neoway

Get connected Get smart

S726

LCD Commissioning Guide

Issue 1.0

Notice

This document provides guide for users to use S726.

This document is intended for system engineers (SEs), development engineers, and test engineers.

THIS GUIDE PROVIDES INSTRUCTIONS FOR CUSTOMERS TO DESIGN THEIR APPLICATIONS. PLEASE FOLLOW THE RULES AND PARAMETERS IN THIS GUIDE TO DESIGN AND COMMISSION. NEOWAY WILL NOT TAKE ANY RESPONSIBILITY OF BODILY HURT OR ASSET LOSS CAUSED BY IMPROPER OPERATIONS.

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE DUE TO PRODUCT VERSION UPDATE OR OTHER REASONS.

EVERY EFFORT HAS BEEN MADE IN PREPARATION OF THIS DOCUMENT TO ENSURE ACCURACY OF THE CONTENTS, BUT ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS DOCUMENT DO NOT CONSTITUTE A WARRANTY OF ANY KIND, EXPRESS OR IMPLIED.

Neoway provides customers complete technical support. If you have any question, please contact your account manager or email to the following email addresses:

Sales@neoway.com

Support@neoway.com

Website: http://www.neoway.com

Contents

About This Document

Scope

This document is applicable to the S726 series.

Audience

This document is intended for system engineers (SEs), development engineers, and test engineers.

Change History

Issue	Change	Changed By
1.0	Initial draft	Leo Shen

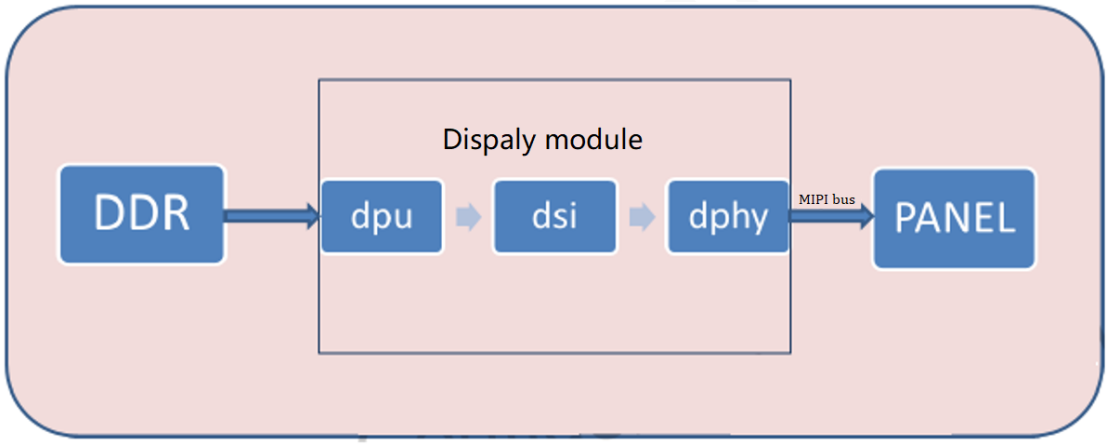
Conventions

Symbol	Indication
	Indicates danger or warning. This information must be followed. Otherwise, a catastrophic module or user device failure or bodily injury may occur.
	Indicates caution. This symbol alerts the user to important points about using the module. If these points are not followed, the module or user device may fail.
	Indicates instructions or tips. This symbol provides advices or suggestions that may be useful when using the module.

1 LCD Architecture Overview

- Platform: UIS8581E
- System: Android 10
- LCD: HX8399C, 1920x1080, RGB888, MIPI-4lane
- Backlight: PWM + backlight boost chip

The display driver module mainly consists of DPU, DSI, and DPHY. The core module DPU is related to display and is mainly responsible for layer composition, image transformation, and screen refresh. The DSI and DPHY are related to communication and are responsible for packaging and transmitting data. The upper-level program draws the image to be displayed and writes it to the DDR buffer, and then the DPU retrieves the data from the buffer, synthesizes it, and sends it to the DSI for packaging. The DPHY sends the packaged data to the panel through the MIPI bus for display, as shown below.



The LCD driver addition process consists of two parts: u-boot and kernel.

2 u-boot

2.1 Adding the LCD Driver File

Step 1: Copy the existing LCD driver file in the following directory:

bsp\bootloader\uboot15\drivers\video\sprd\lcd\

Step 2: Rename the LCD driver file according to the following naming convention. The format is as follows:

lcd_[DriverIC]_[ModuleVendor]_[BusType]_[Resolution].c
lcd_hx8399c_boe_mipi_fhd_nwy.c

Step 3: Modifying the LCD Driver File

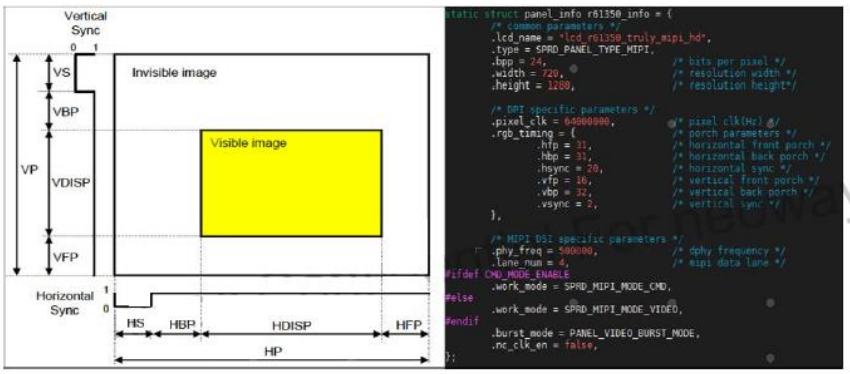
1. Replace all ID-related functions and variable names in the driver file with hx8399c.
2. Configure the initialization code (provided by the module factory) using the following format:

```
static uint8_t init_data[] = {
        //Host I/F Setting
        0x15, 0x00, 0x00, 0x02, 0xB0, 0x00,
        0x15, 0x00, 0x00, 0x02, 0xCC, 0x04,
        0x15, 0x00, 0x00, 0x02, 0xE3, 0x01,

        //DSI control
        0x39, 0x00, 0x00, 0x03, 0xB6, 0x61,

        //display setting
        0x39, 0x00, 0x00, 0x07, 0xC0, 0x23,

        //display h-timing setting
        0x39, 0x00, 0x00, 0x17, 0xC1, 0x0B,
```

the initialization code in the mipi_dsi_send_cmds function will forcibly converts to dsi_cmd_desc structure.

```
struct dsi_cmd_desc {
        uint8_t data_type;  // type of the mipi packet
        uint8_t wait;       // timeout period (ms)
        uint8_t wc_h;       // Last 4 bits of the packet length
        uint8_t wc_l;       // First 4 bits of the packet length
        uint8_t payload[];  // Data
};
```

Step 4: Configure the front and rear porch parameters (the panel_info structure and parameters are provided by the panel manufacturer), as shown below.




```
static struct panel_info r61350_info = {
        /* common parameters */
        .lcd_name = "lcd_r61350_truly_mipi_hd",
        .type = SPRD_PANEL_TYPE_MIPI,
        .bpp = 24,                      /* bits per pixel */
        .width = 720,                   /* resolution width */
        .height = 1280,                 /* resolution height*/

        /* DPI specific parameters */
        .pixel_clk = 64000000,          /* pixel clk(Hz) */
        .rgb_timing = {                 /* porch parameters */
                .hfp = 31,              /* horizontal front porch */
                .hbp = 31,              /* horizontal back porch */
                .hsync = 20,            /* horizontal sync */
                .vfp = 16,              /* vertical front porch */
                .vbp = 32,              /* vertical back porch */
                .vsync = 2,             /* vertical sync */
        },

        /* MIPI DSI specific parameters */
        .phy_freq = 500000,             /* dphy frequency */
        .lane_num = 4,                  /* mipi data lane */
#ifdef CMD_MODE_ENABLE
        .work_mode = SPRD_MIPI_MODE_CMD,
#else
        .work_mode = SPRD_MIPI_MODE_VIDEO,
#endif
        .burst_mode = PANEL_VIDEO_BURST_MODE,
        .nc_clk_en = false,
};
```

lcd_name is the node name. The lcd dts node name in u-boot and the kernel must be the same, otherwise the kernel cannot match the correct LCD driver.

Step 5: Implement the panel_ops callback function, as shown below.

```
static struct panel_ops hx8399c_ops = {
    .init = hx8399c_init,
    .read_id = hx8399c_readid,
    .power = hx8399c_power,
 };
```

Step 6: Register the panel driver, as shown below.

```
struct panel_driver hx8399c_boe_driver = {
    .info = &hx8399c_info,
    .ops = &hx8399c_ops,
 };
```

2.2 Adding Macro Control for LCD Compilation

After adding and configuring the LCD driver file, you can open the added new LCD module in the board configuration file.

Take the S726 project as an example. Find the configuration header file of the corresponding project in u-boot, and add the LCD compilation macro control:

```
bsp\bootloader\u-boot15\include\configs\S726.h
#define CONFIG_LCD_HX8399C_BOE_MIPI_FHD
```

2.3 Adding LCD Compilation Rules

Step 1: Modify the LCD Makefile:

```
u-boot15/drivers/video/sprd/lcd/Makefile
 obj-$(CONFIG_LCD_HX8399C_BOE_MIPI_FHD) += lcd_hx8399c_boe_mipi_fhd_nwy.o
```

Step 2: Modify the LCD configuration file.

```
// Configure the file path
bsp\bootloader\u-boot15\drivers\video\sprd\lcd\panel_cfg.h
// add the Entrance instructions for the LCD driver
```

```
 #ifdef CONFIG_LCD_HX8399C_BOE_MIPI_FHD
 extern struct panel_driver hx8399c_boe_driver;
 #endif
// Add reference
static struct panel_cfg supported_panel[] = {
 #ifdef CONFIG_LCD_HX8399C_BOE_MIPI_FHD
    {
        .lcd_id = 0x8399,
        .drv = &hx8399c_boe_driver,
    },
 #endif
...
}
```

- lcd_id: built-in lcd id, which is used to read the ID to confirm the model.
- drv: pointer to the LCD panel driver structure, which points to the driver structure. Associate the driver structure with supported_panel. During startup, u-boot enumerates the display in this structure until the readid function of a driver returns success.

3 Modifying the Kernel

The configuration of the LCD kernel adopts the new display framework DRM. In the new Android version, the framework is gradually migrated from ADF to DRM. Starting from the kernel 4.14, the display framework is gradually migrated to DRM. The panel is now customized through LCD DTS, the unified SPRD generic panel driver is used, and the modification of the customized driver is currently not supported.

3.1 Adding the LCD Driver File

1. Copy the existing driver file in the following directory:

   bsp\kernel\kernel4.14\arch\arm64\boot\dts\lcd\

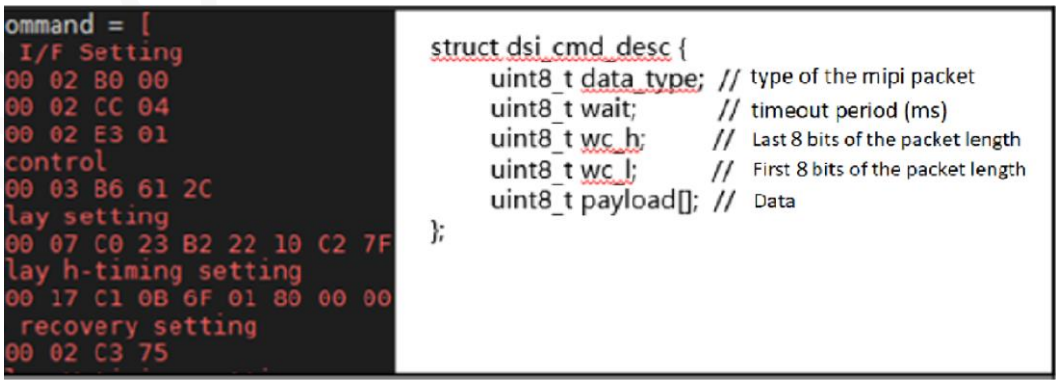
2. Rename the LCD driver file according to the following naming convention.

   For example:
   lcd_[DriverIC]_[ModuleVendor]_[BusType]_[Resolution].dtsi
   lcd_boe_hx8399c_mipi_fhd_nwy.dtsi

3.2 Modifying the LCD Driver File

Step 1: Rename all the original LCD related names in the old DTSI file. Replace them with xxx_hx8399c_xxx.

Step 2: Configure the initialization code (provided by the module factory) using the following format:



Step 3: Configure the DTS attributes.

The reference settings are shown below, and the parameters should be consistent with the configuration on the uboot. The parameters listed in the following table must be configured. Other parameters can be configured according to the actual situation, or you can consult Neoway technical support.

Properties	Description	Remarks
lcd_hx8399c_boe_mipi_fhd	LCD node name	It must be consistent with the lcd name in u-boot, otherwise the kernel cannot match the correct LCD driver.
sprd,dsi-work-mode	DSI mode used by the LCD to transmit image data. Currently, the following values are supported: **0**: cmd mode **1**: video burst mode (Default) 2: video non-burst mode with sync pulse(2) 3: video non-burst mode with sync event(3)	Confirm the DSI mode based on the LCD specifications. Most LCDs use the video mode.
sprd,dsi-lane-number	Total number of data lanes when the LCD receives the transmitted video data. The value ranges from 1 to 4.	Confirm the DSI mode based on the LCD specifications.
sprd,dsi-color-format	Data format used for video stream transmission. Currently, the following values are supported: **rgb888**, **rgb666**, **rgb666_packed**, **rgb565**, and **dsc**. The default format **rgb888**.	Confirm the setting based on the LCD specifications.
sprd,phy-bit-clock	Bit clock, that is, total LCD data clock	It is calculated and filled by the timing table. For details, see Chapter xx.
sprd,reset-on-sequence	Reset sequence before the LCD is turned on	Confirm the setting based on the LCD specifications. The sequence is usually high, low, and high.
sprd,reset-off-sequence	Reset sequence before the LCD is turned off	Confirm the DSI mode based on the LCD specifications. The sequence is usually constant low.
sprd,initial-command	LCD initialization code	It is provided by the module factory and filled based on the format. For details, see section 3.2.

sprd,sleep-in-command	Code that enables the LCD to enter the sleep mode	Provided by the module manufacturer
sprd,sleep-out-command	Code that enables the LCD to exit the sleep mode	Provided by the module manufacturer
display-timings	Configuration of the MIPI clock, LCD width and height, and front and rear porch parameters	Confirm the setting based on the LCD specifications.

3.3 Referencing the LCD Driver File

Locate and open the board configuration header file, and associate the LCD driver file that needs to be adapted with the board through **#include**.

The file path is as follows:

bsp\kernel\kernel4.14\arch\arm64\boot\dts\sprd\S726_uis8581e5h10.dts

Add the LCD header file.

```
#include "uis8581e.dtsi"
#include "sc9863a-wcn.dtsi"
#include "sp9863a-mach-S726.dtsi"
#include "lcd/lcd_hx8399c_boe_mipi_fhd_nwy.dtsi"
```

The panel driver will search for the corresponding LCD child node under the **/lcds** node based on the lcd_name uploaded by uboot, so as to complete LCD matching.

```
&dsi {
    status = "okay";
    #address-cells = <1>;
    #size-cells = <0>;
    panel: panel {
        compatible = "sprd,generic-mipi-panel";
        #address-cells = <1>;
        #size-cells = <0>;
        reg = <0>;
        reset-gpio = <&ap_gpio 50 GPIO_ACTIVE_HIGH>;
        sprd,force-attached = "lcd_hx8399c_boe_mipi_fhd";
        port {
            reg = <1>;
            panel_in: endpoint {
                remote-endpoint = <&dphy_out>;
            };
        };
    };
};
```


4 Timing Description

Timing-related parameters, including resolution, front and rear porches, pixel clock, physical clock, are all related to the display module. Therefore, after replacing the display module with the new one, if the resolution changes, make sure to notify Neoway technical support.

Timing involves modifications of uboot and kernel. Taking the S726 project as an example:

- IC: LCD boe hx8399c
- Resolution: 1920x1080
- Bits per Pixel (BPP): RGB888/16.7K color
- MIPI: Dsi-4lane

4.1 Generating Timing

Open the Timing calculation table **33340_unisoclcdmipifps-phy_feq calculation_c_v1.05.xlsm**, fill in the parameters such as resolution, front porch, and rear porch, select SC9832E as the platform, and then adjust the front and rear porches until the desired frame rate is around 60. Note: If there is a problem with the display, increase or decrease the frame rate appropriately.

- hfp = 16
- hbp = 16
- hsync = 16
- vfp = 10
- vbp = 10
- vsync = 8

Fill the driver parameters such as resolution, front porch, and rear porch in the table, which are marked in read in the figure below.

width	hfp	hbp	hsync	W
1080	16	16	16	1128
height	1920			
vfp	10			
vbp	10		fps(期望帧率)	60
vsync	8		lan_number	4
H	1948		product	SC9832E

计算结果：

fps(实际帧率)	58	58.252	.hfp	16	.vfp	10
phy_feq	921	*1000	.hbp	16	.vbp	10
pixel_clk	128000000		.hsync	16	.vsync	8

The following two parameters are obtained based on calculation of the table:

- phy_feq = 924*1000 // the unit in the code is Kbps.
- pixel_clk = 128000000 // the unit in the code is bps

Then fill two clk values into uboot and kernel to verify the effect. The parameter correspondence is as follows:

uboot:

```
static struct panel_info hx8399c_info = {
    /* common parameters */
    .lcd_name = "lcd_hx8399c_boe_mipi_fhd",
    .type = SPRD_PANEL_TYPE_MIPI,
    .bpp = 24,
    .width = 1080,
    .height = 1920,
    /* DPI specific parameters */
    .pixel_clk = 128000000,
    .rgb_timing = {
        .hfp = 16,
        .hbp = 16,
```

```
        .hsync = 16,
        .vfp = 10,
        .vbp = 10,
        .vsync = 8,
    },
    /* MIPI DSI specific parameters */
    .phy_freq = 921000, /*Kbps*/
    .lane_num = 4,
    .work_mode = SPRD_MIPI_MODE_VIDEO,
    .burst_mode = PANEL_VIDEO_BURST_MODE,
    .nc_clk_en = false,
};
```

kernel:

```
/ { lcds {
        lcd_hx8399c_boe_mipi_fhd: lcd_hx8399c_boe_mipi_fhd {
        sprd,dsi-work-mode = <1>; /* video burst mode */
        sprd,dsi-lane-number = <4>;
        sprd,dsi-color-format = "rgb888";
        sprd,phy-bit-clock = <921000>;  /* kbps */
...
        display-timings {
            timing0 {
                clock-frequency = <128000000>;
                hactive = <1080>;
                vactive = <1920>;
                hfront-porch = <16>;
                hback-porch = <16>;
                hsync-len = <16>;
                vfront-porch = <10>;
                vback-porch = <10>;
                vsync-len = <8>;
            };
        };
```

4.2 Confirming Timing

After the above two timing clk values are filled, compiled and configured, confirm timing based on actual_fps after startup.

Criterion: The actual frame rate in the table should be consistent with that obtained through **cat** here.

```
adb root
adb shell
cat /sys/class/display/dispc0/actual_fps
```