全 志 科 技
Allwinner Technology

Video Decoder API

AWCodecAPI(For OS)

文档履历

版本号	日期	制/修订人	内容描述
V0.1	2015-06-10	BZ	基于《视频解码概要设计》建立初稿

目 录

1. 概述

指导视频解码库的使用。

2. 模块介绍

2.1. 功能介绍

视频解码库是一个提供视频解码功能的库。基于视频解码库，应用程序可以在全志公司的各个 IC 平台上实现高效的、多格式的视频解码功能。

3. 接口和流程设计

3.1. 视频解码库 API

视频解码库的 API 接口如下表所示。

视频解码库 APIs		
1	CreateVideoDecoder	创建一个视频解码器
2	DestroyVideoDecoder	销毁一个视频解码器，释放资源
3	InitializeVideoDecoder	根据视频码流信息（编码格式等）初始化视频解码器
4	ResetVideoDecoder	重置视频解码器，重置后视频宽高等基本信息仍保留，用于播放器跳播等操作
5	DecodeVideoStream	解码一笔视频码流
6	GetVideoStreamInfo	从解码器获取视频信息
7	RequestVideoStreamBuffer	获取码流 Buffer，用于填充码流数据
8	SubmitVideoStreamData	填充完码流数据后，将数据提交给解码器
9	VideoStreamBufferSize	获取码流缓冲区的大小，以字节为单位
10	VideoStreamDataSize	码流缓冲区内有效（未解码）数据的大小，以字节为单位
11	VideoStreamFrameNum	码流缓冲区内有多少笔有效（未解码）数据
12	RequestPicture	获取视频图像
13	ReturnPicture	归还视频图像
14	NextPictureInfo	获取下一帧视频的信息，如时间戳等信息
15	TotalPictureBufferNum	解码器内总共有多少个视频图像 Buffer
16	EmptyPictureBufferNum	目前空闲的图像 Buffer 数量
17	ValidPictureNum	等待显示的视频图像数量
18	ConfigHorizonScaleDownRatio	设置视频输出图像水平方向缩放倍数，支持 2 倍和 4 倍缩放
19	ConfigVerticalScaleDownRatio	设置视频输出图像垂直方向缩放倍数，支持 2 倍和 4 倍缩放
20	ConfigSecHorizonScaleDownRatio	设置从路通道图像水平方向缩放倍数，支持 2/4/8/16/32 倍缩放
21	ConfigSecVerticalScaleDownRatio	设置从路通道图像垂直方向缩放倍数，支持 2/4/8/16/32 倍缩放
22	ReopenVideoEngine	重新打开 Video Engine 模块，用于支持多分辨率视频流的播放
23	AllocatePictureBuffer	申请一个独立的，指定大小的图像 Buffer
24	FreePictureBuffer	释放由 AllocatePictureBuffer 函数申请的图像 Buffer
25	RotatePicture	旋转图像

开始解码前，应用程序首先调用 CreateVideoDecoder 函数创建一个解码器，然后调用 InitializeVideoDecoder 函数，将视频基本信息作为参数，初始化解码器。

初始化后，解码器可以开始解码视频流。

应用程序通过 RequestVideoStreamBuffer 函数从解码器获取码流 Buffer，将数据填入后，通过 SubmitVideoStreamData 函数将码流提交给解码器。

应用程序通过调用 DecodeVideoStream 函数解码视频码流。

应用程序通过调用 RequestPicture 函数获取视频图像，视频图像显示完毕后，应用程序调用 ReturnPicture 将图像 Buffer 归还解码器。

视频解码库支持多线程操作，码流的传送、解码和视频图像的输出工作可以在不同的线程中进行。一般来说，播放器会有 Demux 线程、视频解码线程和视频渲染（Render）等三个线程处理视频相关的工作。Demux 线程不断调用 RequestVideoStreamBuffer 函数和 SubmitVideoStreamData 函数传送数据；视频解码线程通过调用 DecodeVideoStream 函数解码视频流；视频渲染线程不断调用 RequestPicture 函数获取图像用于显示，调用 ReturnPicture 归还已经显示的图像。

视频解码库还支持图像缩放、旋转等功能，这些功能需要调用其他 API 进行配置。下文详细介绍各个 API 函数。

3.1.1. CreateVideoDecoder

函数原型	VideoDecoder* CreateVideoDecoder(void)
功能	创建一个视频解码器
参数	无
返回值	成功：视频解码器指针； 失败：返回 NULL；
调用说明	视频解码库支持创建多个解码器，同时解码多路视频。

3.1.2. DestroyVideoDecoder

函数原型	void DestroyVideoDecoder(VideoDecoder* pDecoder)
功能	销毁一个视频解码器，释放相关软硬件资源
参数	pDecoder：通过 CreateVideoDecoder 函数创建的视频解码器指针
返回值	无
调用说明	无

3.1.3. InitializeVideoDecoder

函数原型	int InitializeVideoDecoder(VideoDecoder* pDecoder VideoStreamInfo* pVideoInfo, VConfig* pVConfig)
功能	初始化视频解码器
参数	pDecoder: 通过 CreateVideoDecoder 函数创建的视频解码器指针 pVideoInfo: 视频码流的基本信息，如编码格式、分辨率、帧率等 pVConfig: 视频解码器配置选项，用于配置旋转、图像缩小、缩略图模式等
返回值	0: 表示成功; -1: 失败，不支持的编码格式或内存资源不足;
调用说明	VideoStreamInfo 中，不是所有信息都是必须的。 对于 H264 和 MPEG2，可以只填写编码格式信息（format），其他视频一般还需要正确填写视频分辨率信息（width 和 height，由于码流中没有该信息）。某些视频解码前需要初始化数据（initData）。不同格式视频对应的初始化信息如何填写，可以参考本文档第 6 节附录部分的“视频初始化信息设置” pVConfig：配置解码器的旋转，缩小等信息 1． bScaleDownEn: 配置缩小输出，取值为 0 或 1， 默认值为 0; 2． bRotationEn：配置旋转输出，取值为 0 或 1， 默认值为 0; 3． nHorizonScaleDownRatio: 视频输出图像水平方向缩小比例，0 表示不缩放，1 表示缩放为 1/2 大小，2 表示缩放为 1/4 大小，3 表示缩放为 1/8 大小; 4． nVerticalScaleDownRatio: 视频输出图像竖直方向缩小比例，0 表示不缩放，1 表示缩放为 1/2 大小，2 表示缩放为 1/4 大小，3 表示缩放为 1/8 大小;（注：缩放比例是针对原始图像设置， 同时进行旋转时需要注意；VP6、WMV1、WMV2 格式的视频不支持 ScaleDown 功能）; 5． nSecHorizonScaleDownRatio: 从通道视频输出图像水平方向缩小比例，0 表示不缩放，1 表示缩放为 1/2 大小，2 表示缩放为 1/4 大小，3 表示缩放为 1/8 大小，4 表示缩放 1/16，5 表示缩放 1/32; 6． nSecVerticalScaleDownRatio: 从通道视频输出图像垂直方向缩小比例，0 表示不缩放，1 表示缩放为 1/2 大小，2 表示缩放为 1/4 大小，3 表示缩放为 1/8 大小，4 表示缩放 1/16，5 表示缩放 1/32; 7． nRotateDegree：视频输出图像的旋转角度，以顺时针方向计算,0 表示不旋转，1 表示 90 度，2 表示 180 度，3 表示 270 度; 8． bThumbnailMode:解码器以缩略图模式工作,当应用程序只是希望解码视频文件的一幅图像作为缩略图显示时，解码器可以只申请一个图像 Buffer，解码输出图像后应用程序关闭解码器，不再继续解码, 取值为 0 或 1， 默认值为 0; 9． eOutputPixelFormat:解码器输出图像的像素格式,像素格式对应的数据存放方式，请参考本文档附录 6.3 节; 10．bNoBFrames:视频源是否没有 B 帧， 取值为 0 或 1(默认值为 0); 11．bDisable3D:不支持 3D 模式，取值 0 或 1(默认值为 0); 12．bSupportedMaf:是否支持 de_interlace 功能，只用于 A20 或 A23 平台;

3.1.4. ResetVideoDecoder

函数原型	void ResetVideoDecoder(VideoDecoder* pDecoder)
功能	重置视频解码器
参数	pDecoder：通过 CreateVideoDecoder 函数创建的视频解码器指针
返回值	无
调用说明	调用本函数后，Video Engine 模块被重置，但视频初始化信息被保留，码流 Buffer 中的数据被清空，图像数据也被清空。 本函数一般用于跳播清除视频解码器数据。

3.1.5. DecodeVideoStream

函数原型	int DecodeVideostream(VideoDecoder* pDecoder, int bEndOfStream, int bDropBFrameIfDelay, int64_t nCurrentTimeUs)
功能	解码一帧图像，解码库会对码流 Buffer 中的码流进行解码
参数	pDecoder：通过 CreateVideoDecoder 函数创建的视频解码器指针； bEndOfStream：是否码流结束，等于 1 表示所有码流数据都已经传送到 Sbm； bDropBFrameIfDelay：码流的时间戳（PTS）比当前时间大时，是否丢弃 B 帧，等于 0 表示不丢弃，等于 1 表示丢弃 B 帧； nCurrentTimeUs：当前时间，用于比较码流是否过时；
返回值	VDECODE_RESULT_FRAME_DECODED(1)：解码成功，输出了一帧图像； VDECODE_RESULT_CONTINUE(2)：码流被解码，但没有图像输出，需继续解码； VDECODE_RESULT_KEYFRAME_DECODED(3)：解码成功，输出了一帧关键帧图像； VDECODE_RESULT_NO_FRAME_BUFFER(4)：当前无法获取到图像 Buffer； VDECODE_RESULT_NO_BITSTREAM(5)：当前无法获取到码流数据； VDECODE_RESULT_RESOLUTION_CHANGE(6)：视频分辨率发生变化，无法继续； VDECODE_RESULT_UNSUPPORTED(-1)：不能支持的格式或申请内存失败，无法继续解码；
调用说明	在解码性能不足的情况下，通过丢弃过时的 B 帧码流，解码器可以追赶视频播放的进度，避免音视频不同步的问题；

3.1.6. GetVideoStreamInfo

函数原型	int GetVideoStreamInfo(VideoDecoder* pDecoder, VideoStreamInfo* pVideoInfo)
功能	获取视频信息
参数	pDecoder: 通过 CreateVideoDecoder 函数创建的视频解码器指针 pVideoInfo: 输出参数，存放视频信息；
返回值	0: 表示成功；-1: 失败；
调用说明	除了初始化时设置的信息，解码器会将解码后获得的信息一起输出。

3.1.7. RequestVideoStreamBuffer

函数原型	int RequestVideoStreamBuffer(VideoDecoder* pDecoder, int nRequireSize, char** ppBuf, int* pBufSize, char** ppRingBuf, int* pRingBufSize, int nStreamBufIndex)
功能	向解码器请求存放码流 Buffer，用于存放数据
参数	pDecoder: 通过 CreateVideoDecoder 函数创建的视频解码器指针 nRequireSize: 请求 Buffer 的大小，以字节为单位； ppBuf: 输出参数，码流 Buffer 起始地址，等于 NULL 表示失败； pBufSize: 输出参数，码流 Buffer ppBuf 的大小； ppRingBuf: 输出参数，第二块 Buffer 的起始地址，等于 NULL 表示没有； pRingBufSize: 第二块 Buffer ppRingBuf 的大小； nStreamBufIndex: 对于蓝光 MVC 等 3D 视频，解码器需要处理两路码流，nStreamBufIndex 指定从第几路视频码流 Buffer 获取 Buffer，0 表示第 0 路（MVC 主码流）、1 表示第 1 路（MVC 从码流）。
返回值	0: 表示成功；-1: 失败；
调用说明	码流 Buffer 是一块循环 Buffer，当 Buffer 回头时，外部请求的 Buffer 被分成两段，ppBuf 和 pBufSize 返回第一段 Buffer 的地址和大小，ppRingBuf 和 pRingBufSize 返回第二段 Buffer 的地址和大小。 Buffer 没有回头时，ppRingBuf 和 pRingBufSize 返回 NULL。

3.1.8. SubmitVideoStreamData

函数原型	int SubmitVideoStreamData(VideoDecoder* pDecoder, VideoStreamDataInfo* pDataInfo, int nStreamBufIndex)
功能	向解码器提交码流数据
参数	pDecoder: 通过 CreateVideoDecoder 函数创建的视频解码器指针; pDataInfo: 码流数据信息，包括地址、长度、时间戳、边界信息等; nStreamBufIndex: 对于蓝光 MVC 等 3D 视频，解码器需要处理两路码流，nStreamBufIndex 指定从第几路视频码流 Buffer 获取 Buffer，0 表示第 0 路（MVC 主码流）、1 表示第 1 路（MVC 从码流）。
返回值	0: 表示成功; -1: 失败;
调用说明	提交数据时，数据可以是一笔包含整数个数据单元的完整码流帧，也可以只包含一个数据单元的部分数据，只需将 VideoStreamDataInfo 结构体中的两个表示数据边界信息的变量正确填写即可。两个边界信息变量为 bIsFirstPart: 表示该笔数据第一个字节是否是一个数据单元的开始; bIsLastPart: 表示该笔数据最后一个有效字节是否是一个数据单元的结束;

3.1.9. VideoStreamBufferSize

函数原型	int VideoStreamBufferSize(VideoDecoder* pDecoder, int nStreamBufIndex)
功能	查询码流 Buffer 的总大小，单位为字节
参数	pDecoder: 通过 CreateVideoDecoder 函数创建的视频解码器指针; nStreamBufIndex: 对于蓝光 MVC 等 3D 视频，解码器需要处理两路码流，nStreamBufIndex 指定从第几路视频码流 Buffer 获取 Buffer，0 表示第 0 路（MVC 主码流）、1 表示第 1 路（MVC 从码流）。
返回值	码流 Buffer 的大小，单位为字节。
调用说明	在解码器初始化后才能正确返回码流 Buffer 的大小，否则返回 0.

3.1.10. VideoStreamDataSize

函数原型	int VideoStreamDataSize(VideoDecoder* pDecoder, int nStreamBufIndex)
功能	查询码流 Buffer 中有效数据（未解码的数据）大小，单位为字节
参数	pDecoder: 通过 CreateVideoDecoder 函数创建的视频解码器指针; nStreamBufIndex: 对于蓝光 MVC 等 3D 视频，解码器需要处理两路码流，nStreamBufIndex 指定从第几路视频码流 Buffer 获取 Buffer。
返回值	码流 Buffer 中未解码的数据量，单位为字节。
调用说明	

3.1.11. VideoStreamFrameNum

函数原型	int VideoStreamFrameNum (VideoDecoder* pDecoder, int nStreamBufIndex)
功能	查询码流 Buffer 中有效数据（未解码的数据）有多少帧；
参数	pDecoder：通过 CreateVideoDecoder 函数创建的视频解码器指针； nStreamBufIndex：对于蓝光 MVC 等 3D 视频，解码器需要处理两路码流，nStreamBufIndex 指定从第几路视频码流 Buffer 获取 Buffer，0 表示第 0 路（MVC 主码流）、1 表示第 1 路（MVC 从码流）。
返回值	码流 Buffer 中未解码的数据有多少帧。
调用说明	

3.1.12. RequestPicture

函数原型	VideoPicture* RequestPicture(VideoDecoder* pDecoder, int nStreamIndex)
功能	获取一帧图像
参数	pDecoder：通过 CreateVideoDecoder 函数创建的视频解码器指针； nStreamIndex：对于蓝光 MVC 等 3D 视频，解码器有两路视频可供显示，nStreamIndex 指定从获取第几路视频的图像。
返回值	成功：返回图像 Buffer 指针； 失败：返回 NULL；
调用说明	

3.1.13. ReturnPicture

函数原型	int ReturnPicture(VideoDecoder* pDecoder, VideoPicture* pPicture)
功能	归还通过 RequestPicture 获取的视频图像给解码器
参数	pDecoder：通过 CreateVideoDecoder 函数创建的视频解码器指针； pPicture：通过 RequestPicture 函数获取的图像 Buffer；
返回值	0： 表示成功；-1：失败；
调用说明	

3.1.14. NextPictureInfo

函数原型	VideoPicture* NextPictureInfo(VideoDecoder* pDecoder,

	int nStreamIndex)
功能	获取下一帧输出图像的信息
参数	pDecoder：通过 CreateVideoDecoder 函数创建的视频解码器指针； nStreamIndex：对于蓝光 MVC 等 3D 视频，解码器有两路视频可供显示，nStreamIndex 指定从获取第几路视频的图像。
返回值	成功：返回下一帧待显示图像 Buffer 的指针； 失败：返回 NULL；
调用说明	与 RequestPicture 函数相比，本函数只是获取视频图像的信息，不会使该图像从解码器的显示队列中删除。

3.1.15. TotalPictureBufferNum

函数原型	int TotalPictureBufferNum(VideoDecoder* pDecoder, int nStreamIndex)
功能	询问解码器内总共有多少个图像 Buffer
参数	pDecoder：通过 CreateVideoDecoder 函数创建的视频解码器指针； nStreamIndex：对于蓝光 MVC 等 3D 视频，解码器有两路视频可供显示，nStreamIndex 指定从获取第几路视频的图像。
返回值	图像 Buffer 个数
调用说明	某些视频格式（H264 和 MPEG2）需要解码部分码流（SPS/PPS、Sequence Header）信息后才申请图像 Buffer，在此之前，本函数返回 0。

3.1.16. EmptyPictureBufferNum

函数原型	int EmptyPictureBufferNum(VideoDecoder* pDecoder, int nStreamIndex)
功能	询问解码器内有多少个空闲的图像 Buffer，即未被解码器和显示占用的图像 Buffer 个数。 nStreamIndex：对于蓝光 MVC 等 3D 视频，解码器有两路视频可供显示，nStreamIndex 指定从获取第几路视频的图像。
参数	pDecoder：通过 CreateVideoDecoder 函数创建的视频解码器指针；
返回值	空闲图像 Buffer 个数
调用说明	

3.1.17. ValidPictureNum

函数原型	int ValidPictureNum(VideoDecoder* pDecoder, int nStreamIndex)

功能	询问解码器内显示队列中有多少个待显示的图像。
参数	pDecoder：通过 CreateVideoDecoder 函数创建的视频解码器指针； nStreamIndex：对于蓝光 MVC 等 3D 视频，解码器有两路视频可供显示，nStreamIndex 指定从获取第几路视频的图像。
返回值	待显示的图像个数
调用说明	

3.1.18. ConfigHorizonScaleDownRatio

函数原型	int ConfigHorizonScaleDownRatio(VideoDecoder* pDecoder, int nScaleDownRatio)
功能	设置图像水平方向缩放比例因子。
参数	pDecoder：通过 CreateVideoDecoder 函数创建的视频解码器指针； nScaleDownRatio：图像缩放比例因子。
返回值	0： 表示成功；-1：失败；
调用说明	

3.1.19. ConfigVerticalScaleDownRatio

函数原型	int ConfigVerticalScaleDownRatio(VideoDecoder* pDecoder, int nScaleDownRatio)
功能	设置图像垂直方向缩放比例因子。
参数	pDecoder：通过 CreateVideoDecoder 函数创建的视频解码器指针； nScaleDownRatio：图像缩放比例因子。
返回值	0： 表示成功；-1：失败；
调用说明	

3.1.20. ConfigSecHorizonScaleDownRatio

函数原型	int ConfigSecHorizonScaleDownRatio(VideoDecoder* pDecoder, int nScaleDownRatio)
功能	设置从路通道图像水平方向缩放比例因子。
参数	pDecoder：通过 CreateVideoDecoder 函数创建的视频解码器指针； nScaleDownRatio：图像缩放比例因子。
返回值	0： 表示成功；-1：失败；
调用说明	当从路通道缩放比例需求与主路通道不同时，可通过调用此接口进行设置；

3.1.21. ConfigSecVerticalScaleDownRatio

函数原型	int ConfigSecVerticalScaleDownRatio(VideoDecoder* pDecoder, int nScaleDownRatio)
功能	设置从路通道图像垂直方向缩放比例因子。
参数	pDecoder：通过 CreateVideoDecoder 函数创建的视频解码器指针； nScaleDownRatio：图像缩放比例因子。
返回值	0：表示成功；-1：失败；
调用说明	当从路通道缩放比例需求与主路通道不同时，可通过调用此接口进行设置；

3.1.22. ReopenVideoEngine

函数原型	int ReopenVideoEngine (VideoDecoder* pDecoder)
功能	重新打开解码器内 Video Engine 模块
参数	pDecoder：通过 CreateVideoDecoder 函数创建的视频解码器指针；
返回值	0：表示成功；-1：失败；
调用说明	如果视频分辨率发生改变，DecodeVideoStream 函数会返回对应的值 VDECODE_RESULT_RESOLUTION_CHANGE，并将码流归还到码流 Buffer； 此时应用应该调用本函数重新打开 Video Engine 模块； 重新打开 Video Engine 模块会导致图像 Buffer 被释放，图像 Buffer 管理模块重新初始化。这一点显示控制需要注意。

3.1.23. AllocatePictureBuffer

函数原型	VideoPicture* AllocatePictureBuffer (int nWidth, int nHeight, int nLineStride, int ePixelFormat)
功能	申请一个图像 Buffer。
参数	nWidth：图像像素宽度； nHeight：图像像素高度； nLineStride：图像在内存中存放的行宽，以像素为单位； ePixelFormat：图像像素格式；
返回值	0：表示成功；-1：失败；
调用说明	本函数独立于 VideoDecoder 模块，是全局函数。

3.1.24. FreePictureBuffer

函数原型	int FreePictureBuffer (VideoPicture* pPicture)
功能	释放一个由 AllocatePictureBuffer 函数申请的图像 Buffer。
参数	pPicture：通过 AllocatePictureBuffer 函数申请的图像 Buffer；

返回值	成功：返回图像 Buffer 指针； 失败：返回 NULL
调用说明	本函数独立于 VideoDecoder 模块，是全局函数。

3.2. 流程设计

3.2.1. 码流数据传输流程

码流数据的传输通过 RequestVideoStreamBuffer 和 SubmitVideoStreamData 两个 API 函数完成。

RequestVideoStreamBuffer 函数从 Stream Buffer Manager 获取 Buffer 给外部程序，SubmitVideoStreamData 函数将外部提交的数据送入 Stream Buffer Manager。

3.2.2. 解码流程

解码工作通过 DecodeVideoStream 这个 API 函数完成。

3.2.3. 视频图像输出流程

视频图像的传送通过 RequestPicture 和 ReturnPicture 两个 API 函数完成。

RequestPicture 函数从 Frame Buffer Manager 模块取出一帧待显示的图像，送给外部程序。

ReturnPicture 函数将图像归还给 Frame Buffer Manager，使解码器可以继续使用该图像 Buffer 解码新的图像。

3.2.4. MB32 排列格式

VE 视频解码硬件默认输出 MB32 格式的图像。MB 是 Macro Block 的意思，MB32 是指以 32*32 大小的宏块作为一个存储单元，宏块的数据连续存放。因此，按照 MB32 格式存放 YUV420 数据时，存放顺序不是按照光栅扫描，而是按照如图 6.3.4-1 所示的扫描顺序：

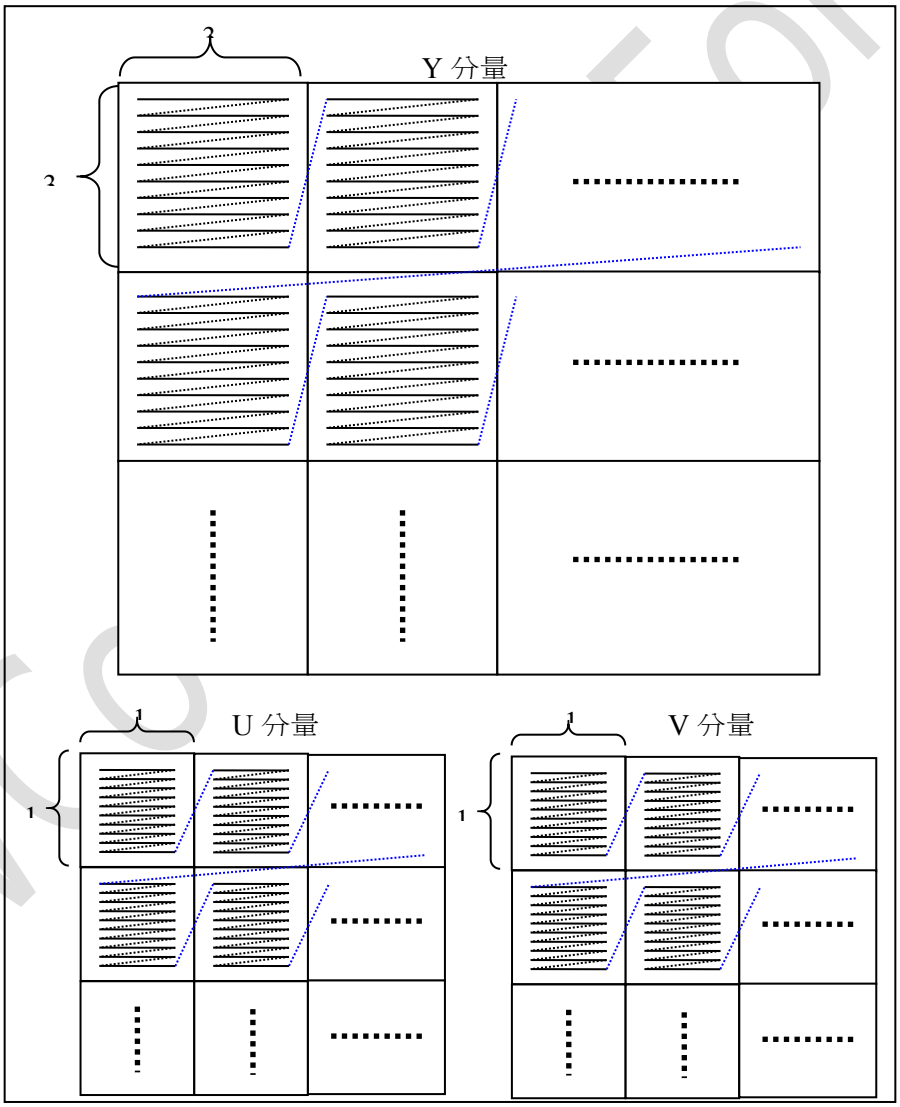

图 6.3.4-1. 按照 MB32 格式存放 YUV 数据时的扫描顺序

按照 MB32 格式存放 YUV 数据时，Y 分量和 UV 分量可以分开存放，但 UV 分量是存放在一起，且是间隔存放的，如图 6.3.4-2 所示：

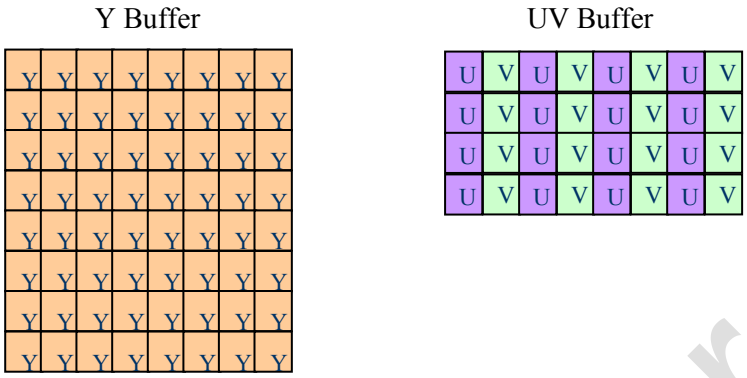

图 6.3.4-2. 按 MB32 格式存放 YUV420 数据

实际应用中，图像的宽度和高度可能不满足 32 像素对齐，此时需要将图像扩展到宽度和高度都满足 32 像素对齐（相应的，U 分量和 V 分量为 16 对齐）。