PVRTC Specification and User Guide

Public. This publication contains proprietary information which is subject to change without notice and is supplied 'as is' without warranty of any kind. Redistribution of this document is permitted with acknowledgement of the source.

Filename	:	PVRTC Specification and User Guide
Version	:	PowerVR SDK REL_18.1@5080009a External Issue
Issue Date	:	31 May 2018
Author	:	Imagination Technologies Limited

Contents

List of Figures

List of Tables

1. Introduction

Texture compression is an important tool in the arsenal of developers that, due to its many advantages over uncompressed formats, should be used whenever it is feasible to do so. This document contains a brief explanation of why texture compression should be used as well as an introduction to PowerVR Texture Compression (PVRTC). PVRTC is available in two versions, namely PVRTC1 and PVRTC2. Each version offers two bit depths per version, 2bpp and 4bpp. Both versions are described in this document.

1.1. Benefits of Texture Compression

Modern applications have become graphically intensive. Certain types of software, such as games or navigation aids, often need large amounts of textures in order to represent a scene with satisfying quality. Texture compression can save or allow better utilisation of bandwidth, power, and memory without noticeably losing graphical quality. The main benefits of texture compression are:

- **Performance improvement**: A smaller texture data size means smaller transfers from memory to the graphics core. Memory bandwidth is precious, particularly in mobile platforms where shared memory architectures are prevalent. In situations where memory bandwidth is the limiting factor in an application's performance, texture compression can provide a significant improvement.
- **Storage footprint vs. memory footprint**: Texture compression reduces the memory footprint of a given texture. This allows applications to fit all their required textures in a constrained amount of texture memory, or to use larger (or more) textures for the same memory budget resulting in, potentially, extra quality. In addition, any savings in memory requirements are very useful for mobile and tablet devices where, as mentioned, memory is shared across an entire System on Chip (SoC).
- **Power consumption**: Memory accesses are expensive in terms of power consumption on mobile devices where battery life is of the upmost importance. The bandwidth savings and better cache usage resulting from run-time texture compression both contribute to decreasing the quantity and magnitude of memory accesses. This in turn reduces the power consumption of an application.

1.2. Image File Storage Compression vs. Texture Compression

Developers are familiar with compressed image file formats such as JPG and PNG. However, it is important to remember that there is a distinction between these forms of 'storage' compression and texture compression discussed in this document.

The primary requirement of 'storage' compression schemes is that files compressed using them should occupy as small an amount of storage in a file system as possible. There is no requirement that the data stay compressed while in use. The result is that 'storage-based' image file formats tend to produce very small file sizes, often for very high (or lossless) image quality, but at the cost of immediate decompression when the image is loaded into an application and used. This immediate decompression (usually to 24bpp or 32bpp) means that the image, while small on disk, consumes large amounts of bandwidth and memory when used as a texture.

Texture compression schemes, such as PVRTC, are designed to be directly usable by the graphics core. The texture data exists in storage, in memory, and when transferred to the graphics hardware itself, in the compressed format. The only step in which full-precision colour values are extracted from a compressed state is when the texture sampling hardware inside the graphics accelerator passes texel values to the shader processing units. This is a process that is entirely separate from the main memory bus. A graphical representation of this can be seen in Figure 1.

This allows all the previously mentioned advantages to be gained, but puts some limits on the form the compression technique may take. In order to allow for direct use by the graphics accelerator, a texture format should be optimised for random access, with a minimal size of data from which to retrieve the values for each texel. Consequently, texture compression schemes are usually fixed bitrate with very high data locality. Image file formats are not constrained by these requirements and thus can often achieve higher compression ratios and image quality for a given data size.

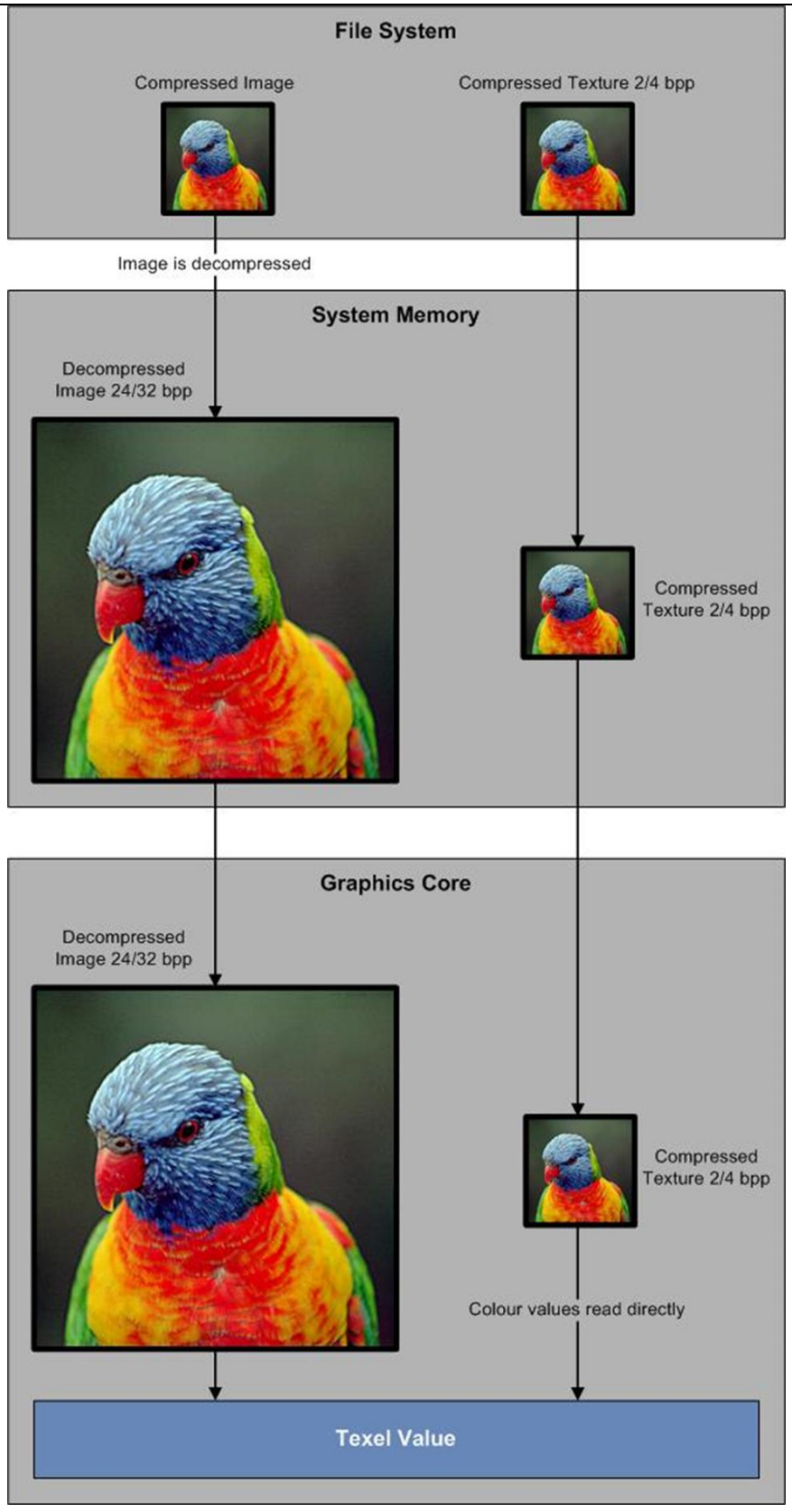

Figure 1. Image file compression vs. texture compression

2. PVRTC Overview

PVRTC is PowerVR's family of proprietary texture compression schemes, providing compression of colour data at 4 or 2 bits per pixel (4/2bpp). It uses an interpolation and modulation scheme to compress texture data. Texture data is encoded as two low-resolution images, along with a full resolution, low bit-precision modulation signal. More information on the specifics of this compression technique can be found in the whitepaper "Texture Compression using Low-Frequency Signal Modulation".

There are two generations of PVRTC, PVRTC1 and PVRTC2, both supporting 4bpp and 2bpp compression ratios. They are broadly similar, but PVRTC2 adds additional features to the format.

PVRTC supports both opaque (RGB) and translucent (RGBA) textures in the same data footprint (unlike other formats such as S3TC that require additional data to support a full alpha channel). PVRTC also boasts a very high image quality for competitive compression ratios: 4 or 2 bits per pixel. These represent savings in memory footprint of 8:1 or 16:1, compared to uncompressed, 32 bit per pixel textures. Alpha channel data is encoded on a per-data-word basis so that fully opaque sections of a texture do not needlessly encode alpha information.

In PVRTC, data is arranged in 64-bit words, each of which contains a pixel from each of the low-resolution images and modulation data for either a 4x4 or 8x4 set of pixels. Unlike traditional block-based formats, PVRTC uses adjacent data-words to reconstruct the original image. This can represent a considerable visual enhancement compared to block-based compression techniques that use the contents of a single block alone to reconstruct the texel values within that individual block.

Imagination's PVRTexTool can be used to generate PVRTC texture data from source images in most formats.

2.1. Benefits of PVRTC

In any given situation, the best texture format to use is the one that gives the required image quality at the highest rate of compression. The smaller the size of the texture data, the less bandwidth is required for texture fetches. This reduces power consumption, can increase performance, and allows for more textures to be used for the same budget. The smallest RGB and RGBA format currently available is PVRTC 2bpp, which should be considered for every texture in an application. A larger format, such as PVRTC 4bpp, should only be used if a texture encoded using PVRTC 2bpp does not have sufficient quality. The following are some considerations to be remembered:

- It is important to consider textures on a case by case basis rather than applying a blanket format to all textures that may result in unnecessary bandwidth wastage.
- Once a texture is loaded into the graphics API that is being used (e.g. OpenGL ES), then the use of this texture is the same irrespective of format so that only the upload code needs to worry about variable formats. For an example of texture loading code that can deal with many formats, please consult the `PVRTTexture` functions found in the PVRTools folder of the PowerVR Graphics Tools and SDK.
- It is suggested that texture encoding be carried out using a script and the PVRTexTool Command Line (also available in the Tools and SDK) in order to ease this selective choice of format per texture.
- PVRTC support can be checked at runtime through the graphics API, for example, look for `GL_IMG_texture_compression_pvrtc` in the OpenGL ES extension string.

2.2. PVRTC1 vs. PVRTC2

As with PVRTC1, PVRTC2 supports RGB and RGBA textures at 4bpp and 2bpp, and also has the following advantages:

- **Image quality**: In general, PVRTC2 will give a better quality image than PVRTC1. In images with large areas of colour discontinuity, such as those in Figure 2, the quality will be significantly higher without the need for texture manipulation techniques such as border expansion. A border around images in a skybox is also no longer needed and non-tiling textures will appear

much improved. This is due to the addition of dedicated modes for dealing with areas of high contrast between sections of a texture.

Figure 2. Texture atlas and normal map

- **Non-Power-of-Two (NPOT) dimensions**: PVRTC2 offers the developer the use of arbitrary-sized NPOT textures, which are textures that do not have dimensions that are limited to powers of two.
- **Sub-texturing**: Unlike PVRTC1, sub-texturing is supported in PVRTC2 at data-word boundaries (4x4 or 8x4 for PVRTC2 4bpp or PVRTC2 2bpp, respectively). This further enables techniques such as texture atlasing for applications. It should be noted that this requires `Hard Transition Flag` mode to be enabled around the sections of a texture to be replaced.

2.3. Comparisons against other Compression Formats

Figure 3 to Figure 6 identify comparison images between different texture compression formats.

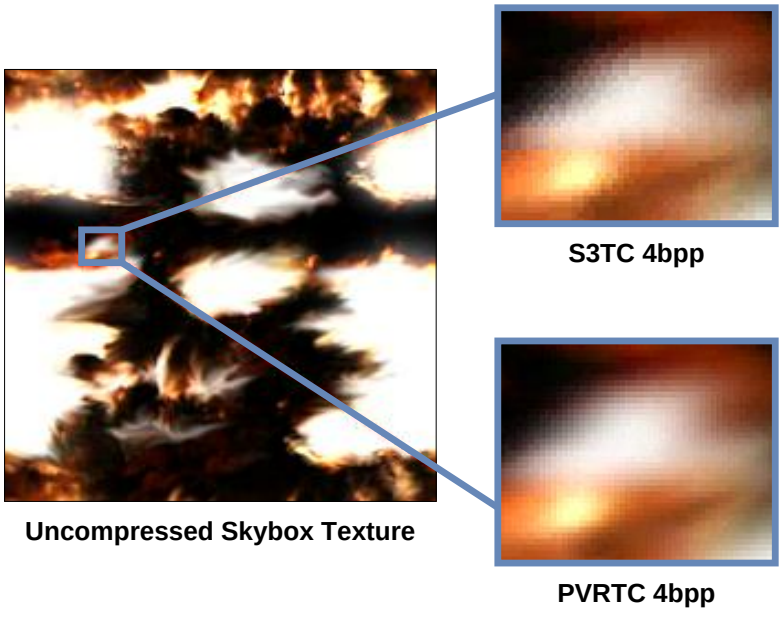

Figure 3. PVRTC1 4bpp vs. S3TC (DXT1): skybox texture

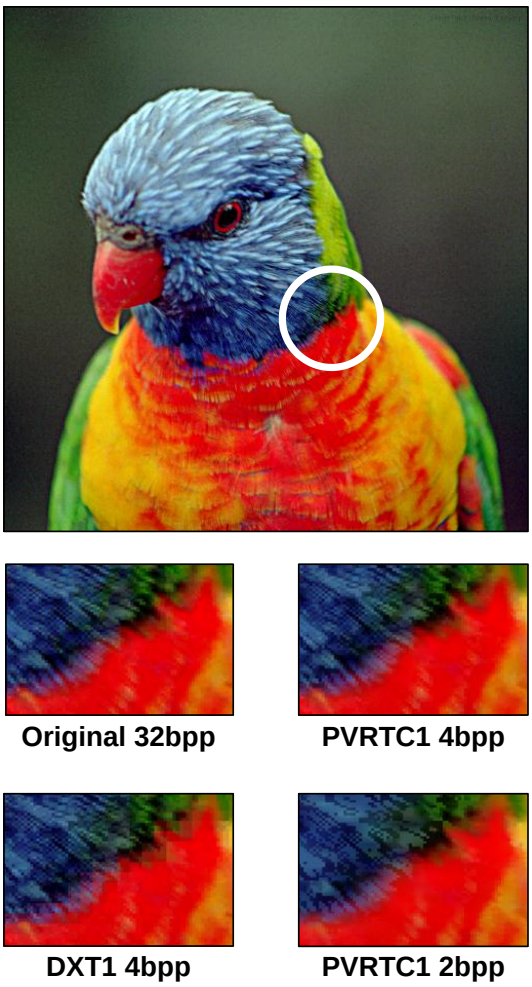

Figure 4. PVRTC1 vs. S3TC (DXT1): photograph

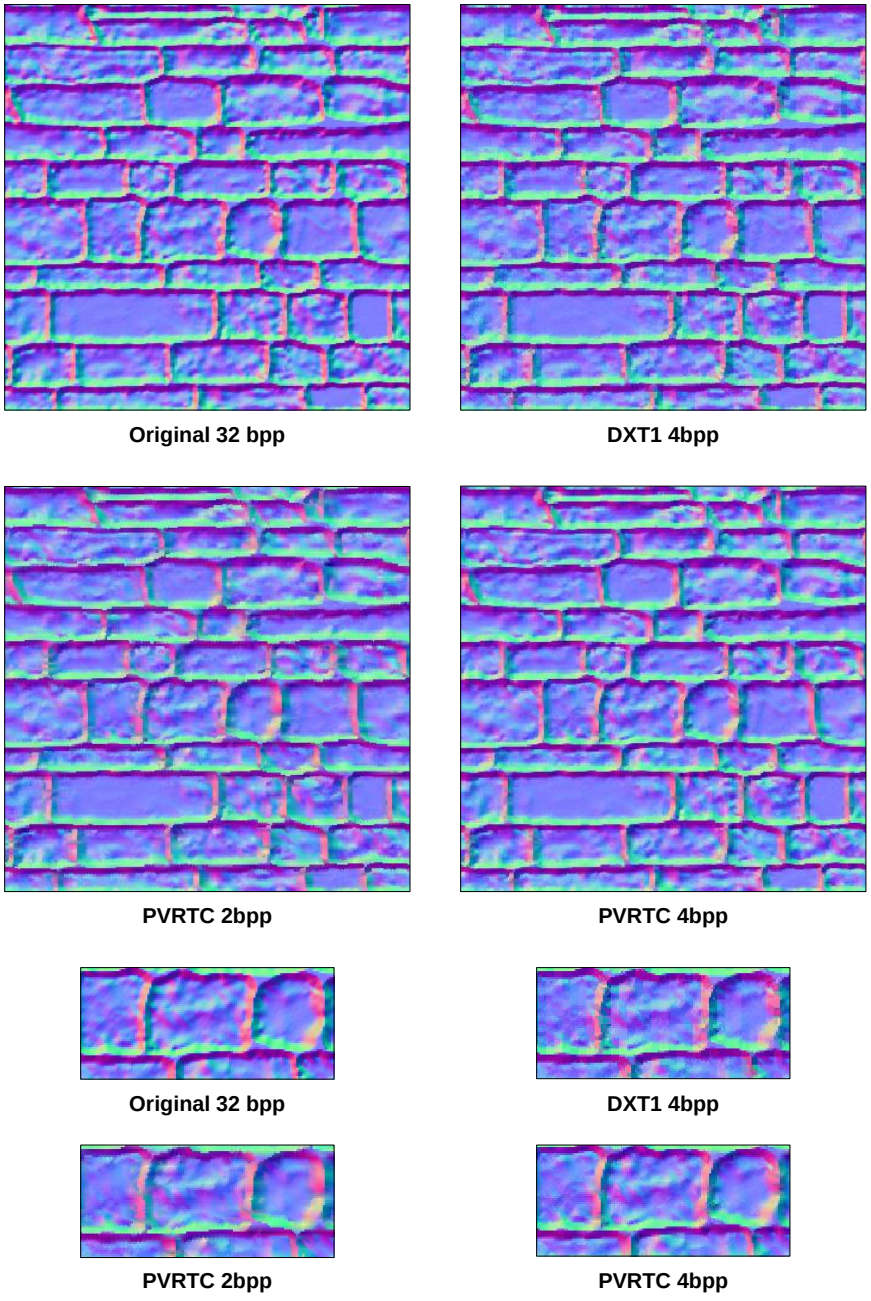

Figure 5. PVRTC1 vs. S3TC (DXT1): normal map

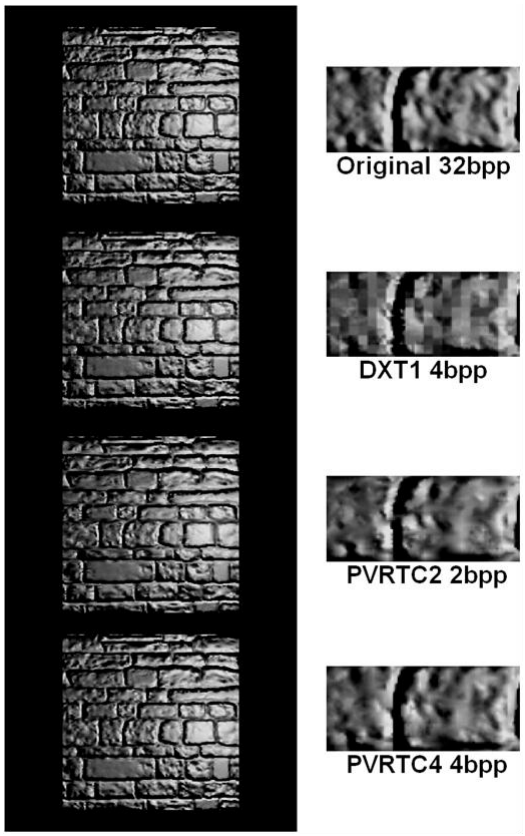

Figure 6. Usage of normal map from Figure 5

3. PVRTC Technical Specification

3.1. Format PVRTC1 4bpp

For PVRTC1 4bpp, each 64-bit word Wi,j contains the modulation information for a 4x4 block of texels beginning at (4i, 4j), and colour information relating to an overlapping 7x7 texel region, centered on (_4i+2, 4j+2). Each word does not entirely describe the matching pixel locations in the original image, as each region in the reconstructed image is produced by examining multiple data words in the compressed image data. At least one word of PVRTC data must exist in each mip level of a texture. If a texture (or a particular mip-level) is smaller than the size of one word in any dimension, reconstruction of an image occurs as if fetching the upper-left most texels from the region covered by those words. PVRTC1 images must have dimensions that are power of two values.

Data words in PVRTC1 formats are stored in a reflected Morton order, as shown in the figure below:

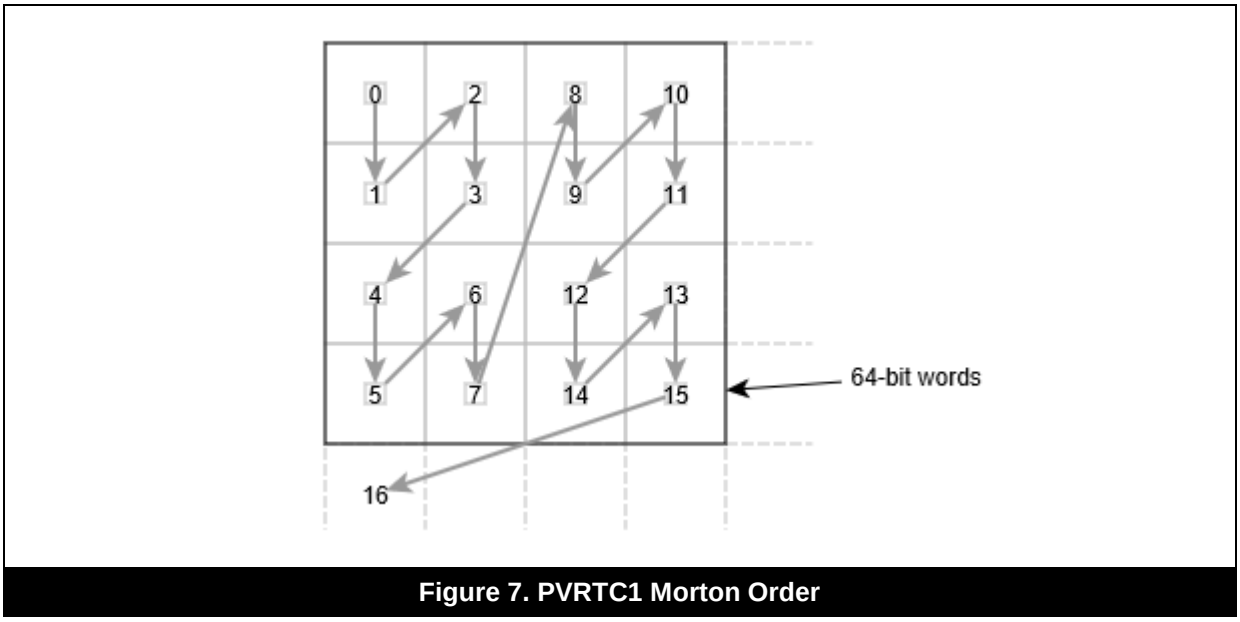

Figure 7. PVRTC1 Morton Order

Expressing each of the U and V indices as an array of bits, the index of a particular PVRTC word can be found by interleaving the bits of the x and y indices as follows:

Word U indexUM-1…UNUN-1…U2U1U0Word V indexVN-1…V2V1V0Reflected Morton offsetUM-1…UNUN-1VN-1…U2V2U1V1U0V0

When reconstructing the image, for a given pixel location, the word containing data for that region is fetched, as well as the three nearest PVRTC words. The texture data is wrapped toroidally, such that the "nearest" word may exist on the opposite side of the image. For example, sampling a pixel in any corner of the image results in the words in all four corners being examined - or sampling a pixel at the top of the image will result in a word from the bottom of the image being examined.

In Figure 8 below, P, Q, R and S are the words corresponding to the region of interest for the texel. The region of interest exceeds the dimensions of the image, so wraps back to the top.

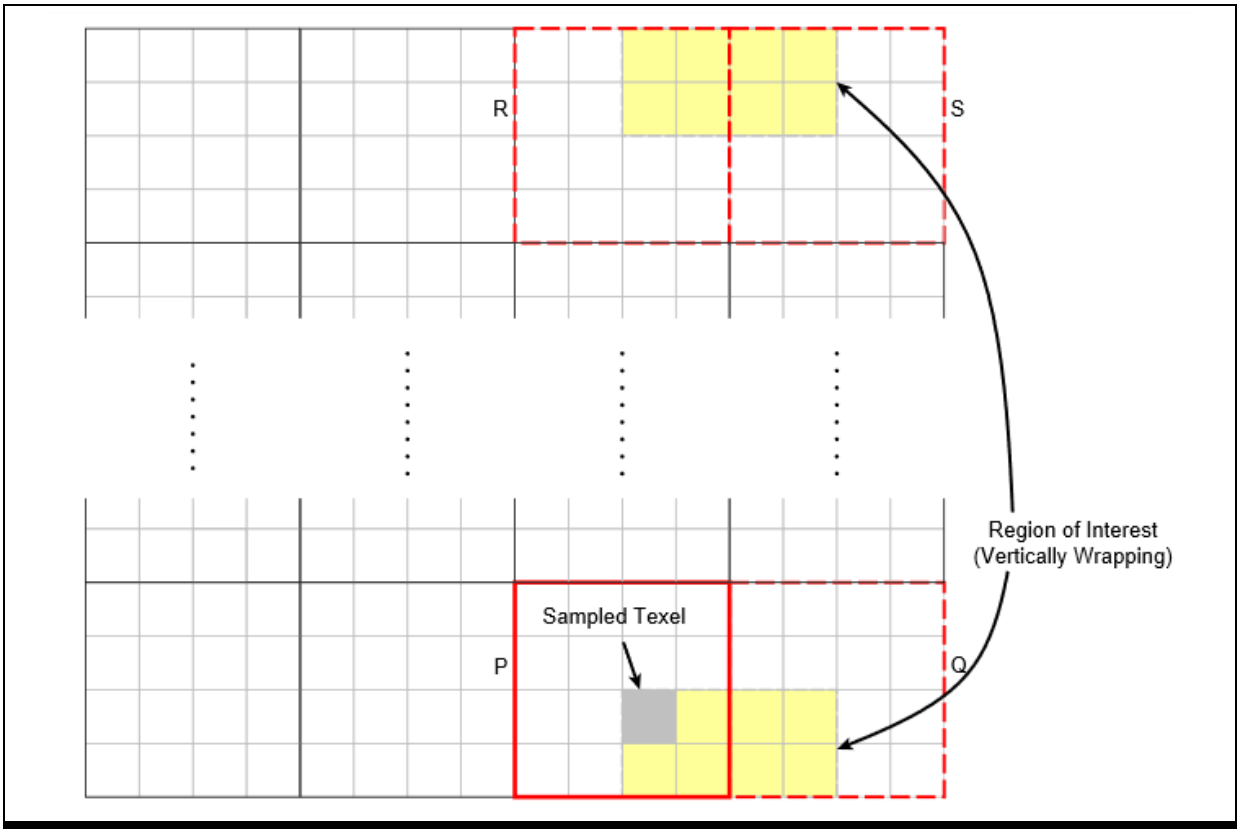

Figure 8. PVRTC Image Reconstruction, showing regions/words fetched when a texel is sampled

Each data word contains two colours, and a set of modulation data describing how to interpolate between them for each pixel location in the original image. A flag describes how the modulation data should be interpreted, and each colour contains a bit to describe whether it is contains alpha data or not.

Bits 63-32: Color data and flags																												
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Color B																Color A															Modulation Flag
Bits 31-0: Modulation Data																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
3,3		*3,2*		*3,1*		*3,0*		*2,3*		*2,2*		*2,1*		*2,0*		*1,3*		*1,2*		*1,1*		*1,0*		*0,3*		*0,2*		*0,1*		*0,0*	

Figure 9. Texel Data format for PVRTC1 4bpp compressed texture formats

The layout of colour data in PVRTC word depends on the value of the opacity flag, stored in the first bit. If the opacity flag is 1, then the colour is treated as opaque (no alpha data), and if it's 0, the colour has alpha data and may be translucent. The exact data layout of each colour is described below:

Table 1. Data layout of colour segments in a PVRTC word

Colour B – Opaque Colour Mode (Opacity Flag = 1)

Colour B – Opaque Colour Mode (Opacity Flag = 1)															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Opacity Flag	Red					Green					Blue				
Colour B - Translucent Colour Mode (Opacity Flag = 0)															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Opacity Flag	Alpha			Red				Green				Blue			
Colour A - Opaque Colour Mode (Opacity Flag = 1)															
14	13		12	11	10	9	8	7	6	5	4	3	2	1	0
Opacity Flag	Red					Green					Blue				
Colour A - Translucent Colour Mode (Opacity Flag = 0)															
14	13		12	11	10	9	8	7	6	5	4	3	2	1	0
Opacity Flag	Alpha				Red				Green				Blue		

The colour A values and colour B values for each word are bilinearly upscaled by a factor of 4 in both dimensions into two virtual images, Image A and Image B. This upscale operation is performed by treating each colour as an ARGB:4555 format, and generating results in an ARGB:8888 format. Any R, G, or B channel in each colour with fewer than 5 bits is initially expanded via bit replication:

- a 3-bit channel, C2C1C0 becomes C2C1C0C2C1,
- a 4-bit channel, C3C2C1C0 becomes C3C2C1C0C3.

The 3-bit alpha channel data is expanded to 4 bits by zero padding:

- A2A1A0 becomes A2A1A00.

The final image is created by linearly interpolating between the Image A and Image B, using the modulation data for each pixel to determine the weighting. The weight for that interpolation is derived from the 2 bits of the modulation data corresponding to the relevant pixel, depending on the modulation flag value.

Table 2. Modulation modes for PVRTC1 4bpp

Flag Value	Mode
0	Standard Bilinear
1	Punch-through

Table 3. Modulation weights for PVRTC1 4bpp

Standard Bilinear		Punch-through	
Modulation Bits	Weight	Modulation Bits	Weight
00	0	00	0
01	3/8	01	4/8
10	5/8	10	4/8 – ‘Punch-through’
11	1	11	1

If punch-through mode is selected, and the modulation bits for a given pixel have a value of 10, the alpha value of the resulting colour is 0. This is irrespective of the presence or values of any alpha channel in the input colours.

Note: For punch-through pixels, the RGB components are 50:50 blends of the corresponding pixels in the upscaled images. For this reason, with PVRTC1 4bpp, it is advised to not use pre-multiplied alpha textures, and to change the colour of fully transparent areas to the average of the local neighbourhood. PVRTexTool provides "alpha bleed" functionality to modify fully transparent areas appropriately.

3.2. Format PVRTC1 2bpp

PVRTC1 2bpp has the same broad data layout as PVRTC1 4bpp, but instead uses an 8x4 bilinear upscale. The modulation data is interpreted differently in order to accomodate the additional pixels. In PVRTC1 2bpp, rather than changing the weight values as in PVRTC1 4bpp, the modulation flag affects the modulation data layout. Optional flags in bit 0 and bit 20 of the modulation data may further affect the layout:

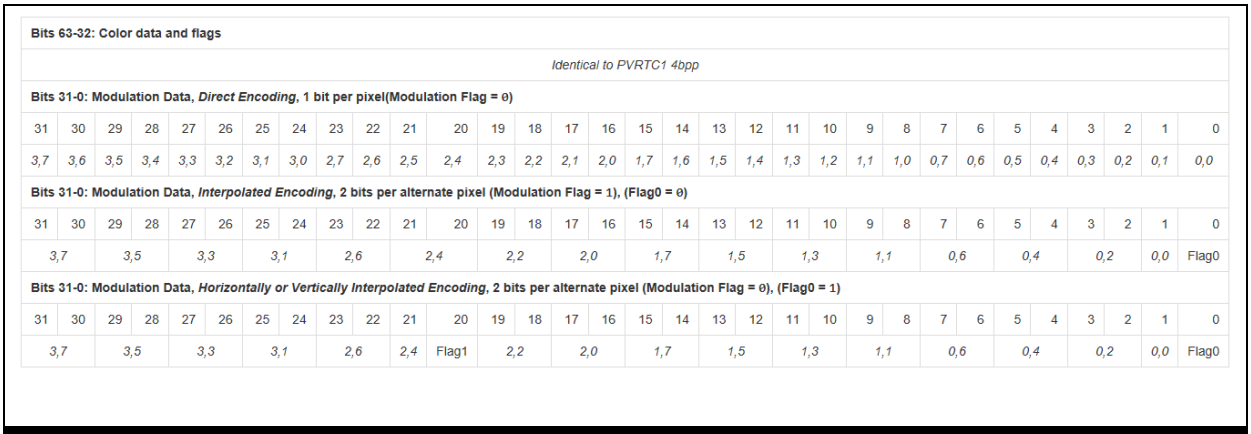

Figure 10. Texel Data format for PVRTC1 2bpp compressed texture formats

If the modulation flag is set to 0, each pixel only has a single bit of modulation. The weight value for interpolation between Image A and Image B is set to 0.0 for bit pattern 0, or 1.0 for bit pattern 1.

Table 4. Modulation modes for PVRTC1 2bpp

Flag Value	Mode
0	Standard Bilinear, 1bpp modulation
1	Punch-through, interpolated modulation

If modulation flag is set to 1, the pixels with 2-bit stored values have modulation weights equal to those of PVRTC1 4bpp for modulation mode 1.

The modulation data for the pixel at 0,0, (and when Flag0 = 1, the pixel at 2,4) is treated as if their value was duplicated - so the single bit 0 encoding becomes 00, and 1 becomes 11, such that they always either take a weight of 0 or 1.

For pixels without stored modulation data, Flag0 and Flag1 determine how they are reconstructed.

- If Flag0 is 0, the value is the mean of the weights of the four horizontally and vertically adjacent pixels, rounded to the nearest 1/8th.
- If Flag0 is 1, and Flag1 is 1, the value is the mean of the weights of the two vertically adjacent pixels, rounded to the nearest 1/8th.

- If Flag0 is 1, and Flag1 is 0, the value is the mean of the weights from the horizontally adjacent pixels, rounded to the nearest 1/8th.

If an adjacent pixel's modulation data is not present in the current word, the value is obtained from the adjacent PVRTC word which does contain that pixel's modulation data, with the location wrapping to the other side of the image if necessary.

3.3. Format PVRTC2 4bpp

PVRTC2 is the second revision of the PVRTC compression scheme, and shares most of its layout and interpretation. The PVRTC2 sections document the differences between the formats.

PVRTC2 images may have logical dimensions of any size greater than zero. If the logical size of the image is not a multiple of the interpolation size, the stored format uses the next-nearest multiple. Any padding texels added by this resize are not accessed when reconstructing the image.

In Figure 11, P, Q, R and S are the words corresponding to the region of interest for the texel. The region of interest exceeds the dimensions of the image, so wraps back to the top.

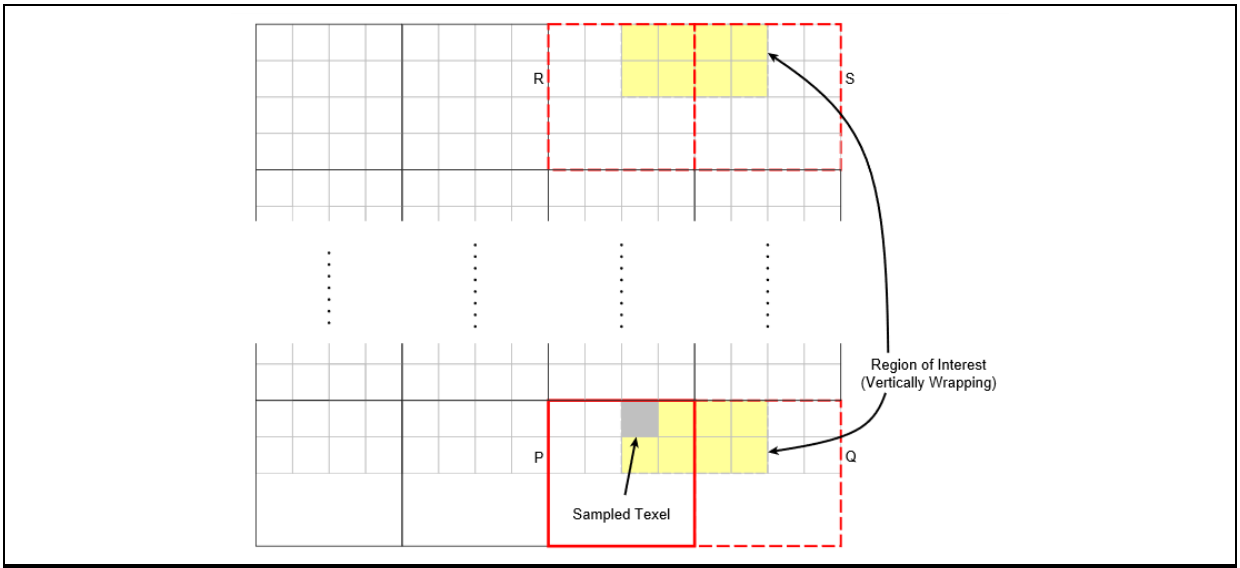

Figure 11. PVRTC2 Non-Power-of-Two Image Reconstruction, showing regions/words fetched when a texel is sampled

PVRTC2 data is not laid out in a reflected Morton order, instead it is laid out in a standard linear order, as with all other formats.

The layout of PVRTC2 data words is very similar to that of PVRTC1, though there are two main differences:

- There is only a single opacity flag that affects both colours, rather than per-colour flags.
- A *Hard Transition* flag is included to aid representing colour discontinuities, or diverse colour distributions.

Bits 63-32: Color data and flags

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Opacity Flag	Color B															Hard Transition	Color A														Modulation Flag

Bits 31-0: Modulation Data

Identical to PVRTC1 4bpp

Figure 12. Texel Data format for PVRTC1 4bpp compressed texture formats

Table 5. Data layout of colour segments in a PVRTC2 word

Colour B – Opaque Colour Mode (Opacity Flag = 1)														
14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Red					Green					Blue				
Colour B - Translucent Colour Mode (Opacity Flag = 0)														
14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Alpha			Red				Green				Blue			
Colour A - Opaque Colour Mode (Opacity Flag = 1)														
13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Red					Green					Blue				
Colour A - Translucent Colour Mode (Opacity Flag = 0)														
13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Alpha			Red				Green				Blue			

There is one change to the interpretation of the colour data relative to PVRTC1. As there is only one opacity flag for each PVRTC2 data word, to allow a local area to span the full alpha gamut when in translucent mode, Colour B's 3-bit alpha channel is expanded to four bits as A2A1A01, instead of A2A1A00.

3.3.1. Hard Transition Flag

The bilinear interpolation scheme of PVRTC1 usually works well as most areas of natural image textures are reasonably 'continuous'. However at the boundaries of non-tiling textures, around sub-textures in texture atlases, or in some areas of hand-drawn graphics, this assumption can break down. To address these issues, PVRTC2 includes the Hard Transition Flag.

Since it can be assumed that such discontinuities are more likely to be centered on boundaries of multiples of 4x4 texel regions, the hard transition flag changes the behavior of the region shown in Figure 4, which shows a set of logical 4x4 pixel regions, P, Q, R, & S that correspond to the modulation data stored in 64-bit data words, WP through WS. The flag for this hard transition region is stored in WP.

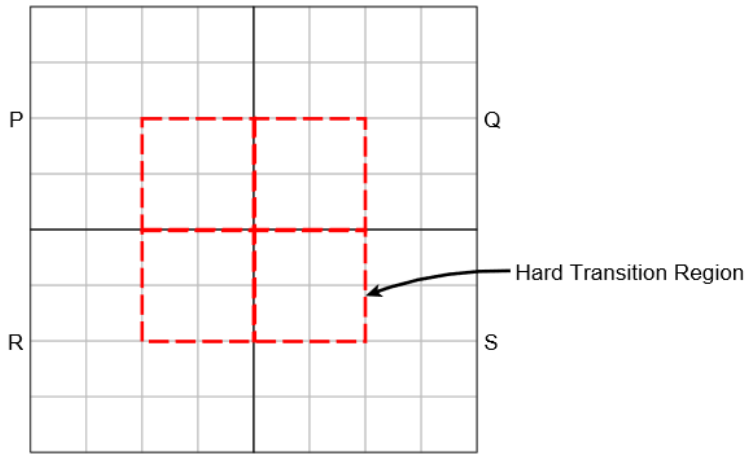

Figure 13. Hard transition region covering a subset of logical pixel regions P, Q, R, and S, with the hard transition flag set in data word WP

The hard transition region is further subdivided into four subregions where it intersects each of P through S. The hard transition flag, coupled with the relevant modulation flag for the texel subregion, determines how the colours for each reconstructed pixel in the subregion are evaluated, as summarised in the table below.

Table 6. Modulation modes for PVRTC2 4bpp

Modulation Flag Value	Hard Transition Flag Value	Mode
0	0	Standard Bilinear
1	0	Punch-through Alpha
0	1	Non-interpolated
1	1	Local Palette

In 'Standard Bilinear' the modulation behaves as described for PVRTC1 4bpp.

In 'Punch-through alpha', the modulation behaves in almost the same manner as the equivalent mode as for PVRTC1 4bpp except that for texels marked as 'punch-through', ie. using the 2-bit encoding 10, the output texel is set to transparent black, which may be better suited to pre-multiplied texture formats.

The remaining two modes are described below.

3.3.2. Non-interpolated

In the non-interpolated mode, the A and B base colours - in the affected regions - are not bilinearly interpolated. Instead, the colours for the word encapsulating each particular pixel are used directly for all affected texels when creating the upscaled images. Modulation is then applied in the same manner as for PVRTC1 4bpp with a modulation flag value of 0.

3.3.3. Local Palette Mode

In local palette mode, the hard transition region is no longer reconstructed by interpolating the upscaled images. Instead, the eight distinct colours from each surrounding word make up a local palette of colours to select from.

Denoting Colour B and Colour A from words WP through WS as described above, with a subscript corresponding to either colour, the following colours are available: Pa, Pb, Qa, Qb, Ra, Rb, Sa, and Sb.

Whilst 8 distinct colours exist in each of those four words, only two bits of modulation data are available for each pixel. Subsequently, each pixel has access to a subset of the palette as follows:

Table 7. Colour mappings in local palette mode for PVRTC2 4bpp

Pa, (5Pa+3Pb)/8, (3Pa+5Pb)/8, Pb[1]	Pa, Pb, Qa, Qb	Pa, Pb, Qa, Qb	Pa, Pb, Qa, Qb
Pa, Pb, Ra, Rb	Pa, Pb, Qa, Rb	Pa, Pb, Qa, Qb	Sa, Pb, Qa, Qb
Pa, Pb, Ra, Rb	Pa, Pb, Ra, Rb	Pa, Sb, Ra, Qb	Sa, Sb, Qa, Qb
Pa, Pb, Ra, Rb	Pa, Sb, Ra, Rb	Sa, Sb, Ra, Rb	Sa, Sb, Ra, Qb

[1] This entry is interpolated as per PVRTC's standard bilinear filtering mode. It will thus only use colours from word WP.

- The modulation values correspond to the same entry in the list for each pixel above. For instance, a mod value of 3 (bit pattern 11) for pixel location *0,1* would correspond to colour Qb being selected.

- The stored colour data is expanded to ARGB8888 from a ARGB4555 format using bit replication:
- The 4-bit alpha channel, A3A2A1A0 becomes A3A2A1A0A3A2A1A0
- Each 5-bit R, G or B channel, C4C3C2C1C0 becomes C4C3C2C1C0C4C3C2

3.4. Format PVRTC2 2bpp

PVRTC2 2bpp data layout has colour data laid out identically to PVRTC2 4bpp, and modulation data equivalent to PVRTC1 2bpp. The only difference between the PVRTC1 and PVRTC2 variants is the addition of the hard transition flag and the single opacity flag, as specified in the PVRTC2 4bpp format.

Table 8. Texel Data format for PVRTC2 2bpp compressed texture formats

Bits 63-32: Colour data and flags
Identical to PVRTC2 4bpp
Bits 31-0: Modulation Data
Identical to PVRTC1 2bpp

Colour data is interpreted in the same manner as for the PVRTC2 4bpp format.

The 2bpp variation of PVRTC2 is slightly simpler than the 4bpp, in that it only uses the non-interpolated mode - no local palette mode exists.

Table 9. Modulation modes for PVRTC2 4bpp

Modulation Flag Value	Hard Transition Flag Value	Mode
0	0	Standard Bilinear, 1bpp modulation
1	0	Standard Bilinear, interpolated modulation
0	1	Non-interpolated, 1bpp modulation
1	1	Non-interpolated, interpolated modulation

If the hard transition flag for PVRTC2 2bpp is equal to 0, the format is interpreted in the same manner as the PVRTC1 2bpp format.

When the hard transition flag for PVRTC2 2bpp is equal to 1, the initial bilinear interpolation of block colours across the hard transition region is skipped as for PVRTC2 4bpp. Subsequently, the format interpreted in the same way as the PVRTC1 2bpp format.

Note: When interpreting modulation data in the interpolated modulation mode, the hard transition flag has no effect on this - the modulation data is always interpolated across PVRTC words.

4. Contact Details

For further support, visit our forum:
http://forum.imgtec.com

Or file a ticket in our support system:
https://pvrsupport.imgtec.com

To learn more about our PowerVR Graphics Tools and SDK and Insider programme, please visit:
http://www.powervrinsider.com

For general enquiries, please visit our website:
http://imgtec.com/corporate/contactus.asp