embedded Linux kernel development report 4

# introduction

This LAB’s purpose is to learn how the Linux framebuffer works and use it through a userspace YUV video decoder application.

As the framebuffer uses RGBa format, we need to convert each YUV pixels to a RGB equivalent.

The format is YUV 4:2:0, meaning that for a 4x4 pixels square we have 4 Y, 2 U and 2 V data info.

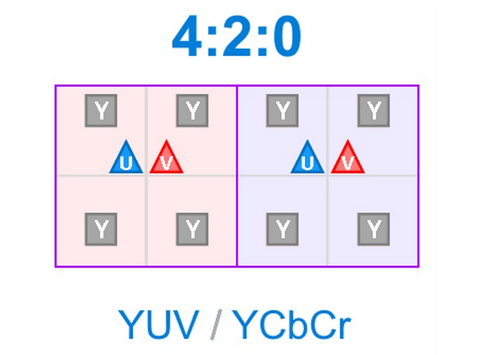


Figure - YUV 4:2:0 format (http://www.latelierducable.com)

The data in each frame is organized as follow.

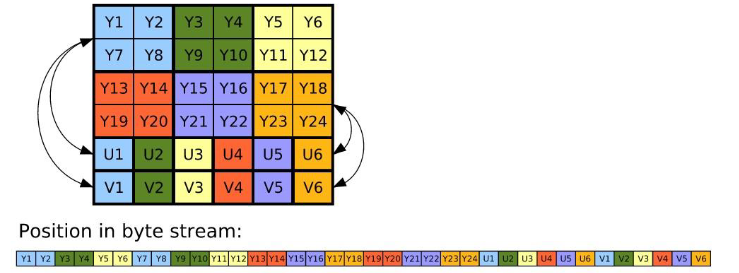


Figure - YUV data layout for one frame

# Floating point conversion

We first load the template code provided in LABS/TP\_FRAMEBUFFER. We can test with the blueRectangle function that the writing is done correctly.

As the PC’s screen is constantly refreshed by the system, we need to switch to console mode (**CTRL+ALT+F1**), then go back to GUI mode (**CTRL+ALT+F7**).

Une image contenant moniteur, équipement électronique, intérieur, télévision

Description générée automatiquement

Figure - blue screen test code

This code uses the *mmap* method to…

We can replace the blueRectangle method by the *YUVtoRGB* method and use the following formula to do the conversion.

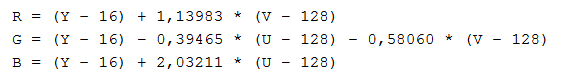


Figure - YUV to RGB formula

Doing the conversion, we need to mind the inc\_Y and inc\_UV values so that they match the correct pixel in buffer.

For inc\_Y we just need to increment by one, but for inc\_UV we can refer to Figure1. We need to increment inc\_UV by 1 for 2 pixels crossed, and reset the counter at the start of the line for each one of two new lines.

[img code]

# fixed point

As the floating point calculus can be resource heavy, we switch to fixed point values.

By doing a 14bits shift on the previous formula, we can avoid *float*.

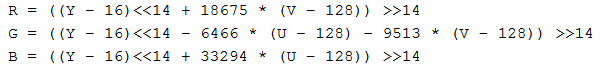


Figure - new fixed point formula

The code is updated as follow to easily compare performance between each mode.

[code]

And we can see the fixed point method is 2 times quicker than the floating point method.

[result]

The resulting video is still correctly displayed in the console mode:

Une image contenant fenêtre, moniteur, vue, télévision

Description générée automatiquement

Figure - YUV video extract

# conclusion

Interacting with Linux framebuffer is fairly easy even when we’re in the userspace. But as it’s an existing Driver, some processes are already using it when in GUI mode; To see our application working we first need to stop the GUI from refreshing the screen.

When developing applications using the framebuffer, we need to mind the resources used as it can heavily impact the application performance.