PVR File Format

Specification (Legacy)

Filename	:	PVR File Format.Specification.Legacy
Version	:	PowerVR SDK REL_3.4@3164023a External Issue
Issue Date	:	30 Sep 2014
Author	:	Imagination Technologies Limited

Contents

List of Figures

List of Tables

1. Introduction

The purpose of this document is to act as a specification for legacy versions the PVR file format (PVR specification version 1 and 2).

2. Format Description

Legacy PVR files contain two elements:

- One Header, of 44 or 52 bytes length.
- One texture data element, whose length can be determined from the header.

The file is laid out as shown in Figure 1.

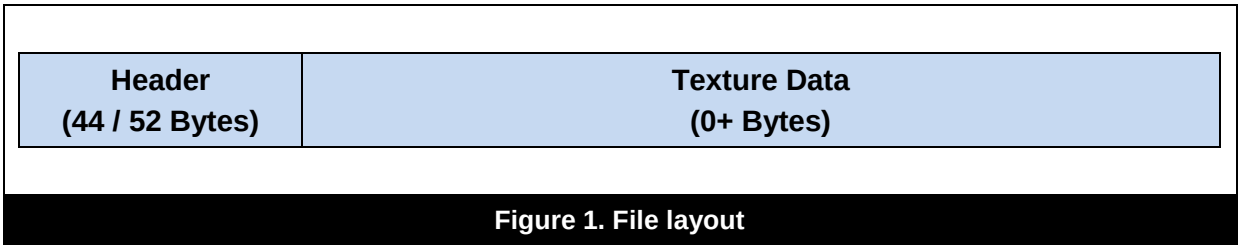

Figure 1. File layout

2.1. Header Format

Table 1 lists details of the header format.

Table 1. Header format

Name	Offset (Bytes)	Size (Bytes)	Content
Header Size	0	4	Integer Value
Height	4	4	Integer Value
Width	8	4	Integer Value
Mip Map Count	12	4	Integer Value
Pixel Format	16	1	Integer Value
Flags	17	3	Bit Field
Surface Size	20	4	Integer Value
Bits Per Pixel	24	4	Integer Value
Red Mask	28	4	Bit Field
Green Mask	32	4	Bit Field
Blue Mask	36	4	Bit Field
Alpha Mask	40	4	Bit Field
PVR Identifier	44	4	4 Characters
Number of Surfaces	48	4	Integer Value

2.1.1. Header Size

`Header Size` is a 32bit unsigned integer representing the number of bytes in the header. For PVR version 1, this should be 44, and for version 2 it should be 52.

2.1.2. Height

`Height` is a 32bit unsigned integer representing the height of the texture stored in the texture data, in pixels.

2.1.3. Width

`Width` is a 32bit unsigned integer representing the width of the texture stored in the texture data, in pixels.

2.1.4. MIP-Map Count

`MIP-Map Count` is a 32bit unsigned integer representing the number of MIP-Map levels present, excluding the top level. A value of zero, therefore, means that only the top level texture exists. If this value is anything other than 0, the MIP Map flag should be set (see Table 3. Flags).

Note: This differs from the version 3 definition, which includes the top level in the MIP Map count.

2.1.5. Pixel Format

`Pixel Format` is an 8-bit unsigned integer containing the pixel format of the texture data. The format is a specific enumerated value with a meaning corresponding to Table 2.

Table 2. Pixel format

Value	Format
0x0	ARGB 4444
0x1	ARGB 1555
0x2	RGB 565
0x3	RGB 555
0x4	RGB 888
0x5	ARGB 8888
0x6	ARGB 8332
0x7	I 8
0x8	AI 88
0x9	1BPP
0xA	(V,Y1,U,Y0)
0xB	(Y1,V,Y0,U)
0xC	PVRTC2
0xD	PVRTC4
0x10	ARGB 4444
0x11	ARGB 1555
0x12	ARGB 8888
0x13	RGB 565
0x14	RGB 555
0x15	RGB 888
0x16	I 8
0x17	AI 88
0x18	PVRTC2
0x19	PVRTC4
0x1A	BGRA 8888
0x20	DXT1
0x21	DXT2
0x22	DXT3
0x23	DXT4
0x24	DXT5
0x25	RGB 332
0x26	AL 44
0x27	LVU 655

Value	Format
0x28	XLVU 8888
0x29	QWVU 8888
0x2A	ABGR 2101010
0x2B	ARGB 2101010
0x2C	AWVU 2101010
0x2D	GR 1616
0x2E	VU 1616
0x2F	ABGR 16161616
0x30	R 16F
0x31	GR 1616F
0x32	ABGR 16161616F
0x33	R 32F
0x34	GR 3232F
0x35	ABGR 32323232F
0x36	ETC
0x40	A 8
0x41	VU 88
0x42	L16
0x43	L8
0x44	AL 88
0x45	UYVY
0x46	YUY2

2.1.6. Flags

The flags field adds a number of bit flags describing how the texture data should be treated. The flags identified in Table 3 are supported.

Table 3. Flags

Description	Value
MIP-Maps are present	0x00000100
Data is twiddled	0x00000200
Contains normal data	0x00000400
Has a border	0x00000800
Is a cube map (Every 6 surfaces make up one cube map)	0x00001000
MIP-Maps have debug colouring	0x00002000
Is a volume (3D) texture (numSurfaces is interpreted as a depth value)	0x00004000
Alpha channel data is present (PVRTC only)	0x00008000

Description	Value
Texture data is vertically flipped	0x00010000

2.1.7. Surface Size

This is a 32bit unsigned integer representing the number of bytes per single surface (see Number of Surfaces) within the texture.

2.1.8. Bits Per Pixel

This is a 32bit unsigned integer representing the total number of bits of data that make up a single pixel.

2.1.9. Channel Masks

Each of the Red, Green, Blue and Alpha channel masks are used to determine the bits occupied by each channel in a colour format. Each mask is a bitfield value where a value of 1 indicates that this bit is used by the relevant channel. E.g. a value of '0xff00000' in `Red Mask` would indicate that the last 8 bits in each pixel are used as the red value.

This only applies to pixel formats that are less (or equal to) 32-bits. In practice, these masks are usually ignored, but if an unknown Pixel Format is encountered, these values can be used to describe the channel layout.

2.1.10. PVR File Identifier

The file identifier is a late addition to the file format, and is only present in version 2 files. It is a series of 4 characters spelling out 'PVR!', or the hexadecimal value '0x21525650' if read on a little-endian processor.

2.1.11. Number of Surfaces

The number of surfaces is a 32bit unsigned integer, signifying the number of distinct surfaces in the texture data. It is only present in version 2 files. A distinct surface is either a cube map face, a slice of a volume texture, or a member of a texture array, depending on the flags present:

- If the cube map flag is present, there are (Num. Surfaces / 6) cube maps within the texture.
- If the volume texture flag is present, then each surface should be treated as a z-slice within a 3D texture.
- If neither flag is present, each surface is a member of a 2D texture array.
- If both flags are present, the texture is invalid, and it is up to the reader to decide what to do.

Note: A bug existed in some early writers which caused this value to be overwritten with a null terminating character from the file identifier – forcing the value to 0. If this value is zero, the number of surfaces can be worked out programmatically from the size of the file, surface size, and the cube map flag.

3. Texture Data

The remainder of the file, after the header and metadata, is texture data. The format and size of this texture data can be found in the header (see Section 2.1).

3.1. Uncompressed Texture Data Structure

The uncompressed texture data is laid out as follows:

```
for each Surface in Num. Surfaces
    for each Face in 6
        for each MIP-Map Level in MIP-Map Count
            for each Row in Height
                for each Pixel in Width
                    Byte data[Size_Based_On_PixelFormat]
                end
            end
        end
    end
end
```

3.2. Compressed Texture Data Structure

All compressed data formats have a "minimum width/height" which is the lowest number of pixels that can be represented by any given region in a compressed image, according to the data format.

The compressed texture data is laid out as follows:

```
for each Surface in Num. Surfaces
    for each Face in 6
        for each MIP-Map Level in MIP-Map Count
            for each Region by aligned Height (Based On PixelFormat)
                for each Region by aligned Width (Based On PixelFormat)
                    Byte data[Size Based On PixelFormat]
                end
            end
        end
    end
end
```

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 SDK and Insider programme, please visit:
http://www.powervrinsider.com

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