Video Formats

This article contains information regarding the organisation of data in video formats supported by Decklink.

'UYVY' 4:2:2 Compression

Four 8-bit unsigned components (CCIR 601) are packed into one 32-bit **little-endian** word.

Word			
Decreasing Address Order			
Byte 3	**Byte 2**	**Byte 1**	**Byte 0**
Y' 1	Cr 0	Y' 0	Cb 0
7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0

framesize = *(Width * 16 / 8) * Height* NOTE: This is an INTEGER calculation.
= *rowbytes * Height*

E.g. for 720 x 486:

rowbytes = *(Width * 16 / 8)*
= *(720 * 16 / 8)*
= *1440*
framesize = *rowbytes * Height*
= *1440 * 486*
= *699840*

In this format two pixels pixel fits into 32 bits or 4 bytes, so one pixel fits into 16 bits or 2 bytes. For the row bytes calculation the image width is multiplied by the number of bytes per pixel. For the frame size calculation the rowbytes are simply multiplied by the number of rows in the frame.

'YUY2' 4:2:2 Compression

Four 8-bit unsigned components (CCIR 601) are packed into one 32-bit **little-endian** word.

Word			
Decreasing Address Order			
Byte 3	**Byte 2**	**Byte 1**	**Byte 0**
Cr 0	Y' 1	Cb 0	Y' 0
7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0

framesize = *(Width * 16 / 8) * Height* NOTE: This is an INTEGER calculation.
= *rowbytes * Height*

E.g. for 720 x 576:

rowbytes = *(Width * 16 / 8)*
= *(720 * 16 / 8)*
= *1440*

framesize = *rowbytes * Height*
= *1440 * 576*
= *829440*

In this format two pixels pixel fits into 32 bits or 4 bytes, so one pixel fits into 16 bits or 2 bytes. For the row bytes calculation the image width is multiplied by the number of bytes per pixel. For the frame size calculation the rowbytes are simply multiplied by the number of rows in the frame.

'HDYC' 4:2:2 Compression

Four 8-bit unsigned components (CCIR 709) are packed into one 32-bit **little-endian** word.

Word			
Decreasing Address Order			
Byte 3	**Byte 2**	**Byte 1**	**Byte 0**
Y' 1	Cr 0	Y' 0	Cb 0
7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0

framesize = *(Width * 16 / 8) * Height* NOTE: This is an INTEGER calculation.
= *rowbytes * Height*

E.g. for 720 x 576:

rowbytes = *(Width * 16 / 8)*
= *(720 * 16 / 8)*
= *1440*

framesize = *rowbytes * Height*
= *1440 * 576*
= *829440*

In this format two pixels pixel fits into 32 bits or 4 bytes, so one pixel fits into 16 bits or 2 bytes. For the row bytes calculation the image width is multiplied by the number of bytes per pixel. For the frame size calculation the rowbytes are simply multiplied by the number of rows in the frame.

BI_RGB (or RGB24) 4:4:4 No compression

Three 8-bit unsigned components are packed into three 8-bit **little-endian** words.

Word			
Decreasing Address Order			
Byte 2	**Byte 1**	**Byte 0**	
R	G	B	
7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	

framesize = *(Width * 24 / 8) * Height* NOTE: This is an INTEGER calculation.
= *rowbytes * Height*

E.g. for 1920 x 1080:

rowbytes = *(Width * 24 / 8)*
= *(1920 * 24 / 8)*

$= 5760$

$framesize = rowbytes * Height$

$= 5760 * 1080$

$= 6220800$

In this format each pixel fits into 24 bits or 3 bytes. For the row bytes calculation the image width is multiplied by the number of bytes per pixel. For the frame size calculation the rowbytes are simply multiplied by the number of rows in the frame.

BI_RGB (or RGB32) 4:4:4:x No compression

Four 8-bit unsigned components are packed into one 32-bit **little-endian** word. Alpha channel *may* be valid.

Word			
Decreasing Address Order			
Byte 3	Byte 2	Byte 1	Byte 0
X	R	G	B
7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0

$framesize = (Width * 32 / 8) * Height$ NOTE: This is an INTEGER calculation.

$= rowbytes * Height$

E.g. for 1920 x 1080:

$rowbytes = (Width * 32 / 8)$

$= (1920 * 32 / 8)$

$= 7680$

$framesize = rowbytes * Height$

$= 7680 * 1080$

$= 8294400$

In this format each pixel fits into 32 bits or 4 bytes. For the row bytes calculation the image width is multiplied by the number of bytes per pixel. For the frame size calculation the rowbytes are simply multiplied by the number of rows in the frame.

BI_RGB (or ARGB32) 4:4:4:4 No compression

Four 8-bit unsigned components are packed into one 32-bit **little-endian** word. Alpha channel **is** valid.

Word			
Decreasing Address Order			
Byte 3	Byte 2	Byte 1	Byte 0
A	R	G	B
7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0

$framesize = (Width * 32 / 8) * Height$ NOTE: This is an INTEGER calculation.

$= rowbytes * Height$

E.g. for 1920 x 1080:

rowbytes = *(Width * 32 / 8)*
= *(1920 * 32 / 8)*
= *7680*
framesize = *rowbytes * Height*
= *7680 * 1080*
= *8294400*

In this format each pixel fits into 32 bits or 4 bytes. For the row bytes calculation the image width is multiplied by the number of bytes per pixel. For the frame size calculation the rowbytes are simply multiplied by the number of rows in the frame.

'v210' 4:2:2 Compression

Twelve 10-bit unsigned components are packed into four 32-bit **little-endian** words.

Word 0																															
Decreasing Address Order																															
Byte 3								**Byte 2**								**Byte 1**								**Byte 0**							
X	X	Cr 0										Y' 0										Cb 0									
		9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0

Word 1																															
Decreasing Address Order																															
Byte 3								**Byte 2**								**Byte 1**								**Byte 0**							
X	X	Y' 2										Cb 2										Y' 1									
		9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0

Word 2																															
Decreasing Address Order																															
Byte 3								**Byte 2**								**Byte 1**								**Byte 0**							
X	X	Cb 4										Y' 3										Cr 2									
		9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0

Word 3																															
Decreasing Address Order																															
Byte 3								**Byte 2**								**Byte 1**								**Byte 0**							
X	X	Y' 5										Cr 4										Y' 4									
		9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0

framesize = *((Width + 47) / 48) * 128 * Height* NOTE: This is an INTEGER calculation.
= *rowbytes * Height*

E.g. for 1920 x 1080:

rowbytes = *((Width + 47) / 48) * 128*
= *((1920 + 47) / 48) * 128*
= *40 * 128*

$= 5120$

$framesize = rowbytes * Height$

$= 5120 * 1080$

$= 5529600$

In this format each line of video must be aligned on 128 byte boundaries. Six pixels fit into 16 bytes so 48 pixels fit in 128 bytes. For the row bytes calculation the image width is rounded to the nearest 48 pixel boundary and multiplied by 128. For the frame size calculation the rowbytes are simply multiplied by the number of rows in the frame.

'r210' 4:4:4 No compression

Three 10-bit unsigned components are packed into one 32-bit **big-endian** word.

Word																													
Decreasing Address Order																													
Byte 3								Byte 2								Byte 1								Byte 0							
B_{lo}								G_{lo}						B_{hi}		R_{lo}						G_{hi}		X	X	R_{hi}					
7	6	5	4	3	2	1	0	5	4	3	2	1	0	9	8	3	2	1	0	9	8	7	6	x	x	9	8	7	6	5	4

$framesize = ((Width + 63) / 64) * 256 * Height$ NOTE: This is an INTEGER calculation.

$= rowbytes * Height$

E.g. for 1920 x 1080:

$rowbytes = ((Width + 63) / 64) * 256$

$= ((1920 + 63) / 64) * 256$

$= 30 * 256$

$= 7680$

$framesize = rowbytes * Height$

$= 7680 * 1080$

$= 8294400$

In this format each line of video must be aligned on 256 byte boundaries. One pixel fits into 4 bytes so 64 pixels fit in 256 bytes. For the row bytes calculation the image width is rounded to the nearest 64 pixel boundary and multiplied by 256. For the frame size calculation the rowbytes are simply multiplied by the number of rows in the frame.