

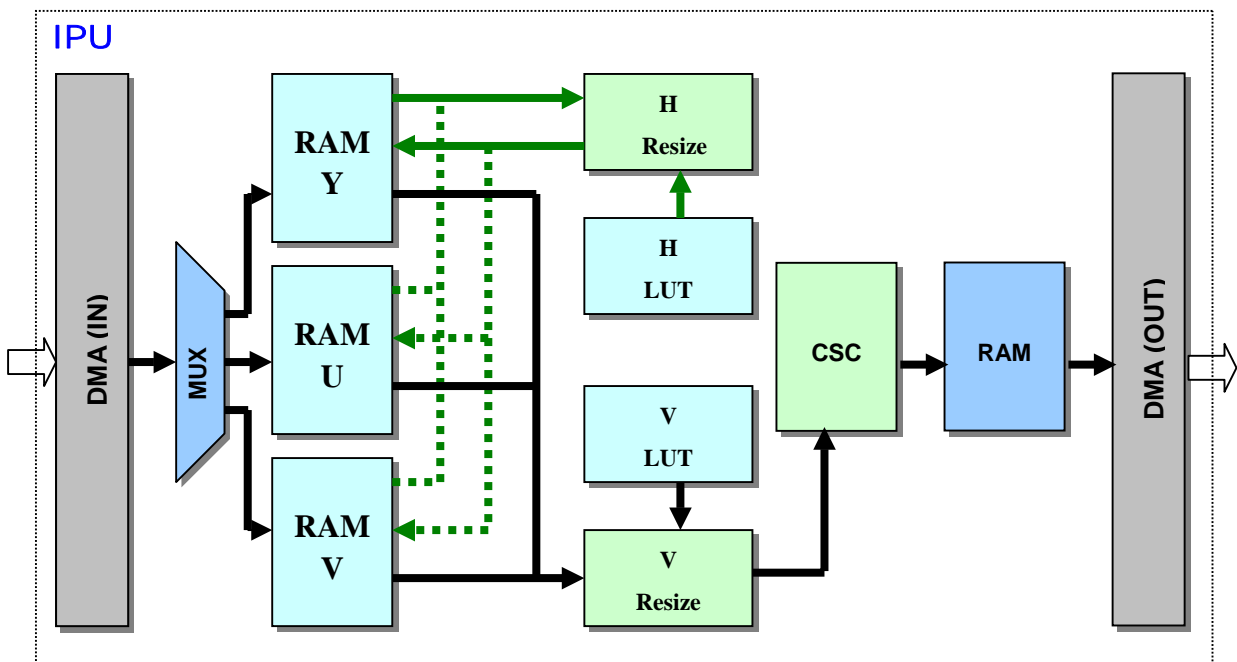
1 Overview

IPU (Image process unit) contains Resize and CSC (color space conversion), which is used for image post processing.

Features

- Input data: from external memory
- Input format: YUV /YCbCr (4:2:0, 4:2:2, 4:4:4, 4:1:1)
- Output format: RGB (565, 555, 888)
- Minimum input image size: 33x33
- Maximum input image size: 2047x2047
- Image resizing:
 - Up scaling ratios up to 1:2 in fractional steps
 - Down scaling ratios up to 20:1 in fractional steps

Block Diagram



2 Data flow

Input Data

Y, U, V (or Y, Cb, Cr; the following use YUV for convenience) data would be fetched from external memory by DMA burst read operation respectively.

Output Data

The data format after CSC could be RGB (565, 555, 888), and the data would be stored to the external memory by DMA burst write operation.

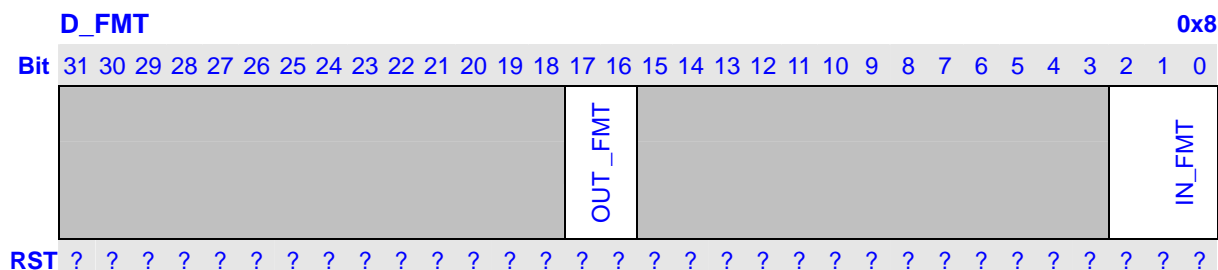
Resize Coefficients LUT

The resize coefficients look up table is preset by software according to specific format (see later chapter for detail). There are 2 tables support independent horizontal and vertical scaling. Each table has 20 entries that can accommodate up to 20 coefficients.

NOTES:

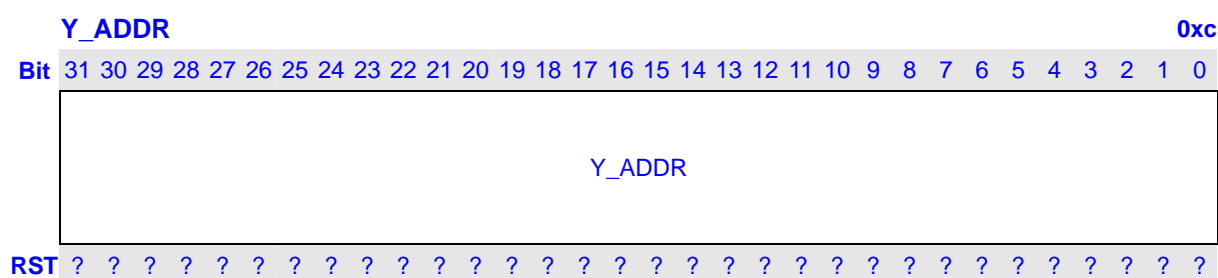
If IPU_CONTROL.FM_IRQ_EN has been set 1, once OUT_END is set value 1 which denotes a frame's post process done, an low level active interrupt request will be issued until corresponding software handler clear OUT_END to value 0.

Data Format Register



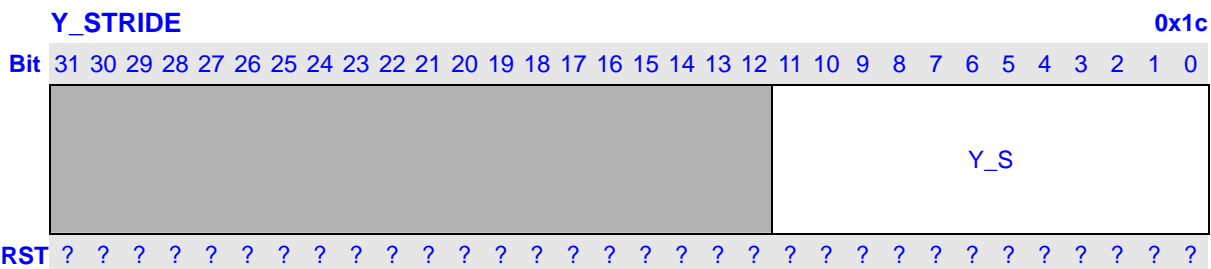
Bits	Name	Description	R/W
31:18	Reserved	Writing has no effect, read as zero.	R
17:16	OUT_FMT	Indicates the destination data format: 00: RGB555 01: RGB565 10: RGB888 11: reserved	RW
15:3	Reserved	Writing has no effect, read as zero.	R
2:0	IN_FMT	Indicates the source data format: 000: YUV 4:2:0 001: YUV 4:2:2 010: YUV 4:4:4 011: YUV 4:1:1 100: YCbCr 4:2:0 101: YCbCr 4:2:2 110: YCbCr 4:4:4 111: YCbCr 4:1:1	RW

Input Y Data Address Register



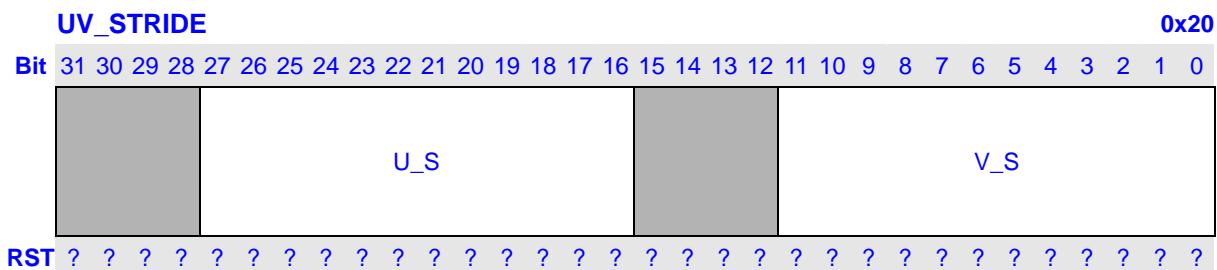
Bits	Name	Description	R/W
31:0	Y_ADDR	The source Y data buffer's start address	RW

Input Y Data Line Stride Register



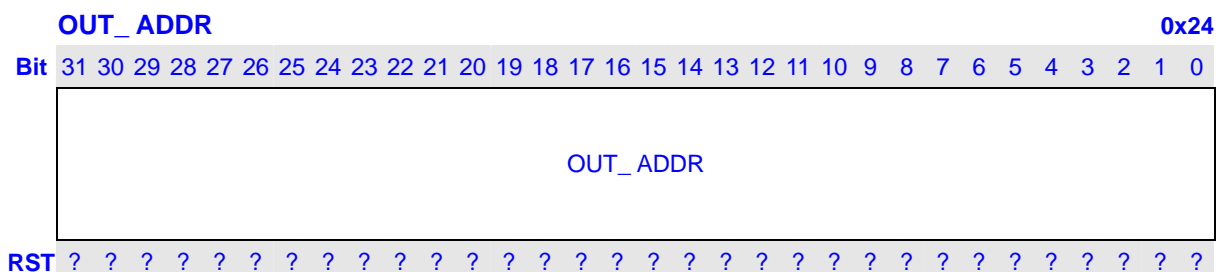
Bits	Name	Description	R/W
31:12	Reserved	Writing has no effect, read as zero.	R
11:0	Y_S	The line stride of the source Y data in the external memory. (unit: byte)	RW

Input UV Data Line Stride Register



Bits	Name	Description	R/W
31:28	Reserved	Writing has no effect, read as zero.	R
27:16	U_S	The line stride of the source U data in the external memory. (unit: byte)	RW
15:12	Reserved	Writing has no effect, read as zero.	R
11:0	V_S	The line stride of the source V data in the external memory. (unit: byte)	RW

Output Frame Start Address Register



Bits	Name	Description	R/W
31:0	OUT_ADDR	The output buffer's start address.	RW

Resized width and height calculation

For software, to preset correct value for register OUT_GS, please refer to following formula.

Set IW stand for original input frame width, IH stand for original input frame height, OW stand for new frame width after resize, OH stand for new frame height after resize.

In Up-scale case ($n < m$):

If $[(IW - 1) * (m/n)] * (n/m) == (IW-1)$ then

$$OW = [(IW - 1) * (m/n)] + 1;$$

Else $OW = [(IW - 1) * (m/n)] + 2;$

If $[(IH - 1) * (m/n)] * (n/m) == (IH-1)$ then

$$OH = [(IH - 1) * (m/n)] + 1;$$

Else $OH = [(IH - 1) * (m/n)] + 2;$

In Down-scale case ($n > m$):

If $[(IW - 1) * (m/n)] * (n/m) == (IW-1)$ then

$$OW = [(IW - 1) * (m/n)];$$

Else $OW = [(IW - 1) * (m/n)] + 1;$

If $[(IH - 1) * (m/n)] * (n/m) == (IH-1)$ then

$$OH = [(IH - 1) * (m/n)];$$

Else $OH = [(IH - 1) * (m/n)] + 1;$

For example:

A 36x46 frame with the horizontal resize ratio of 4:5 (up-scale) and vertical resize ratio of 7:6 (down-scale), by the expressions above we get its new size after resize from the following process.

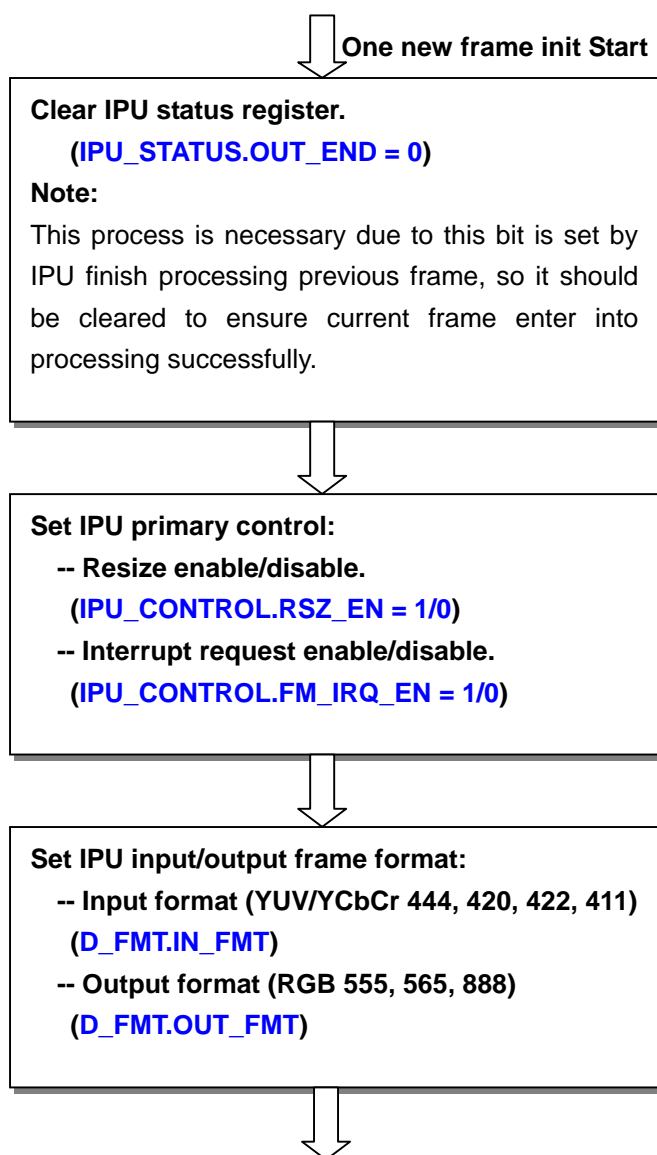
For Width: $[(36 - 1) * (5/4)] * (4/5) = 34.4 \neq (36-1)$

$$\text{So } OW = [(36 - 1) * (5/4)] + 2 = 45$$

For Height: $[(46 - 1) * (6/7)] * (7/6) = 44.33 \neq (46 - 1)$

$$\text{So } OH = [(46 - 1) * (6/7)] + 1 = 39$$

IPU Initialization Flow



↓

Set input frame size:

- Input frame width (Eg: 288x188 frame)
(**IN_FM_GS.IN_FM_W = 288**)
- Input frame height (Eg: 288x188 frame)
(**IN_FM_GS.IN_FM_H = 188**)
- Y frame stride (**Y_STRIDE.Y_S**)
- U frame stride (**UV_STRIDE.U_S**)
- V frame stride (**UV_STRIDE.V_S**)

Note: Frame width/height value should be restricted according to frame format (ensure it is a legal size). In 411 case the value should be multiple of 4. In 420/422 case the value should be multiple of 2. 444 case, the value can be any integer in the legal range (33 ~ 2047). Moreover, all the three stride values should be set to ensure frames' every line start address are word aligned, reference to attached Figure A-1.

↓

Set input/output data start address:

- Y frame start address (**Y_ADDR.Y_ADDR**)
- U frame start address (**U_ADDR.U_ADDR**)
- V frame start address (**V_ADDR.V_ADDR**)
- Output frame start address
(**OUT_ADDR.OUT_ADDR**)

Note: Y/U/V frame start address must be word aligned. Output frame start address can be half-word or word aligned except RGB888 format, which should be aligned by word.

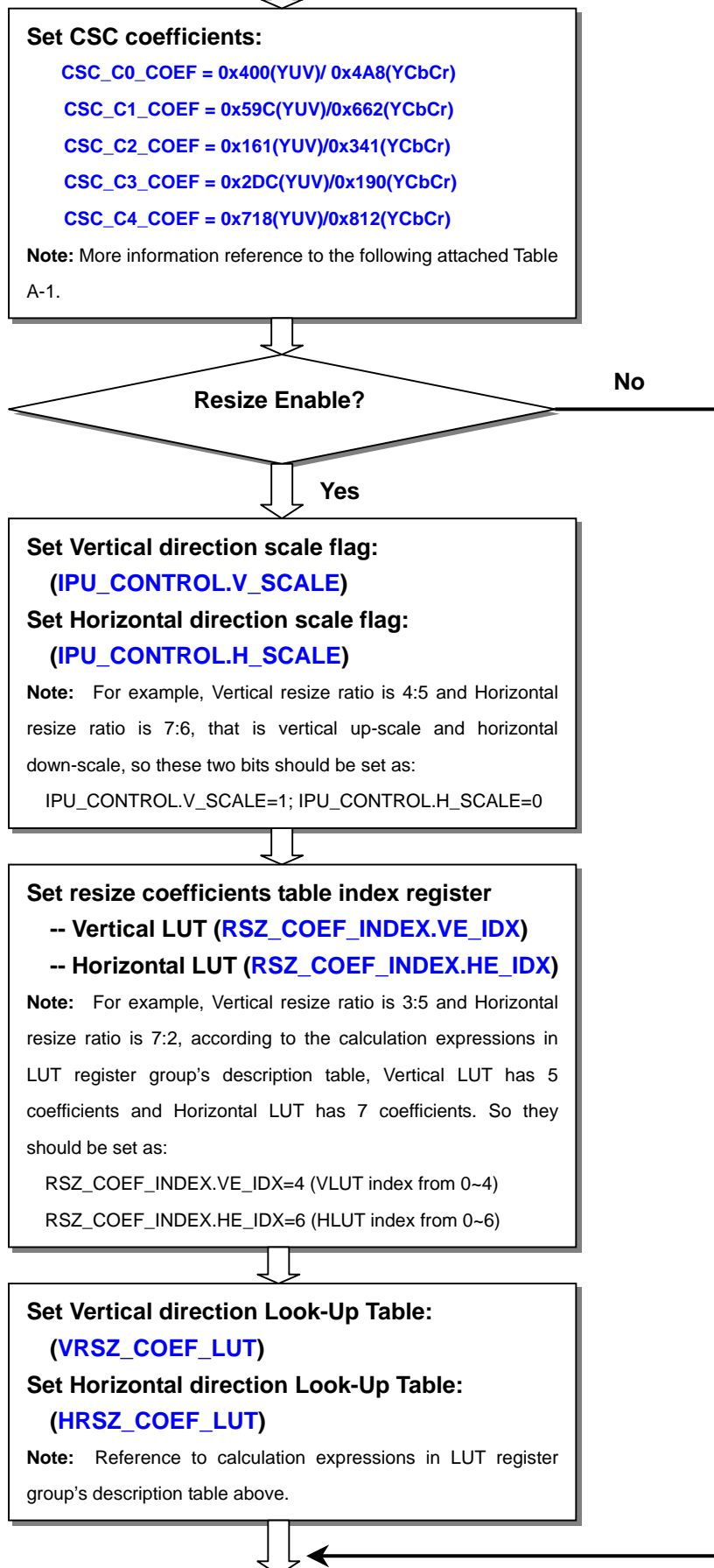
↓

Set output frame size:

- Output frame width
(**OUT_GS.OUT_FM_W**)
- Output frame height
(**OUT_GS.OUT_FM_H**)
- Output frame stride
(**OUT_STRIDE.OUT_S**)

Note: All values above are united by byte, so the values filled in are the pixel numbers left shifted by 1 or 2 according to output format. For example: RGB888, each pixel takes 4 bytes, so the width value is pixel numbers * 4. Such as a RGB888 frame with size of 120x80 and stride of 124, their value should be filled as: **OUT_GS.OUT_FM_W=120<<2;**
OUT_GS.OUT_FM_H=80<<2;OUT_STRIDE.OUT_S=124<<2

↓



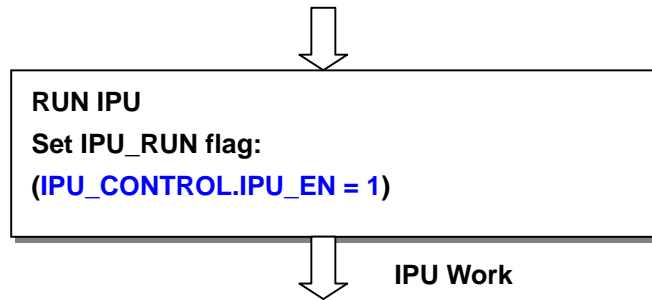


Table A-1. YUV/YCbCr to RGB CSC Equations

Input data	Matrix	CSC_COEF register values
YUV	$R = C0*(Y - X0) + C1*(V-128)$ $G = C0*(Y - X0) - C2*(U-128) - C3*(V-128)$ $B = C0*(Y - X0) + C4*(U-128)$ X0: 0 C0: 1 C1: 1.4026 C2: 0.3444 C3: 0.7144 C4: 1.7730	CSC_C0_COEF = 0x400 CSC_C1_COEF = 0x59C CSC_C2_COEF = 0x161 CSC_C3_COEF = 0x2DC CSC_C4_COEF = 0x718
YCbCr	$R = C0*(Y - X0) + C1*(Cr-128)$ $G = C0*(Y - X0) - C2*(Cb-128) - C3*(Cr-128)$ $B = C0*(Y - X0) + C4*(Cb-128)$ X0: 16 C0: 1.164 C1: 1.596 C2: 0.813 C3: 0.391 C4: 2.018	CSC_C0_COEF = 0x4A8 CSC_C1_COEF = 0x662 CSC_C2_COEF = 0x341 CSC_C3_COEF = 0x190 CSC_C4_COEF = 0x812

Table A-2. Output data package format

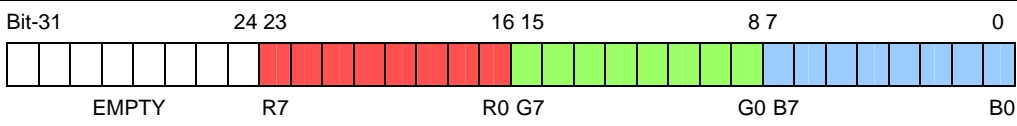
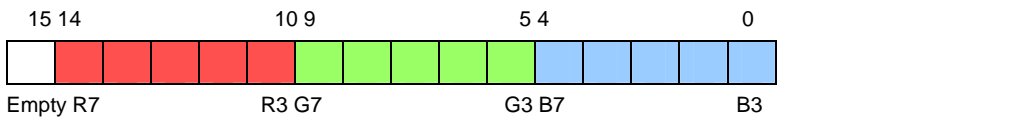
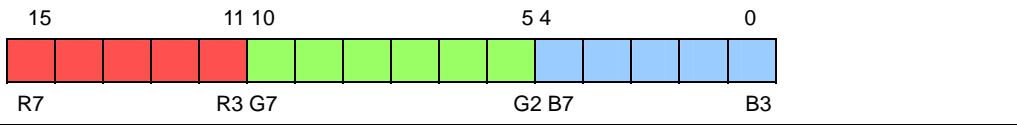
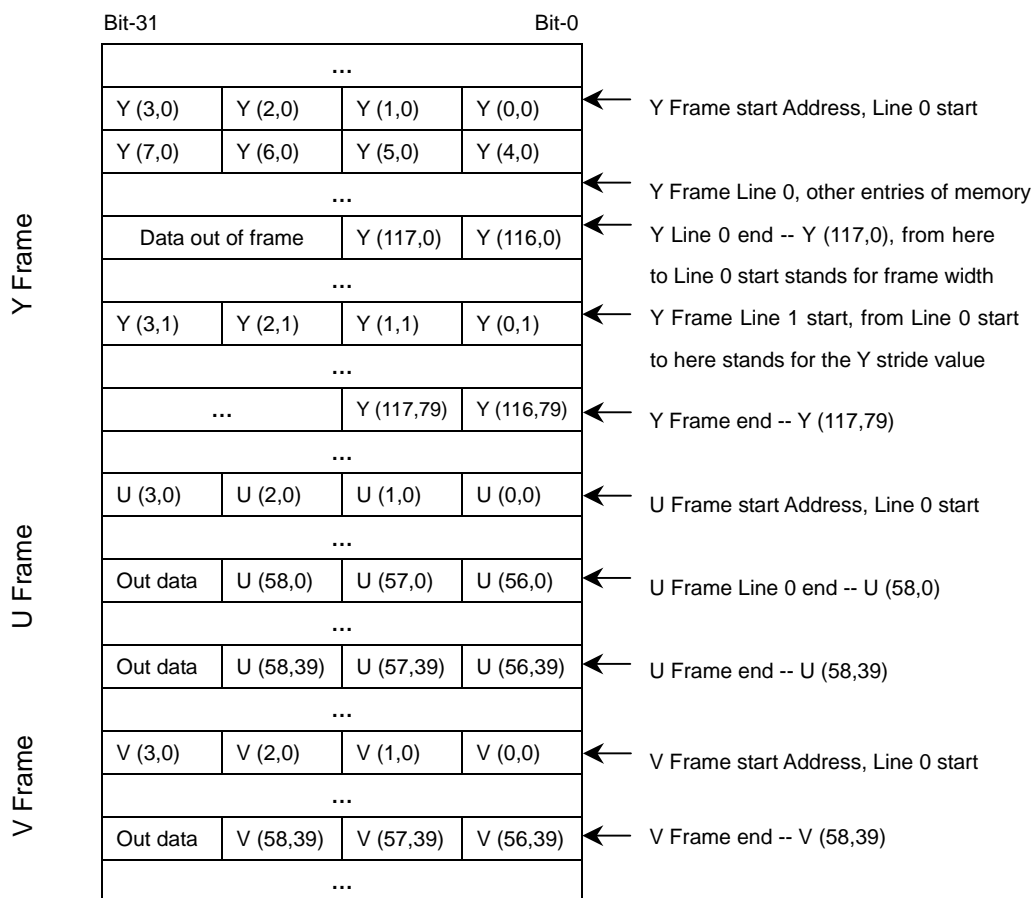
Format	Package
RGB888	 <p>Bit-31 24 23 16 15 8 7 0</p> <p>EMPTY R7 R0 G7 G0 B7 B0</p>
RGB555	 <p>15 14 10 9 5 4 0</p> <p>Empty R7 R3 G7 G3 B7 B3</p>
RGB565	 <p>15 11 10 5 4 0</p> <p>R7 R3 G7 G2 B7 B3</p>
Note:	All R/G/B data are little-endian type; all the empty bits in the above figure are filled with 0.

Figure A-1 Source Data storing format in external memory.

Example: YUV420 118x80 frame



- Note:
1. Every line's start address should be word aligned.
 2. All pixel data should be stored as little-endian type.
 3. Destination data (RGB) storing format in external memory is similar with above figure, but RGB555 and RGB565 frame's every line start address can be half-word aligned (RGB888 frame still need word aligned).