GM8136

LCD CONTROLLER

User Guide
Rev.: 1.0
Issue Date: September 2014

REVISION HISTORY

GM8136 LCD User Guide

Date	Rev.	From	To
Sept. 2014	1.0	-	Original

Printed in Taiwan 2014

Grain Media, Inc.
5F, No. 5, Li-Hsin Road III, Hsinchu Science Park, Hsinchu City, Taiwan 300, R.O.C.

Grain Media's home page can be found at:
http://www.grain-media.com

TABLE OF CONTENTS

LIST OF TABLES

Chapter 1

Introduction

LCD210 is a LCD controller that is compliant with the Advanced Microcontroller Bus Architecture (AMBA) protocol, Version 2.0. LCD210 supports two PIPs (Total three planes) and each PIP supports four quarters. LCD210 can output the RGB888 signals or the CCIR656/BT1120 signals, but it cannot simultaneously output both signal groups. For the detailed functions of LCD210, please refer to the related data sheet. Because Linux provides the frame buffer management in the kernel, AP can easily use these interfaces to access the information of LCD210 through the Linux frame buffer driver. For the Linux generic frame buffer interfaces provided by the Linux middleware, it is not addressed in this document.

The LCD210 module provides many useful functions, such as PIP, mouse, and so on. They can be controlled by using the proc notes. This document will describe how to use these proc nodes and how to install the LCD module.

Chapter 2

Installation of Frame Buffer Driver

This chapter contains the following sections:

The LCD210 module is a subsystem of the Linux frame buffer driver. That is, the LCD210 module works along with the Linux frame buffer driver. To activate the frame buffer function, some options in the kernel configuration menu must be selected. Please refer to the pictures in the subsections for details. Please set up the frame buffer driver by using the command, "make menuconfig".

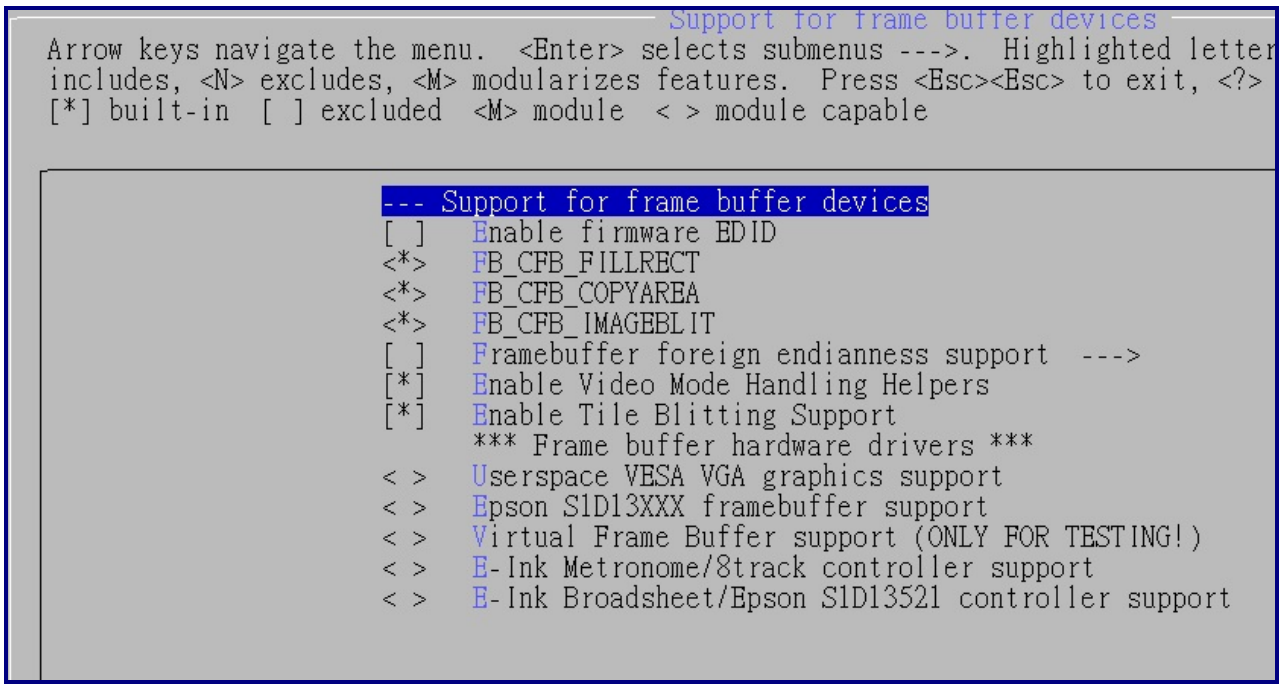
2.1 Configuration

This driver should be enabled in menuconfig.

→Device Drivers

→ Graphics support

→ Support for frame buffer devices



2.2 Device Nodes

After installing the LCD210 module, the device nodes will be created. If there are three planes, three device nodes will be created:

- /dev/fb0 is associated with the frame buffer driver 0, and the plane plays the pictures.
- /dev/fb1 is associated with the frame buffer driver 1, and it indicates the first layer PIP or GUI.
- /dev/fb2 is associated with the frame buffer driver 2, and it indicates the second layer PIP or GUI. If this PIP is enabled, it will usually be disabled for saving the bandwidth.

If there are two LCD controllers and each controller is configured as two planes; then, a total of four device nodes will be created. They include:

- /dev/fb0 is associated with the frame buffer driver 0 of first LCD.
- /dev/fb1 is associated with the frame buffer driver 1 of first LCD.
- /dev/fb2 is associated with the frame buffer driver 0 of second LCD or it can be frame buffer driver 2 of first LCD. It depends on how many planes are configured in the LCD driver. If the LCD driver supports three planes, the next frame buffer driver 0 of second LCD will start from fb3.
- /dev/fb3 is associated with the frame buffer driver 1 of second LCD.

Device Name	Major	Minor
/dev/fb0	29	0
/dev/fb1	29	1
/dev/fb2	29	2
/dev/fb3	29	3
/dev/fbosd	10	X
/dev/fbosd2	10	X

2.3 Driver Compilation Option

A compilation option file is provided to select the included functions when compiling the LCD driver. This file is platform-dependent. Taking GM8136 as an example, the configuration file will be config_8136.

The options are listed below:

Name	Description	Valid Value
DEBUG_MESSAGE	Embedded debug message in a driver	Y/N
SIMPLE_OSD	Build the feature of simple OSD in a driver	Y/N
PIP_OUTPUT_TYPE	Set the default output for the PIP device	Please refer to the later chapter for the description of output_type.
PIP_FB0_NUM PIP_FB1_NUM PIP_FB2_NUM	Set the default number of frame buffers for the PIP device Users can set the number to '0' to disable the frame buffer. User must disable the frame buffer from FB2.	0 ~ 1
PIP_FB0_DEF_MODE PIP_FB1_DEF_MODE PIP_FB2_DEF_MODE	Default color format of a frame buffer for each frame buffer driver. If ARGB or RGB888 is selected (Each pixel is four bytes), it will consume more memory. To save the memory and performance, RGB565 or YUV422 is preferred.	0: YUV422 16: ARGB 17: RGB888 18: RGB565
PIP_FB0_SUPPORT_YUV422 PIP_FB0_SUPPORT_ARGB PIP_FB0_SUPPORT_RGB888 PIP_FB0_SUPPORT_RGB565 PIP_FB1_SUPPORT_YUV422 PIP_FB1_SUPPORT_ARGB PIP_FB1_SUPPORT_RGB888 PIP_FB1_SUPPORT_RGB565 PIP_FB2_SUPPORT_YUV422 PIP_FB2_SUPPORT_ARGB PIP_FB2_SUPPORT_RGB888 PIP_FB2_SUPPORT_RGB565	The color format of a frame buffer supports the frame buffer driver. If ARGB or RGB888 is selected (Each pixel is four bytes), it will consume more memory. To save the memory and improve the performance, RGB565 or YUV422 is preferred.	Y/N
D1_WIDTH_704	The width is 704 for D1.	-
CURSOR_64x64	Support the hardware cursor It depends on hardware.	Y/N
SUPPORT_GRAPHIC_COMPRESS	Support the graphic compression/decompression	Y/N

Chapter 3

Module Parameters

This chapter contains the following sections:

Table 3-1 lists the module parameters when inserting the LCD210 driver.

Table 3-1. Module Parameter

Name	Default value	Description
fb0_num	1	Number of fb0 frame buffers
fb1_num	1	Number of fb1 frame buffers
fb2_num	0	Number of fb2 frame buffers
output_type	0	Output resolution and output target For example, 0 is CVBS while 8 is 1024x768. This is also a proc node for users to dynamically change the output type.
input_res	Associated with output_type	Input resolution For example, 0 is D1 (NTSC) while 3 is 1024x768. This is also a proc node for users to dynamically change the input resolution.

Both output_type and input_res are very important when inserting the LCD210 module. Therefore, they will be explained in more details.

3.1 fbx_num

x means 0 ~ 1. It indicates the needed frame buffers when the driver is working. If the maximum input resolution is 1024x768 with 16-bit pixel, the consumed memory for one frame buffer will be 1024x768x2 bytes. Therefore, when x frame buffers are needed, the total memory will be 1024x768x2 times the count of the frame buffers.

3.2 output_type

output_type consists of two parts. One is the resolution and the other is the device. For example, if output_type = 0, the output resolution is D1 and the backend device is TV DAC. That is, the LCD controller outputs 720x480 or 720x576 pixels to TV DAC, which connects to the LCD displayer. Given another example, if output_type = 29, the LCD controller outputs 1920x1080 pixels and the backend device is the CAT6612 device with RGB565 interface, which is a HDMI device.

Users can execute the following command to find out the supported outputs by reading the proc node, "cat /proc/flcd200/pip/output_type".

In addition, users can dynamically change output_type at any time. For example,
"echo 8 > /proc/flcd200/pip/output_type".

This command means that users change output_type to VGA DAC with a resolution of 1024x768.

Table 3-2 lists the currently supported output_type.

Table 3-2. Supported Output Types

output_type	Description
0	Composite DAC(TVE100 DAC): NTSC
1	Composite DAC(TVE100 DAC): PAL
20	CAT6612 with 480P BT1120 interface
21	CAT6612 576P BT1120 interface
22	CAT6612 720P BT1120 interface
23	CAT6612 1080P BT1120 interface
29	CAT6612 1080P RGB565 interface

Note: Because the LCD210 driver has already added new resolutions, the resolutions in Table 3-2 may not be updated. Users can issue the following command to list the supported resolutions:
cat /proc/flcd200/pip/output_type

3.3 input_res

Compared to output_type, input_res is very simple. It indicates the input resolution. When the input resolution is changed, the frame buffer will be cleared. In addition, the input resolution also indicates the frame buffer size; therefore, users can use the input resolution as the frame buffer size. When the input resolution is bigger, the allocated memory for the frame buffer will also be bigger. Users can execute the following command to find out the supported input resolutions by reading the proc nodes. The following example is for LCD0:
"cat /proc/flcd200/pip/input_res"

For LCD2, users can issue “cat /proc/flcd200_2/pip/input_res”. It is similar to LCD2.

In addition, users can also dynamically change the input resolution at any time. For example, “echo 3 > /proc/flcd200/pip/input_res”. This command means that users change the input resolution to 1024x768 pixels. Table 3-3 lists the currently supported input resolutions.

Table 3-3. Supported Input Resolutions

input_res	Description
0	D1 (NTSC)
1	D1 (PAL)

Notes:

1. Because the LCD210 driver has already added new resolutions, the resolutions in Table 3-2 may not be updated. Users can issue the following command to list the supported resolutions:
   cat /proc/flcd200/pip/input_res.
2. When the input resolution is higher, the consumed memory will also be bigger. Thus, a configure file is provided to determine the maximum input resolution. Please use the following declaration in the module/LCD200_v3 directory.
   static VIN_RES_T res_support_list[] = { VIN_NTSC, VIN_PAL, VIN_640x480, …..};

Users can find out the supported resolutions for the content in the support_list array.

```
/lib/modules # cat /proc/flcd200/pip/input_res

id =   0, NTSC                    id =   1, PAL
id =   2, 640x480(VGA)            id =   3, 1024x768(XVGA)
id =   4, 1440x1080(HD)           id =   5, 1280x800(WXGA)
id =   6, 1280x960                id =   7, 1280x1024(SXGA)
id =   8, 1360x768(HD)            id =   9, 1280x720(720P)
id =  10, 1600x1200               id =  11, 640x360
id =  12, 1680x1050(SXGA)         id =  14, 800x600(SVGA)
id =  15, 1920x1080               id =  16, 1440x960
id =  17, 1440x1152               id =  18, 1440x900
id =  19, 540x360

Current input_res value = 3

      Input Res.          Output Res.          Max Res.
-------------------------------------------------------
xres:1024                 1024                 1280
yres:768                  768                  1024
(input_res = 1024x768(XVGA), output_type = 1024x768)

LCD Scalar           : Disabled
Cursor postion scale: Disabled

Input resolution support list ==>
NTSC
PAL
640x480(VGA)
1024x768(XVGA)
1280x1024(SXGA)
800x600(SVGA)

/lib/modules #
```

3.4 d1_3frame

For GM8136, this function is disabled.

3.5 Example

The following example shows the operation procedure.

```
/mnt/nfs # insmod flcd200-common.ko
LCD platform: Hook GM8136 driver.

/mnt/nfs # insmod flcd200-pip.ko input_res=3 output_type=8
GM_ffb: [ver:0.1.14]INIT flcd200 OK.
LCD0: PA = 0x9a800000, VA = 0xcf8a0000, size:0xd000 bytes
LCD0: Output resolution is 1024x768/60HZ.
Register codec 1024x768/60HZ.
LCD0: Input resolution is 1024x768(XVGA)
LCD0: Standard: 65.0MHZ
LCD0: PVALUE = 9, pixel clock = 64.8MHZ, scalar clock = 216MHZ
LCD driver[Ver: 1.1.1] init ok.

/lib/modules # insmod flcd200-pip2.ko output_type=1 d1_3frame=1
GM_ffb: [ver:0.1.14]INIT flcd200 OK.
LCD2: PA = 0x9b200000, VA = 0xcf890000, size:0xd000 bytes
LCD2: Output resolution is D1 PAL.
LCD2: Input resolution is PAL
LCD2 joins D1 of 3frame.
LCD2: Driver[Ver: 1.2.0] init ok.
```

Chapter 4

LCD GUI

This chapter contains the following sections:

This chapter describes how to use the GUI plane to draw GUI. LCDC provides several planes. Plane 0 is dedicated for the video display. Plane 1 is a graphic plane and users can use IOCTL to apply some functions on that, such as the mmap() function. Before using GUI, users MUST turn on the PIP plane for the GUI display. When GUI plane is turned on, it must have higher blending value than plane 0. Thus, users can turn it off by default.

Users can turn it on by the using the following proc commands for LCD0:

(1) echo x > /proc/flcd200/pip/pip_mode, where x: 0, 1, and 2
(2) echo y > /proc/flcd200/pip/blend, where y: 0 ~ 255. Less value gets lower blending value. It is zero by default, this means that PIP1 is off. Usually, users should set this value to higher value and gets high blending value than PIP0.

For LCD2, the proc commands are:
(1) echo x > /proc/flcd200_2/pip/pip_mode, where x: 0, 1, and 2
(2) echo y > /proc/flcd200_2/pip/blend, where y: 0 ~ 255. Less value gets lower blending value. It is zero by default, which means that PIP1 is off. Usually, users should set this value to higher value and gets high blending value than PIP0.

For the PIP1 GUI plane, the device node is /dev/fb1 for LCD0 and /dev/fb4 for LCD2. If users cannot find these device nodes, it will not be available to use GUI.

These proc commands should be enabled behind the LCD drivers. Users should add these two proc commands in the boot script file. The boot script file should be vg_boot.sh in /mnt/mtd.

For example:

```
insmod flcd200-common.ko
insmod flcd200-pip.ko output_type=51
insmod flcd200-pip2.ko
…
echo 1 > /proc/flcd200/pip/pip_mode
echo 255 > /proc/flcd200/pip/blend
echo 1 > /proc/flcd200_2/pip/pip_mode
echo 255 > /proc/flcd200_2/pip/blend
```

Linux graphic middleware provides several IOCTLs and users can find the related information on the internet. Below lists the frequently-used IOCTLs:

1. FBIOGET_FSCREENINFO
   This IOCTL is used to get the memory information such as smem_start, smem_len, and line_length.
2. FBIOGET_VSCREENINFO
   This IOCTL is used to get screen related information such as xres, yres, and bits_per_pixel.

In addition to the IOCTLs provided by Linux, LCD driver also provides some private IOCTLs, which are the following:

1. FLCD_GET_DATA_SEP/FFB_GET_DATA_SEP: It is used to get the LCD driver information, such as buffer length and number of frame buffers.

```
void function(void)
{
 struct flcd_data f_data;
 unsigned char *gui_vaddr; /* The virtual address of GUI frame buffer */

 ioctl(fb1_fd, FLCD_GET_DATA_SEP, &f_data);
 gui_vaddr = (unsigned char *)mmap(0, f_data.frame_no * f_data.buf_len,
PROT_READ|PROT_WRITE, MAP_SHARED, fb1_fd, 0);
}
```

2. FLCD_SET_FB_NUM/FFB_SET_FB_NUM: If the number of frame buffers exceeds one, then it indicates which frame buffer is ready to display. Currently, this value is zero due to only one frame buffer in GUI. For this case, users do not need to set this IOCTL. As to how many frames supported for GUI, it is configurable in LCD driver.

```
void function(void)
{
 int fbn = 0;

 ioctl(fb1_fd, FLCD_SET_FB_NUM, &fbn);
```

```
}
```

3. COLOR_KEY1: Color key for plane 1 and it is a 32-bit value.
   If the plane is a RGB565 one, users should transform it into a 32-bit value.

```
void function(void)
{
 unsigned short rgb565_color = 0x1F; /* Blue color */
 unsigned int rgb888_color;

 rgb888_color = ((565_color & 0xF800) << 3) | ((565_color & 0x7E0) << 2) | ((565_color & 0x1F) << 3);    /* to RGB888 */
 ioctl(fb1_fd, COLOR_KEY1, & rgb888_color);
}
```

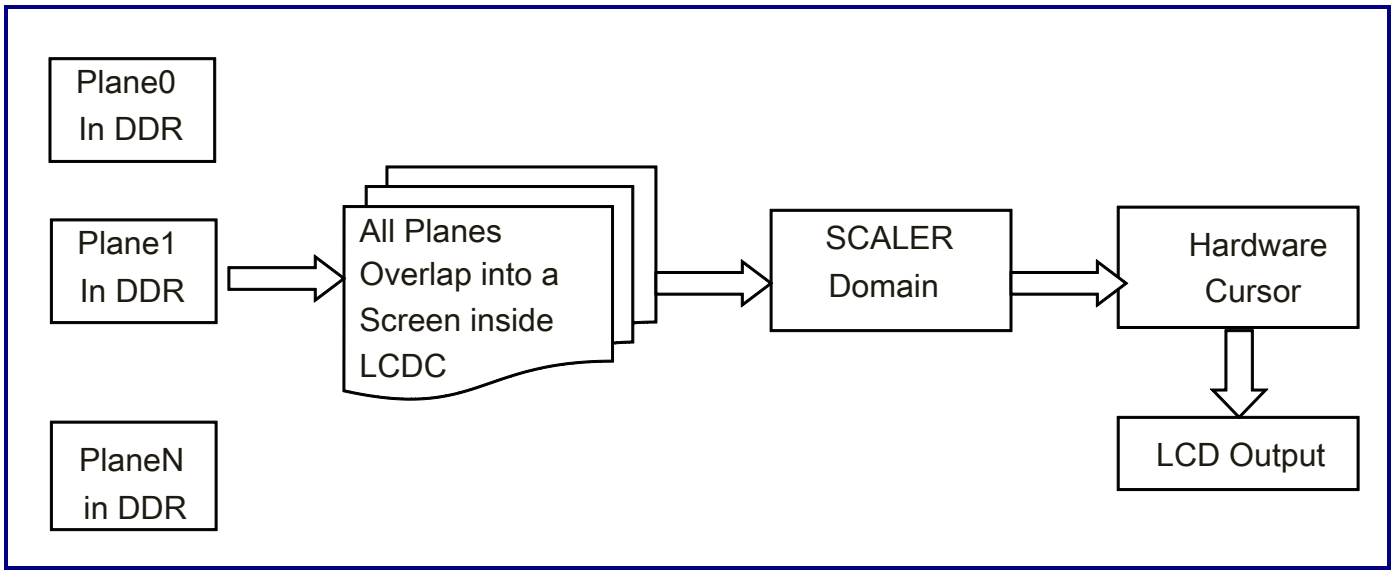
4. COLOR_KEY1_EN: Color key1 is enabled or disabled. 1 for enabled and 0 for disabled.
5. COLOR_KEY2: Color key for plane 2 and it is a 32bit value
6. COLOR_KEY2_EN: Color key2 is enabled or disabled. 1 for enabled and 0 for disabled.
7. FFB_IOCURSOR: Used to set the position of hardware cursor. A sample code will be provided to the customers for reference.

4.1 GUI Frame Format

In general, users use RGB color to draw the pictures. LCD driver provides RGB565, RGB888, ARGB888, and YUV422 for this GUI frame buffer. For the bandwidth consideration, Grain Media suggests users using RGB565 plus color key to implement it. Thus, the default frame format is RGB565. Users can read the format through the proc functions, such as "cat /proc/flcd200/pip/fb1_input". It will display which format is working now.

4.2 LCD Scaler

When the input resolution is not equal to the output resolution, the scaling capability is necessary. LCDC supports the global scaling capability instead of per plane capability. A frame overlapped with all planes will go through scaler if the input resolution and output one is different. The following picture shows the position of the SCALER domain. Please note that LCD2 does not provide the scaler capability.



When LCD driver detects the input resolution is not equal to output one, the scaler will be enabled automatically. Please be aware that the scaler capability is only maximized to 2x2. That means the horizontal scaler is two times and so is vertical.

4.3 Hardware Cursor

For the cursor, users can either use CPU to draw the software cursor or use hardware cursor provided by the LCD driver. When the latter case is used, it can save CPU resource and users only need to set the x/y positions to the LCD driver via IOCTL_FFB_IOCURSOR.

Before using the hardware cursor, users need to set its shape, color and so on. A sample is provided to use the hardware cursor. Please refer to arm-linux-3.3/module/LCD200_v3/sample/test_lcd_cursor.

4.4 Graphic Accelerator

When users would like to draw a picture, such as a rectangle, users can either uses CPU to draw the frame buffer directly or uses 2D graphic engine accelerator to offload CPU. 2D graphic engine accelerator supports line drawing, rectangle drawing, and so on. Users can refer to the 2D engine user guide to get more information.

4.5 Share GUI Contents among LCDs

In GM8136, this function is disabled.,

4.5.1 Hardware Cursor Share

While sharing the GUI content, the hardware cursor can also be shared if users use the hardware cursor to control the cursor position. The procedure is shown as below (Assume that LCD0 is the main GUI screen):

(1) Configure the hardware cursor including its shape, color, and so on through the device node of LCD0
(2) Configure the hardware cursor including its shape, color, and so on through the device node of LCD2
(3) Update the cursor position of LCD0, the position of LCD2 will be updated automatically inside the LCD driver.

For the hardware cursor usage, please refer to arm-linux-3.3/module/LCD200_v3/sample/test_lcd_cusor.

Chapter 5

Proc Nodes

This chapter contains the following section:

The LCD module provides several proc nodes to directly access the LCD setting without using a function call. The proc nodes are divided into two categories, one is related to the LCD controller self-setting, and the other is the backend DAC connected to the LCD controller. These categories associated with the directories, flcd200 and flcd200_cmn, are shown below:

```
/proc # cd flcd200_cmn/
/proc/flcd200_cmn # ls
codec_reg_info    io_link1_vga    sli10121    vg_info

/proc/flcd200_cmn # cd /proc/flcd200
/proc/flcd200 # ls
brightness        dbg_mode          mouse_pose_scale
contrast          dbg_msgcnt        pattern_gen
cvbs_screen_pos   dump_reg          pip
cvbs_screen_sz    frame_compesation saturation
dac_input_sel     hue               sharpness
dbg_level         lcd_on_off
/proc/flcd200 #
```

The following table shows the proc nodes. For the latter case, the proc nodes depend on the connected DAC. For example, when using CAT6611, a HDMI chip is connected to the LCD controller, the driver will have inserted into the system, and users can find information for this codec in the flcd200_cmn directory. In addition, the PIN MUX setting can also be found in this directory, if necessary.

/proc/flcd200	brightness	
	contrast	
	hue	
	saturation	
	sharpness	
	pattern_gen	
	mouse_pos_scale	
	cvbs_screen_pos	
	pip	blend
		fb0_input
		fb0_win

		fb1_input
		fb1_win
		input_res
		output_type
		property
		pip_mode
		vg_dbg
/proc/flcd200_cmn	codec_reg_info	
	cat6611	
	vg_info	

Please note that some nodes in /proc/flcd200_cmn are optional. These nodes are platform-depended. For example, iolink1_vga can only be appeared in GM8136. The codec_reg_info node is only for codec, such ascat6612.

5.1 /proc/flcd200 Proc Nodes

The following subsections describe these proc nodes.

5.1.1 Brightness

Users can use the "/proc/flcd200/brightness" node to change the brightness value.
Usage:

- Get the current value
  cat/proc/flcd200/brightness
- Set the brightness value
  echo [value] > /proc/flcd200/brightness, where the value is between -127 and +127

5.1.2 Contrast

Users can use the "/proc/flcd200/contrast" node to change the contrast value.

Usage:

- Get the current value

  cat/proc/flcd200/contrast
- Set the contrast value

  echo [value] > /proc/flcd200/contrast, where the value is between 1 and 31.

5.1.3 Hue

Users can use the "/proc/flcd200/hue" node to change the hue value.

Usage:

- Get the current value

  cat/proc/flcd200/hue
- Set the hue value

  echo [value] > /proc/flcd200/hue, where the value is between -32 and +32

5.1.4 Saturation

Users can use the "/proc/flcd200/saturation" node to change the saturation value.

Usage:

- Get the current value

  cat/proc/flcd200/saturation
- Set the saturation value

  echo [value] > /proc/flcd200/saturation, where the value is between 0 and 63.

5.1.5 Sharpness

Users can use the "/proc/flcd200/sharpness" node to change the sharpness value.

Usage:

- Get the current value

  cat/proc/flcd200/sharpness
- Set the sharpness value

  echo [k0] [k1] [threshold0] [threshold1] > /proc/flcd200/sharpness.

  k0/k1: Range 0 ~ 15

  threshold0/threahold1: Range 0 ~ 255

5.1.6 Pattern Generation

Users can use the "/proc/flcd200/pattern_gen" node to control the pattern generation. It is only for debugging. The pattern is generated from the LCD controller and is not generated from SRAM or DDR.

Usage:

- Get the current value

  cat/proc/flcd200/pattern_gen
- Set the pattern_gen value

  echo [value] > /proc/flcd200/pattern_gen.

  Value: 0 for disable or 1 for enable

5.1.7 mouse_pos_scale

Users can use the "/proc/flcd200/mouse_pos_scale" node to control the mouse position scale. When the input resolution is not equal to the output resolution, the LCD module will automatically perform the scale operation. This also effects the mouse position due to the scaling factor.

Usage:

- Get the current value

  cat/proc/flcd200/mouse_pos_scale
- Set the mouse position scaling value

  echo [value] > /proc/flcd200/mouse_pos_scale.

  Value: 0 for disable or 1 for enable

5.1.8 Blend

Users can use the "/proc/flcd200/pip/blend" node to change the PIP alpha blending value and blending priority.

Usages:

- Get the current value
  cat/proc/flcd200/pip/blend
- Set the blending value and blending priority
  Blend_*: 0 ~ 255. Limitation is Blend_H+Blend_L ≤ 255.
  Img*_P: 0 ~ 2. The blending priority of each buffer.
  echo [Blend_H] [Blend_L] [Img0_P] [Img1_P] [Img2_P] > /proc/flcd200/pip/blend

5.1.9 Switch Input Color Format

Users can use the "/proc/flcd200/pip/fb*_input" node to switch the input color format. (*: 0 ~ 2)

Usage:

- Get the input color format that is supported by the driver
  cat/proc/flcd200/pip/fb*_input
- Set the input color format
  echo [color index] > /proc/flcd200/pip/fb*_input

5.1.10 Change Sub-window Position and Dimension of PIP

Users can use the "/proc/flcd200/pip/fb*_win" node to change the sub-window position and dimension of PIP

(*: 1 ~ 2)

Usage:

- Get the current value
  cat/proc/flcd200/pip/fb*_win
- Set the sub-window position and dimension of PIP; the dimension must be less than the dimension of frame buffer 0.
  echo [X] [Y] [Width] [Height] > /proc/flcd200/pip/fb*_win

5.1.11 Input Resolution

Users can use "/proc/flcd200/pip/input_res" to change the input resolution.

Usage:

- Get the current value

  cat/proc/flcd200/pip/input_res

- Set the input resolution

  echo [value] > /proc/flcd200/pip/input_res.

  Value: Please refer to the following example.

```
# /proc/flcd200/pip # cat input_res

id =  0, NTSC                        id =  1, PAL
id =  2, 640x480(VGA)          id =  3, 1024x768(XVGA)
id =  4, 1440x1080(HD)        id =  5, 1280x800(WXGA)
id =  6, 1280x960                 id =  7, 1280x1024(SXGA)
id =  8, 1360x768(HD)          id =  9, 1280x720(720P)
id = 10, 1600x1200              id = 11, 640x360
id = 12, 1680x1050(SXGA)  id = 14, 800x600(SVGA)
id = 15, 1920x1080              id = 16, 1440x1152
id = 17, 1440x960

Current input_res value = 3

     Input Res.       Output Res.       Max Res.
-----------------------------------------------
xres:1024             1024                1024
yres:768               768                  768
(input_res = 1024x768(XVGA), output_type = 1024x768)

LCD Scalar          : Disabled
Cursor position scale: Disabled

Input resolution support list ==>
NTSC
PAL
640x480(VGA)
1024x768(XVGA)
800x600(SVGA)


/proc/flcd200/pip # echo 0 > input_res LCD: Input resolution is NTSC

Input resolution is changed!
```

Please note that if users input a non-supported resolution, an error will pop out.

5.1.12 Output Resolution

Users can use the "/proc/flcd200/pip/output_type" node to change the output resolution. output_type is addressed in the previous chapter.

Usage:

- Get the current value

  cat/proc/flcd200/pip/output_type

- Set the output type value

  echo [value] > /proc/flcd200/pip/output_type

  Value: Please refer to the following example.

```
/proc/flcd200/pip # cat output_type

id =  0, D1(NTSC)                id =  1, D1(PAL)
id =  2, CS4954                   id =  3, PVI2003A
id =  4, FS453                    id =  5, MDIN200_HD(1440x1080)
id =  6, MDIN200_D1            id =  8, VGA_1024x768
id =  9, VGA_1280x800         id = 10, VGA_1280x960
id = 13, VGA_1280x1024     id = 14, VGA_1360x768
id = 15, VGA_800x600          id = 16, VGA_1600x1200
id = 17, VGA_1680x1050     id = 18, Cascase 1024x768x25
id = 19, Cascase 1024x768x30  id = 20, CAT6611_480P
id = 21, CAT6611_576P              id = 22, CAT6611_720P
id = 23, CAT6611_1080P           id = 24, CAT6611_1024x768_RGB
id = 25, CAT6611_1024x768P    id = 26, CAT6611_1280x1024P
id = 27, CAT6611_1680x1050P  id = 28, CAT6611_1280x1024_RGB
id = 29, CAT6611_1920x1080_RGB     id = 35, MDIN240_HD(1440x1080)
id = 36, MDIN240_SD

Current output_type value = 8

     Input Res.       Output Res.       Max Res.
-------------------------------------------------
xres:1024             1024              1024
yres:768              768               768
(input_res = 1024x768(XVGA), output_type = 1024x768)
```

```
LCD Scalar          : Disabled
Cursor position scale: Disabled
```

5.1.13 Property

Users can use the "/proc/flcd200/pip/property" node to read the current setting configured from the videograph layer. It is read-only and is only for debugging purpose.

Usage:

- Get the current value

  cat/proc/flcd200/pip/property

5.1.14 Switch PIP Mode

Users can use the "/proc/flcd200/pip/pip_mode" node to change the PIP mode of the LCD module.

Usages:

- Get the current value

  cat/proc/flcd200/pip/pip_mode

- Set the PIP mode

  echo [mode] > /proc/flcd200/pip/pip_mode

  mode: 0: Disable, 1: Single PIP window, 2: Double PIP windows

5.1.15 Vg_dbg

Users can use the "/proc/flcd200/pip/vg_dbg" node to debug the job status of the LCD module. It is only for debugging purpose.

Usages:

- Get the current value

  cat/proc/flcd200/pip/vg_dbg

Chapter 6

LCD Outputs

The LCD driver can output screen through TV DAC, BT1120 or RGB565 interface. Currently, the BT1120/RGB565 interfaces are pin muxed with MAC interface. Thus is, when BT1120 or RGB565 interface is used, MAC interface must be disabled.

Besides, when either BT1120 or RGB565 interface is used, the PLL3 frequency must be taken into consideration. Currently, for 720P or 1080P the PLL3 must be 594 MHz. For 540M PLL3, only D1 resolution can be supported. For the PLL3 configuration, it is done in nsboot.bin.