AVC Decoder Overview

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Document serial number: 000004892117

Table of Contents

1 Library Overview

Scope of This Document

This document describes the AVC Decoder library, which decodes AVC Elementary Stream data. It also describes the basic procedures and restrictions of the buffer region used for decoding.

Purpose and Features

The AVC Decoder library provides functions for decoding AVC Elementary Stream ("ES") data. Input the ES data format of the split BSF format for each access unit ("AU") accompanying PTD/DTS in units of 90 kHz as decodable AVC ES data. Dedicated hardware is used to perform AVC decoding and color space conversion ("CSC") from YCbCr to RGBA and to output the decoded results in RGBA or YCbCr in single frame units.

Main Functions

The AVC Decoder library provides the following main functions.

- Supports up to Baseline/Main/High Profile Lv3.1
  (Baseline does not include error tolerance tools (ASO, FMO, RS))
- Supports up to 1280 x 720 pixels picture frame (in 16-pixel units, horizontal: 64 to 1920, vertical: 64 to 1088)
- Supports up to the Lv 3.1 maximum number of reference images
- 960 x 544 pixels picture frame and 1 to 3 reference images are recommended
- The frame rate is 29.970 Hz
- Supports AVC decoding and high-speed color space conversion functions (conversion from YCbCr420 color space sampling to RGBA) using dedicated hardware

Embedding in Program

The files required for using the AVC Decoder library are as follows.

Filename	Description
videodec.h	Header file
libSceVideodec_stub.a	Stub library file

Include videodec.h in source program (some other header files will also be included automatically). When building the program, link libSceVideodec_stub.a.

Sample Programs

Refer to the following sample programs for the AVC Decoder library.

sample_code/audio_video/api_avcdec/decode/

This sample shows basic uses of the AVC Decoder library.

References

For the AVC format, refer to the following standards as necessary.

- ISO/IEC 14496-10:2012 Information technology -- Coding of audio-visual objects -- Part 10: Advanced Video Coding
  http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=61490

(The above reference destination has been confirmed as of May 29, 2013. Note that pages may have been subsequently moved or its contents modified.)

2 Usage

Basic Procedure

This describes the basic procedure for AVC decode processing. The following is an overview of the process flow.

(1) Initialize the library.
(2) Obtain the required buffer size for decoding and allocate the buffer.
(3) Use the allocated buffer to create an AVC decoder instance.
(4) Perform AVC decoding for each AU and obtain the output results for each frame.
(5) Use the decoded results for display.
(6) Repeat steps (4) and (5).
(7) When the end of the AVC ES data is reached, call the function for stopping AVC decoding, and remove the frames remaining in the AVC decoder until there are no more frames.
(8) Use the frames in step (7) for display, as needed.
(9) Delete the instance of the AVC decoder.
(10) Terminate the library.

(1) Initialize the library

Call `sceVideodecInitLibrary()` to perform initialization. To *`codecType`*, specify `SCE_VIDEODEC_TYPE_HW_AVCDEC` representing the video decoder type. When setting the initialization parameters, specify the AVC decodable area (up to a maximum of 1280 x 720 pixels in 16-pixel units, horizontal: 64 to 1920, vertical: 64 to 1088) to generate an AVC decode instance and the number of reference frames (up to the maximum number of Lv3.1) (960 x 544 pixels and up to 3 reference images are recommended) to the parameters of the `SceVideodecQueryInitInfoHwAvcdec` structure of the `SceVideodecQueryInitInfo` union. In addition, specify 1 to *`numOfStreams`* and specify `sizeof(SceVideodecQueryInitInfoHwAvcdec)` to *`size`*.

(2) Obtain the required buffer size for decoding and secure the buffer.

Call `sceAvcdecQueryDecoderMemSize()` to obtain the size of the frame memory buffer provided to the library for creating an AVC decoder instance. To *`codecType`*, specify `SCE_VIDEODEC_TYPE_HW_AVCDEC` representing the video decoder type. Specify the AVC decodable area (up to a maximum of 1280 x 720 pixels in 16-pixel units, horizontal: 64 to 1920, vertical: 64 to 1088) and the number of reference frames (up to the maximum number of Lv3.1) (960 x 544 pixels and up to 3 reference images are recommended) to the `SceAvcdecQueryDecoderInfo` structure. The size of the frame memory is returned to *`frameMemSize`* of the `SceAvcdecDecoderInfo` structure. Refer to "Precautions" and allocate the required buffer as the frame memory.

(3) Create an instance of the AVC decoder.

Call `sceAvcdecCreateDecoder()` to create an AVC decoder instance. Allocate the buffer for the size obtained in step (2), and specify the pointer and size to the *`frameBuf`* member of the `SceAvcdecCtrl` structure. Specify the same parameters specified in step (2) to the `SceAvcdecDecoderInfo` structure.

(4) Perform AVC decoding for each AU and obtain the output results for each frame.

Call `sceAvcdecDecode()` to perform AVC decoding. Use the AVC decode instance (`SceAvcdecCtrl` structure) obtained in step (3).

Input the AVC ES data format of the split BSF format for each AU accompanying PTD/DTS in units of 90 kHz to the `SceAvcdecAu` structure. See "Time Stamp of Input Access Unit" for more information on the time stamp.

Configure the memory so that the pointer to the `SceAvcdecPicture` structure is stored at the destination of the *`pPicture`* pointer of the `SceAvcdecArrayPicture` structure, and specify 1 to *`numOfElm`*. When decode output is obtained, *`numOfOutput`* becomes 1. Assign `sizeof(SceAvcdecPicture)` to the *`size`* member variable of the `SceAvcdecPicture` structure. In addition, set the parameters for *`pixelType`*, *`framePitch`*, *`frameWidth`*, and *`frameHeight`* of the `SceAvcdecFrame` structure, which are *`frame`* member variables, according to how they are used in step (5). In addition, allocate the buffer of the frame data that is output with the above parameters, set the pointer at the start of the buffer to *`pPicture`*`[0]`, and specify NULL to *`pPicture`*`[1]`. See "Frame Data of Decoded Result" for the required buffer size and restrictions.

To control the output information using the `SceAvcdecFrameOption` structure, add `SCE_AVCDEC_OPTION_ENABLE` to *`pixelType`*.

If no information is output due to the status of calling `sceAvcdecDecode()`, *`numOfOutput`* becomes 0. In addition, the ES buffer may become full depending on the internal status of the AVC decoder. In this case, the `SCE_AVCDEC_ERROR_ES_BUFFER_FULL` error is returned, so input the same AU again.

Unlike other errors, when the `SCE_AVCDEC_ERROR_ES_BUFFER_FULL` error occurs, the decoded output may be stored. Check the number of decoding information units in *`numOfOutput`* of the `SceAvcdecArrayPicture` structure.

(5) Use the decoded results.

Use the decoded results for screen display.

The address to the frame that was decode output is stored to the `SceAvcdecFrame` structure, which is a *`frame`* member variable of the `SceAvcdecPicture` structure and the frame data to *`pPicture`*`[0]`.

At the time the frame data is stored, if the decoded result is greater than either *`frameWidth`* or *`frameHeight`*, which were provided in step (4), the decoded result is reduced in size by the hardware with a bilinear conversion and is output. Refer to the values of other `SceAvcdecFrame` structures and the value of the `SceAvcdecInfo` structure of the *`info`* member variable, as needed.

(6) Repeat steps (4) and (5).

Repeat steps (4) and (5) until the end of the AVC stream is reached.

(7) Stop AVC decoding.

When the end of the AVC ES data to be decoded is reached, or to stop decoding midway, call `sceAvcdecDecodeStop()` or `sceAvcdecDecodeFlush()` to stop the AVC decoder.

To obtain all output frame data from the AVC ES data that was input to the AVC decoder, repeatedly call `sceAvcdecDecodeStop()` and use step (8) to display the data until *`numOfOutput`* of the `SceAvcdecArrayPicture` structure reaches 0.

To delete all output frame data remaining in the AVC decoder, call `sceAvcdecDecodeFlush()`.

(8) Display the frames in step (7), as needed.

To obtain all output frame data from the AVC ES data that was input to the AVC decoder, repeatedly call `sceAvcdecDecodeStop()`, and obtain and display the data until *`numOfOutput`* of the `SceAvcdecArrayPicture` structure reaches 0.

(9) Delete the instance of the AVC decoder.

Call `sceAvcdecDeleteDecoder()` to delete the AVC decoder instance. This will release resources allocated when the AVC decoder instance was created.

(10) Terminate the library.

Call `sceVideodecTermLibrary()` to terminate the library. This will release resources allocated at initialization.

Major APIs Used in Basic Processing

API	Description
`sceVideodecInitLibrary()`	Initializes the library
`sceVideodecTermLibrary()`	Terminates the library
`sceAvcdecQueryDecoderMemSize()`	Obtains the frame memory size required for creating the AVC decoder instance
`sceAvcdecCreateDecoder()`	Creates an instance of the AVC decoder
`sceAvcdecDeleteDecoder()`	Deletes the instance of the AVC decoder
`sceAvcdecDecode()`	Decodes the AVC ES data for one AU
`sceAvcdecDecodeStop()`	Stops the AVC decoder (Removes the frame data residing in the AVC decoder)
`sceAvcdecDecodeFlush()`	Stops the AVC decoder (Deletes the frame data residing in the AVC decoder)

Time Stamp of Input Access Unit

When decoding with `sceAvcdecDecode()`, either PTD/DTS must be input in units of 90 kHz together with ES data for one AU, or a Picture Timing SEI/Buffering Period SEI must be stored in the stream. If both are set, the PTS/DTS specification has priority.

To use only a Picture Timing SEI/Buffering Period SEI, store `SCE_VIDEODEC_VOID_TIMESTAMP(0xffffffff)` to the *`upper`* and *`lower`* members of the `SceVideodecTimeStamp` structure used to store PTS/DTS.

If there is neither a PTS/DTS specification nor Picture Timing SEI/Buffering Period SEI, the output order cannot be guaranteed.

PTS Added to the Decode Output Result

The display time PTS of the decode output result is added to the *`outputPts`* member variable of the `SceAvcdecInfo` structure and is output to the decode output result. When decoding is performed with `sceAvcdecDecode()` and PTS/DTS is entered in units of 90 kHz together with the ES data of one AU, the PTS rearranged in order of output is stored, and the values synchronized with display time of the frame data for the decode result are stored.

During operations with only Picture Timing SEI/Buffering Period SEI, at a frame rate of 29.97 Hz, the value is interpolated and stored, and at other frame rates, the interpolated PTS value cannot be guaranteed. Take note of this during operation.

Frame Data of Decoded Result

The frames of the decoded result obtained with `sceAvcdecDecode()` and `sceAvcdecDecodeStop()` are stored in RGBA, BGRA (32-bit) or YCbCr format as images that were raster scanned from top left to bottom right.

YCbCr has two formats. When `SCE_AVCDEC_PIXEL_YUV420_RASTER` is specified, Y, Cb, and Cr are raster images in separate formats. When `SCE_AVCDEC_PIXEL_YUV420_PACKED_RASTER` is specified, Y is separate, but Cb and Cr are raster images in a format separated by chroma in 1-pixel units.

The decoded result is written to the pointer address stored in *`pPicture`*`[0]` of the `SceAvcdecFrame` structure, which is the *`frame`* member variable of the `SceAvcdecPicture` structure. Allocate the buffer with a size of *`framePitch`* x *`frameHeight`* x *`pixelType`* with a 256-byte alignment in an uncached continuous area (custom DRAM or physical continuous memory on the main memory) of the physical address. (Refer also to "Buffer Restrictions for Positioning Data".)

The value of *`pixelType`* becomes 4 when *`pixelType`* is `SCE_AVCDEC_PIXEL_RGBA8888` or `SCE_AVCDEC_PIXEL_BGRA8888`, or 1.5 when `SCE_AVCDEC_PIXEL_YUV420_RASTER` or `SCE_AVCDEC_PIXEL_YUV420_PACKED_RASTER`. For example, when *`framePitch`* = 512 and *`frameHeight`* = 272, and *`pixelType`* = `SCE_AVCDEC_PIXEL_RGBA8888`, allocate a buffer with a size of 557056 bytes (512 x 272 x 4).

The α value is 255 by default. To change the value, add `SCE_AVCDEC_OPTION_ENABLE` to the *`pixelType`* member variable of the `SceAvcdecFrame` structure, and change the *`alpha`* member variable of the `SceAvcdecFrameOptionRGBA` structure. In addition, adding `SCE_AVCDEC_OPTION_ENABLE` enables the *`cscCoefficient`* member of the `SceAvcdecFrameOptionRGBA` structure. To reset to the default value, use `SCE_AVCDEC_CSC_COEFFICIENT_DEFAULT`.

Set *`framePitch`* and *`frameWidth`* in multiples of 16, from 64 to 1920.

However, when *`pixelType`* is `SCE_AVCDEC_PIXEL_YUV420_RASTER`, *`framePitch`* must be in multiples of 32. Set *`frameHeight`* in multiples of 16, from 64 to 1088. In addition, the *`framePitch`* x *`frameHeight`* area must be no greater than 1280 x 720.

The number of horizontal and vertical pixels of the decoded result is stored to *`horizontalSize`* and *`verticalSize`* of the `SceAvcdecFrame` structure. Set *`frameWidth`* and *`frameHeight`* with the value between one-fourth and four times the value of *`horizontalSize`* and *`verticalSize`*.

When *`frameWidth`* or *`frameHeight`* of the storage destination frame of the decoded result is smaller than *`horizontalSize`* or *`verticalSize`*, the number of horizontal or vertical pixels, of the decoded result, the decoded result is output according to *`frameWidth`* or *`frameHeight`*. When the decoded result is the same value or less than *`frameWidth`* or *`frameHeight`*, the decoded result is output with the number of pixels of *`horizontalSize`* or *`verticalSize`*.

Memory Required for Decoding

The following memory areas are required to perform decoding.

- Memory area used with `sceVideodecInitLibrary()`
- Memory area used with `sceAvcdecCreateDecoder()`

In all other cases, there must be a memory area for positioning input ES data and an area for positioning frame data for fetching the decode result, and these are used with `sceAvcdecDecode()` and `sceAvcdecDecodeStop()`. In cases other than input ES data, the memory area has restrictions, so refer to "Buffer Restrictions for Positioning Data" in chapter 3 for details.

Interlaced Streams

When handling interlaced streams, always call `sceAvcdecSetInterlacedStreamMode()` before executing `sceAvcdecQueryDecoderMemSize()`. When only using progressive streams without handling any interlaced streams, do not call `sceAvcdecSetInterlacedStreamMode()`.

The `SceAvcdecArrayPicture` structure is used for obtaining the frames of decoded results and decoding information using `sceAvcdecDecode()` and `sceAvcdecDecodeStop()`, but when handling interlaced streams, in *`pPicture`* in the `SceAvcdecArrayPicture` structure, input the start pointer value of a pointer array with a pointer to an `SceAvcdecPictureForInterlaced` structure stored instead of an `SceAvcdecPicture` structure.

In the frames for the results decoded with `sceAvcdecDecode()` and `sceAvcdecDecodeStop()`, two fields (top field and bottom field) are simultaneously output in one frame in interlaced streams.

If decoded output exists after decoding, a pointer to the earlier field from among the two fields will be output to *`pPicture`*`[0]` in the `SceAvcdecFrame` structure, and a pointer to the later field will be output to *`pPicture`*`[1]`. Determine whether the images stored in *`pPicture`*`[0]` and *`pPicture`*`[1]` are top fields or bottom fields using the values of *`picStruct`* and *`ctType`* in the `SceAvcdecPictureForInterlaced` structure.

Even though interlaced streams can be handled by calling `sceAvcdecSetInterlacedStreamMode()`, if the input stream is progressive, a pointer input for *`pPicture`*`[0]` in the `SceAvcdecFrame` structure will return as-is. A valid value will not be input to *`pPicture`*`[1]`, so if NULL is set before executing `sceAvcdecDecode()` or `sceAvcdecDecodeStop()`, NULL will return as-is.

For interlaced streams, pointers to two fields with frames that have half the number of vertical pixels of a progressive stream will be stored in *`pPicture`*`[0]` and *`pPicture`*`[1]`.

Note that the AVC decoder library does not support progressive conversion of interlaced output results. Perform appropriate conversion and display with the GPU, etc.

The output image in memory for the storage destination frame when decoding an interlaced stream is shown in the following.

Note that if the decoded results are equal to *`frameWidth`* and *`frameHeight`* or less, the decoded result output will be in the number of pixels for *`horizontalSize`* and *`verticalSize`*. For details, refer to the "Frame Data of Decoded Result" section.

Figure 1 Frame Storage Image For Progressive Streams

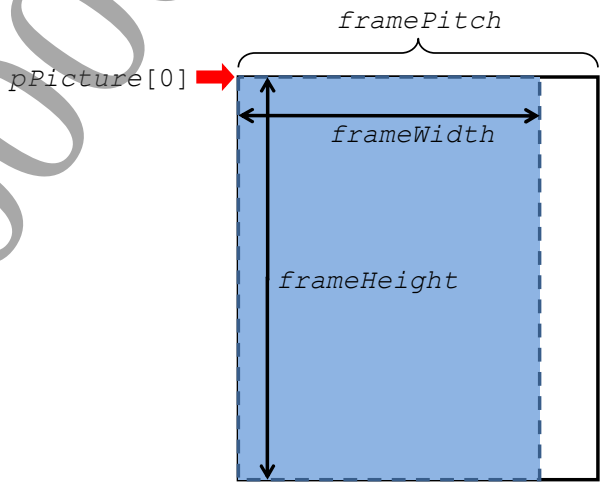


Figure 2 Frame Storage Image For Interlaced Streams

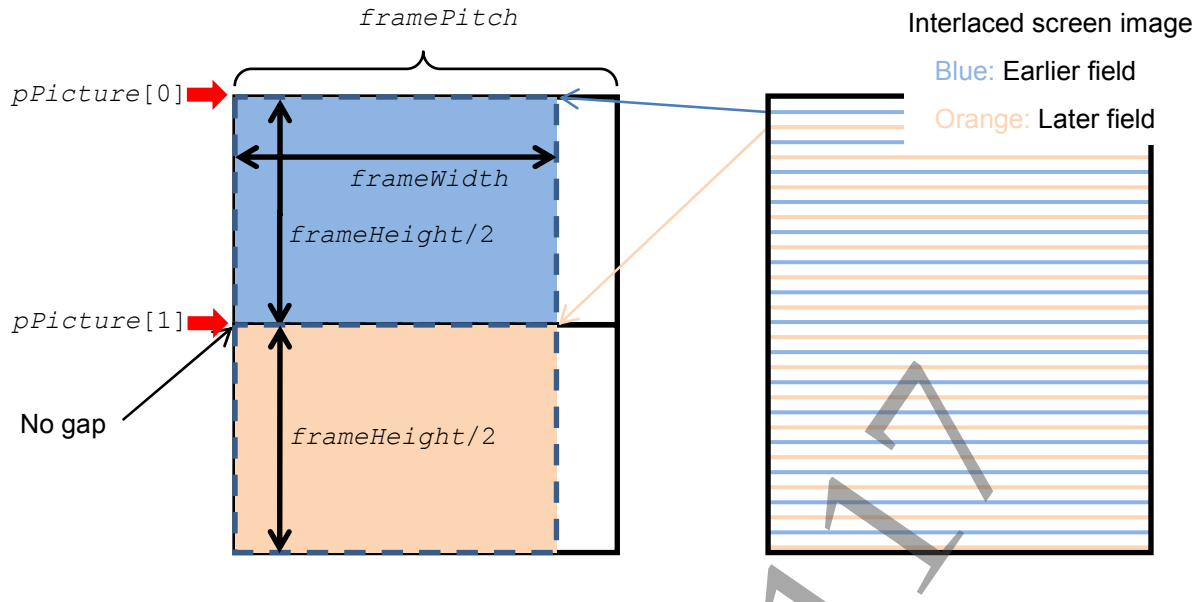


Figure 3 Interlaced Streams: 720 x 240 2-field Output Example With SCE_AVCDEC_PIXEL_YUV420_PACKED_RASTER and *frame(Pitch,Width,Height)*=(720,720,480) for the Output

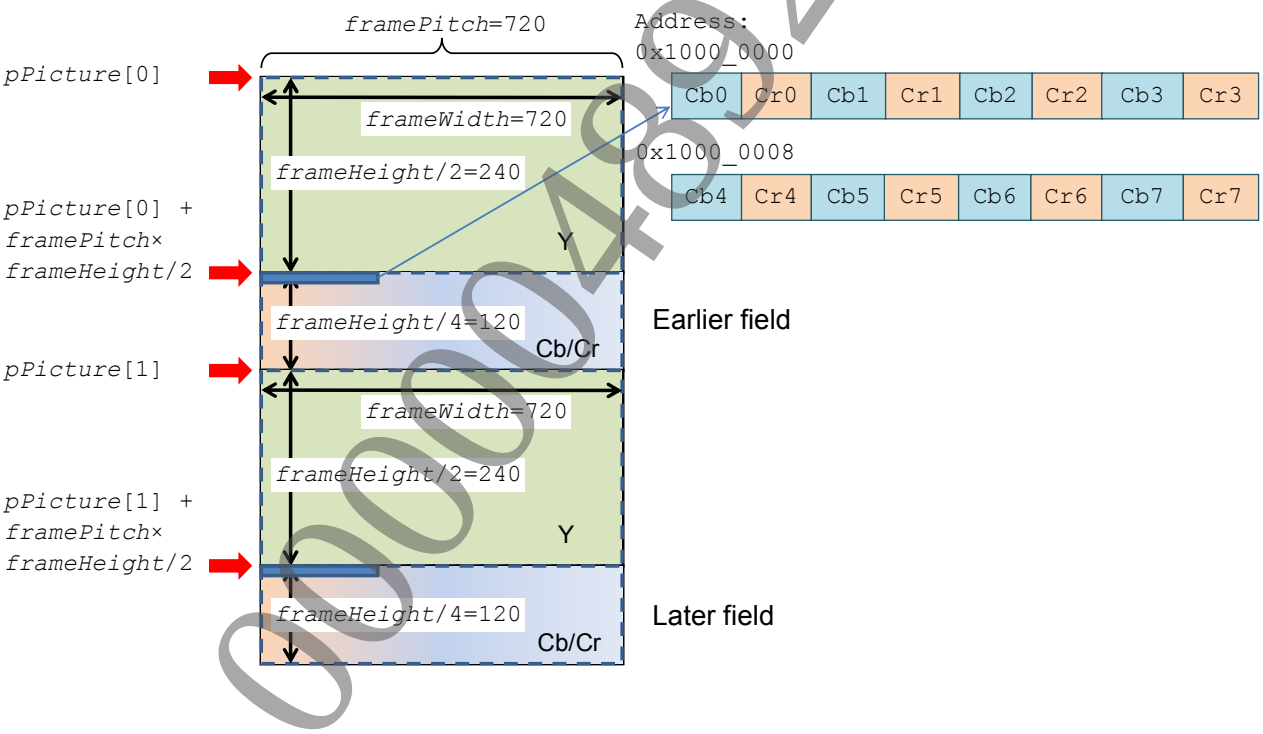
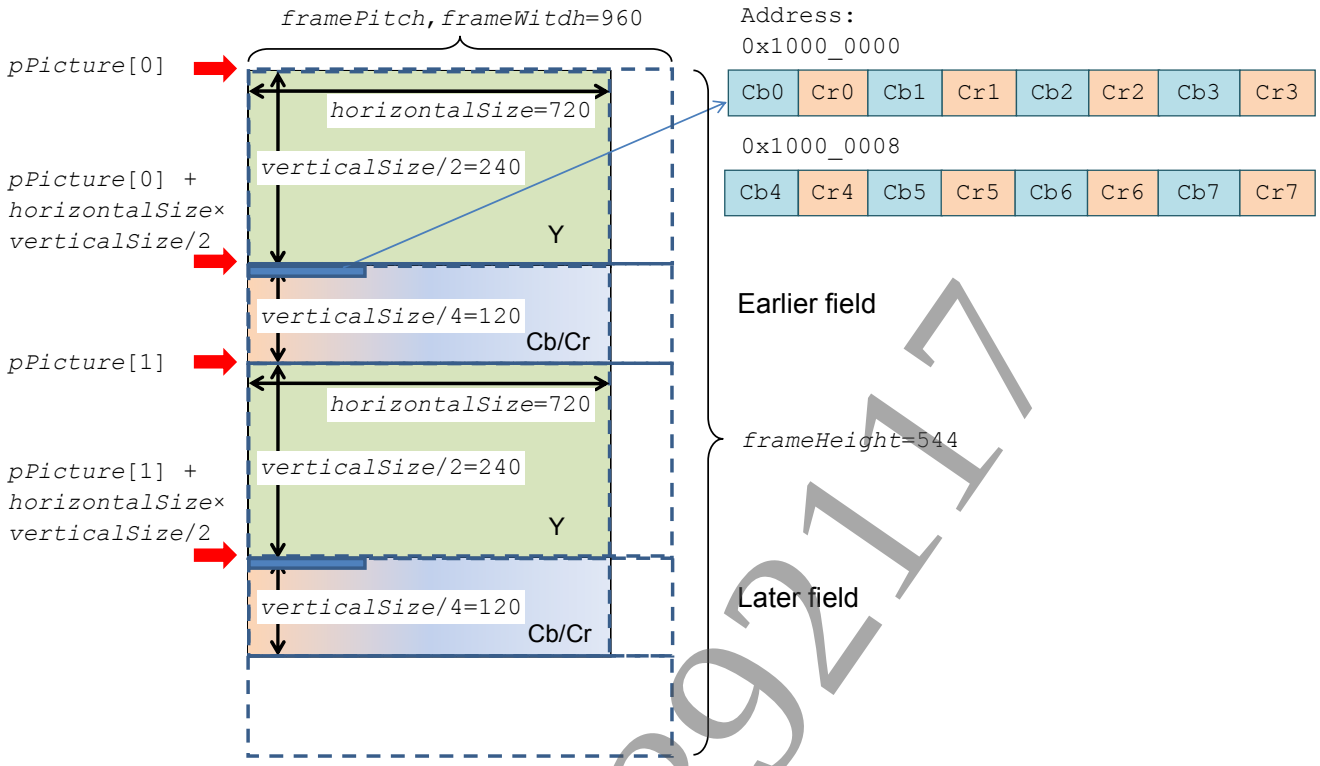


Figure 4 Interlaced Streams: 720 x 240 2-field Output Example With SCE_AVCDEC_PIXEL_YUV420_PACKED_RASTER and *frame(Pitch,Width,Height)*=(960,960,544) for the Output



3 Precautions

Buffer Restrictions for Positioning Data

The output frame buffer must have an uncached continuous physical address space (custom DRAM or physical continuous memory on the main memory) and must also fulfill 256-byte alignment. If these conditions are not met, an error will be returned.

The buffer specified to `sceAvcdecCreateDecoder()`, which is used to create an AVC decoder instance, is allocated in an uncached continuous physical address space (custom DRAM) by the AVC Decoder library when NULL and 0 are specified to the pointer and size of the *`frameBuf`* member variable of the `SceAvcdecCtrl` structure respectively. When directly specifying the buffer, set a region that has an uncached continuous physical address space (custom DRAM or physical continuous memory on the main memory) and fulfills 1-MiB alignment.

The maximum number of reference frames of Lv 3.1 can be specified. Imprudently specifying a large number will cause an increase in used memory. The recommended value for normal cases is 3. With the exception of cases requiring more frames, do not specify a value greater than 3.

Although an uncached continuous physical address space (custom DRAM or physical continuous memory on the main memory) can be specified with a physical address for the output frame buffer or buffer specified to `sceAvcdecCreateDecoder()`, if an uncached continuous physical address space on the main memory is specified, the process for decoding will become slow. Specify a custom DRAM if high-speed decoding is required.

With `sceVideodecInitLibrary()`, follow the parameters specified with the `SceVideodecQueryInitInfo` union to create a video decode instance. At this time, the video decoder allocates a memory area of a maximum 6 MiB and with a 256-KiB alignment in the uncached continuous physical address space on the available main memory.

Streams Greater than 1200 Horizontal Pixels at Lv 3.0 or Lower

The correct frame data value of the decode output result cannot be obtained when the input ES is Lv 3.0 or lower and the number of horizontal pixels of the picture frame of the decode result exceed 1200. When the number of horizontal pixels exceeds 1200, use an input stream of Lv 3.1. If the stream in question is decoded, the error code `SCE_AVCDEC_ERROR_GREATER_THAN_1200_AT_LV30` is returned.

1field=1AU Type Interlaced Streams

When handling `1field=1AU` type interlaced streams, if `sceAvcdecDecodeStop()` is issued after only 1AU is decoded, an `SCE_AVCDEC_ERROR_INVALID_PICTURE` error will occur. Always input `2AU`. Use caution when implementing features that decode close to the GOP start, such as fast forward and rewind features.