] = '&';
    skipHeader[0] = (datalength >> 24) & 0xff;
    skipHeader[1] = (datalength >> 16) & 0xff;
    skipHeader[2] = (datalength >> 8) & 0xff;
    skipHeader[3] = datalength & 0xff;
    fwrite(skipTail, 1, 8, fp);
    //
    fclose(fp);
}
```

FAQ 常见问题

Q： G-sensor 数据如何写入

A: 1.g-sensor 数据与 gps 数据在一条记录中一并调用 MakeEncryptDataBlock。不要单独分开

2. g-sensor 数据与 gps 数据之间需要有空格。

Q:每个数据之间是否需要有空格

A：是的。每个数据，包括 g-sensor 数据与 gps 之间都需要有空格。

Q:怎么避免地图上显示的线路有非常明显的毛刺

A: 1.第一次写入数据时，确保 GPS 已定到位

2.当前 GPS 数据经纬度与上一个经纬度进行比较防抖，如果相邻的两个经度或者纬度变化过大，则取上一个的数据写入。

Q:为什么模块的数据与实际坐标不匹配

A:有很多模块是有加密的，GPS 本身输出的数据就是经过加密处理的，需要脱密之后才与实际坐标一致。

PS：当前我们主推的 Gxplayer 播放器匹配的就是国兴北斗的双模模块。

Q:为什么模块串口出来的数据都比较实际坐标要大

A：因为 GPS 国际标准输出的 GPS 经纬度是“度分”结构的，开发人员需要自己转换成度，文档上半部分有参考的算法。

PS：此处转换需很谨慎。没转换好很容易坐标偏差。

Q:为什么地图坐标进度与实际视频进度不一致

A：请用 GpsPaserv 工具分析视频文件，通常是因为视频秒数与 GPS 数据数量不一致，比如说 60 秒数据就需要至少 59 组 GPS 数据。

1.GPS 数据多于视频长度，就会出现地图显示坐标滞后。

2.GPS 数据少于视频长充，就会出现视频还没播放完，但是地图已到达终点。

Q:为什么播放器速度与实际速度不一致

A：播放器默认 GPS 数据是海里，所以存储到视频中的数据一定单位是海里，不是公里，请忽略 GPS 协议中的 “km/h”,此标记仅于数据识别。

。

GPS 数据格式：

$GPRMC,<1>,<2>,<3>,<4>,<5>,<6>,<7>,<8>,<9>,<10>,<11>,<12>*hh
<1>UTC 时间，hhmmss.sss(时分秒.毫秒)格式
<2>定位状态，A=有效定位，V=无效定位
<3>纬度 ddmm.mmmm(度分)格式(前面的 0 也将被传输)
<4>纬度半球 N(北半球)或 S(南半球)
<5>经度 dddmm.mmmm(度分)格式(前面的 0 也将被传输)
<6>经度半球 E(东经)或 W(西经)
<7>地面速率(000.0~999.9 节，前面的 0 也将被传输)
<8>地面航向(000.0~359.9 度，以正北为参考基准，前面的 0 也将被传输)
<9>UTC 日期，ddmmyy(日月年)格式
<10>磁偏角(000.0~180.0 度，前面的 0 也将被传输)
<11>磁偏角方向，E(东)或 W(西)
<12>模式指示(仅 NMEA01833.00 版本输出，A=自主定位，D=差分，E=估算，N=数据无效)

$GPGGA 语句包括 17 个字段：语句标识头，世界时间，纬度，纬度半球，经度，经度半球，定位质量指示，使用卫星数量，水平精确度，海拔高度，高度单位，大地水准面高度，高度单位，差分 GPS 数据期限，差分参考基站标号，校验和结束标记(用回车符<CR>和换行符<LF>)，分别用 14 个逗号进行分隔。该数据帧的结构及各字段释义如下：
$GPGGA,<1>,<2>,<3>,<4>,<5>,<6>,<7>,<8>,<9>,M,<10>,M,<11>,<12>*xx<CR><LF>
$GPGGA：起始引导符及语句格式说明(本句为 GPS 定位数据)：
<1> UTC 时间，格式为 hhmmss.sss；
<2> 纬度，格式为 ddmm.mmmm(第一位是零也将传送)；
<3> 纬度半球，N 或 S(北纬或南纬)
<4> 经度，格式为 dddmm.mmmm(第一位零也将传送)；
<5> 经度半球，E 或 W(东经或西经)
<6> 定位质量指示，0=定位无效，1=定位有效；
<7> 使用卫星数量，从 00 到 12(第一个零也将传送)
<8> 水平精确度，0.5 到 99.9
<9> 天线离海平面的高度，-9999.9 到 9999.9 米
M 指单位米
<10> 大地水准面高度，-9999.9 到 9999.9 米
M 指单位米
<11> 差分 GPS 数据期限(RTCM SC-104)，最后设立 RTCM 传送的秒数量
<12> 差分参考基站标号，从 0000 到 1023(首位 0 也将传送)。
* 语句结束标志符
xx 从$开始到*之间的所有 ASCII 码的异或校验和
<CR> 回车
<LF> 换行

附录七：

多路文件播放，参考文件命名方法

文件命名方法： 日期_时间+标志位.文件格式
比如： 20230919_120406A.ts
日期：年月日；20240919
时间：120406
标志位：A，B，C，D，四种类型
文件格式：ts,mov,mp4,avi 等