



Application Notes

AML8726-MX/MXL/MXS Panel Tuning User Guide
Revision 0.2

Amlogic Confidential!

Amlogic, Inc.
3930 Freedom Circle
Santa Clara, CA 95054
U.S.A.
www.amlogic.com

Legal Notices

© 2013 Amlogic, Inc. All rights reserved. Amlogic® is registered trademarks of Amlogic, Inc. All other registered trademarks, trademarks and service marks are property of their respective owners.

This document is Amlogic Company confidential and is not intended for any external distribution.

Index

1. Overview	4
2. LCD interface knowledge	5
2.1 TTL interface	5
2.2 LVDS interface.....	7
3. LCD basic information	9
3.1 Panel actual size.....	9
3.2 Panel type and bit width.....	9
4. LCD timing config	10
4.1 Display area definition	10
4.2 Pixel clock and spread spectrum setting.....	11
4.3 Synchronization signals and polarity setting	12
5. LCD interface config	13
5.1 TTL interface RGB data swapping	13
5.2 LVDS interface data mapping and PN swapping.....	13
6. LCD recommend setting.....	14
7. LCD data processing	15
7.1 Offset	15
7.2 Gain	15
7.3 Gamma.....	16
8. LCD power control	17
9. LCD online debug	18
9.1 LCD timing online tuning.....	18
9.2 LCD gamma online debug	20
10. LCD clock calculate.....	22
10.1 TTL interface pixel clock.....	22
10.2 LVDS interface pixel clock.....	23
11. Example.....	24
11.1 Power control	24
11.2 Display control.....	26

Amlogic Application Notes

Revision History

Revision	Date	Owner	Changes
0.1	March 29, 2013	Evoke Zhang	Draft
0.2	May 20, 2013	Evoke Zhang	Change timing parameters' description for new lcd driver

Amlogic Confidential!

1. Overview

This document is an user guide of AML8726-MX/MXL/MXS LCD panel tuning, it describes:

- LCD interface knowledge
- LCD basic information
- LCD timing config
- LCD interface control
- LCD recommend setting
- LCD data processing
- LCD power control
- LCD online debug
- LCD clock calculate
- Example

Amlogic Confidential!

2. LCD interface knowledge

2.1 TTL interface

- RGB data

AML8726-MX has up to 24 RGB data for 8-bit TTL panel (RGB888). AML8726-MXL/MXS have up to 18 RGB data for 6-bit TTL panel (RGB666). All RGB data can be multiplexed to GPIO pins shown in Table 2.1.

Pad	Signal	Pad Name	Signal	Pad	Signal	Chip Support
GPIOB_0	LCD_R0	GPIOB_8	LCD_G0	GPIOB_1	LCD_B0	MX
GPIOB_1	LCD_R1	GPIOB_9	LCD_G1	GPIOB_1	LCD_B1	MX
GPIOB_2	LCD_R2	GPIOB_10	LCD_G2	GPIOB_1	LCD_B2	MX/MXL/MXS
GPIOB_3	LCD_R3	GPIOB_11	LCD_G3	GPIOB_1	LCD_B3	MX/MXL/MXS
GPIOB_4	LCD_R4	GPIOB_12	LCD_G4	GPIOB_2	LCD_B4	MX/MXL/MXS
GPIOB_5	LCD_R5	GPIOB_13	LCD_G5	GPIOB_2	LCD_B5	MX/MXL/MXS
GPIOB_6	LCD_R6	GPIOB_14	LCD_G6	GPIOB_2	LCD_B6	MX/MXL/MXS
GPIOB_7	LCD_R7	GPIOB_15	LCD_G7	GPIOB_2	LCD_B7	MX/MXL/MXS

Table 2.1 - RGB data multiplexing table (LCD_RGB[1:0] are invalid in AML8726-MXL/MXS)

<Note>:

6-bit TTL panel (RGB666) must be connected LCD_RGB[7:2], or it will display abnormally.

When set the right lcd bit width, the lcd driver will set the right RGB data pinmux automatically. Please refer to Chapter 3.2.

Figure 2.1 shows the correct connection for 6-bit TTL panel (RGB666) and 8-bit TTL panel (RGB888).

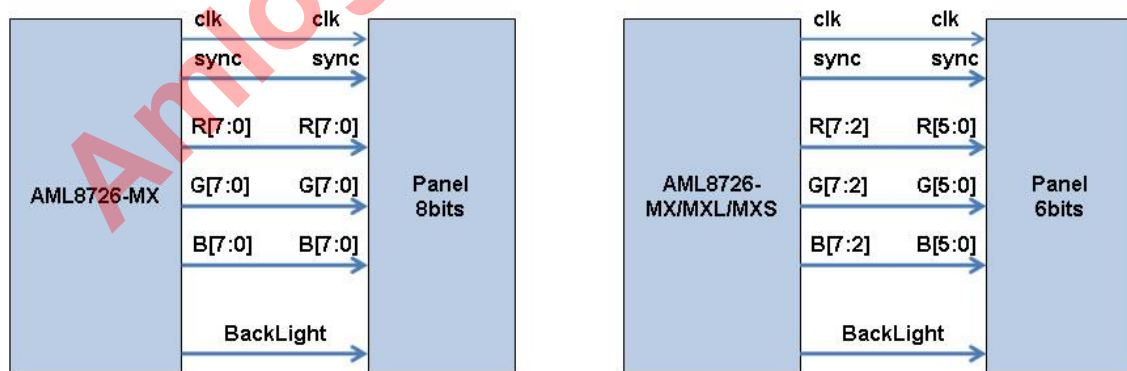


Figure 2.1 - RGB Data Connection (Left: RGB888 to 8-bit TTL panel. Right: RGB666 to 6-bit TTL panel)

- RGB data swapping

AML8726-MX support R/B data swapping and RGB MSB/LSB bit swap. AML8726-MXL/MXS only support R/B data swapping. It is convenient for hardware circuit connection design. Please refer to Chapter 5.1.

RGB data swapping is shown in Figure 2.1.

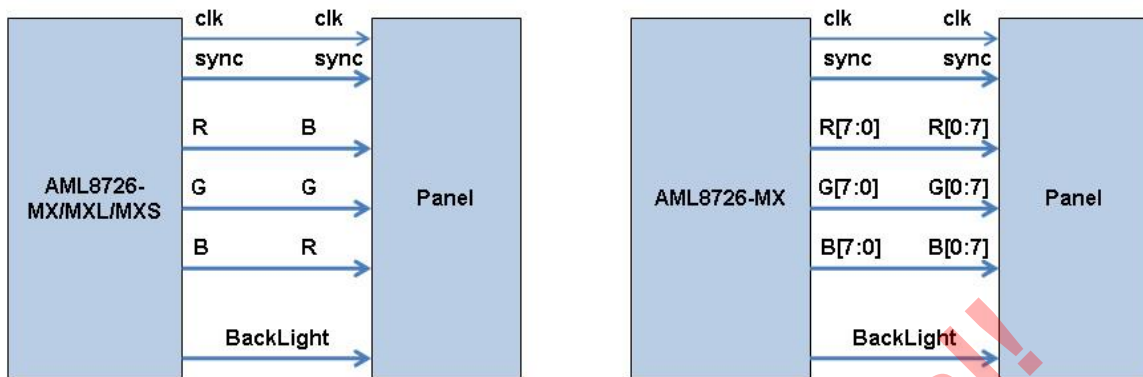


Figure 2.2 - RGB Data Swapping (Lef: R/B data swapping. Right: RGB MSB/LSB bit swapping)

- TCON signals

AML8726-MX/MXL/MXS has built-in TCON module for TTL interface. All TCON signals can be multiplexed to GPIO pins shown in Table 2.2. They are automatically set by lcd driver.

	Signal	Default TTL	Pinmux	Chip Support
GPIO_D_2	TCON_STH1	Hsync	reg1[19]	MX/MXL/MXS
GPIO_D_3	TCON_STV1	Vsync	reg1[18]	MX/MXL/MXS
GPIO_D_4	TCON_OEH	DE	reg1[17]	MX/MXL/MXS
GPIO_D_7	TCON_CPH1	Pclk	reg1[14]	MX/MXL/MXS

Table 2.2 - TCON signals multiplexing table

2.2 LVDS interface

- LVDS signals

AML8726-MX/MXL/MXS has 1 pair of clock signals and 5 pairs of data signals, which can support up to 10-bits RGB data. We usually use 6-bits or 8-bits RGB data for panel required.

<Note>:

When set the right lcd bit width, the lcd driver will set the right lvds channels automaticlly. Please refer to Charppter 3.2.

- LVDS data PN swapping

AML8726-MX/MXL/MXS support LVDS channels P/N swap.

<Note>:

It can only support swap all the LVDS channels at the same time. Please refer to Charppter 5.2.

- LVDS data mapping

AML8726-MX/MXL/MXS support 2 options for LVDS data mapping:

JEDIA mode shown in Figure 2.3

VESA mode shown in Figure 2.4

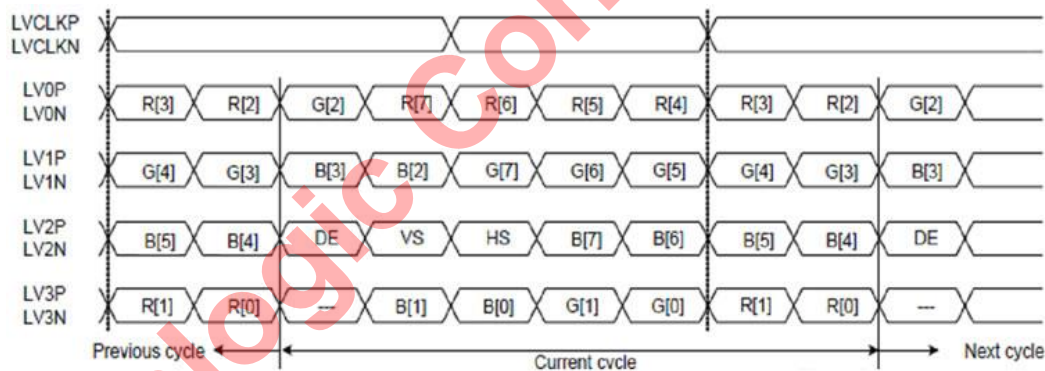


Figure 2.3 - JEIDA mode

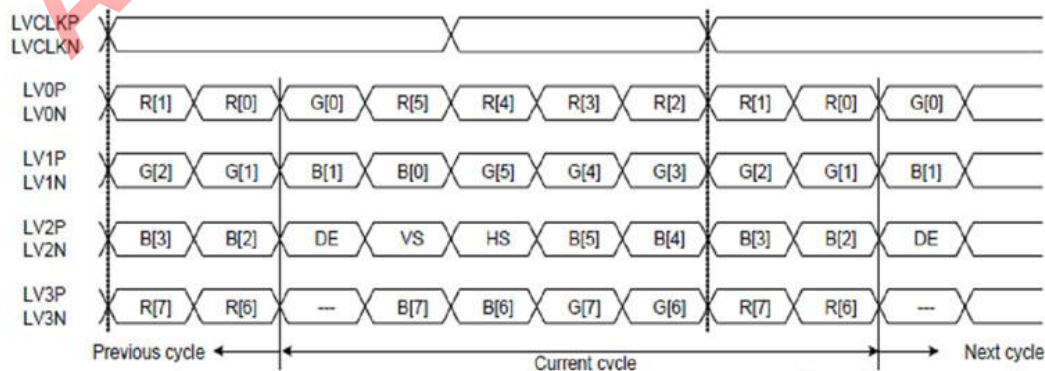


Figure 2.4 - VESA mode

<Note>:

The LVDS panel will be display abnormal if you use the wrong data mapping. Please refer to Chapter 5.2.

Amlogic Confidential!

3. LCD basic information

3.1 Panel actual size

- Set panel actual size correctly to meet the OSD display ratio and Android DPI calculation.
- Those values are controlled by:

```
#define ACITVE_AREA_WIDTH      197    //unit: mm  
#define ACITVE_AREA_HEIGHT    147    //unit: mm
```

<Note>:

You will find the parameters in panel spec, they usually called “Active Area” or “Display Area”.

3.2 Panel type and bit width

- Set panel interface information correctly to match panel type and bit width.
- We support 2 options of panel type:
LCD_DIGITAL_TTL: TTL digital panel
LCD_DIGITAL_LVDS: LVDS digital panel
- We support 2 options of panel bit width:
8: RGB888 for 8-bit panel
6: RGB666 for 6-bit panel
- Those values are controlled by:

```
#define LCD_TYPE                LCD_DIGITAL_LVDS //LCD_DIGITAL_TTL //LCD_DIGITAL_LVDS  
#define LCD_BITS                6              //6          //8
```

<Note>:

You can get the right lcd bit width by the discriptions as below.

- TTL RGB bit:
24bit, 3x8 -> 8bit
18bit, 3x6 -> 6bit
- LVDS data channels:
data channel 0~3 -> 8bit
data channel 0~2 -> 6bit
- color depth:
16.7M, 2^8^3 -> 8bit
262K, 2^6^3 -> 6bit

4. LCD timing config

4.1 Display area definition

- Display area is illustrated in Figure 4.1.

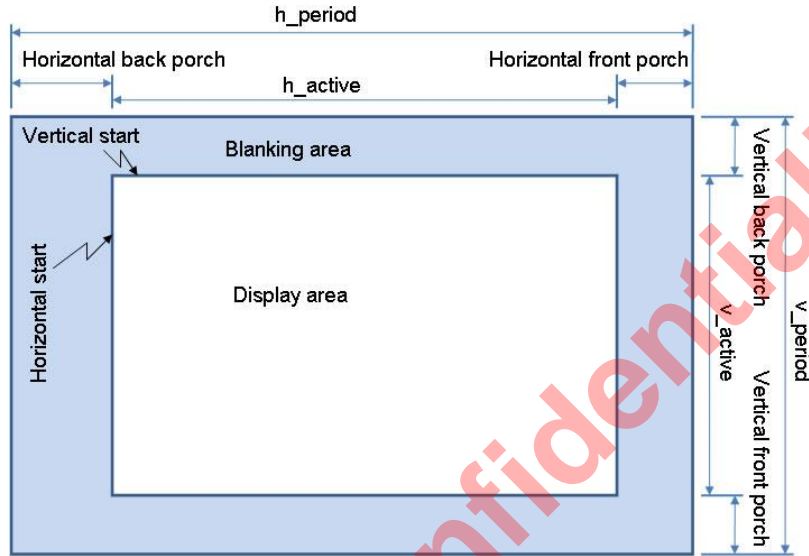


Figure 4.1 - Panel timing diagram

- Panel timing is available in panel specification. An example is shown in Table 4.1.

Item	Value	Unit	Related Parameter of Display Area Definition	Related LCD Synchronous Signals
Horizontal Active	1024	pixel	h_active	Hsync, DE
Horizontal Period	1344	pixel	h_period	Hsync, DE
Horizontal Blanking	320	pixel	Horizontal back porch + Horizontal front porch	DE
Horizontal pulse width	20	pixel		Hsync
Horizontal back porch	160	pixel	Horizontal back porch	Hsync
Vertical Active	600	line	v_active	Vsync, DE
Vertical Period	635	line	v_period	Vsync, DE
Vertical Blanking	35	line	Vertical back porch + Vertical front porch	DE
VS pulse width	3	line		Vsync
VS back porch	23	line	Vertical back porch	Vsync

Table 4.1 - Panel video timing table

- Software video timing parameters need match the above panel timing:
`#define H_ACTIVE 1024`
`#define V_ACTIVE 600`
`#define H_PERIOD 1344 //Remember, H_PERIOD > H_ACTIVE + VIDEO_ON_PIXEL`
`#define V_PERIOD 635 //Remember, V_PERIOD > V_ACTIVE + VIDEO_ON_LINE`

<Note>:

- `H_ACTIVE` and `V_ACTIVE` define the display area for real image
- `H_PERIOD` and `V_PERIOD` define the whole area. They are used to calculate panel line frequency and frame rate:
$$\text{Line frequency} = \text{pixel clock} / H_PERIOD$$
$$\text{Frame rate} = \text{pixel clock} / H_PERIOD / V_PERIOD$$
- `VIDEO_ON_PIXEL/LINE` please refer to Chapter 6.

4.2 Pixel clock and spread spectrum setting

- Set panel aspect ratio correctly to match the display area.
- Those values are controlled by:
`//#define FRAME_RATE 50`
`#define LCD_CLK 85700000/(H_PERIOD * V_PERIOD * FRAME_RATE) //unit: Hz`
`#define CLK_SS_LEVEL 0 //0~5, 0 for disable spread spectrum`
`#define CLK_AUTO_GEN 1 //1, auto generate clk parameters //0, user set clk parameters`

<Note>:

- You can directly set the pixel clock frequency in unit Hz, or you can set the frame rate, and use the formula "`H_PERIOD * V_PERIOD * FRAME_RATE`" to calculate the pixel clock automatically.
- Spread spectrum is for EMI reducing, the `ss_level` value means the spread spectrum width.
- `CLK_AUTO_GEN` usually set to 1. When set it to 0, you must manual setting the `pll_ctrl`, `div_ctrl` and `clk_ctrl` in lcd config, to generate the specified lcd pixel clock (please refer to Chapter 10).

4.3 Synchronization signals and polarity setting

- The Hsync and Vsync signals are illustrated in Figure 4.2.

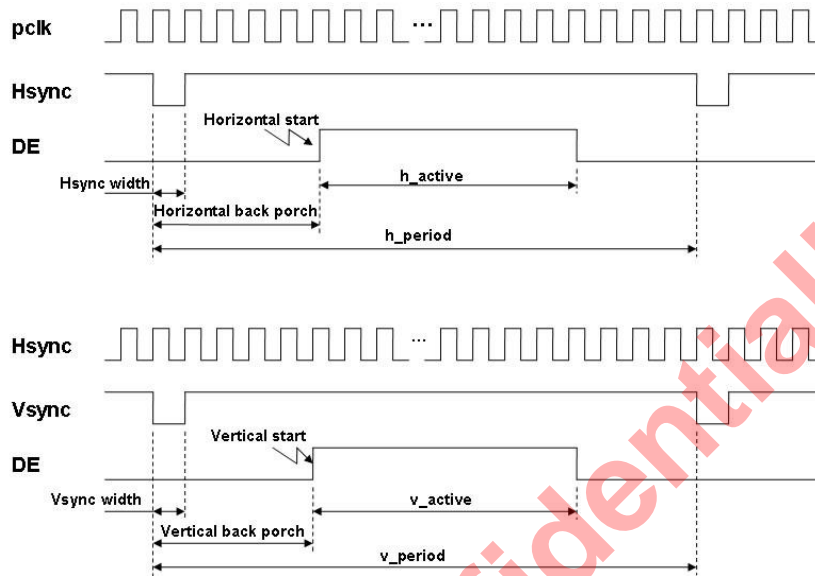


Figure 4.2 - TTL interface TCON signal timing

- The Hsync and Vsync signals, lcd pixel clock and sync signals polarity are controlled by:
//modify below settings if needed

```
#define CLK_POL          0      //0: negative, 1: positive
#define HS_WIDTH         20
#define HS_BACK_PORCH    160    //include HS_WIDTH
#define HS_POL           0      //0: negative, 1: positive
#define VS_WIDTH         3
#define VS_BACK_PORCH    23     //include VS_WIDTH
#define VS_POL           0      //0: negative, 1: positive
```

5. LCD interface config

5.1 TTL interface RGB data swapping

- AML8726-MX supports both R/B data swapping and RGB MSB/LSB data bit swapping.
- AML8726-MXL/MXS support only R/B data swapping.
- RGB data swapping is controlled by:

```
#define TTL_RB_SWAP          0      //0: normal, 1: swap  
#define TTL_RGB_BIT_SWAP    0      //0: normal, 1: swap
```

<Note>:

- RGB data swapping are determined by hardware circuit connection.

5.2 LVDS interface data mapping and PN swapping

- LVDS data mapping and PN swapping are controlled by:

```
#define LVDS_REPACK          0  //lvs data mapping //0:JEIDA mode, 1:VESA mode  
#define LVDS_PN_SWAP        0  //0:normal, 1:swap
```

<Note>:

- LVDS data mapping, please refer to Chapter 2.2.
- LVDS PN swapping is determined by hardware circuit connection, please refer to Chapter 2.2.

6. LCD recommend setting

- There are some parameters related to MX/MXL/MXS internal modules. Usually they don't need to modify. But sometime there maybe display problems, then we can modify them to fine tune lcd display.

- The parameters are controlled by:

//recommend settings, don't modify them unless there is a display problem

```
#define TTL_H_OFFSET      0      //adjust ttl display h_offset
#define H_OFFSET_SIGN    1      //0: negative value, 1: positive value
#define TTL_V_OFFSET      0      //adjust ttl display v_offset
#define V_OFFSET_SIGN    1      //0: negative value, 1: positive value
#define VIDEO_ON_PIXEL   80
#define VIDEO_ON_LINE    32
```

<Note>:

- *H/V_OFFSET* can adjust TTL display offset. When the panel is work in DE mode, you can modify these parameters to adjust the diplay position. They are no effect to LVDS panel.
- *VIDEO_ON_PIXEL/LINE* only can modify in spcial condition, it doesn't recommend to modify them.

7. LCD data processing

- AML8726-MX/MXL/MXS integrates a build-in data processing engine to compensate LCD panel display characteristics and improve the display quality
- All data processing is only in RGB color gamut
- All data processing is in 10-bit even though the final output to panel is only up to 8-bit
- The data processing includes three steps:
offset (base)
gain (co-efficient)
gamma
- The data processing is illustrated in Figure 7.1

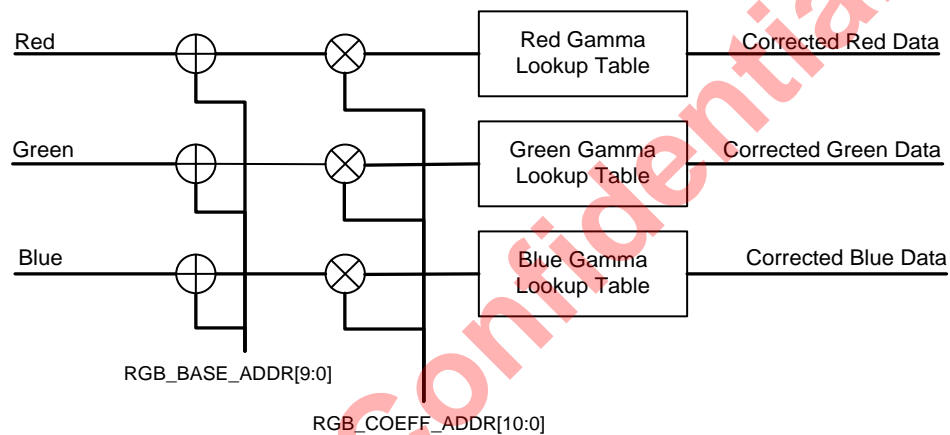


Figure 7.1 - LCD data processing flowchart

7.1 Offset

- Offset is controlled by:
`struct lcd_effect. rgb_base_addr = 0xf0,`

<Note>:

Offset will be added to R/G/B all the time.

7.2 Gain

- Gain is controlled by:
`struct lcd_effect. rgb_coeff_addr = 0x74a,`

<Note>:

Gain will be multiplied to R/G/B all the time.

7.3 Gamma

- 3*256 10-bit registers are used for R/G/B gamma lookup table. They are set as below:

```
static void lcd_setup_gamma_table(Lcd_Conf_t *pConf)
{
    int i;
    const unsigned short gamma_adjust[256] = {
        0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,
        .....
        ..... 248,249,250,251,252,253,254,255
    };

    for (i=0; i<256; i++) {
        pConf->GammaTableR[i] = gamma_adjust[i] << 2;
        pConf->GammaTableG[i] = gamma_adjust[i] << 2;
        pConf->GammaTableB[i] = gamma_adjust[i] << 2;
    }
}
```
- Gamma is controlled by:
`struct lcd_effect.gamma_cntl_port[LCD_GAMMA_EN] // 1: enable, 0: disable`

For example:

- Enable gamma:
`struct lcd_effect.gamma_cntl_port = (1<<LCD_GAMMA_EN) | ... ,`
- Disable gamma:
`struct lcd_effect.gamma_cntl_port = (0<<LCD_GAMMA_EN) | ... ,`

8. LCD power control

- LCD power control contains:
LCD backlight power on/off control
LCD power on/off control
LCD power on/off sequence
- LCD backlight power on/off control and LCD power on/off control are determined by hardware design
- LCD power on/off sequence are determined by panel specification. An example is shown in Figure 8.1 and 8.2.

a. Power on:

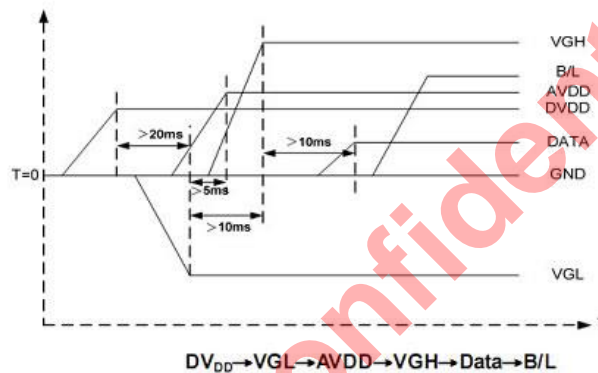


Figure 8.1 - LCD power on sequence

b. Power off:

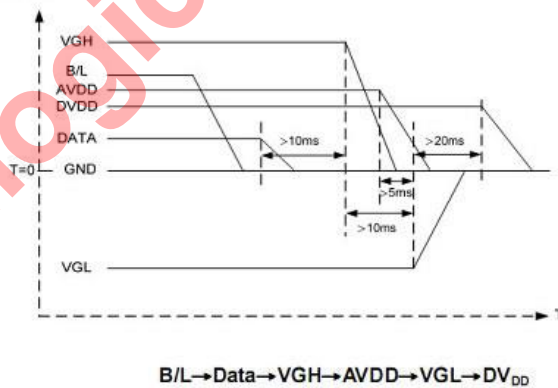


Figure 8.2 - LCD power off sequence

9. LCD online debug

AML8726-MX/MXL/MXS platform supports LCD timing and gamma online debug.

9.1 LCD timing online tuning

- There is a node in kernel sysfs directory `/sys/class/lcd/`, you can type serial commands as below to tune lcd configs:

`echo basic <h_active> <v_active> <h_period> <v_period> > debug`

Function: write lcd basic config

`echo type <lcd_type> <lcd_bits> > debug`

Function: write lcd type and bit width

`echo clock <lcd_clk> <ss_level> <clk_pol> > debug`

Function: write lcd clock config

`echo sync <hs_width> <hs_backporch> <hs_pol> <vs_width> <vs_backporch> <vs_pol> > debug`

Function: write lcd sync timing

Data format:

`<lcd_type>` : 1 for TTL, 2 for LVDS

`<lcd_bits>` : 6 for 6bit(RGB18bit), 8 for 8bit(RGB24bit)

`<lcd_clk>` : decimal numbers unit in Hz

`<ss_level>` : lcd clock spread spectrum level, 0~5, 0 for disable

`<xxx_pol>` : 0 for negative, 1 for positive

`echo ttl <rb_swap> <bit_swap> > debug`

Function: write TTL RGB swap config

`echo offset <h_sign> <h_offset> <v_sign> <v_offset> > debug`

Function: write TTL display offset

`echo lvds <lvds_repack> <pn_swap> > debug`

Function: write LVDS config

Data format:

`<xx_swap>` : 0 for normal, 1 for swap

`<xx_sign>` : 0 for negative, 1 for positive

`<lvds_repack>` : 0 for JEIDA mode, 1 for VESA mode

`echo write > debug`

Function: update LCD config

`echo reset > debug`

Function: reset LCD config to default

`echo read > debug`

Function: read LCD config

`echo disable > debug`

Function: power off LCD

`echo enable > debug`

Function: power on LCD

cat help

Function: look for help

- After set the LCD display timing, **DON'T FORGET** to type "*echo write > debug*" to validate it. For example:

```
cd /sys/class/lcd
```

```
echo basic 1024 600 1344 635 > debug
```

```
echo type 1 6 > debug
```

```
echo clock 42700000 0 0 > debug
```

```
echo sync 20 160 0 3 23 0 > debug
```

```
echo ttl 0 0 > debug
```

```
echo write > debug
```

Amlogic Confidential!

9.2 LCD gamma online debug

- There is a node in kernel sysfs directory `/sys/class/gamma/`, you can type serial commands as below to tune lcd gamma curve:

echo coeff <R_coeff> <G_coeff> <B_coeff> > write

Function: set R/G/B scale factor

Data format:

<R/G/B_coeff>: 0~100 in decimal representing 0%~100%

echo [r|g|b] <step> <value> <value> <value> <value> <value> <value> <value> <value> > write

Function: write R/G/B gamma table with grid values

Data format:

<step>: 4-bit hex, there are 8 steps (0~7, 8bit gamma) or 16 steps (0~f, 10bit gamma) for a single command

<value> : 32-bit hex, 2 (10-bit gamma) or 4 (8-bit gamma) values combined into a single <value>

echo w [0|8|10] > write

Function: update the original/8-bit/10-bit gamma table

echo f[r|g|b|w] <level_value> > write

Function: write R/G/B/White gamma table with fixed level_value

Data format:

<level_value>: 0~255 in decimal

echo [0|1] > read

Function: read the original/current gamma table

cat help

Function: look for help

- You can type “`echo coeff ... > /sys/class/gamma/write`” to change RGB gamma scale factors to adjust panel white balance. When you tune out a group of suitable RGB gamma scale factors, you can update them to gamma table in lcd driver. It is not recommended to use RGB gamma scale factors less than 90. For example:

Command: `echo coeff 96 98 100 > /sys/class/gamma/write`

Code in LCD driver:

```
static unsigned short r_coeff=96; g_coeff=98; b_coeff=100;
```

```
static void lcd_setup_gamma_table(Lcd_Config_t *pConf)
```

```
{
```

```
    int i;
```

```
    const unsigned short gamma_adjust[256] = {... };
```

```
    for (i=0; i<256; i++) {
```

```
        pConf->lcd_effect.GammaTableR[i] = (gamma_adjust[i]*r_coeff/100) << 2;
```

```
        pConf->lcd_effect.GammaTableG[i] = (gamma_adjust[i]*g_coeff/100) << 2;
```

```
        pConf->lcd_effect.GammaTableB[i] = (gamma_adjust[i]*b_coeff/100) << 2;
```

```
    }
```

}

- You can also tune RGB gamma table. It needs 24 (3*8) step for R/G/B 8-bit gamma table, or 48 (3*16) steps for R/G/B 10-bit gamma table. After tune the gamma table, **DON'T FORGET** to type “echo w[0/8/10] > /sys/class/gamma/write” to validate it. For example:

```
cd /sys/class/gamma/
```

```
echo r 0 00020406 080A0C0E 10111315 17191B1F 21222324 25262729 2A2B2C2D 2E2F3031 >write
```

```
.....
```

```
echo r 7 EAEAEBEB ECEDEDEE EFEFF0F1 F1F2F2F3 ..... FDFDFEFF >write
```

```
echo g 0 00020406 080A0C0E 10111315 17191B1F ..... 2E2F3031 >write
```

```
.....
```

```
echo g 7 EAEAEBEB ECEDEDEE EFEFF0F1 F1F2F2F3 ..... FDFDFEFF >write
```

```
echo b 0 00020406 080A0C0E 10111315 17191B1F ..... 2E2F3031 >write
```

```
.....
```

```
echo b 7 EAEAEBEB ECEDEDEE EFEFF0F1 F1F2F2F3 ..... FDFDFEFF >write
```

```
echo w 8 >write
```

10. LCD clock calculate

When the parameter *CLK_AUTO_GEN* setn to 0, you must manual setting the *pll_ctrl*, *div_ctrl* and *clk_ctrl* in *lcd config*, to generate the specified *lcd pixel clock*. Please refer to Chapter4.2.

10.1 TTL interface pixel clock

- TTL interface pixel clock is calculated in the below formula:

$$TTL_interface_pixel_clock = M * XTAL / (2^{OD} * (pre_div + 1) * XD)$$

<Note>:

- XTAL is the crystal frequency, which typically is 24M
- All the other four parameters are illustrated in Table 10.1

Struct	Member	Symbol	Range
<i>lcd_timing</i>	<i>pll_ctrl</i> [17:16]	OD	0~2
	<i>pll_ctrl</i> [8:0]	M	$750M < (M * 24 / N) < 1500M$
	<i>div_ctrl</i> [7:4]	pre_div	0~5
	<i>clk_ctrl</i> [3:0]	XD	1~15

Table 10.1 - TTL interface pixel clock parameter table

For example:

- Given:

```
struct lcd_timing.pll_ctrl=0x10220,
                .div_ctrl=0x18803,
                .clk_ctrl=0x1119,
```
- We have:
OD=1, M=32, pre_div=0, XD=9.
- So:
 $TTL_interface_pixel_clock = 32 * 24 / (2^{1} * (0 + 1) * 9) = 42.7MHz.$

- <Note>:

- | Struct | Member | Symbol | Range |
|-------------------|-----------------|---------|-------------------------------|
| <i>lcd_timing</i> | pll_ctrl[17:16] | OD | 0~2 |
| | pll_ctrl[8:0] | M | $750M < (M * 24 / N) < 1500M$ |
| | div_ctrl[7:4] | pre_div | 0~5 |

For example:

- Given:
 $struct\ lcd_timing.pll_ctrl=0x10232,$
 $.div_ctrl=0x18813,$
- We have:
OD=1, M=50, pre_div=1.
- So:
 $LVDS_interface_pixel_clock=50*24/(2*(1+1)*7)=42.9MHz.$

11. Example

- In Amlogic source code, panel driver directory is:
[uboot root]/customer/board/[board directory]/lcd.c
[kernel root]/customer/boards/board-m6xxx-panel.c
- They mainly contain 2 parts:
Power Control
Display Control

11.1 Power control

- LCD backlight power control

```

//*****
// Define backlight control method
//*****
#define BL_CTL_GPIO      0
#define BL_CTL_PWM      1
#define BL_CTL           BL_CTL_GPIO

//backlight controlled parameters in driver, define the real backlight level
#if (BL_CTL==BL_CTL_GPIO)
#define DIM_MAX          0x0
#define DIM_MIN          0xd
#elif (BL_CTL==BL_CTL_PWM)
#define PWM_CNT          600    //PWM_CNT <= 65535
#define PWM_PRE_DIV      0      //pwm_freq = 24M / (pre_div + 1) / PWM_CNT
#define PWM_MAX          (PWM_CNT * 100 / 100)
#define PWM_MIN          (PWM_CNT * 10 / 100)
#endif

//brightness level in Android UI menu
#define BL_MAX_LEVEL      255
#define BL_MIN_LEVEL      10
#define DEFAULT_BL_LEVEL  102    //please keep this value the same as uboot

```

- LCD power sequence

```

static void lcd_power_ctrl(Bool_t status)
{
    printk(KERN_INFO "%s() Power %s\n", __FUNCTION__, (status ? "ON" : "OFF"));
    if (status) {
        gpio_out(PAD_GPIOD_8, 0);    //reset signal to low
        gpio_set_status(PAD_GPIOD_8, gpio_status_out);
    }
}

```



```
gpio_out(PAD_GPIOA_27, 0);          // open LCD power
gpio_set_status(PAD_GPIOA_27,gpio_status_out);
mdelay (10);
gpio_out(PAD_GPIOD_8, 1); //reset signal to high
mdelay (20);

#ifdef CONFIG_AW_AXP
    axp_gpio_set_io(3,1);
    axp_gpio_set_value(3, 0);
#endif
    mdelay (20);

    ttl_ports_ctrl(ON);
    mdelay (200);
    data_status = status;
}
else {
    data_status = status;
    mdelay (30);
    ttl_ports_ctrl(OFF);
    mdelay (20);

#ifdef CONFIG_AW_AXP
    axp_gpio_set_io(3,0);
#endif
    mdelay (20);
    gpio_out(PAD_GPIOD_8, 0);
    //gpio_out(PAD_GPIOA_27, 1);
    gpio_set_status(PAD_GPIOA_27,gpio_status_in);
    mdelay (100); //power down sequence, needed
}
}
```

11.2 Display control

```
//*****
// Define LCD Timing Parameters
//*****
#define ACITVE_AREA_WIDTH 154 //unit: mm
#define ACITVE_AREA_HEIGHT 86 //unit: mm
#define LCD_TYPE LCD_DIGITAL_TTL //LCD_DIGITAL_TTL //LCD_DIGITAL_LVDS
#define LCD_BITS 6 //6 //8

#define H_ACTIVE 1024
#define V_ACTIVE 600
#define H_PERIOD 1344
#define V_PERIOD 635
//define FRAME_RATE 50
#define LCD_CLK 42700000//{(H_PERIOD * V_PERIOD * FRAME_RATE)//unit: Hz
#define CLK_SS_LEVEL 0 //0~5, 0 for disable spread spectrum
#define CLK_AUTO_GEN 1 //1, auto generate clk parameters //0, user set pll_ctrl, div_ctrl
& clk_ctrl

//modify below settings if needed
#define CLK_POL 0
#define HS_WIDTH 20
#define HS_BACK_PORCH 160 //include HS_WIDTH
#define HS_POL 0 //0: negative, 1: positive
#define VS_WIDTH 3
#define VS_BACK_PORCH 23 //include VS_WIDTH
#define VS_POL 0 //0: negative, 1: positive
#define TTL_RB_SWAP 0 //0: normal, 1: swap
#define TTL_RGB_BIT_SWAP 0 //0: normal, 1: swap

#define LVDS_REPACK 0 //lvds data mapping //0:JEIDA mode, 1:VESA mode
#define LVDS_PN_SWAP 0 //0:normal, 1:swap

//recommend settings, don't modify them unless there is display problem
#define TTL_H_OFFSET 0 //adjust ttl display h_offset
#define H_OFFSET_SIGN 1 //0: negative value, 1: positive value
#define TTL_V_OFFSET 0 //adjust ttl display v_offset
#define V_OFFSET_SIGN 1 //0: negative value, 1: positive value
#define VIDEO_ON_PIXEL 80
#define VIDEO_ON_LINE 32
//*****

static Lvds_Phy_Control_t lcd_lvds_phy_control = {
    .lvds_prem_ctl = 0x0,
```

```
.lvds_swing_ctl = 0x4,  
.lvds_vcm_ctl = 0x7,  
.lvds_ref_ctl = 0x15,  
};  
  
static Lvds_Config_t lcd_lvds_config = {  
.lvds_repack = LVDS_REPACK,  
.pn_swap = LVDS_PN_SWAP,  
};  
  
Lcd_Config_t lcd_config = {  
.lcd_basic = {  
.h_active = H_ACTIVE,  
.v_active = V_ACTIVE,  
.h_period = H_PERIOD,  
.v_period = V_PERIOD,  
.screen_ratio_width = ACITVE_AREA_WIDTH,  
.screen_ratio_height = ACITVE_AREA_HEIGHT,  
.screen_actual_width = ACITVE_AREA_WIDTH,  
.screen_actual_height = ACITVE_AREA_HEIGHT,  
.lcd_type = LCD_TYPE,  
.lcd_bits = LCD_BITS,  
},  
  
.lcd_timing = {  
.lcd_clk = LCD_CLK,  
//.pll_ctrl = 0x10232, //clk=42.9MHz, vfreq=50.2Hz  
//.div_ctrl = 0x18813, //[7:4]div  
//.clk_ctrl = 0x1111, //[19:16]ss_ctrl, [12]pll_sel, [8]div_sel, [4]vclk_sel, [3:0]xd  
.clk_ctrl = 0x1111 | (CLK_AUTO_GEN << CLK_CTRL_AUTO) | (CLK_SS_LEVEL << CLK_CTRL_SS),  
  
.video_on_pixel = VIDEO_ON_PIXEL,  
.video_on_line = VIDEO_ON_LINE,  
  
.hsync_width = HS_WIDTH,  
.hsync_bp = HS_BACK_PORCH,  
.vsync_width = VS_WIDTH,  
.vsync_bp = VS_BACK_PORCH,  
  
.pol_cntl_addr = (CLK_POL<<LCD_CPH1_POL) | (HS_POL<<LCD_HS_POL) | (VS_POL<<LCD_VS_POL),  
.inv_cnt_addr = (0<<LCD_INV_EN) | (0<<LCD_INV_CNT),  
.tcon_misc_sel_addr = (1<<LCD_STV1_SEL) | (1<<LCD_STV2_SEL),  
.dual_port_cntl_addr = (TTL_RB_SWAP<<LCD_RGB_SWP) | (TTL_RGB_BIT_SWAP<<LCD_BIT_SWP),  
},  
};
```

```
.lcd_effect = {
    .gamma_cntl_port = (1<<LCD_GAMMA_EN) | (0<<LCD_GAMMA_RVS_OUT) |
        (1<<LCD_GAMMA_VCOM_POL),
    .gamma_vcom_hswitch_addr = 0,
    .rgb_base_addr = 0xf0,
    .rgb_coeff_addr = 0x74a,
},

.lvds_mlvds_config = {
    .lvds_config = &lcd_lvds_config,
    .lvds_phy_control = &lcd_lvds_phy_control,
},

.lcd_power_ctrl = {
    .cur_bl_level = 0,
    .power_ctrl = lcd_power_ctrl,
    .backlight_ctrl = backlight_power_ctrl,
    .get_bl_level = get_backlight_level,
    .set_bl_level = set_backlight_level,
    .lcd_suspend = lcd_suspend,
    .lcd_resume = lcd_resume,
},
};

static void lcd_setup_gamma_table(Lcd_Config_t *pConf)
{
    int i;

    const unsigned short gamma_adjust[256] = {
        0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,
        32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,
        .....,
        .....,250,251,252,253,254,255
    };

    for (i=0; i<256; i++) {
        pConf->lcd_effect.GammaTableR[i] = gamma_adjust[i] << 2;
        pConf->lcd_effect.GammaTableG[i] = gamma_adjust[i] << 2;
        pConf->lcd_effect.GammaTableB[i] = gamma_adjust[i] << 2;
    }
}
```