StarFive 赛昉科技

JH7110 MIPI LCD Developing and Porting Guide

VisionFive 2

Version: 1.1

Date: 2023/06/09

Doc ID: JH7110-PGEN-004

Legal Statements

Important legal notice before reading this documentation.

PROPRIETARY NOTICE

Information in this document is provided "as is," with all faults. Contents may be periodically updated or revised due to product development. Shanghai StarFive Technology Co., Ltd. (hereinafter "StarFive") reserves the right to make changes without further notice to any products herein.

StarFive expressly disclaims all warranties, representations, and conditions of any kind, whether express or implied, including, but not limited to, the implied warranties or conditions of merchantability, fitness for a particular purpose, and non-infringement.

StarFive does not assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation indirect, incidental, special, exemplary, or consequential damages.

Contact Us

Address: Room 502, Building 2, No. 61 Shengxia Rd., China (Shanghai) Pilot Free Trade Zone, Shanghai, 201203, China

Website: http://www.starfivetech.com

Email:

- Sales: sales@starfivetech.com
- Support: support@starfivetech.com

Contents

List of Tables

List of Figures

Preface

About this guide and technical support information.

About this document

This document mainly provides the SDK developers with the developing and porting instructions for the LCD module of the StarFive next generation SoC platform - JH7110.

Audience

This document mainly serves the LCD relevant driver developers. If you are developing and porting other modules, place a request to your sales or support consultant for our complete documentation set on JH7110.

Revision History

Table 0-1 Revision History

Version	Released	Revision
1.1	2023/06/09	Added MIPI Parameter Configuration *(on page 22)*.
1.0	2023/03/10	First official release.

Notes and notices

The following notes and notices might appear in this guide:

- **Tip:**
  Suggests how to apply the information in a topic or step.
- **Note:**
  Explains a special case or expands on an important point.
- **Important:**
  Points out critical information concerning a topic or step.

- **CAUTION:**
  Indicates that an action or step can cause loss of data, security problems, or performance issues.

- **Warning:**
  Indicates that an action or step can result in physical harm or cause damage to hardware.

1. Introduction

This document is intended to:

- Introduce the porting procedures of the StarFive JH7110 Software Development Kit (SDK) to a new platform.
- Provide instructions for porting a new LCD screen to the JH7110 SoC platform.
- Instruct on how to write an LCD driver.
- Provide an example of a typical LCD interface configuration.

The code sources referenced in this document are based on the following conditions:

- SDK version: 3.0
- U-Boot version: 3.0
- Linux Kernel version: 5.15
- For different U-Boot or Linux Kernel versions, these references may be slightly different.

2. IC Specification

The StarFive JH7110 LCD module uses Cadence IP. The following is the information about Cadence IP.

MIPI DSITX CADENCE

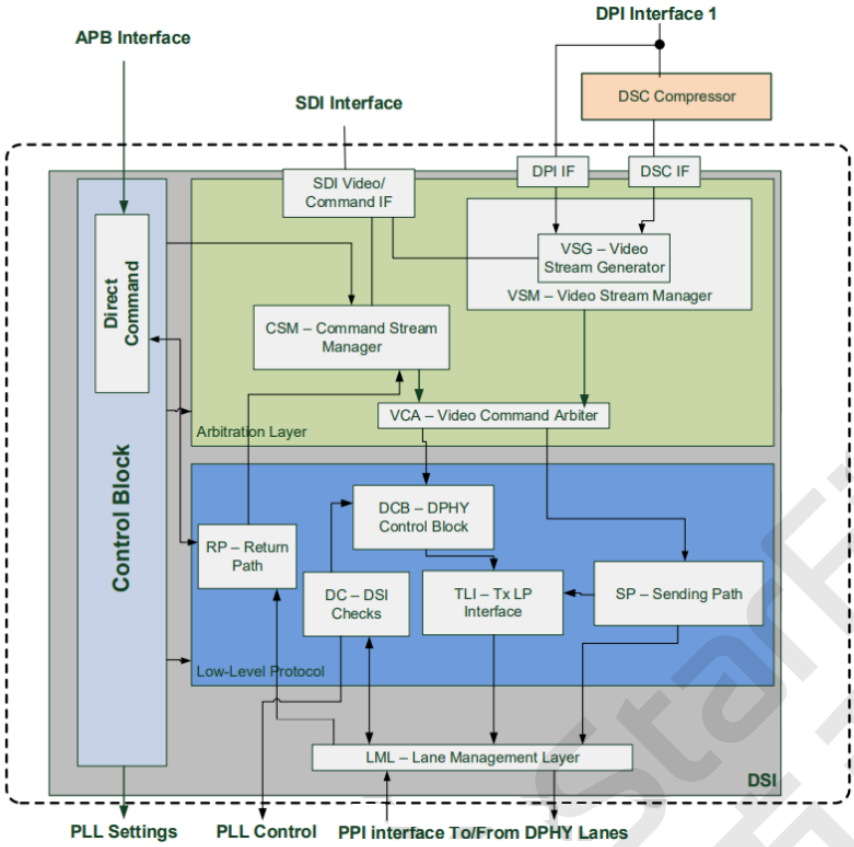
The Cadence MIPI DSI v1.3.1 TX Controller IP (DSITX) provides an interface that receives data and control from the host processor display system using either the DPI, DSC or SDI input bus interfaces. The DSITX will translate the incoming pixel information and control signals into an internal packed byte format, in the case of DPI and DSC, or pass in the pre-packed SDI byte format, before the internal byte format data is packeted and sent to the MIPI DSI Compatible display via the D-PHY physical interface. It supports video and command mode displays and can work in dual-display mode using virtual channel identification on the packets.

The supported MIPI display types are (at least) type 1 (command only), type 2 (video plus a partial-frame buffer in command mode), type 3 (video display with some programming capability in command mode).

In video mode, all the regular modes are supported (Non-Burst with Sync Pulses, Non-Burst with Sync Events and same for Burst mode).

Command mode is used for any panel that integrates a display controller and frame buffer. Data is passed to the display using a command message followed by data pixels and/or parameter messages. The host side can also perform read and write to the panel registers and frame buffer using the bidirectional lane on the DPHY.

Figure 2-1 Block Digram of DSITX



MIPI DPHY M31

MIPI DPHY M31 has the following features:

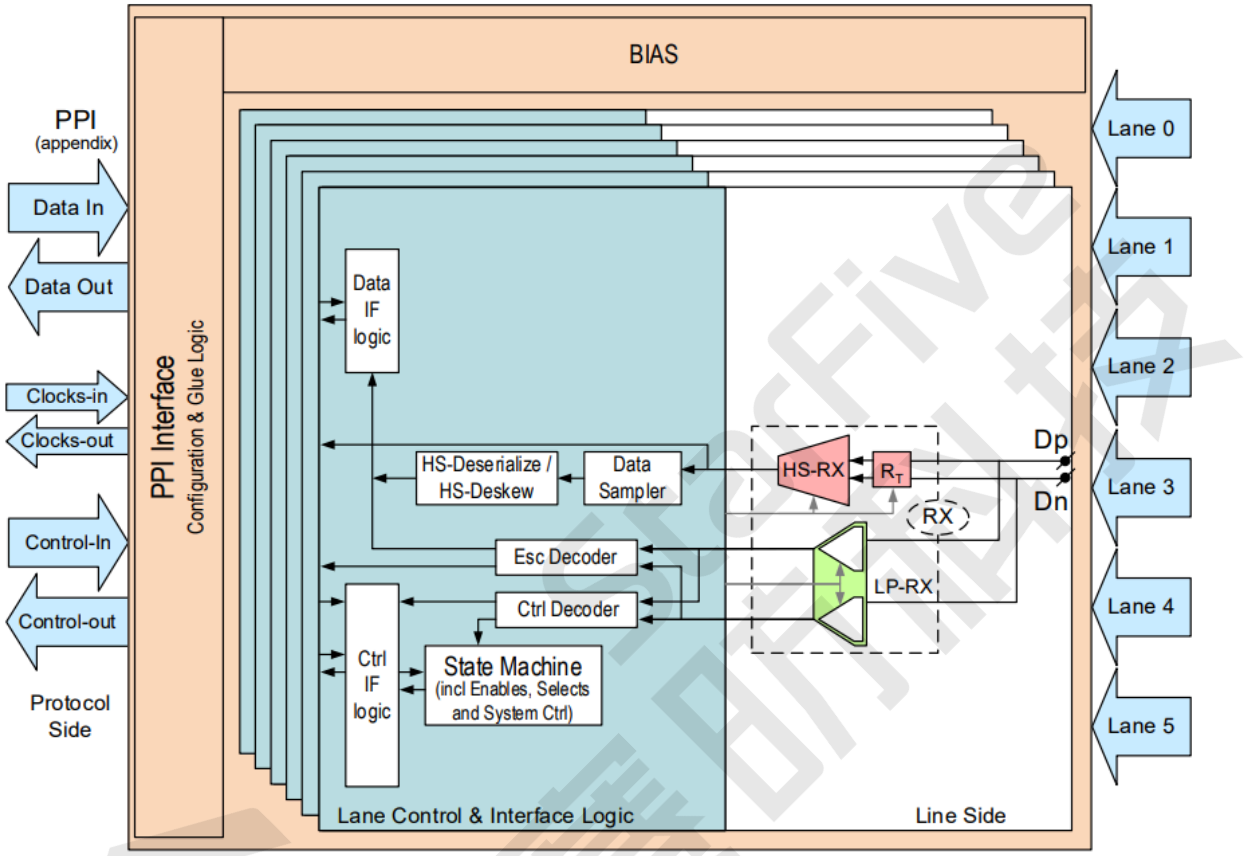
- Supports standard 8b PPI interface compliant of MIPI D-PHY Specification.
- Supports 1 Clock Lane and up to 4 Data Lanes scalability in DPHY mode.
- Supports independent (1 Clock Lane & up to 2 Data Lanes) X 2 in DPHY mode.
- Supports HS-Rx Data rate from 80Mbps up to 1.5 Gbps. (DPHY)
- Supports LS-Rx Data rate of 10Mbps & Ultra-low power mode.
- Supports Triggers, ULPS and LPDT.
- Supports on-die terminated and non-terminated operation, integrates switchable termination.
- Supports Fault Detection of Sequence Error. (Error Report)
- Supports clock and data lane swapping function.
- Build-in BISTTX for at-speed testing.

IP Overview

M31DPHYRX611TL028D_00151501 integrates 2 Clock Lane and up to 4 Data Lanes MIPI D-PHY compatible high-speed receiver that supports 80 Mbps up to 1.5 GHz data transfer rate, and low-

power receiver that supports data transfer in 10 Mbps. This IP supports D-PHY v1.1 specification. It is optimized with low power design for mobile CSI-2/DSI slave application. The interface of M31D-PHY is designed for standard MIPI D-PHY PPI interface. It can be easily integrated into the lane management of each customer's CSI-2 controller. The following is the block digram of IP.

Figure 2-2 Block Digram of M31D-PHY



3. Module Introduction

3.1. Display Driver

3.1.1. Display Driver Locations

The following list shows the address of the display drivers.

- Linux Kernel Display Driver:

```
linux-5.15/linux/drivers/gpu/drm/verisilicon
```

- Device tree:

```
linux-5.15/arch/riscv/boot/dts/starfive/jh7110.dtsi
linux-5.15/arch/riscv/boot/dts/starfive/jh7110-common.dtsi
```

3.1.2. Device Tree configuration

A DTS/DTSI file is used to store all the device tree configuration.

The device tree of MIPI LCD is stored in the following path:

```
linux-5.10/arch/riscv/boot/dts/starfive/
```

The following code block shows the DTS file structure for MIPI LCD.

```
linux-5.15.0
└-- arch
└-- | -- riscv
| -- | -- | -- boot
| -- | -- | -- | -- dts
| -- | -- | -- | -- | -- starfive
| -- | -- | -- | -- | -- | -- jh7110-common.dtsi
| -- | -- | -- | -- | -- | -- jh7110.dtsi
```

MIPI DSI

In the file `jh7110.dtsi`, you can find the device tree configuration of MIPI DSI as the following code block:

```
linux/arch/riscv/boot/dts/starfive/jh7110.dtsi:
   mipi_dsi: mipi@295d0000 {
   compatible = "starfive,jh7110-mipi_dsi","cdns,dsi";
```

```
    reg = <0x0 0x295d0000 0x0 0x10000>;
    interrupts = <98>;
    reg-names = "dsi";
    clocks = <&clkvout JH7110_U0_CDNS_DSITX_CLK_SYS>,
      <&clkvout JH7110_U0_CDNS_DSITX_CLK_APB>,
      <&clkvout JH7110_U0_CDNS_DSITX_CLK_TXESC>,
      <&clkvout JH7110_U0_CDNS_DSITX_CLK_DPI>;
    clock-names = "sys", "apb", "txesc", "dpi";
    resets = <&rstgen RSTN_U0_CDNS_DSITX_DPI>,
      <&rstgen RSTN_U0_CDNS_DSITX_APB>,
      <&rstgen RSTN_U0_CDNS_DSITX_RXESC>,
      <&rstgen RSTN_U0_CDNS_DSITX_SYS>,
      <&rstgen RSTN_U0_CDNS_DSITX_TXBYTEHS>,
      <&rstgen RSTN_U0_CDNS_DSITX_TXESC>;
    reset-names = "dsi_dpi", "dsi_apb", "dsi_rxesc",
      "dsi_sys", "dsi_txbytehs", "dsi_txesc";
    phys = <&mipi_dphy>;
                         phy-names = "dphy";
    status = "disabled";

    port {
     dsi_out_port: endpoint@0 {
      remote-endpoint = <&panel_dsi_port>;
     };
     dsi_in_port: endpoint@1 {
      remote-endpoint = <&mipi_out>;
     };
    };

    mipi_panel: panel@0 {
     /*compatible = "";*/
     status = "okay";
    };
   };

linux/arch/riscv/boot/dts/starfive/jh7110-common.dtsi:

&mipi_dsi {
 status = "okay";
};
```

The following list provides explanations for the parameters included in the above code block.

- **compatible**: Compatibility information, used to associate the driver and its target device.
- **reg**: Register base address "`0x295d0000`" and range "`0x10000`".
- **interrupts**: Hardware interrupt ID.
- **reg-name**: The name of the above register.

- **clocks**: The clocks used by the LCD module.
- **clock-names**: The names of the above clocks.
- **resets**: The reset signals used by the LCD module.
- **reset-names**: The names of the above reset signals.
- **phys**: The phys used by the LCD module.
- **phy-names**: The name of the phys.
- **status**: The work status of the LCD module. To enable the module, set this bit as "`okay`" or to disable the module, set this bit as "`disabled`".
- **port**: The port(s) used by the LCD driver.

MIPI DPHY

In the file `jh7110.dtsi`, you can find the device tree configuration of MIPI DPHY as the following code block:

```
linux/arch/riscv/boot/dts/starfive/jh7110.dts:
  mipi_dphy: mipi-dphy@295e0000{
   compatible = "starfive,jh7110-mipi-dphy-tx","m31,mipi-dphy-tx";
   reg = <0x0 0x295e0000 0x0 0x10000>;
   clocks = <&clkvout JH7110_U0_MIPITX_DPHY_CLK_TXESC>;
   clock-names = "dphy_txesc";
   resets = <&rstgen RSTN_U0_MIPITX_DPHY_SYS>,
     <&rstgen RSTN_U0_MIPITX_DPHY_TXBYTEHS>;
   reset-names = "dphy_sys", "dphy_txbytehs";
   #phy-cells = <0>;
   status = "disabled";
  };
linux/arch/riscv/boot/dts/starfive/jh7110-common.dts:
&mipi_dphy {
 status = "okay";
};
```

The following list provides explanations for the parameters included in the above code block.

- **compatible**: Compatibility information, used to associate the driver and its target device.
- **reg**: Register base address "`0x295e0000`" and range "`0x10000`".
- **clocks**: The clocks used by the LCD module.
- **clock-names**: The names of the above clocks.
- **resets**: The reset signals used by the LCD module.

- **reset-names**: The names of the above reset signals.
- **status**: The work status of the LCD module. To enable the module, set this bit as "`okay`" or to disable the module, set this bit as "`disabled`".

I2C2

In the file `jh7110-common.dtsi`, in order to configure LCD DTS port, the `seeed_panel dts` port should be added into `i2c2`. You can find the device tree configuration of i2c2 as the following code block:

```
linux/arch/riscv/boot/dts/starfive/jh7110-common.dts:
&i2c2 {
 clock-frequency = <100000>;
 i2c-sda-hold-time-ns = <300>;
 i2c-sda-falling-time-ns = <510>;
 i2c-scl-falling-time-ns = <510>;
 auto_calc_scl_lhcnt;
 pinctrl-names = "default";
 pinctrl-0 = <&i2c2_pins>;
 status = "okay";

 seeed_plane_i2c@45 {
  compatible = "seeed_panel";
  reg = <0x45>;

  port {
   panel_dsi_port: endpoint {
    remote-endpoint = <&dsi_out_port>;
   };
  };
 };

};
```

In the above code block, the parameters of **pinctrl-names** and **pinctrl-0** are used to configure the i2c2 pin configuration settings.

3.1.3. Driver Configuration

The following code block shows the driver configuration.

```
CONFIG_DRM_VERISILICON=y
```

3.1.4. Kernel Menu Configuration

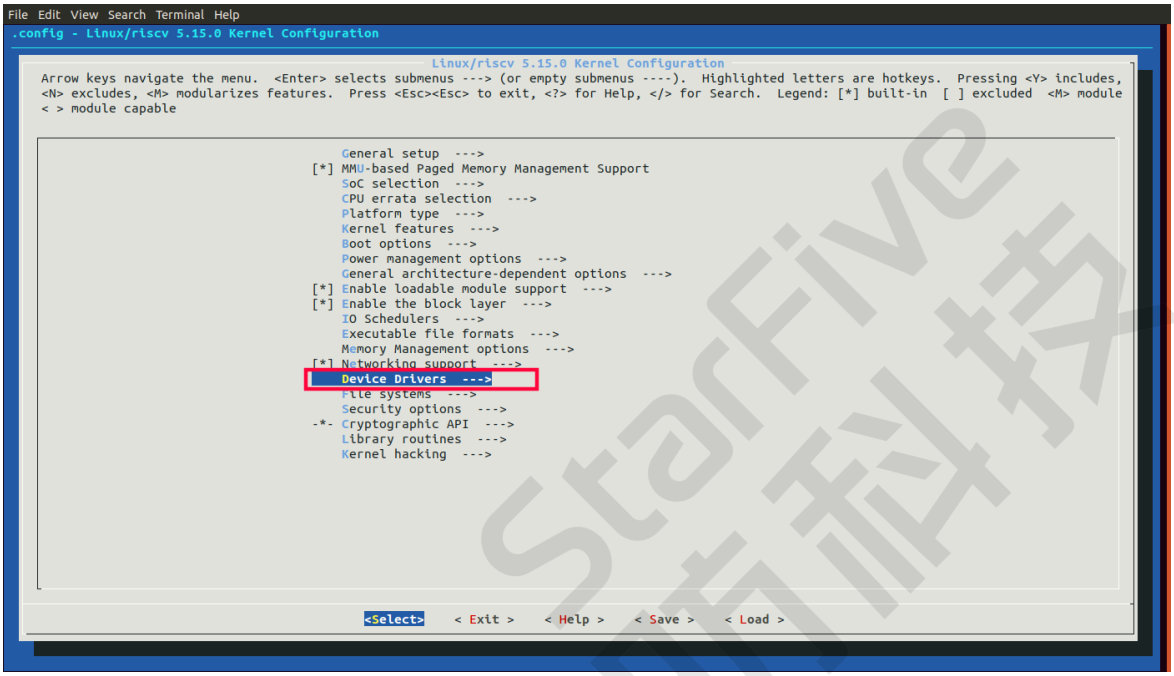
Follow the steps below to enable the kernel configuration for LCD.

1. Under the root directory of `freelight-u-sdk`, type the following command to enter the kernel menu configuration GUI.

```
make linux-menuconfig
```

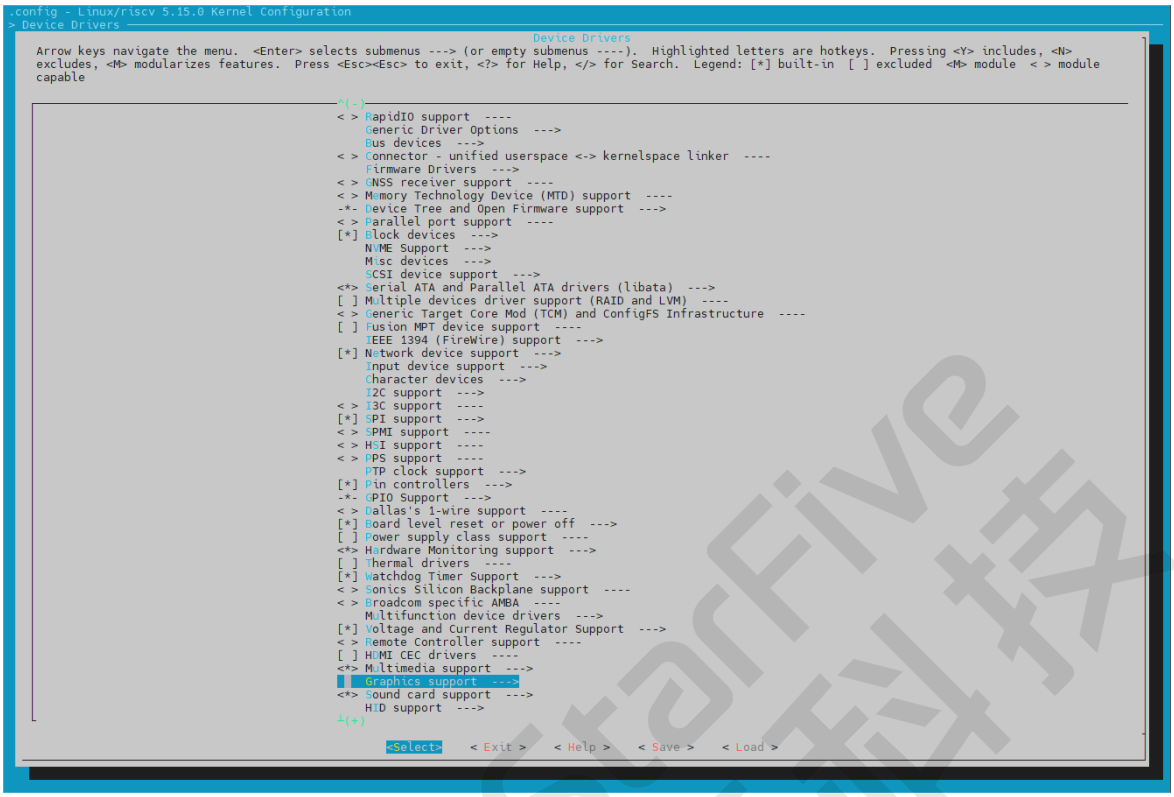
2. Enter the **Device Drivers** menu.

Figure 3-1 Device Drivers



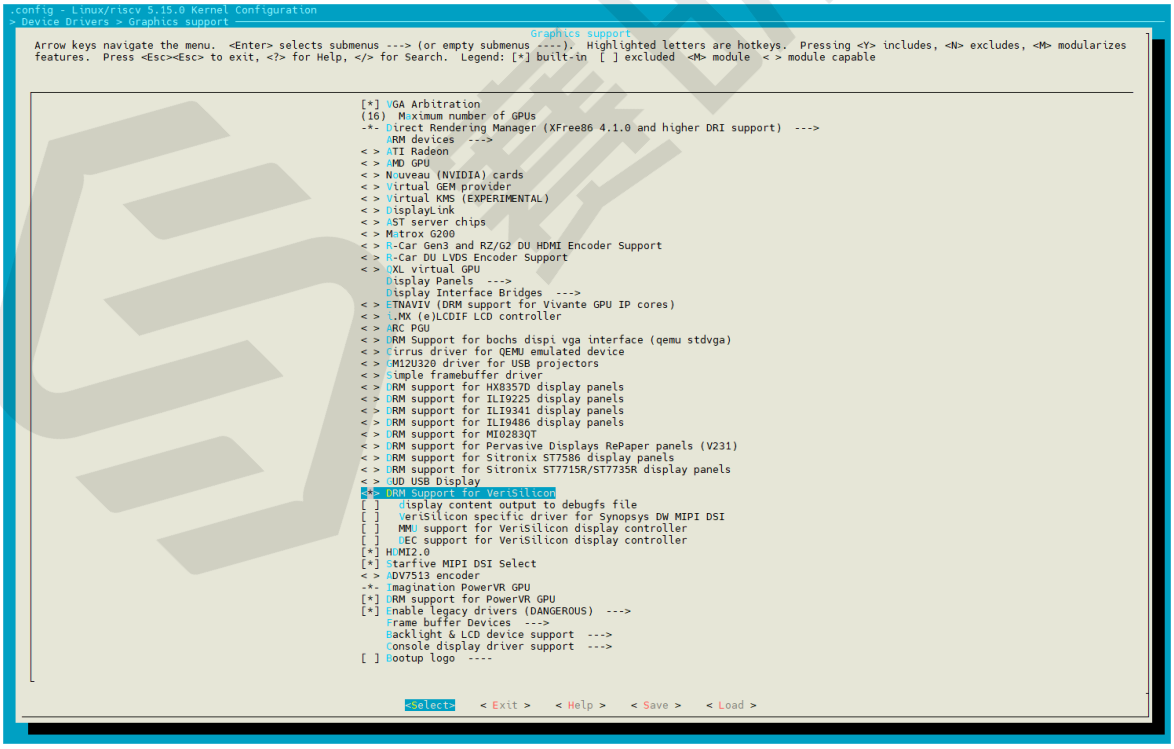
3. Enter the **Graphics support** menu.

Figure 3-2 Graphics support



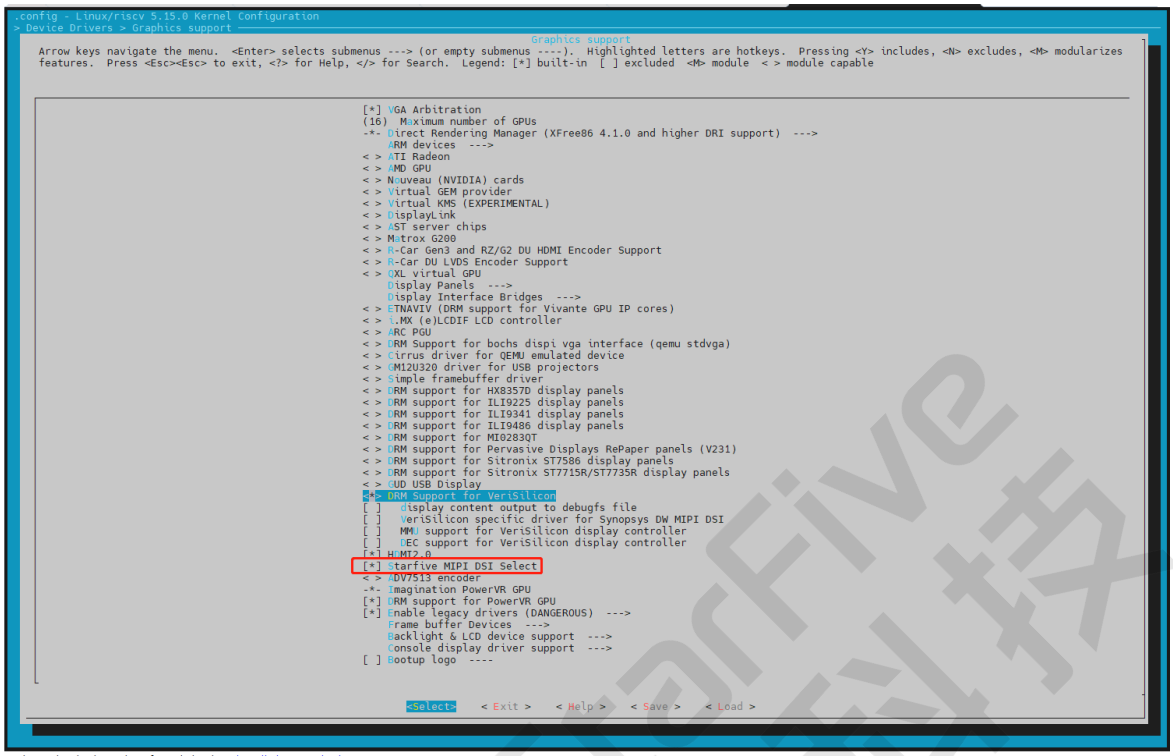
4. In the **Graphics support** menu, select the **DRM Support** option to enable video output.

Figure 3-3 DRM Support



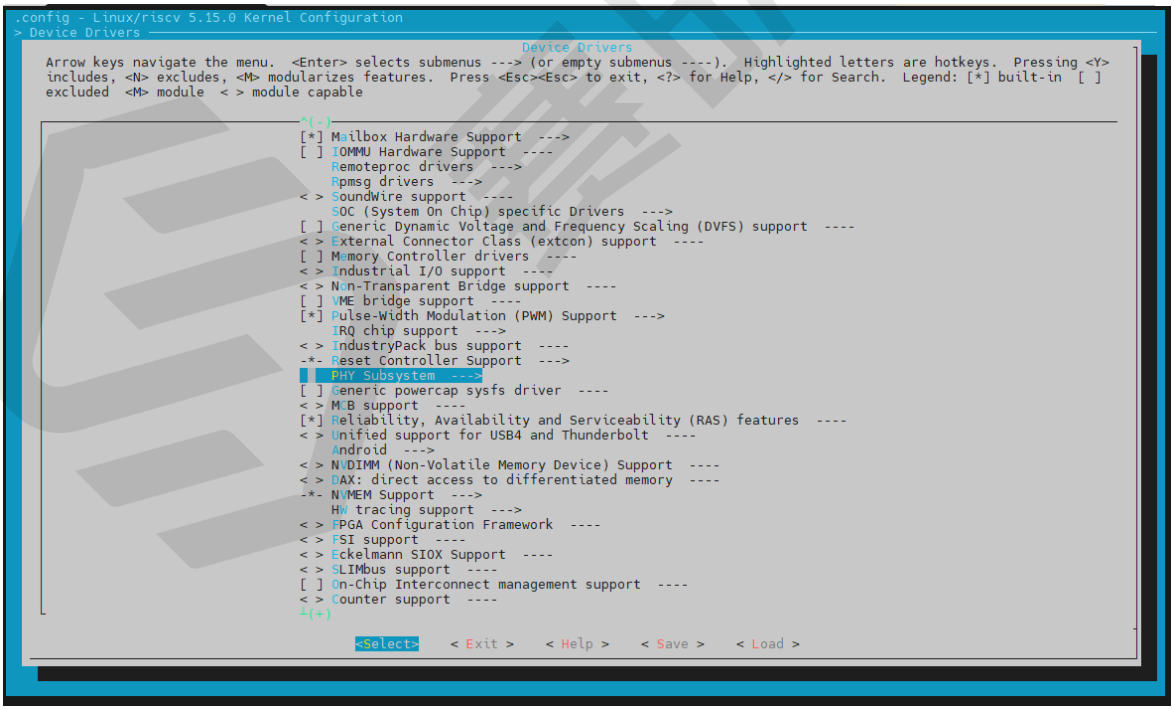
5. In the **Graphics support** menu, for MIPI output, select the **Starfive MIPI DSI Select** option.

Figure 3-4 Starfive MIPI DSI Select



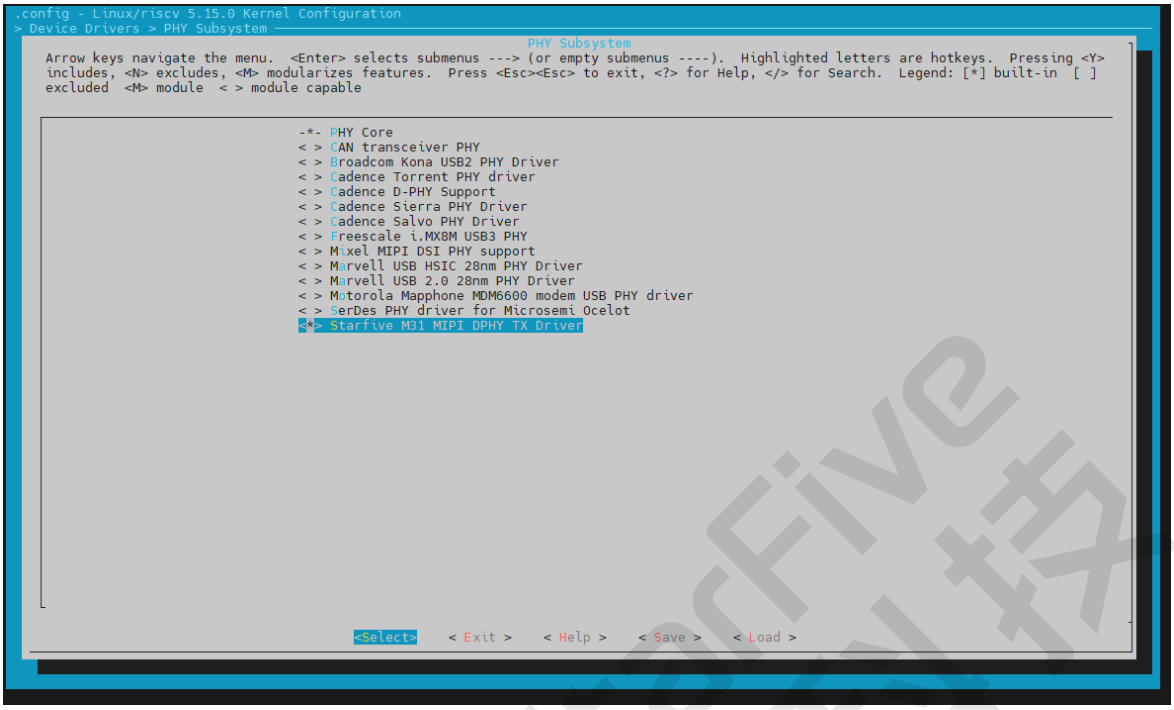
6. Then, to configure the M31 DPHY, back to **Device Drivers** menu to enter the **PHY Subsystem** menu.

Figure 3-5 Device Drivers



7. Select **Starfive M31 MIPI DPHY TX Driver**.

Figure 3-6 Starfive M31 MIPI DPHY TX Driver



8. Save your change before you exit the kernel configuration dialog.

3.2. Display Interface Description

3.2.1. Enable LCD

The function has the following parameters.

- **Function**: `seeed_panel_enable`.
- **Description**: The function is used to enable the display of LCD, initialize lane configuration and DSI configuration, and then turn on the backlight and power of LCD.
- **Prototype**: `Static int seeed_panel_enable(struct drm_panel* panel)`.

3.2.2. Disable LCD

The function has the following parameters.

- **Function**: `seeed_panel_enable`.
- **Description**: The function is used to turn down the backlight and power of LCD.
- **Prototype**: `Static int seeed_panel_disable(struct drm_panel* panel)`.

3.2.3. Obtain LCD Information

The function has the following parameters.

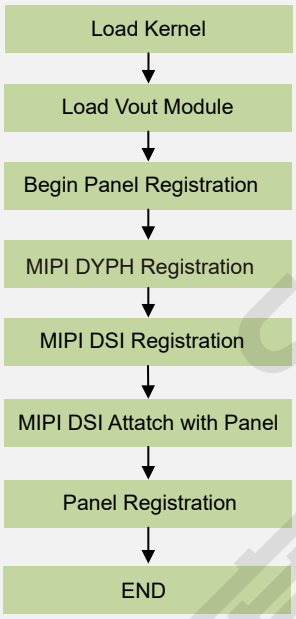
- **Function**: `seeed_panel_get_modes`.
- **Description**: The function is used to get registration information of panel.
- **Prototype**: `Static int seeed_panel_get_modes(struct drm_panel* panel, struct drm_connector* connector)`.

4. Work Process

4.1. Initialization Process

The following diagram shows the LCD initialization process for JH7110.

Figure 4-1 Initialization Process



5. MIPI Parameter Configuration

VisionFive 2 supports two MIPI display output channel, one is 1C2L (2-lane MIPI DSI), the other is 1C4L (4-lane MIPI DSI). This chapter mainly provides a tool for users to calculate JH7110 DSI timing and gives an instruction on how to set the calculated timing into the Linux driver to light the screen.

5.1. DSI Tool

This chapter mainly provides a tool for users to calculate JH7110 DSI timing and gives an instruction on how to set the calculated timing into the Linux driver to light screen. The tool package (`dsi_tool_v2.0`) includes the following parts:

- `StarFive_DSI_Tool_v2.0.exe`
- `input.md`
- `output.txt`
- `./run.sh`: Script to be used to generate `output.txt`

5.1.1. The `input.md` File

This file provides example parameters for MIPI DSI timing calculate, which can be modified and configured by users based on actual parameters.

Figure 5-1 The `input.md` File

input_gen_dsi.md

```
;bpp pixelclock hactive hfront_porch hback_porch hsync_len vactive vfront_porch vback_porch vsync_len dlanes dsi_hblank_ratio:r_hsa r_hbp r_hfp pixclk_source
24 148500000 1920 88 148 44 1080 4 36 5 4 2 2 2 1188000000
```

The description of the parameters in `input.md` file are as follows:

- **bpp**: Bit per pixel
- **pixelclock***: The pixel clock
- **hactive***: Horizontal effective pixels
- **hfront_porch***: Horizontal front porch, delay before horizontal synchronization signal
- **hback_porch***: Horizontal back porch, delay after horizontal synchronization signal
- **hsync_len***: Horizontal pulse width, the length of the horizontal synchronization signal
- **vactive***: Vertical effective pixels
- **vfront_porch***: Vertical front porch, delay before vertical synchronization signal
- **vback_porch***: Vertical back porch, delay after vertical synchronization signal
- **vsync_len***: Vertical pulse width, the length of the vertical synchronization signal

- **dlanes**: MIPI DSI lane number
- **dsi_hblank_ratio**: The ratio of **hsa**, **hbp**, and **hfp** of DSI timing. It is used to allocate the horizontal blanking of the final calculated DSI timing, which can be `0`. If any of the three values are `0`, it means you will use the default ratio.
- **r_hsa**: The ratio of **hsa** of DSI timing, which can be `0`. If the value is `0`, the DSI **hsa**, **hbp**, **hfp** will use the default ratio.
- **r_hbp**: The ratio of **hbp** of DSI timing, which can be `0`. If the value is `0`, the DSI **hsa**, **hbp**, **hfp** will use the default ratio.
- **r_hfp**: The ratio of **hfp** of DSI timing, which can be `0`. If the value is `0`, the DSI **hsa**, **hbp**, **hfp** will use the default ratio.
- **pixclk_source**: Pixel clock source, corresponding to PLL2 of JH7110 (default is 1188M). To make changes to PLL2, you will have to modify this parameter.

Note:
"*" means you can get the value from the screen manual or manufacturer.

5.1.2. The `output.txt` File

This file is the timing result generated by the tool, which is calculated based on `input.md` after running the `./run.sh` script. The following is an example `output.txt`.

Figure 5-2 Example Output

dsi_output.txt

```
Welcome to use StarFive DSI timing generation tool v2.0

{
.dpi_timing = { //1920x1080, 24 bits, 60.00 Hz
    .pixelclock    = 148500000,
    .hactive       = 1920      ,
    .hfront_porch  = 88        ,
    .hback_porch   = 148       ,
    .hsync_len     = 44        ,
    .vactive       = 1080      ,
    .vfront_porch  = 4         ,
    .vback_porch   = 36        ,
    .vsync_len     = 5         ,
},
.dsi_timing = {
    .dlanes        = 4         ,
    .bitrate       = 900000000,
    .hsa           = 306       ,
    .hbp           = 304       ,
    .hfp           = 297       ,
    .hact          = 5760      ,
},
},
```

Due to the fact that the frequency of `pixelclock` is integer divided by JH7110 PLL2, the `pixelclock` in `output.txt` here may change accordingly.

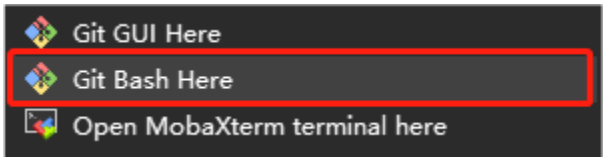
To apply this timing in Linux, refer to Configuration *(on page 24)*.

5.2. Configuration

5.2.1. Use DSI Tool

1. Install Git and open the `dsi_tool_v2.0` folder.
2. Right-click under the `dsi_tool_v2.0` folder to show options.
3. Choose **Git Bash Here** option, and open it.

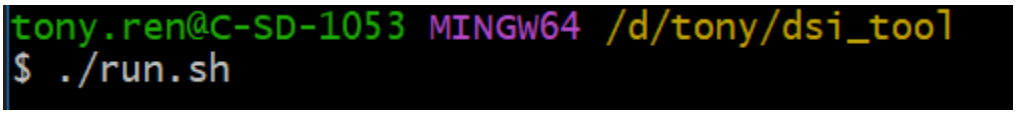
Figure 5-3 Git Bash Here



4. Run the following to generate `output.txt`:

```
./run.sh
```

Figure 5-4 ./run.sh



Note:

If the screen manufacturer does not provide the ratio of **dsi_hblank_ratio**, you can use the default ratio in the following two methods:

- Set the value of **r_hsa**, **r_hbp**, or **r_hfp** in the `input.md` file to 0.
- Generally use **r_hsa (2)**, **r_ hbp(2)**, **r_ hfp(2)**.

5.2.2. Configuration for 1C2L

This section provides an example to configure the parameters in 1C2L MIPI driver. The following code block is an example of the output file:

```
{
.dpi_timing = { //800x480, 24 bits, 60.00 Hz
    .pixelclock   = 29700000,
    .hactive      = 800     ,
    .hfront_porch = 90      ,
    .hback_porch  = 5       ,
    .hsync_len    = 5       ,
    .vactive      = 480     ,
    .vfront_porch = 60      ,
    .vback_porch  = 5       ,
    .vsync_len    = 5       ,
},
.dsi_timing = {
    .dlanes       = 1         ,
    .bitrate      = 750000000,
    .hsa          = 36      ,
    .hbp          = 108       ,
    .hfp          = 288       ,
    .hact         = 2400      ,
},
},
```

1. For the 1C2L MIPI channel, to support a new screen, it is necessary to modify the driver file `starfive_drm_seeedpanel.c` under `/Linux/drivers/gpu/drm/verisilicon`.

   This driver file servers as an `i2c` device by default, and the `probe` interface will use the `i2c` command to read the panel ID, and verify whether the current screen is connected properly based on the return value. When debugging a new screen, there are two cases:

   - If the `i2c` command can read the panel ID, the `probe` function needs to support the `i2c` interface; In `seed_ panel_ enable` function, the panel should be enabled according to the specific command of `i2c`.
   - If the `i2c` command cannot read panel ID, you are recommand to remove the `i2c` related command, then the `probe` process can be completed, so that the screen is in connected status by default.

   Note:
   It is necessary to remove all `i2c` read/write command.

2. `dpi_timing` is the timing of the panel, which needs to be added in the panel driver. It corresponds to `starfive_drm_seedpanel.c` under the path of `\linux\drivers\gpu\drm\verisilicon`, which maps `seeed_panel_modes`.

   a. Enter `starfive_drm_seedpanel.c` file.

   b. Modify the parameters of PLL 1188M under this function according to the output file.

   Figure 5-5 Modify the Parameters

```
static const struct drm_display_mode seeed_panel_modes[] = {
#ifdef PLL_1228M
    {
        .clock = 27306666 / 1000,
        .hdisplay = 800,
        .hsync_start = 800 + 93,
        .hsync_end = 800 + 93 + 5,
        .htotal = 800 + 93 + 5 + 5,
        .vdisplay = 480,
        .vsync_start = 480 + 14,
        .vsync_end = 480 + 14 + 5,
        .vtotal = 480 + 14 + 5 + 5,
    },
#endif
    {// pll 1188M
        .clock = 29700000 / 1000,
        .hdisplay = 800,
        .hsync_start = 800 + 90,
        .hsync_end = 800 + 90 + 5,
        .htotal = 800 + 90 + 5 + 5,
        .vdisplay = 480,
        .vsync_start = 480 + 60,
        .vsync_end = 480 + 60 + 5,
        .vtotal = 480 + 60 + 5 + 5,
    },
};
```

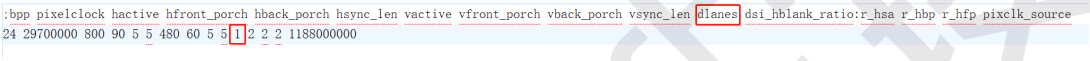
   Tip:

   - clock = pixelclock/1000
   - hdisplay = hactive
   - hsync_start = hactive + hfp

- hsync_end = hactive + hfp + hbp
- htotal = hactive + hfp + hbp+ hsync_len
- vdisplay = vactive
- vsync_start = vactive + vfp
- vsync_end = vactive + vfp + vbp
- vtotal = vactive + vfp + vbp + vsync_len

c. After configuration, it will be synchronized to the DC controller driver and MIPI DSI driver. For example, to set the lanes to **1**, you can change the corresponding parameters in the input file.

Figure 5-6 Example Input



The corresponding driver code is shown in the following figure:

Figure 5-7 Driver Code

```
static int seeed_dsi_probe(struct mipi_dsi_device *dsi)
{
    int ret;

    dsi->mode_flags = (MIPI_DSI_MODE_VIDEO |
                MIPI_DSI_MODE_VIDEO_SYNC_PULSE |
                MIPI_DSI_MODE_LPM);
    dsi->format = MIPI_DSI_FMT_RGB888;
    dsi->lanes = 1;

    ret = mipi_dsi_attach(dsi);
    if (ret)
        dev_err(&dsi->dev, "failed to attach dsi to host: %d\n", ret);

    return ret;
}
```

3. Follow the steps below to configure MIPI DSI.

a. Open `starfive_drm_dsi.c` file under the path of `\linux\drivers\gpu\drm\verisilicon` in Linux.

b. Locate `cdns_dsi_mode2cfg` function to modify the value of **hsa**, **hbp** and **hfp** in channel 0 to **36**, **108**, **288**.

Note:
Modifying parameters should based on the corresponding channel. 1C2L corresponds to channel 0 while 1C4L corresponds to channel 1.

Figure 5-8 hsa hbp hfp

```
if (output->dev->channel == 0) {//seeed
    dsi_cfg->hsa = 117-DSI_HSA_FRAME_OVERHEAD;
    dsi_cfg->hbp = 115-DSI_HBP_FRAME_OVERHEAD;
    dsi_cfg->hfp = 209-DSI_HFP_FRAME_OVERHEAD;
} else if (output->dev->channel == 1){//raxda 8 inch config
    dsi_cfg->hsa = 45-DSI_HSA_FRAME_OVERHEAD;
    dsi_cfg->hbp = 134-DSI_HBP_FRAME_OVERHEAD;
    dsi_cfg->hfp = 356-DSI_HFP_FRAME_OVERHEAD;

}
else if (output->dev->channel == 2){//raxda 10 inch config
    dsi_cfg->hsa = 405-DSI_HSA_FRAME_OVERHEAD;
    dsi_cfg->hbp = 403-DSI_HBP_FRAME_OVERHEAD;
    dsi_cfg->hfp = 396-DSI_HFP_FRAME_OVERHEAD;

}
```

c. Then locate `cdns_dsi_adjust_phy_config` function to modify the **bitrate** to **750000000**.

Figure 5-9 Bitrate

```
if (output->dev->channel == 0) {
    phy_cfg->hs_clk_rate = 750000000;//seeed
} else if (output->dev->channel == 1){
    phy_cfg->hs_clk_rate = 490000000;//8 inch
} else if (output->dev->channel == 2){
    phy_cfg->hs_clk_rate = 980000000;//10 inch
}
```

Tip:

The **hs_clk_rate** in the figure means **bitrate**.

5.2.3. Configuration for 1C4L

This section provides an example to configure the parameters in 1C4L MIPI driver. The following code block is an example of the output file:

```
dpi_timing = { //800x1280, 24 bits, 59.68 Hz
    .pixelclock   = 66000000,
    .hactive      = 800     ,
    .hfront_porch = 44      ,
    .hback_porch  = 5       ,
    .hsync_len    = 5       ,
    .vactive      = 1280    ,
    .vfront_porch = 5       ,
    .vback_porch  = 5       ,
    .vsync_len    = 5       ,
},
```

```
.dsi_timing = {
    .dlanes       = 4         ,
    .bitrate      = 400000000,
    .hsa          = 66        ,
    .hbp          = 64        ,
    .hfp          = 57        ,
    .hact         = 2400      ,
},
```

1. For the 1C4L MIPI channel, to support a new screen, it is necessary to modify the driver file `panel-jadard-jd9365da-h3.c` under `/Linux/drivers/gpu/drm/panel`.

   This driver file servers as an `i2c` device by default, and the `probe` interface will use the `i2c` command to read the panel ID, and verify whether the current screen is connected properly based on the return value. When debugging a new screen, there are two cases:

   - If the `i2c` command can read the panel ID, the `probe` function needs to support the `i2c` interface; In `seed_ panel_ enable` function, the panel should be enabled according to the specific command of `i2c`.
   - If the `i2c` command cannot read panel ID, you are recommand to remove the `i2c` related command, then the `probe` process can be completed, so that the screen is in connected status by default.

   Note:
   It is necessary to remove all `i2c` read/write command.

2. `dpi_timing` is the timing of the panel, which needs to be added in the panel driver. It corresponds to `cz101b4001_desc` under the path of `\linux\drivers\gpu\drm\panel`.

   a. Enter `starfive_drm_seedpanel.c` file.

   b. Modify the parameters under this function according to the output file.

Figure 5-10 Modify the Parameters

```
static const struct jadard_panel_desc cz101b4001_desc = {
    .mode = {
        .clock      = 79200,

        .hdisplay   = 800,
        .hsync_start    = 800 + 139,
        .hsync_end  = 800 + 139 + 5,
        .htotal     = 800 + 139 + 5 + 5,

        .vdisplay   = 1280,
        .vsync_start    = 1280 + 84,
        .vsync_end  = 1280 + 84 + 20,
        .vtotal     = 1280 + 84+ 20 + 7,

        .width_mm   = 62,
        .height_mm  = 110,
        .type       = DRM_MODE_TYPE_DRIVER | DRM_MODE_TYPE_PREFERRED,
    },
    .lanes = 4,
    .format = MIPI_DSI_FMT_RGB888,
    .init_cmds = cz101b4001_init_cmds,
    .num_init_cmds = ARRAY_SIZE(cz101b4001_init_cmds),
    .timings = &jadard_timing,
    .num_timings = 1,
};
```

Tip:

- clock = pixelclock/1000
- hdisplay = hactive
- hsync_start = hactive + hfp
- hsync_end = hactive + hfp + hbp
- htotal = hactive + hfp + hbp+ hsync_len
- vdisplay = vactive
- vsync_start = vactive + vfp
- vsync_end = vactive + vfp + vbp
- vtotal = vactive + vfp + vbp + vsync_len

c. After configuration, it will be synchronized to the DC controller driver and MIPI DSI driver. For example, to set the lanes to **4**, you can change the corresponding parameters in the input file. The corresponding driver code is shown in the following figure:

Figure 5-11 Driver Code

```
static int jadard_dsi_probe(struct mipi_dsi_devic
{
    struct device *dev = &dsi->dev;
    struct jadard *jadard = mipi_dsi_get_drvdata(dsi);

    int ret;

    dsi->mode_flags = MIPI_DSI_MODE_LPM | MIPI_DSI_MODE_
    dsi->format = MIPI_DSI_FMT_RGB888;
    dsi->lanes = 4;
    dsi->channel = 1;
    dsi->hs_rate = 490000000;

    ret = mipi_dsi_attach(dsi);
    if (ret < 0) {
        return ret;
    }

    return 0;
```

3. Follow the steps below to configure MIPI DSI.

a. Open `starfive_drm_dsi.c` file under the path of `\linux\drivers\gpu\drm\verisilicon` in Linux.

b. Locate `cdns_dsi_mode2cfg` function to modify the value of **hsa**, **hbp** and **hfp** in channel 1 to **66**, **64**, **57**.

Note:
Modifying parameters should based on the corresponding channel. 1C2L corresponds to channel 0 while 1C4L corresponds to channel 1.

Figure 5-12 hsa hbp hfp

```
if (output->dev->channel == 0) {//seeed
    dsi_cfg->hsa = 117-DSI_HSA_FRAME_OVERHEAD;
    dsi_cfg->hbp = 115-DSI_HBP_FRAME_OVERHEAD;
    dsi_cfg->hfp = 209-DSI_HFP_FRAME_OVERHEAD;
} else if (output->dev->channel == 1){//raxda 8 inch config
    dsi_cfg->hsa = 45-DSI_HSA_FRAME_OVERHEAD;
    dsi_cfg->hbp = 134-DSI_HBP_FRAME_OVERHEAD;
    dsi_cfg->hfp = 356-DSI_HFP_FRAME_OVERHEAD;

}
else if (output->dev->channel == 2){//raxda 10 inch config
    dsi_cfg->hsa = 405-DSI_HSA_FRAME_OVERHEAD;
    dsi_cfg->hbp = 403-DSI_HBP_FRAME_OVERHEAD;
    dsi_cfg->hfp = 396-DSI_HFP_FRAME_OVERHEAD;

}
```

c. Then locate `cdns_dsi_adjust_phy_config` function to modify the **bitrate** to **400000000**.

Figure 5-13 Bitrate

```
if (output->dev->channel == 0) {
    phy_cfg->hs_clk_rate = 750000000;//seeed
} else if (output->dev->channel == 1){
    phy_cfg->hs_clk_rate = 490000000;//8 inch
} else if (output->dev->channel == 2){
    phy_cfg->hs_clk_rate = 980000000;//10 inch
}
```

Tip:
The **hs_clk_rate** in the figure means **bitrate**.

6. Debug LCD

6.1. Test Case Configuration

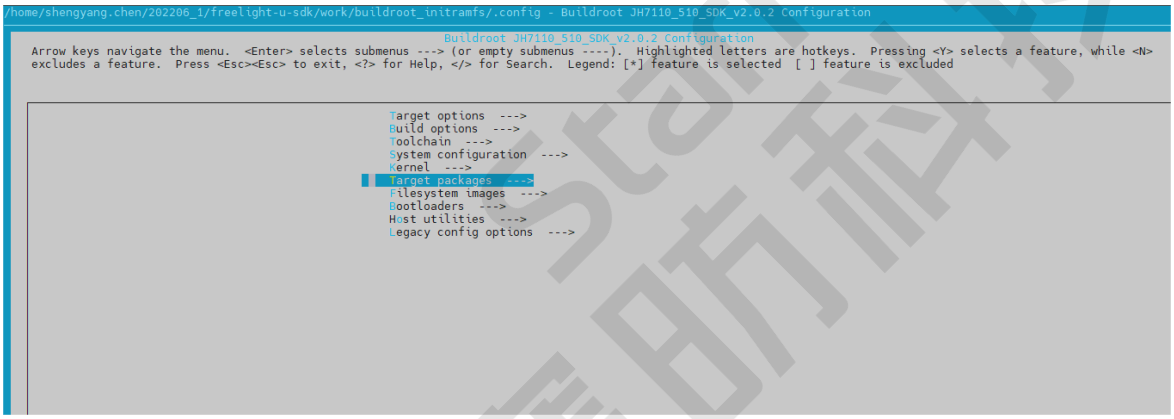
Follow the steps below to enable the kernel configuration for MIPI LCD.

1. Under the root directory of `freelight-u-sdk`, type the following command to enter the kernel menu configuration GUI.

```
make linux-menuconfig
```

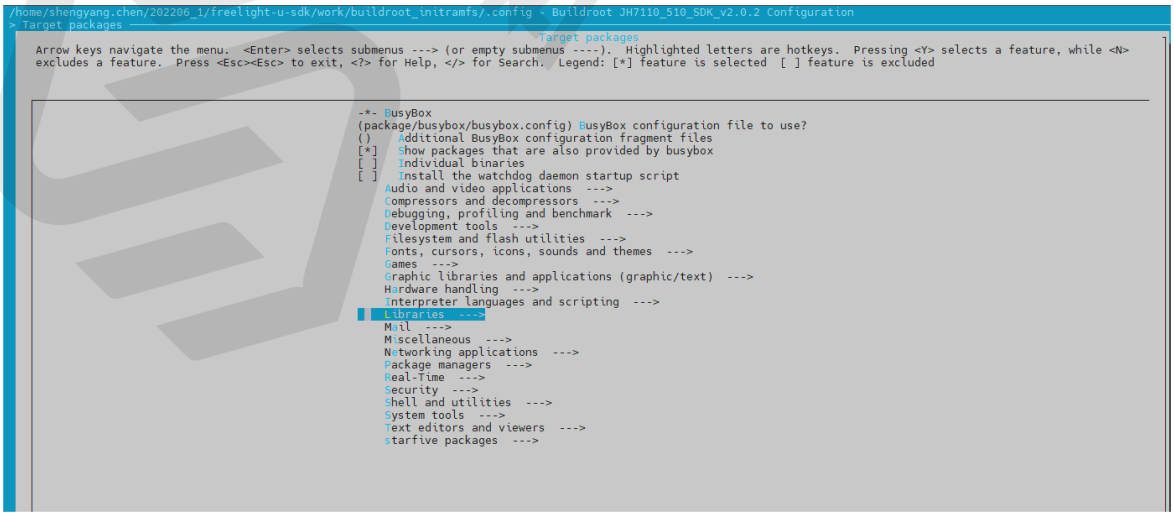
2. Enter the **Target packages** menu.

Figure 6-1 Target Packages



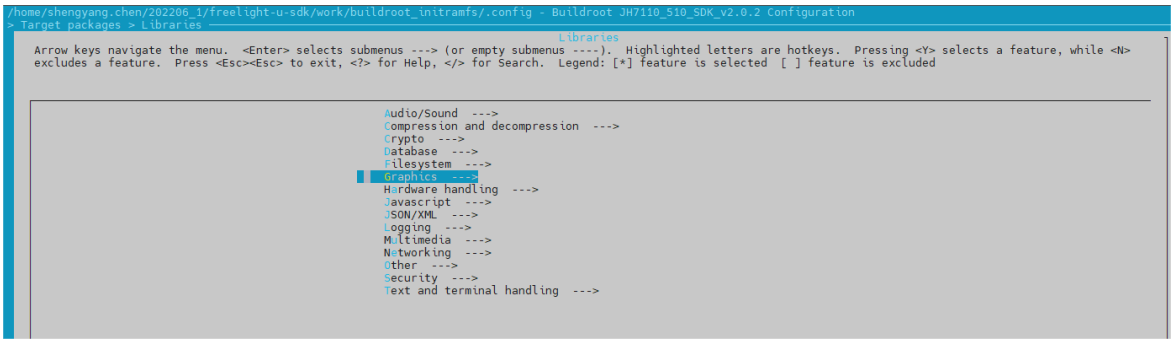
3. Enter the **Libraries** menu.

Figure 6-2 Libraries



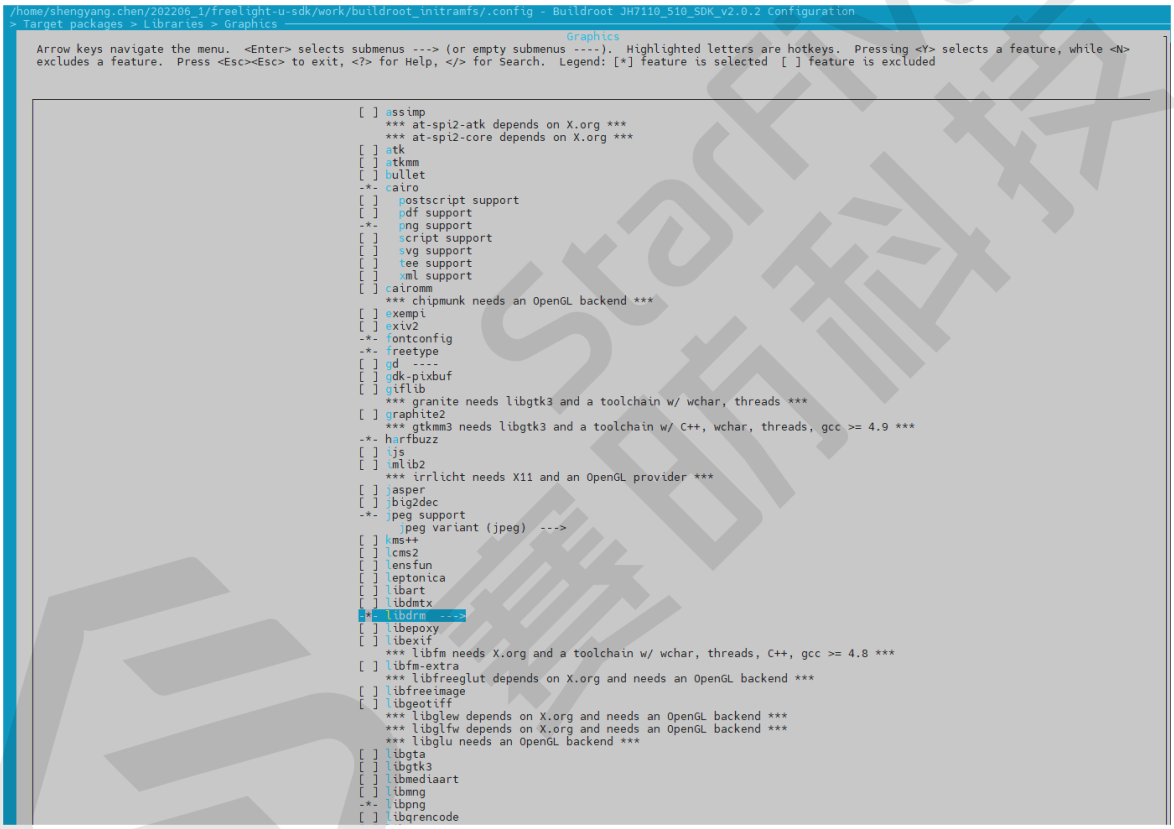
4. Enter the **Graphics** menu.

Figure 6-3 Graphics



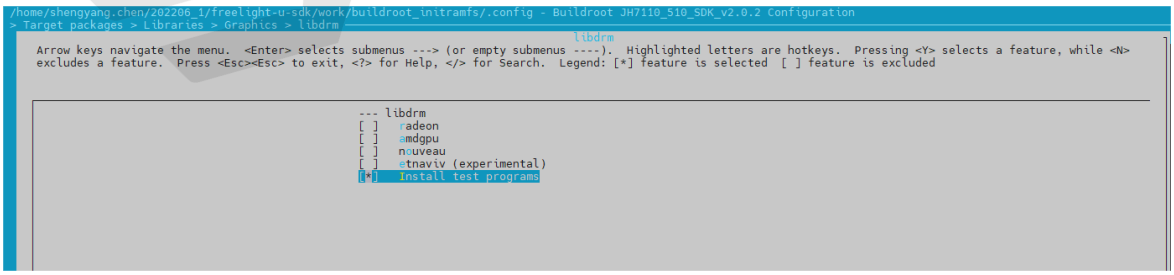
5. Enter the **libdrm** menu.

Figure 6-4 libdrm



6. Select the **Install test programs** option, or you may select ALL options under this menu.

Figure 6-5 Install Test Programs



Result: After you have completed all the above configuration, you can use the `modetest` tool in kernel for testing.

7. Save your change before you exit the kernel configuration dialog.

6.2. Before Debug

Before debugging the MIPI controller, make sure you see the following screen in the start-up logs.

Figure 6-6 Start-up Logs

The content in the red box indicates the MIPI work status. Make sure the log lines showing in the red box are printed before debug.

6.3. Debug LCD

1. Follow the steps in Test Case Configuration *(on page 32)* to configure the test environment.

   Note:
   Make sure you have configured **libdrm** and **modetest** before compiling and burning an image.

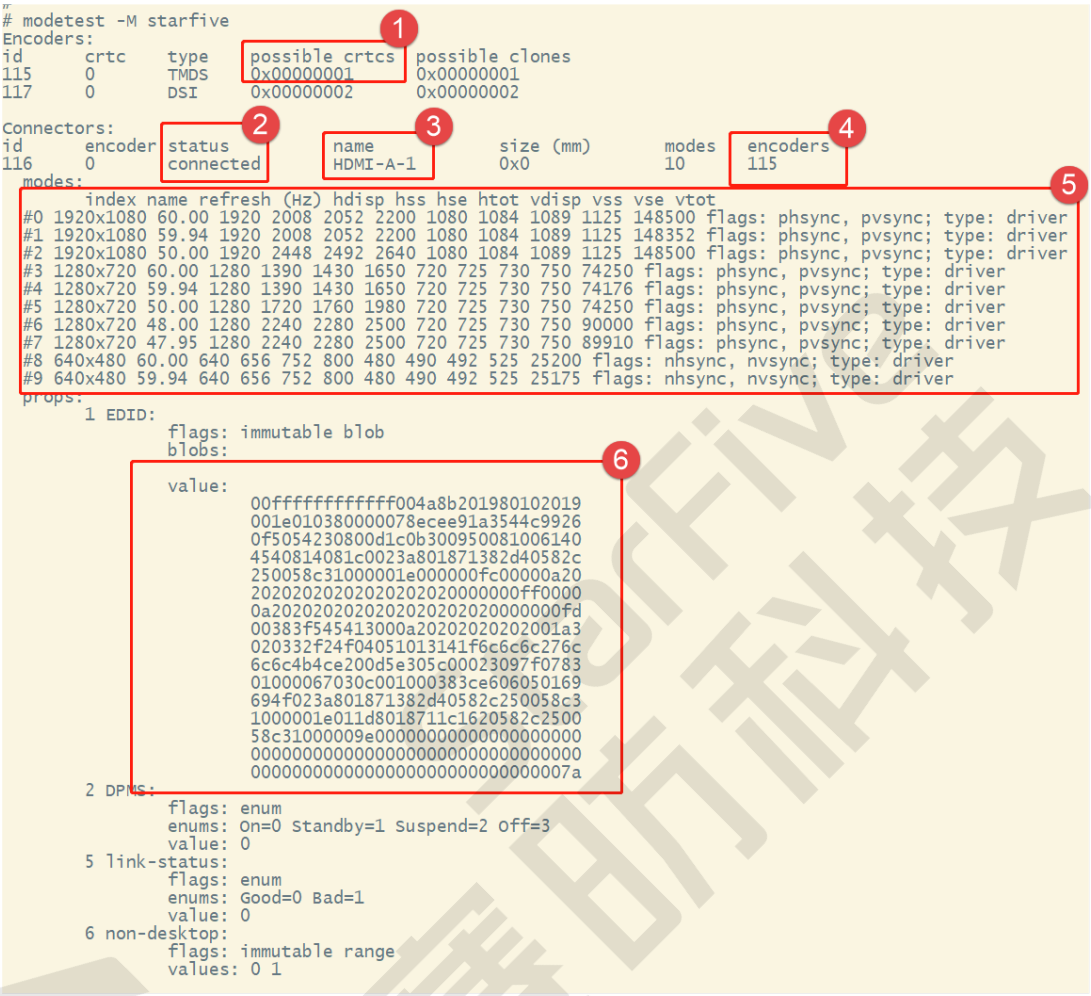
2. After you have completed the kernel start-up, use the following command to verify the display functions and connection status.

   ```
   modetest -M starfive
   ```

   The following legends and tables display an example output and descriptions.

- Debug output 1:

Figure 6-7 Debug Display 1




```
# modetest -M starfive
Encoders:
id      crtc    type    possible crtcs  possible clones
115     0       TMDS    0x00000001      0x00000001
117     0       DSI     0x00000002      0x00000002

Connectors:
id      encoder status          name            size (mm)       modes   encoders
116     0       connected       HDMI-A-1        0x0             10      115
  modes:
        index name refresh (Hz) hdisp hss hse htot vdisp vss vse vtot
  #0 1920x1080 60.00 1920 2008 2052 2200 1080 1084 1089 1125 148500 flags: phsync, pvsync; type: driver
  #1 1920x1080 59.94 1920 2008 2052 2200 1080 1084 1089 1125 148352 flags: phsync, pvsync; type: driver
  #2 1920x1080 50.00 1920 2448 2492 2640 1080 1084 1089 1125 148500 flags: phsync, pvsync; type: driver
  #3 1280x720 60.00 1280 1390 1430 1650 720 725 730 750 74250 flags: phsync, pvsync; type: driver
  #4 1280x720 59.94 1280 1390 1430 1650 720 725 730 750 74176 flags: phsync, pvsync; type: driver
  #5 1280x720 50.00 1280 1720 1760 1980 720 725 730 750 74250 flags: phsync, pvsync; type: driver
  #6 1280x720 48.00 1280 2240 2280 2500 720 725 730 750 90000 flags: phsync, pvsync; type: driver
  #7 1280x720 47.95 1280 2240 2280 2500 720 725 730 750 89910 flags: phsync, pvsync; type: driver
  #8 640x480 60.00 640 656 752 800 480 490 492 525 25200 flags: nhsync, nvsync; type: driver
  #9 640x480 59.94 640 656 752 800 480 490 492 525 25175 flags: nhsync, nvsync; type: driver
  props:
        1 EDID:
                flags: immutable blob
                blobs:

                value:
                        00ffffffffffff004a8b201980102019
                        001e010380000078ecee91a3544c9926
                        0f5054230800d1c0b300950081006140
                        4540814081c0023a801871382d40582c
                        250058c31000001e000000fc00000a20
                        2020202020202020202020000000ff0000
                        0a2020202020202020202020000000fd
                        00383f545413000a202020202020001a3
                        020332f24f04051013141f6c6c6c276c
                        6c6c4b4ce200d5e305c00023097f0783
                        01000067030c001000383ce606050169
                        694f023a801871382d40582c250058c3
                        1000001e011d8018711c1620582c2500
                        58c31000009e000000000000000000000
                        000000000000000000000000000000000
                        000000000000000000000000000007a
        2 DPMS:
                flags: enum
                enums: On=0 Standby=1 Suspend=2 Off=3
                value: 0
        5 link-status:
                flags: enum
                enums: Good=0 Bad=1
                value: 0
        6 non-desktop:
                flags: immutable range
                values: 0 1
```

Table 6-1 Debug Display 1

Legend	Label	Description
①	**possible crtcs**	Available *Cathode Ray Tube Controller (CRTC)* devices
②	**status**	Whether the display connector is connected or not
③	**name**	The name (type) of the display connector
④	**encoders**	The connected encoders
⑤	**modes**	The supported display modes
⑥	**value**	The *Extended Display Identification Data (EDID)* of the screen

- Debug output 2:

Figure 6-8 Debug Display 2

Table 6-2 Debug Display 2

Legend	Label	Description
①	**id**	The CRTC 0x00000001 mentioned in row ① of table Table 6-1 : Debug Display 1 *(on page 35)*, which means the CRTC is available for use.
②	**id**	The CRTC 0x00000002 mentioned in row ① of table Table 6-1 : Debug Display 1 *(on page 35)*, which means the CRTC is available for use.

Note:
If the displayed CRTC is 0x00000003, both of the CRTCs are available for use.

- Debug output 3:

Figure 6-9 Debug Display 3

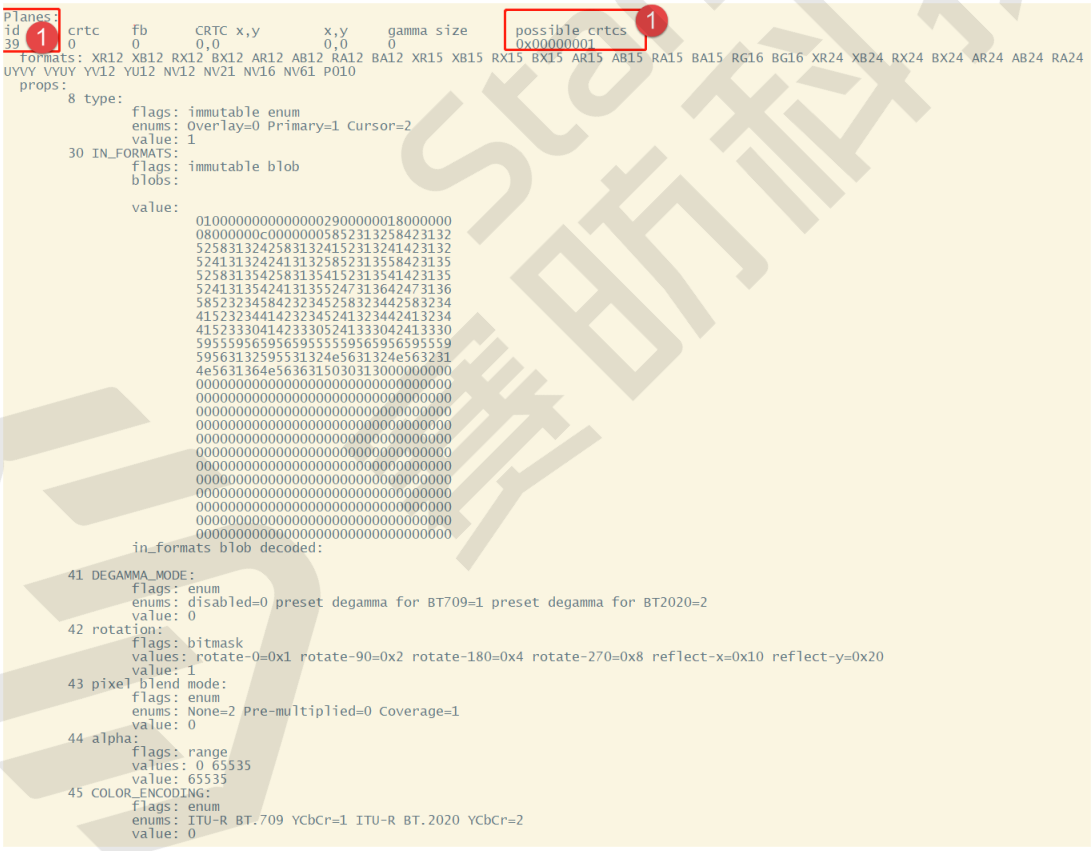
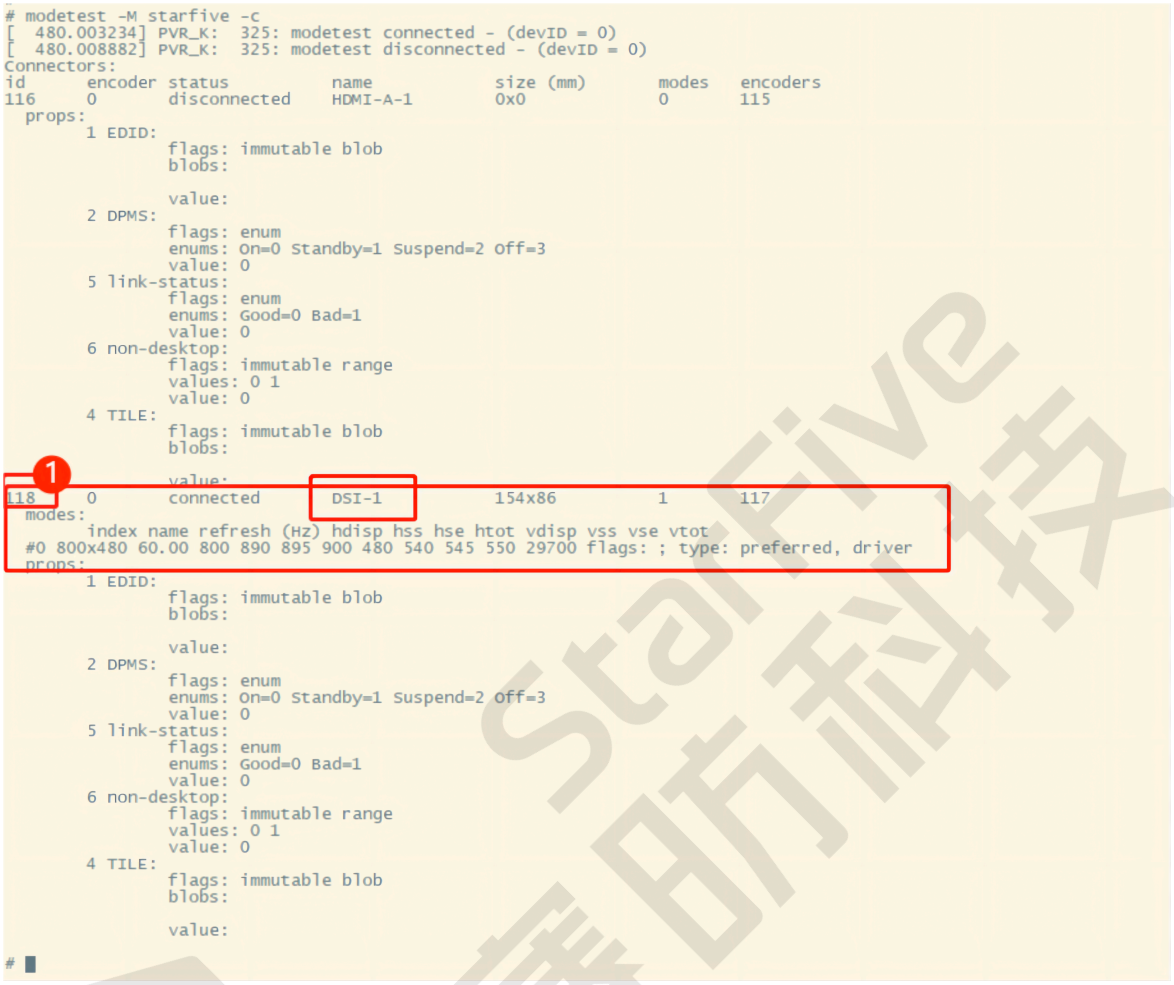


Table 6-3 Debug Display 3

Legend	Description
①	The CRTC and its connected plane

- Check MIPI connect status:

Figure 6-10 MIPI Connect Status




```
# modetest -M starfive -c
[  480.003234] PVR_K:  325: modetest connected - (devID = 0)
[  480.008882] PVR_K:  325: modetest disconnected - (devID = 0)
Connectors:
id      encoder status          name            size (mm)       modes   encoders
116     0       disconnected    HDMI-A-1        0x0             0       115
  props:
        1 EDID:
                flags: immutable blob
                blobs:

                value:
        2 DPMS:
                flags: enum
                enums: On=0 Standby=1 Suspend=2 Off=3
                value: 0
        5 link-status:
                flags: enum
                enums: Good=0 Bad=1
                value: 0
        6 non-desktop:
                flags: immutable range
                values: 0 1
                value: 0
        4 TILE:
                flags: immutable blob
                blobs:

                value:
118     0       connected       DSI-1           154x86          1       117
  modes:
        index name refresh (Hz) hdisp hss hse htot vdisp vss vse vtot
  #0 800x480 60.00 800 890 895 900 480 540 545 550 29700 flags: ; type: preferred, driver
  props:
        1 EDID:
                flags: immutable blob
                blobs:

                value:
        2 DPMS:
                flags: enum
                enums: On=0 Standby=1 Suspend=2 Off=3
                value: 0
        5 link-status:
                flags: enum
                enums: Good=0 Bad=1
                value: 0
        6 non-desktop:
                flags: immutable range
                values: 0 1
                value: 0
        4 TILE:
                flags: immutable blob
                blobs:

                value:

#
```

The common connector ID of MIPI is 118. If both `rgb2hdmi` and MIPI panel are registered, the connector ID of MIPI will be assigned as 120.

6.4. Test Example

For LCD Output

The following command shows an example for testing the LCD output.

```
modetest -M starfive -D 0 -a -s 118@35:800x480 -P 74@35:800x480@RG16 -Ftiles
```

The following list provides explanations for the parameters in the above example command.

- **118@35:800x480** - <Connector ID>@<CRTC ID>: <Resolution>
- **74@35:800x480@RG16** - <Plane ID>@<CRTC ID>: <Resolution>@<Format>

Output Result

The following photo shows the output generated from the above example command.

Figure 6-11 Test Example