音视频帧包装格式定义

文档起草	陈世彬
使用范围	内部文档
密级	绝密
创建日期	2007-06-11
最后修改日期	2008-02-28
所有参与者	龙凯、黄延冬、刘文涛、望西淀、雷堂祖、陈世彬、蒋文军、孙继业
版本号	1.00.00

深圳市锐明视讯技术有限公司

Shenzhen Streaming Video Technology Co., Ltd.

目录

一、概述

1.1 背景

为了能够把同一平台的工作标准化和统一化，加快项目的开发，现对音视频包装格式进行定义，以达到未来的最大兼容。

1.2 文档适用范围

此文档不但规定音视频帧包装格式的框架结构，也定义了具体的类型信息及相关内容，但不包含网传部分的音视频包装格式

所有结构都以 8 字节对齐，此长度用于设备端的方便(如果是 windows 上，用 4 字节对齐就够了)。

注：只规定单视频帧的包装格式，而不是规定编码格式。

- 目前只对 C2 和 Mini 有效。

1.3 数据说明

所有的结构都采用 4 个字节对齐。

二、视频帧包装格式

2.1 视频帧包装头结构

视频帧都有一个固定 24 字节的帧头，此帧头结构定义如下：

```
struct RMVFI_HEADER
{
    unsigned long       lFrameType;   /*I 帧为 x0dc，p 帧为 x1dc*/
    unsigned long       lStreamType;   /*流格式：暂时固定为 H264，ASCII*/
    unsigned long       lFrameLen;    /*仅视频帧数据长度*/
    unsigned short      usExtendLen;  /*仅扩展数据长度*/
    unsigned short      usExtendMode;     /*扩展模式*/
    ULONGLONG       ullPts;               /*时间戳*/
};
```

说明：

VFI ：Video Frame Information

lFrameType：帧的类型标识，I 帧为 x0dc ，p 帧为 x1dc，其中 x 表示通道号，通道号取值为 0 到 15，该字节的值减去 0x30 的结果，例如：1 通道 I 帧为 00dc。这个表述都是 16 进制的数值型方式。

lStreamType：流格式定义。在正常的帧标识情况下，可能不能完全满足查询帧头的要求，于是增加了此成员，用于进一步区分帧头。对于不同的平台，该成员有不同的值，但也可能不用。以下为此列表：

平台	StreamType			
	16进制	字符串	ASCII码	
(HI)3510	0x	“462H”		

lFrameLen： 仅视频帧数据长度；

usExtendLen： 仅扩展数据长度，如果没有扩展数据，则填充 0，如果有扩展数据，则长度需 8 字节对齐；

usExtendMode：扩展模式，规定每种模式，对应一种信息结构。分高低字节进行分类，高字节表示大的分类，低字节表示小的分类。具体定义请参考下面的扩展模式

ullPts： 时间戳信息，单位（微秒），1 秒 = 1000,000 微秒

以下为相关的长度说明图：

包装头(帧头)	括展数据	视频数据
RMVFI_HEADER	RMVFI_HEADER::ExtendLen	RMVFI_HEADER::FrameLen

视频帧和音频帧只用第一个成员进行区别

2.2 扩展数据

扩展数据结构体的选择根据 RMVFI_HEADER::ExtendMode 定义。ExtendMode 采用高低字节功能定义，即进行大致的分类表示。
ExtendMode 用 16 进制表示为：0xABCD
则 0xAB、0xCD 的定义如下：
0xAB：表示大的分类。其中，A 保留，总是为 0。B：1 – 车载应用，3 – 非移动应用，即一般的 DVR，F – 不区分应用类型，即对所有应用都有效的，比如日期时间信息等
0xCD：表示在大的分类里的模式编号，编号从 1 开始。

2.2.1 模式一：车载基本信息

ExtendMode ：0x0101
struct **RMFIM_MBASE**
{
    RMFI_MACCEL accel;
    RMFI_ALARM   alarm;
};
说明：
accel：加速度信息
alarm：报警信息
此结构只包含一般的行车信息，即用户不需要其它信息

2.2.2 模式二：车载信息

ExtendMode ：0x0102
struct **RMFIM_MINFO**
{
    RMFI_MACCEL accel;
    RMFI_ALARM   alarm;
    RMFI_MSTATUS mStatus;
    RMFI_GPS   gps;
}
说明：
accel：加速度信息
alarm：报警信息
mStatus：车辆状态，比如温度等
gps：gps 信息，其中 RMFI_GPS 结构中有 gps 信息是否有效的开关
此结构包含了车辆的大多数信息，相当全面。虽然在结构的内容上包含了 RMFIM_MBASE，但在模式定义的思想上，并不包含该结构，它们只算做是两种不同的模式

2.2.3 模式三：加卡号的车载信息

ExtendMode ： 0x0103
struct **RMFIM_MICARD**
{
    RMFI_MACCEL accel;
    RMFI_ALARM alarm;
    RMFI_MSTATUS mStatus;
    RMFI_GPS gps;
    RMFI_CARDI cardInfo;
}
说明：
accel：加速度信息
alarm：报警信息
mStatus：车辆状态，比如温度等
gps：gps 信息，其中 RMFI_GPS 结构中有 gps 信息是否有效的开关
此结构包含了车辆的大多数信息，相当全面。虽然在结构的内容上包含了 RMFIM_MBASE，但在模式定义的思想上，并不包含该结构，它们只算做是两种不同的模式
cardInfo：卡号信息
此模式只比模式二多了一个卡号信息，目前用于 G11

2.2.4 模式四：日期时间信息

ExtendMode ： 0x0F01
struct **RMFIM_DATETIME**
{
    RMFI_DATETIMEdatetime
};
说明：
datetime：日期和时间

2.2.5 模式五：车载信息版本 2

ExtendMode ： 0x0104
struct **RMFIM_MINFO2**
{
    RMFI_MACCEL accel;
    RMFI_ALARM alarm;
    RMFI_MSTATUS mStatus;
    RMFI_GPS2 gps;
}
说明：
accel：加速度信息

alarm：报警信息
mStatus：车辆状态，比如温度等
gps：gps 信息，其中 RMFI_GPS2 结构中有 gps 信息是否有效的开关
此结构包含了车辆的大多数信息，相当全面。虽然在结构的内容上包含了 RMFIM_MBASE，但在模式定义的思想上，并不包含该结构，它们只算做是两种不同的模式

描述

本模式的定义，主要是由于 GPS 信息的结构做了修改

2.2.6 模式六：加卡号的车载信息版本 2

ExtendMode ：0x0105
struct **RMFIM_MICARD2**
{
    RMFI_MACCEL accel;
    RMFI_ALARM alarm;
    RMFI_MSTATUS mStatus;
    RMFI_GPS2 gps;
    RMFI_CARDI cardInfo;
}
说明：
accel：加速度信息
alarm：报警信息
mStatus：车辆状态，比如温度等
gps：gps 信息，其中 RMFI_GPS2 结构中有 gps 信息是否有效的开关
此结构包含了车辆的大多数信息，相当全面。虽然在结构的内容上包含了 RMFIM_MBASE，但在模式定义的思想上，并不包含该结构，它们只算做是两种不同的模式
cardInfo：卡号信息
此模式只比模式二多了一个卡号信息，目前用于 G11

描述

本模式的定义，主要是由于 GPS 信息的结构做了修改

2.3 单位结构体

注意：每个结构都需要 8 字节对齐

2.3.1 传感器信息

主要包括加速度和角度信息，这两种信息在固定的场所基本上是用不到的，所以指定为车载设备使用，特此前面加上了“M”

struct **RMFI_MACCEL**
{
	unsigned short	usDirectionData;	/*罗盘数据，表示罗盘角度*/
	unsigned char	cAccelerateX;	/*加速度传感器 X 方向的读数，单位 1/64G*/
	unsigned char	cAccelerateY;	/*加速度传感器 Y 方向的读数，单位 1/64G */
	unsigned char	cAccelerateZ;	/*加速度传感器 Z 方向的读数，单位 1/64G */
	char	reserved[3];
};
说明：
M = Movable，表示该结构用于车载设备
FI = Frame Information
说明：
usDirectionData：罗盘数据，表示罗盘角度
cAccelerateX：　HEX，此字节的最高位是符号位，1 表示负，0，表示正，比如 0x40 ,,表示 1G，单位 1/64G
cAccelerateY：加速度 Y 方向的读数，HEX，此字节的最高位是符号位，1 表示负，0，表示正，比如 0x40 ,, 表示 1G，单位 1/64G
cAccelerateZ：加速度 Z 方向的读数，HEX，此字节的最高位是符号位，1 表示负，0，表示正，比如 0x40 ,, 表示 1G，单位 1/64G

2.3.2 车辆状态

struct **RMFI_MSTATUS**
{
	char	cTemperature;	/*温度值*/
	char	cSimulateVal;	/*模拟量值*/
	char	reserved[6];
}’
说明：
cTemperature：温度值，有正负之分
cSimulateVal：模拟量，暂时保留，单位未定义

2.3.3 报警信息

报警信息中的一部分报警类型也是通过传感器来获取的，但为了和其它传感器所取得的数据区分开来，这里用报警这个名称
struct **RMFI_ALARM**
{
	unsigned char	cSensorAlarm;	/*每一位代表一个传感器报警状态*/
	unsigned char	cOtherAlarm;	/*其他报警，如移动侦测、超速报警、遮挡*/
	char	reserved[6];
};
说明：
cSensorAlarm：每一位表示一个传感器的报警状态，0—无报警，1—有报警；
cOtherAlarm：其他报警状态，bit0：移动侦测报警，bit1：轻微超速报警，bit2：严重超速报警，bit3：视频

丢失报警，bit4：遮挡报警，其他位暂时保留；

2.3.4 GPS 信息

struct **RMFI_GPS**
{

```
char            cGpsStatus;     /*gps 是否有效标识*/
char            cSpeedUnit;     /*速度单位*/
unsigned short  usSpeed;        /*速度值*/
char            szLatitude [12]; /*维度值*/
char            szLongitude[12]; /*经度值*/
unsigned short  usGpsAngle;     /*gps 夹角*/
char            reserved[2];
```

};
说明：
cGpsStatus：gps 是否有效标识。‘A’表示有效，‘V’表示无效
cSpeedUnit：速度单位，0 – Mile，英里，1 – KM，公里
usSpeed：速度值
szLatitude：纬度值，ACSII 码，比如“3723.24N”表示北纬 37 度 23 分 24 秒；比如“3723.24S”表示南纬 37 度 23 分 24 秒
szLongitude：经度值，ASCII 码，比如“12158.34W”表示西经 121 度 58 分 34 秒；比如“12158.34E”表示东经 121 度 58 分 34 秒
usGpsAngle：gps 夹角

2.3.5 GPS 信息版本 2

由于 RMFI_GPS 定义的缺陷，只有重新定义一种新的模式来代替之，但 RMFI_GPS 还会存在，只是以后会用的越来越少，而且提倡只使用该新结构
struct **RMFI_GPS2**
{

```
char            cGpsStatus;     /*gps 是否有效标识*/
char            cSpeedUnit;     /*速度单位*/
unsigned short  usSpeed;        /*速度值*/
char            cLatitudeDegree; /*纬度值的度*/
char            cLatitudeCent;  /*纬度值的分*/
char            cLongitudeDegree; /*经度值的度*/
char            cLongitudeCent; /*经度值的分*/
long            lLatitudeSec;   /*纬度值的秒*/
long            lLongitudeSec;  /*经度值的秒*/
unsigned short  usGpsAngle;     /*gps 夹角*/
char            cDirectionLatitude; /*纬度的方向*/
char            cDirectionLongitude;    // 经度的方向
```

};

说明：
cGpsStatus：gps 是否有效标识。‘A’表示有效，‘V’表示无效
cSpeedUnit：速度单位，0 – Mile，英里，1 – KM，公里
usSpeed：速度值
cLatitudeDegree：纬度值的度，ACSII 码，范围：0 ~ 89
cLatitudeCent：纬度值的分，ASCII 码，范围：0 ~ 59
cLongitudeDegree：经度值的度，ASCII 码，范围：0 ~ 179
cLatitudeCent：经度值的分，ASCII 码，范围：0 ~ 59
lLatitudeSec：纬度值的秒，范围：0 ~ 999999
lLongitudeSec：经度值的秒，范围：0 ~ 999999
usGpsAngle：gps 夹角
cDirectionLatitude：纬度的方向，‘N’表示北纬，‘S’表示南纬
cDirectionLongitude：经度的方向，‘E’表示东经，‘W’表示西经

描述

对于经纬度的计算，只需要把它们的度分秒三个字段相加，然后根据方向标示来判断是东西经和南北纬即可。

2.3.6 卡号信息

struct **RMFI_CARDI**
{
    char   szCardNo[16];
    char   reserved[7];
    char   cFlagRes;
};
说明：
最后的 I = Information
szCardNo：卡号。
cFlagRes：对保留字节的意义定义。按位来分解：
    bit0：保留字节是否启用。0 – 没有启用，1 – 启用
    bit1：保留字节的用途(只在 bit0 为 1 时有效)。0 – 卡号的尾数，即完整的卡号是由 szCardNo 和保留字节连起来的结果，这是由于卡号太长，szCardNo 无法保存完毕。1 – 暂不做定义
    bit2 ~ bit7：保留

2.3.7 时间信息

struct **RMS_DATETIME**
{
    BYTE   cYear;
    BYTE   cMonth;
    BYTE   cDay;
    BYTE   cHour;

```
    BYTE    cMinute;
    BYTE    cSecond;
    unsigned short usMilliSecond;
};
typedef struct  RMS_DATETIME  RMFI_DATETIME;
```

说明：

cYear：年份，减去 2000 后的结果。例如，11 表示 2011 年，5 表示 2005 年，113 表示 2113 年

cMonth：月份。范围：1 ~ 12

cDay：日期。范围：1 ~ 31

cHour：小时。范围：0 ~ 23

cMinute：分钟。范围：0 ~ 59

cSecond：秒。范围：0 ~ 59

usMilliSecond：毫秒，范围：0 ~ 999。由于该成员并不一定总是有效，所以定义其最高位表示该毫秒成员是否有效，1 – 无效，0 – 有效

2.4 视频帧中包装信息规则

真对(HI)3510 平台，有以下规则：

（1）.每秒钟必需至少有一个模式二

（2）.每一个 I 帧都必需使用模式二

（3）.对每一个 P 帧，在模式一和模式二中，至少要包含其中一种模式

三、音频帧格式定义

注：音频帧与视频帧的分辨，只按照其最开始的 4 个字节来区分，即 FrameType。

音频数据帧头结构体定义如下：

```
struct RMAFI_HEADER
{
    unsigned long     lFrameType;     /*音频帧标识*/
    unsigned short    usFrameLen      /*一个音频帧长度*/
    unsigned short    usPacketLen;    /*此包音频数据长度*/
};
```

说明：

FrameType：XYwb，X 表示通道号，计算方式和视频帧的包装通道一样，从 0x30 开始为通道一，Y 表示音频格式：

Y == 1,　音频格式为 AMR。MIME，4.75kbit/s；

Y == 2，　音频格式为 G726；

Y == 3，　音频格式为 ADPCM；

Y ==…　可根据应用进行相应扩充；

FrameLen：　有效音频数据长度，单帧的音频帧长度

PacketLen：本音频包的长度。

如果后面的音频数据长度不是 8 字节对齐，则按 8 字节对齐。

为了节省带宽，一个音频包可能包含多个音频帧，即多个音频帧放在一起打成一个包，其帧数 = PacketLen / FrameLen

例：第一通道音频格式为 amr，则为 01wb。

包格式如下图所示：

帧包装头	音频数据			
	音频帧1	音频帧2	... …	音频帧n
RMAFI_HEADER	帧数据	帧数据	… …	帧数据

如果是音频帧不等长，则每一包就一个音频帧即可，如果音频帧长度一致，则可以多个音频帧放在一起打包

四、附录

4.1 数据类型

约定各种数据类型及其长度
int、unsigned int：4 个字节的整型数
short、unsigned short：2 个字节的整型数
char、unsigned char：1 个字节的整型数
long long：8 个字节，在 windows 中为__int64，或者为其它 8 字节的数值型数据类型
字节顺序为 Intel 规范，即高字节在后

4.2 缩写

VFI：Video Frame Information
FI ：Frame Information
ACCEL：Acceleration
FIM：Frame Information Mode

4.2 修改日志

<2007.06.14>：
继第二次讨论后形成的定稿，并交由黄延冬维护视频帧包装格式部分。音频帧包装格式仍然由陈世彬维护

<2007.08.16>：
1. 视频帧部分增加了模式三，其只比模式二多了一个卡号信息
2. 对 RMFI_CARDI 的保留字节做了修改，分离出最后一个字节做为指示性成员，以增加对卡号中更多信息的控制

<2007.08.17>
1. 由于需要 8 字节对齐，所以修改了下列内容：
   A. 车辆状态结构 RMFI_MSTATUS 中最后增加了一个新的成员，此成员保留，使整个结构 8 字节对齐
2. 为了关于字节对齐的部分不产生疑义，调整了 GPS 信息结构 RMFI_GPS 中第二个成员 Speed 和第三个成员 SpeedUnit 的顺序

<2007.08.22>
修改了 GPS 结构中的保留字段，由原来的保留长度为 10，修改为现在的保留长度为 2，即少了 8 个字节

<2007.09.04>

1. 修改了 0 视频帧头的通道号码标示，原来是如果大于 10，就用 A ~ F 来表示，这样的话，以后就不好表示大于 16 个通道的通道号码了，而且在通道号计算时，如果大于 10 个通道，则通道号码的计算方式就不一样了，所以为了统一，就改为以 0x30 为基数的 ascii 来表示，这样通道号码的计算就变得非常简单，不用担心大于 10 的问题了
2. 整理了音频帧的帧头结构说明，并修改了其通道号码的标示，原来是从 1 开始计算，现改为和视频帧一样的计算方式，而且也从 0 开始计算

<2008.01.10>

1. 增加了日期结构
2. 增加了模式四，即日期和时间信息

<**2008．02．28**>

1. 由于原来 GPS 信息的结构缺陷，只好新定义一个相同的结构，只是后面增加了一个数字编号“2”，所以对于两个涉及到该结构的模式，也都新增加了一个相对应的模式，所以就增加了两个模式，而且建议只使用新的模式，而不建议使用旧的模式

<2008.03.04>

1. 前次增加的 GPS2 结构中，由于经度不正确，此次专门对此进行了修改，去掉了纬度和经度上的符号位，并启用保留的两个字节以表示经纬度的方向
2.