StarFive 赛昉科技

Software SDK Developer Guide for Display Controller

VisionFive 2

Version: 1.0

Date: 2022/11/24

Doc ID: JH7110-DGEN-014

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 programing basics and debugging know-how for the display module of the StarFive next generation SoC platform - JH7110.

Audience

This document mainly serves the display module relevant driver developers. If you are developing 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
1.0		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

The display subsystem, named as **dom_vout_top** in the JH7110 system, includes front-end video data capture, display controller and display interface, such as RGB IF, HDMI, and MIPI.

In the display subsystem, DC8200 display controller works as a third party high-performance optimized-area *Display Processor Unit (DPU)* IP that can be used for reading rendered images from the frame buffer to the display.

See Block Diagram *(on page 7)* for more information.

1.1. Function Introduction

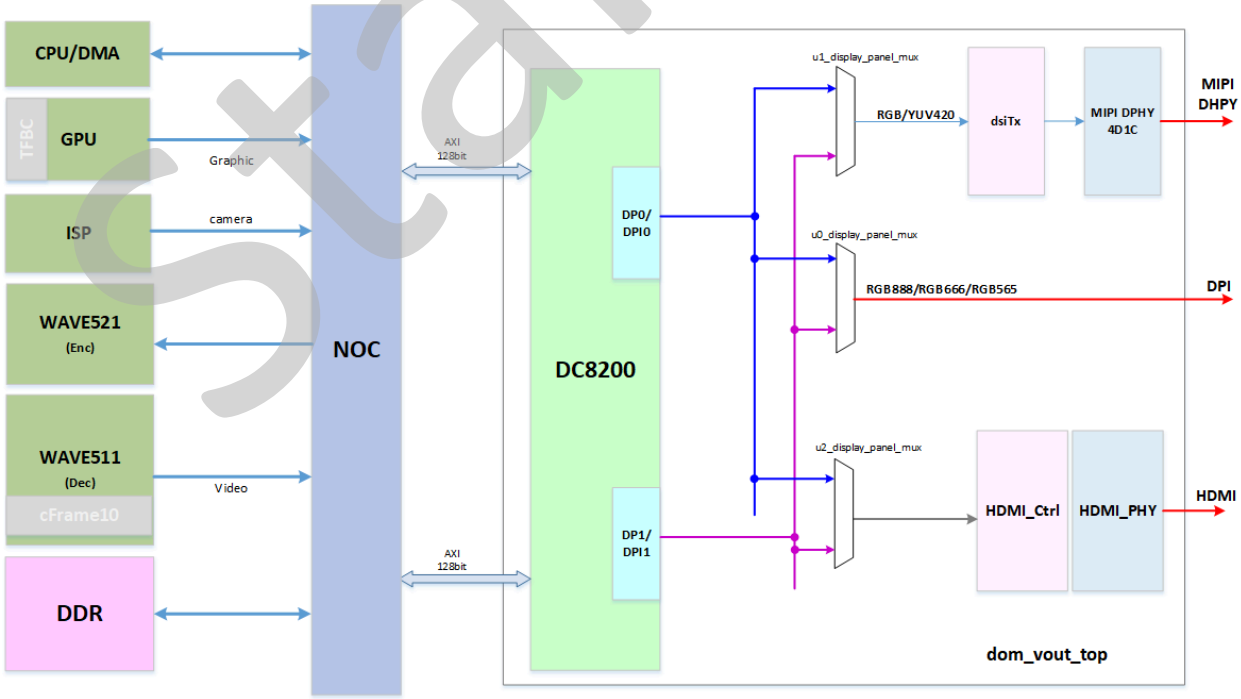
The display controller supports the following features:

- Support 2 display output panels
- Support 1 video/graphic layer per output panel
- Support 2 overlay layers per output panel
- Support 6 video/graphic and overlay layers
- Support 2 cursor layers
- Programmable cursor sizes: 32 × 32, 64 × 64
- Dynamic layer allocation support for video/graphic and overlay layers
- Support 2-screen display
- Support output interfaces: DP (RGB, YUV), DPI (RGB)

1.2. Block Diagram

The block diagram of the display subsystem is displayed in following diagram.

Figure 1-1 Display Subsystem Block Diagram



Data Mapping

The DSI transmitter's pixel data could be from panel 0 or panel 1 interface of DC8200, and could be selected from DP or DPI interface. The RGB PAD and HDMI have similar mechanism.

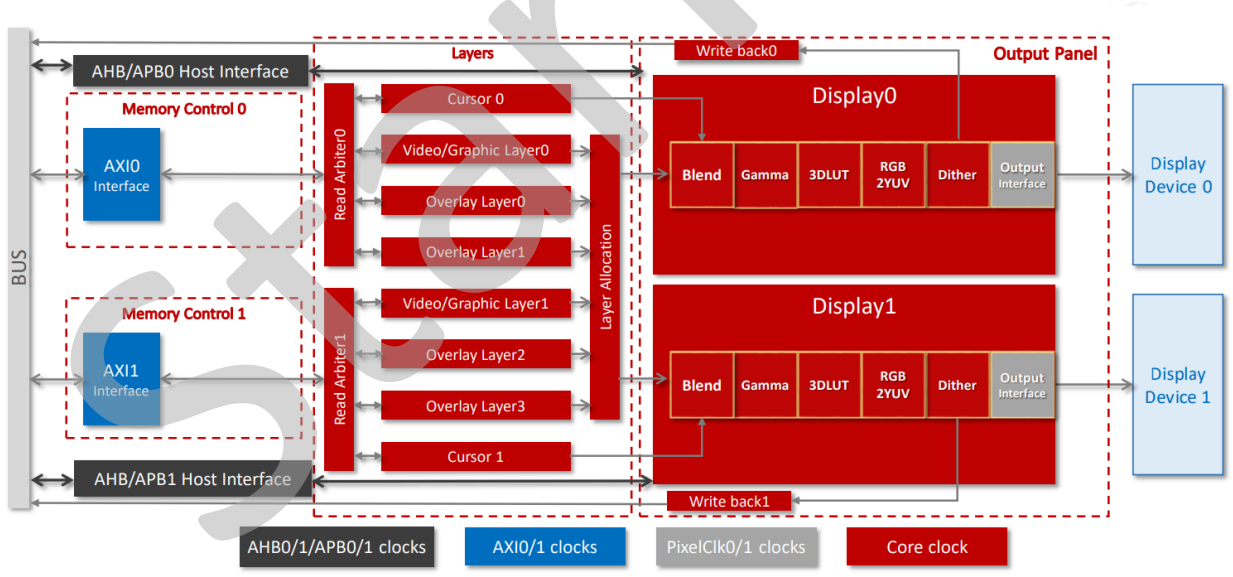
Table 1-1 Display Subsystem Data Mapping

Destination	Supported Data Mapping	Comment
DPI to PAD	• DP0/DP1 or DPI0/1 is used, default DPI is used. • RGB24, RGB666 (CFG1), RGB565 (CFG1) when DPI is used.	For flexibility
DSI Tx Data from DC8200	• Both DPI and DP are supported. • YUV420 8-bit only (CFG3). • YUV422 8-bit only (CFG1).	Default DPI
HDMI Data from DC8200	• Both DP0 and DP1 are used for RGB and YUV. • YUV444 and YUV422 8-bit/10-bit (CFG1). • YUV420 8-bit/10-bit (CFG3).	DP by default, and DPI for back-up

1.3. DC8200 Display Controller

The following image shows the block diagram of the DC8200 display controller.

Figure 1-2 DC8200 Display Controller Block Diagram



The following components are included.

- **Host Interface**: Allows communication with the system and the DC controller. The host interfaces include the AXI, AHB, and APB. In this block, data crosses clock domain boundaries.
- **Memory Control**: Contains the AXI interface to manage the access between the system memory and layers of the DC8200.
- **Write Back**: For debug use only.
- **Layers**: Include video/graphic, overlay, and cursor layers.

- Video/graphic layers support both video and graphic configurations. Video/graphic and overlay layers support dynamic layer allocation and de-gamma.
- Overlay 1 and overlay 3 do not support scaling, rotation, and line buffers.
- Cursor layers provide hardware cursor functionality.

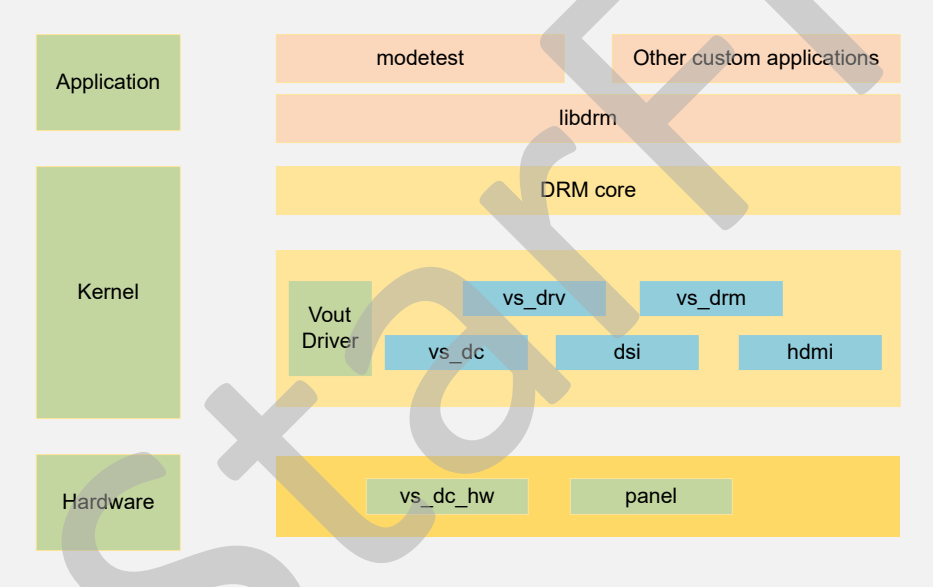
- **Dither**: Provides a *Lookup Table (LUT)*.
- **Gamma**: Performs gamma correction.
- **Output Panels**: Support two output panels, as shown by Display0 and Display1.
- **Output Interfaces**: Support parallel pixel output with 30-bit Data, Horizontal Sync, Vertical Sync, and Data Enable. Support easy adaptation to external serialization logic, for example, HDMI.
- **Pixel Pipelines**: Reside in the layers and output panels. Two display pipelines support linear and tiled frame buffers for RGB and YUV inputs. Optional enhancements include multiple overlay layers, composition and blending, up/down scaling with multi-tap filtering, and color space conversions.

If you need more information, you may contact StarFive technical support and request documentation from the third-party IP.

1.4. Video Output Driver Framework

The following figure shows the framework of the video output driver and the display controller.

Figure 1-3 Driver Framework



The video output driver framework has the following 3 layers.

- **Application** layer consists of application code and test code and communicate with kernel layer through **libdrm**.
- **Kernel** layer consists of **DRM core** and **Vout driver**. **DRM core** receives commands from **libdrm** and transfer to **Vout driver**.
- **Hardware** layer is connected with **Vout driver**, and it operates the hardware directly.

1.5. Device Tree Overview

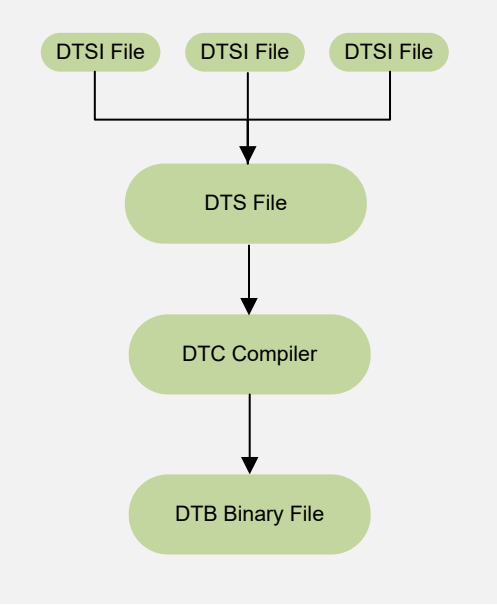
Since Linux 3.x, device tree is introduced as a data structure and language to describe hardware configuration. It is a system-readable description of hardware settings so that the operating system doesn't have to hard code details of the machine.

A device tree is primarily represented in the following forms.

- *Device Tree Compiler (DTC)*: The tool used to compile device tree into system-readable binaries.
- *Device Tree Source (DTS)*: The human-readable device tree description file. You can locate the target parameters and modify hardware configuration in this file.
- *Device Tree Source Information (DTSI)*: The human-readable header file which you can include in device tree description. You can locate the target parameters and modify hardware configuration in this file.
- *Device Tree Blob (DTB)*: The system-readable device tree binary blob files which is burned in system for execution.

The following diagram shows the relationship (workflow) of the above forms.

Figure 1-4 Device Tree Workflow



1.6. Source Code Structure

The following code block shows the source code structure of the display controller.

```
linux-5.15.0
└-- drivers
└-- | --gpu
| -- | -- | -- drm
| -- | -- | -- |-- verisilicon
| -- | -- | -- | -- | -- vs_dc.c
| -- | -- | -- | -- | -- vs_dc.h
| -- | -- | -- | -- | -- vs_dc_hw.c
| -- | -- | -- | -- | -- vs_dc_hw.h
| -- | -- | -- | -- | -- vs_drv.c
| -- | -- | -- | -- | -- vs_drv.h
| -- | -- | -- | -- | -- vs_crtc.c
| -- | -- | -- | -- | -- vs_crtc.h
| -- | -- | -- | -- | -- vs_plane.c
| -- | -- | -- | -- | -- vs_plane.h
| -- | -- | -- | -- | -- vs_simple_enc.c
| -- | -- | -- | -- | -- vs_simple_enc.h
| -- | -- | -- | -- | -- vs_gem.c
| -- | -- | -- | -- | -- vs_gem.h
| -- | -- | -- | -- | -- vs_virtual.c
| -- | -- | -- | -- | -- vs_virtual.h
| -- | -- | -- | -- | -- vs_dc_dec.c
| -- | -- | -- | -- | -- vs_dc_dec.h
```

2. Configuration

2.1. Device Tree Configuration

Overview

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

The device tree of JH7110 is stored in the following path:

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

The following code block shows the DTS file structure for JH7110.

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

Display Subsystem

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

```
display: display-subsystem {
        compatible = "startive,jh7110-display","verisilicon,display-subsystem";
        ports = <&dc_out_dpi0>;
        status = "disabled";
dssctrl: dssctrl@295B0000 {
        compatible = "starfive,jh7110-dssctrl","verisilicon,dss-ctrl", "syscon";
        reg = <0 0X295B0000 0 0x90>;
};
```

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

- **compatible**: Compatibility information, used to associate the display controller and its target device.
- **ports**: The port(s) used by the display controller.
- **status**: The work status of the display controller module. To enable the module, set this bit as "`okay`" or to disable the module, set this bit as "`disabled`".
- **reg**: Register base address "`0X295B0000`" and range "`0x90`".

DC8200

In the file `jh7110.dtsi`, you can find the device tree configuration of DC8200 (host) as the following code block:

```
dc8200: dc8200@29400000 {
        compatible = "verisilicon,dc8200";
        verisilicon,dss-syscon = <&dssctrl>;
        reg = <0x0 0x29400000 0x0 0x100>,
              <0x0 0x29400800 0x0 0x2000>,
              <0x0 0x17030000 0x0 0x1000>;
        interrupts = <95>;
        status = "disabled";
        clocks = <&clkgen JH7110_NOC_BUS_CLK_CPU_AXI>,
                 <&clkgen JH7110_NOC_BUS_CLK_AXICFG0_AXI>,
                 <&clkgen JH7110_NOC_BUS_CLK_GPU_AXI>,
                 <&clkgen JH7110_NOC_BUS_CLK_VDEC_AXI>,
                 <&clkgen JH7110_NOC_BUS_CLK_VENC_AXI>,
                 <&clkgen JH7110_NOC_BUS_CLK_DISP_AXI>,
                 <&clkgen JH7110_NOC_BUS_CLK_ISP_AXI>,
                 <&clkgen JH7110_NOC_BUS_CLK_STG_AXI>,
```

```
                <&clkgen JH7110_VOUT_SRC>,
                <&clkgen JH7110_VOUT_TOP_CLK_VOUT_AXI>,
                <&clkgen JH7110_AHB1>,
                <&clkgen JH7110_VOUT_TOP_CLK_VOUT_AHB>,
                <&clkgen JH7110_VOUT_TOP_CLK_HDMITX0_MCLK>,
                <&clkgen JH7110_I2STX_4CH0_BCLK_MST>,
                <&clkvout JH7110_U0_DC8200_CLK_PIX0>,
                <&clkvout JH7110_U0_DC8200_CLK_PIX1>,
                <&clkvout JH7110_U0_DC8200_CLK_AXI>,
                <&clkvout JH7110_U0_DC8200_CLK_CORE>,
                <&clkvout JH7110_U0_DC8200_CLK_AHB>,
                <&clkgen JH7110_VOUT_TOP_CLK_VOUT_AXI>,
                <&clkvout JH7110_DOM_VOUT_TOP_LCD_CLK>,
                <&hdmitx0_pixelclk>,
                <&clkvout JH7110_DC8200_PIX0>,
                <&clkvout JH7110_U0_DC8200_CLK_PIX0_OUT>,
                <&clkvout JH7110_U0_DC8200_CLK_PIX1_OUT>;
        clock-names = "noc_cpu","noc_cfg0","noc_gpu","noc_vdec","noc_venc",
                      "noc_disp","noc_isp","noc_stg","vout_src",
                      "top_vout_axi","ahb1","top_vout_ahb",
                      "top_vout_hdmiTX0","i2stx","pix_clk","vout_pix1",
                      "axi_clk","core_clk","vout_ahb",
                      "vout_top_axi","vout_top_lcd","hdmitx0_pixelclk","dc8200_pix0",
                      "dc8200_pix0_out","dc8200_pix1_out";
        resets = <&rstgen RSTN_U0_DOM_VOUT_TOP_SRC>,
                 <&rstgen RSTN_U0_DC8200_AXI>,
                 <&rstgen RSTN_U0_DC8200_AHB>,
                 <&rstgen RSTN_U0_DC8200_CORE>,
                 <&rstgen RSTN_U0_NOC_BUS_CPU_AXI_N>,
                 <&rstgen RSTN_U0_NOC_BUS_AXICFG0_AXI_N>,
                 <&rstgen RSTN_U0_NOC_BUS_APB_BUS_N>,
                 <&rstgen RSTN_U0_NOC_BUS_DISP_AXI_N>,
                 <&rstgen RSTN_U0_NOC_BUS_STG_AXI_N>;
        reset-names = "rst_vout_src","rst_axi","rst_ahb","rst_core",
                      "rst_noc_cpu","rst_noc_axicfg0","rst_noc_apb",
                      "rst_noc_disp","rst_noc_stg";
        power-domains = <&pwrc JH7110_PD_VOUT>;
};
```

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

- **compatible**: Compatibility information, used to associate the display controller and its target device.
- **dss-syscon**: The SYSCON register(s) of the display panel.
- **reg**: Register base address "`0x29400000`" and range "`0x100`".
- **interrupts**: Hardware interrupt ID.
- **status**: The work status of the display controller module. To enable the module, set this bit as "`okay`" or to disable the module, set this bit as "`disabled`".
- **clocks**: The clocks used by the display controller module.
- **clock-names**: The names of the above clocks.
- **resets**: The reset signals used by the display controller module.
- **reset-names**: The names of the above reset signals.
- **power-domains**: The power supply domain of the display controller module.

In the file `jh7110-common.dtsi`, you can find the device tree configuration of DC8200 (endpoint) as the following code block:

```
&dc8200 {
 status = "okay";

 dc_out: port {
  #address-cells = <1>;
  #size-cells = <0>;
  dc_out_dpi0: endpoint@0 {
   reg = <0>;
```

```
   remote-endpoint = <&hdmi_input0>;
  };
  dc_out_dpi1: endpoint@1 {
   reg = <1>;
   remote-endpoint = <&hdmi_in_lcdc>;
  };

  dc_out_dpi2: endpoint@2 {
   reg = <2>;
   remote-endpoint = <&mipi_in>;
  };
 };
};
```

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

- **reg**: The register ID of the endpoint device.
- **remote-endpoint**: The endpoint device type of the display controller output.

2.2. Kernel Menu Configuration

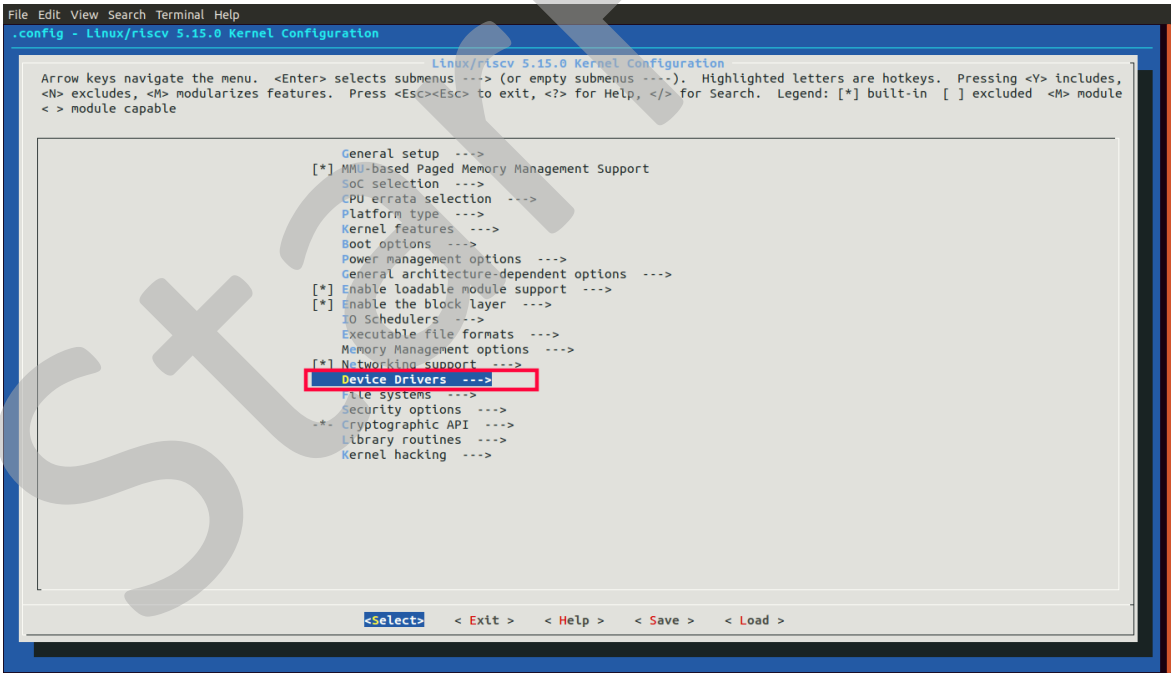
Follow the steps below to enable the kernel configuration for display controller.

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

   ```
   make linux-menuconfig
   ```

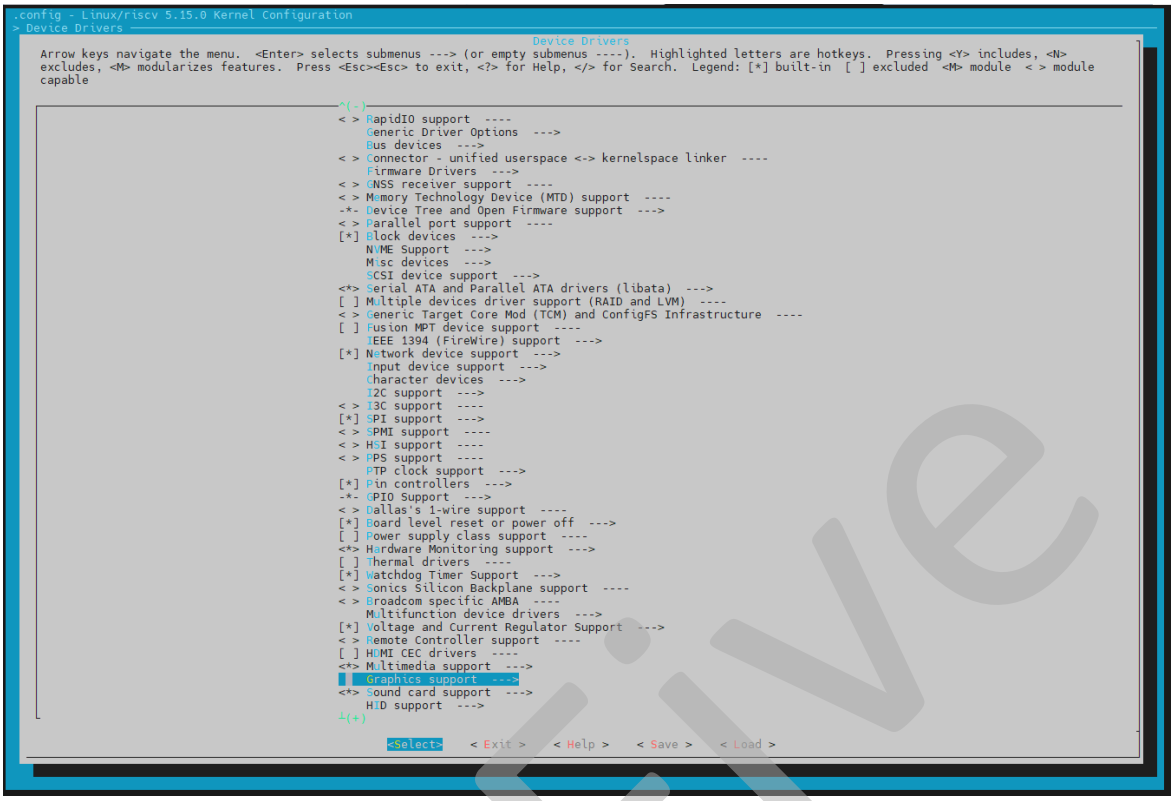
2. Enter the **Device Drivers** menu.

   Figure 2-1 Device Drivers



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

Figure 2-2 Graphics Support



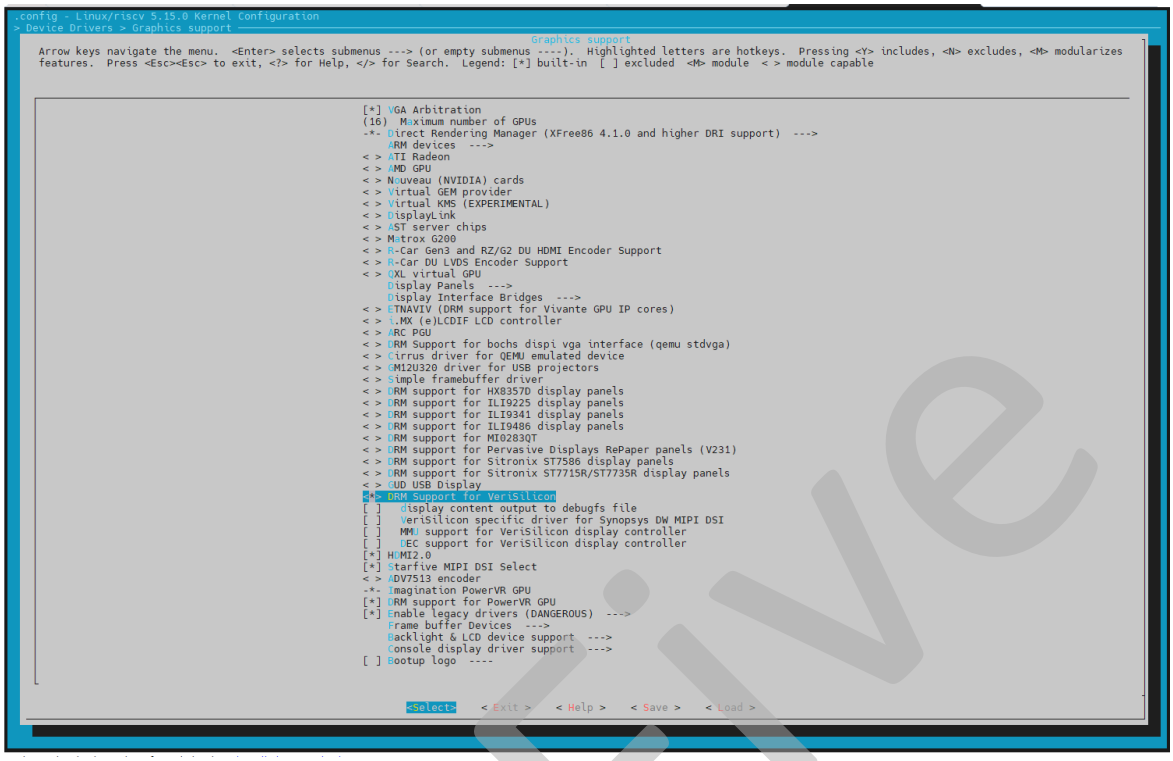
4. Continue your settings per the sections below depending on your target output devices.
    - For HDMI output *(on page 14)*
    - For MIPI output *(on page 15)*
    - For RGB2HDMI output *(on page 16)*

For HDMI Output

Continue your settings with the following steps to enable the kernel configuration for HDMI output.

1. In the **Graphics support** menu, select the **HDMI2.0** option.

Figure 2-3 HDMI2.0



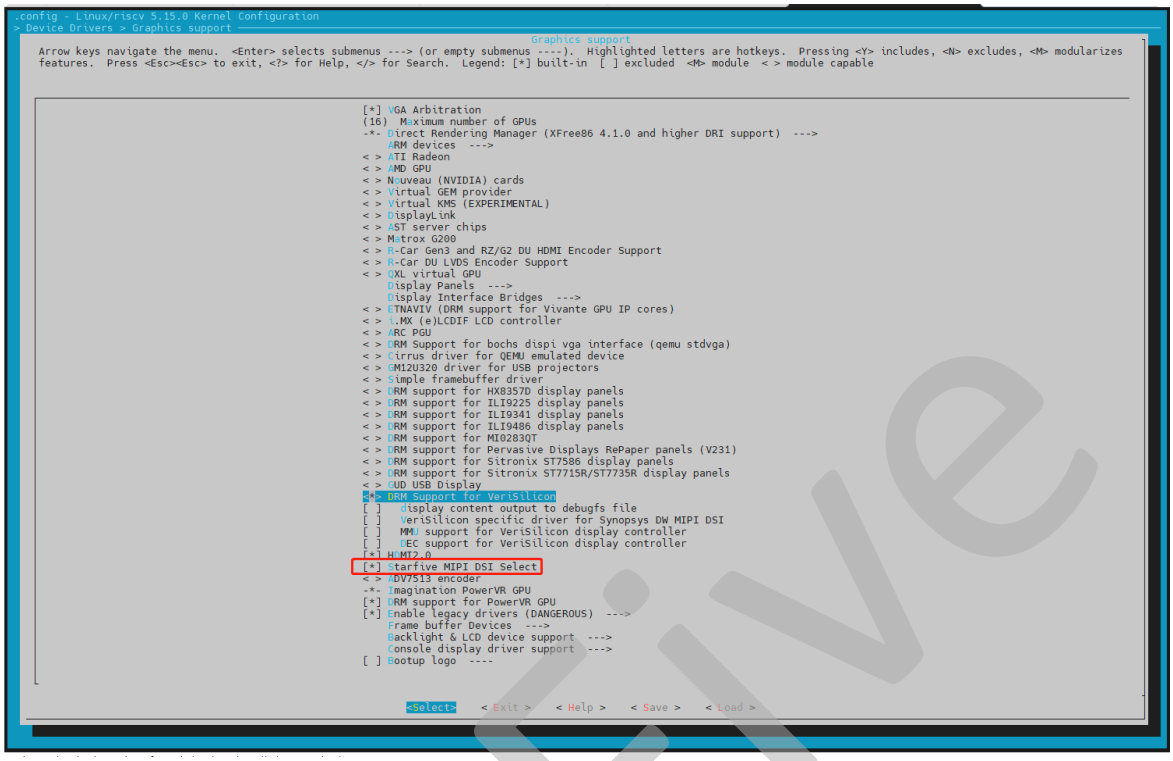
2. Save your change before you exit the kernel configuration dialog.

For MIPI Output

Continue your settings with the following steps to enable the kernel configuration for MIPI output.

1. In the **Graphics support** menu, select the **Starfive MIPI DSI Select** option.

Figure 2-4 MIPI DSI Select



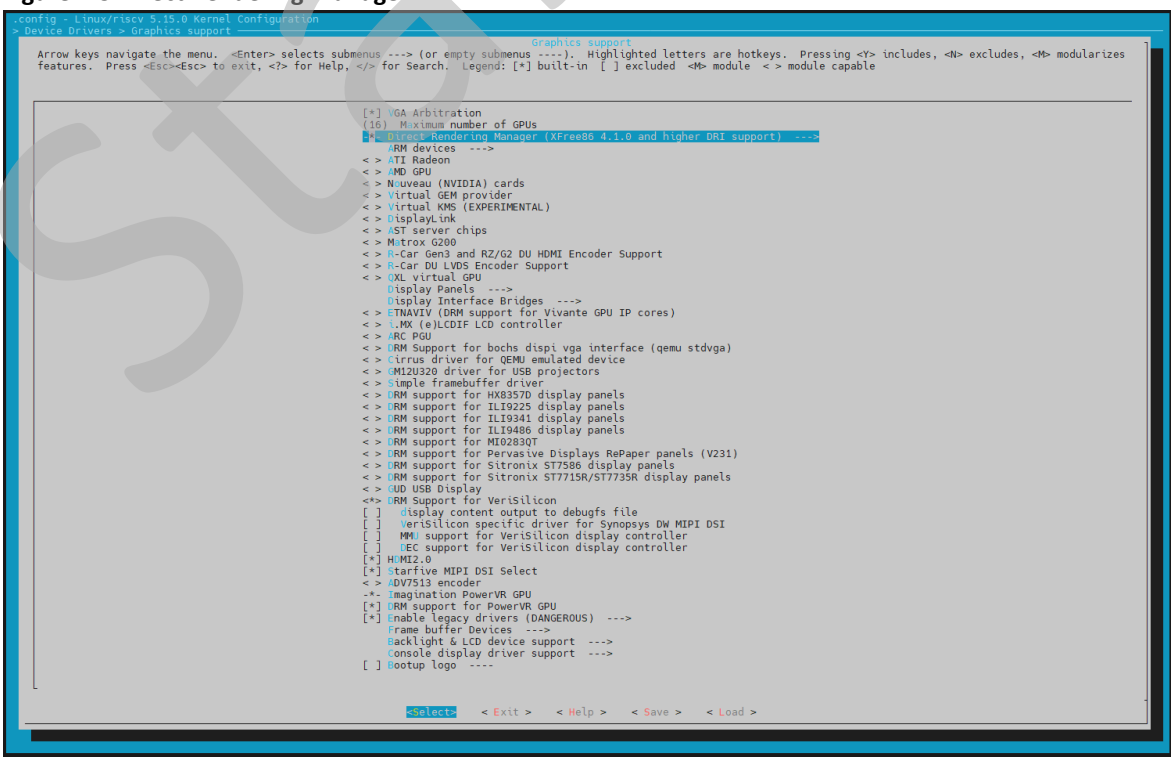
2. Save your change before you exit the kernel configuration dialog.

For RGB2HDMI Output

Continue your settings with the following steps to enable the kernel configuration for RGB2HDMI output.

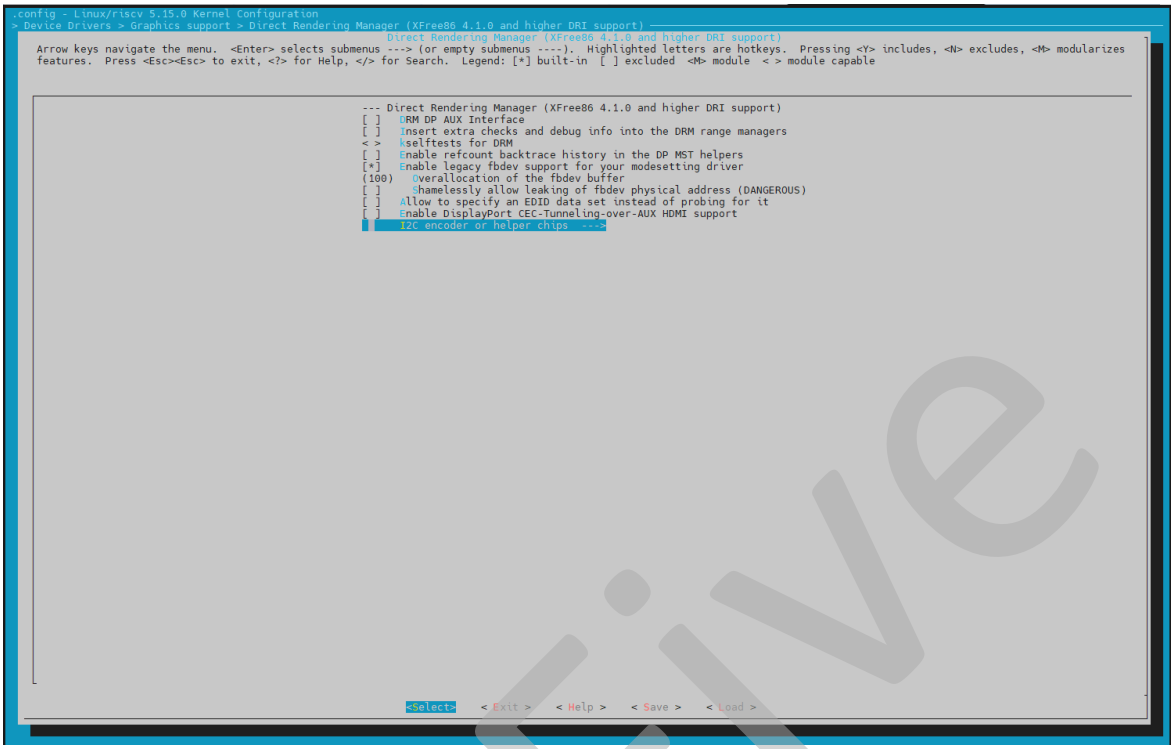
1. In the **Graphics support** menu, select and enter the **Direct Rendering Manager** menu.

Figure 2-5 Direct Rendering Manager



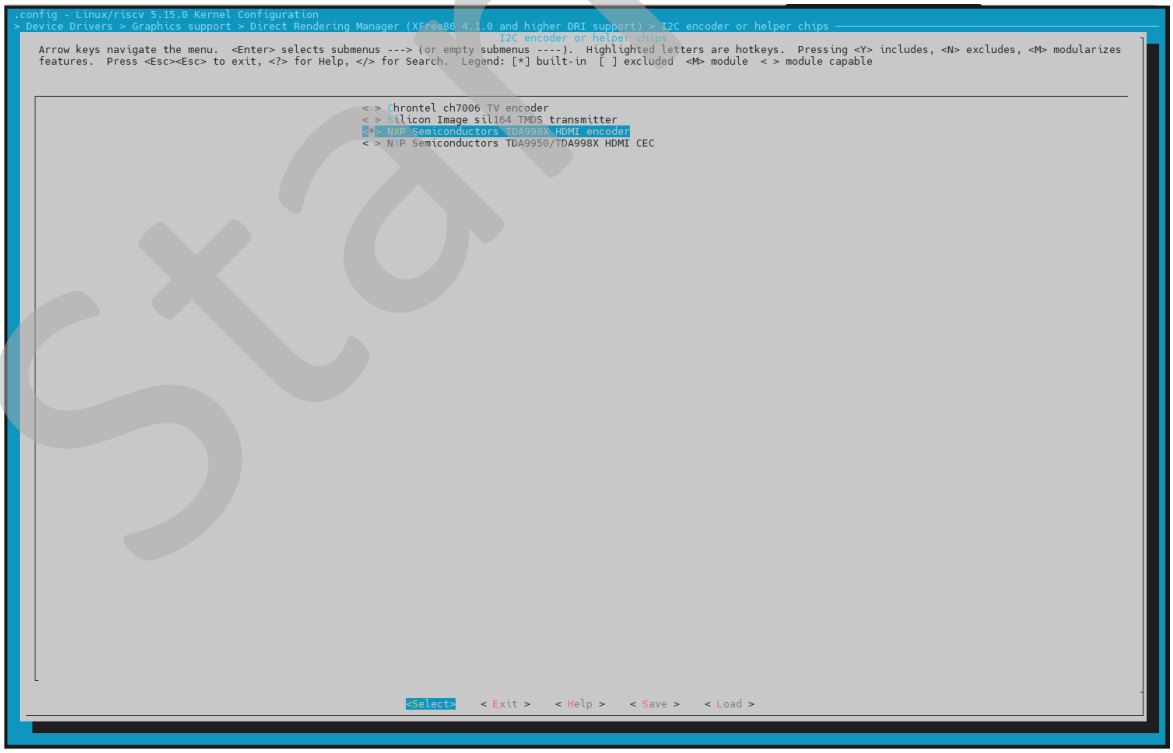
2. In the **Direct Rendering Manager** menu, select and enter the **I2C encoder or helper chips** menu.

Figure 2-6 I2C Encoder or Other Helper



3. In the **I2C encoder or helper chips** menu, select the **NXP Semiconductors TDA998X HDMI encoder** option.

Figure 2-7 NXP Semiconductors TDA998X



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

2.3. Driver Configuration

The following code block shows the driver configuration.

```
CONFIG_DRM_VERISILICON=y
```

3. Debug Method

3.1. Test Case Configuration

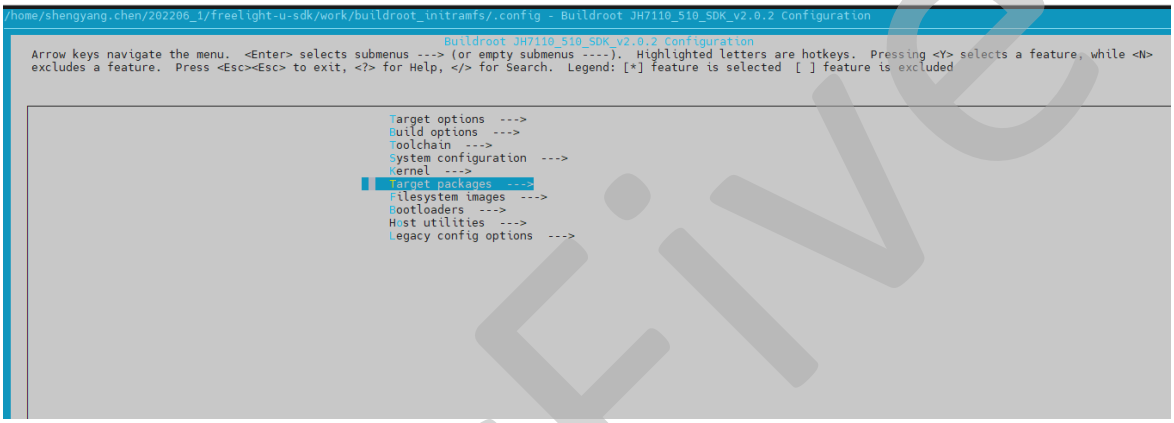
Follow the steps below to enable the kernel configuration for HDMI.

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

```
make linux-menuconfig
```

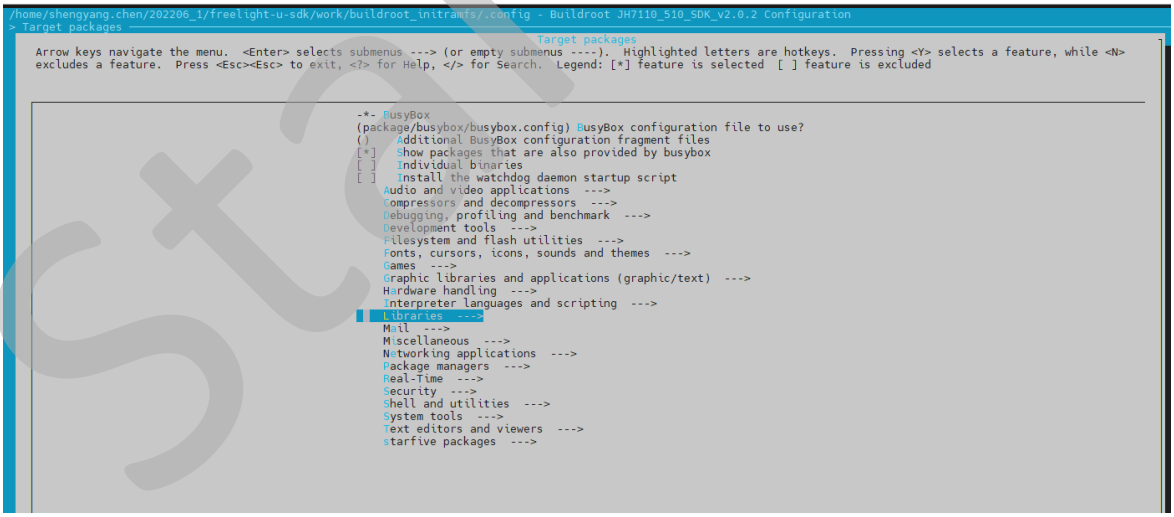
2. Enter the **Target packages** menu.

Figure 3-1 Target Packages



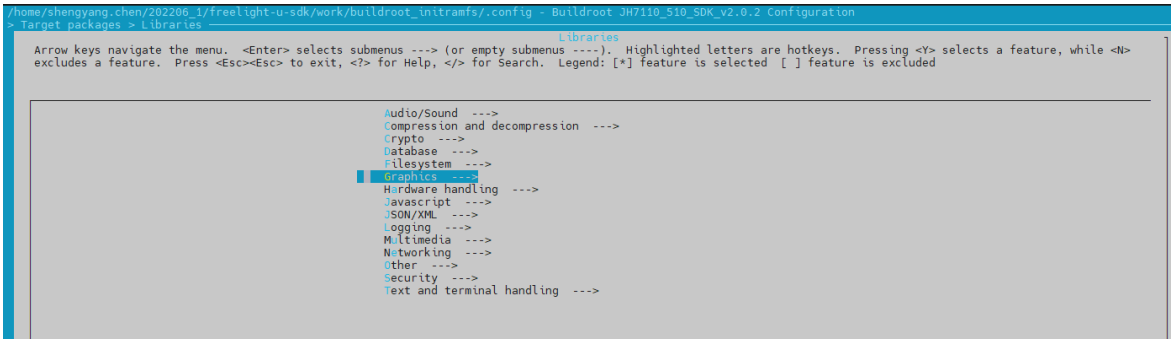
3. Enter the **Libraries** menu.

Figure 3-2 Libraries



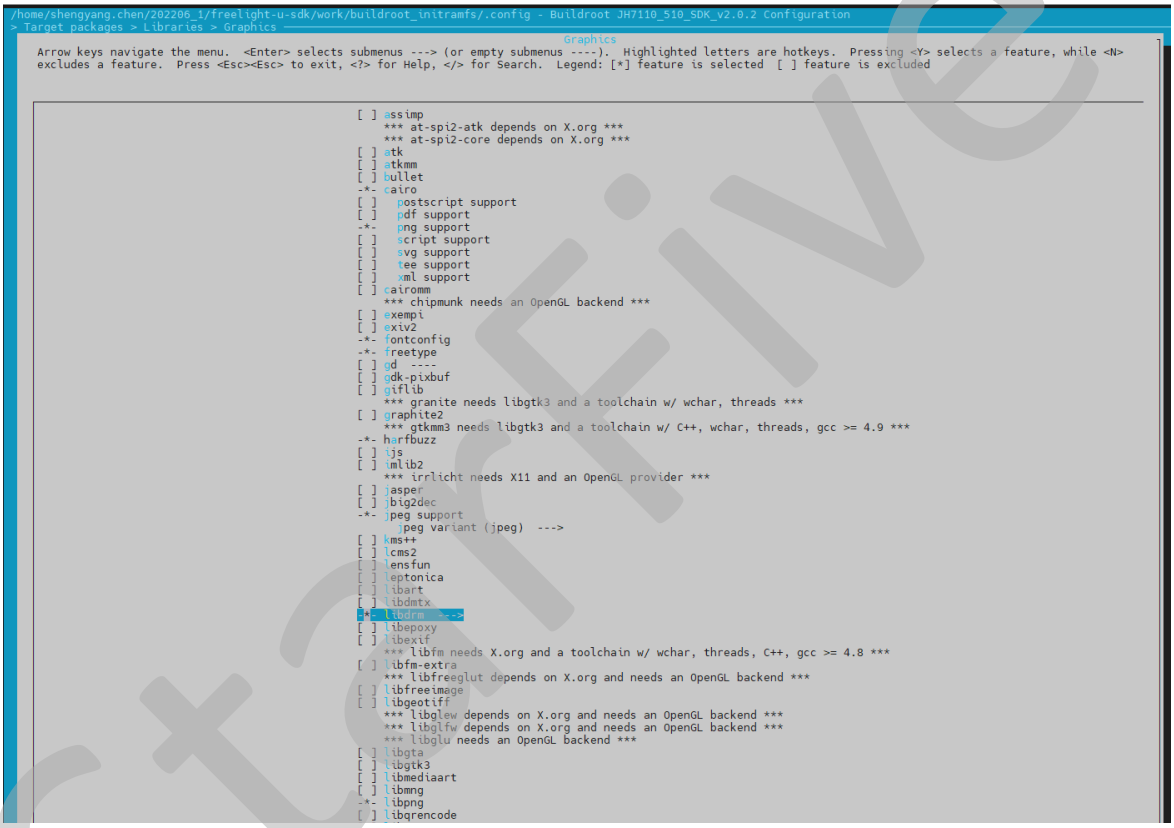
4. Enter the **Graphics** menu.

Figure 3-3 Graphics



5. Enter the **libdrm** menu.

Figure 3-4 libdrm



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

Figure 3-5 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.

3.2. Before Debug

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

Figure 3-6 Start-up Logs

Table 3-1 Start-up Logs

Legend	Description
①	HDMI work status
②	RGB2HDMI work status
③	MIPI work status
④	Display controller work status

The log lines showing display controller and the HDMI are required before the debug.

Note:
Verify the connection status if you cannot find the above log records.

3.3. Debug Display

Follow the steps below to debug the display functions for your JH7110.

1. Follow the steps in Test Case Configuration *(on page 19)* to configure the test environment.

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

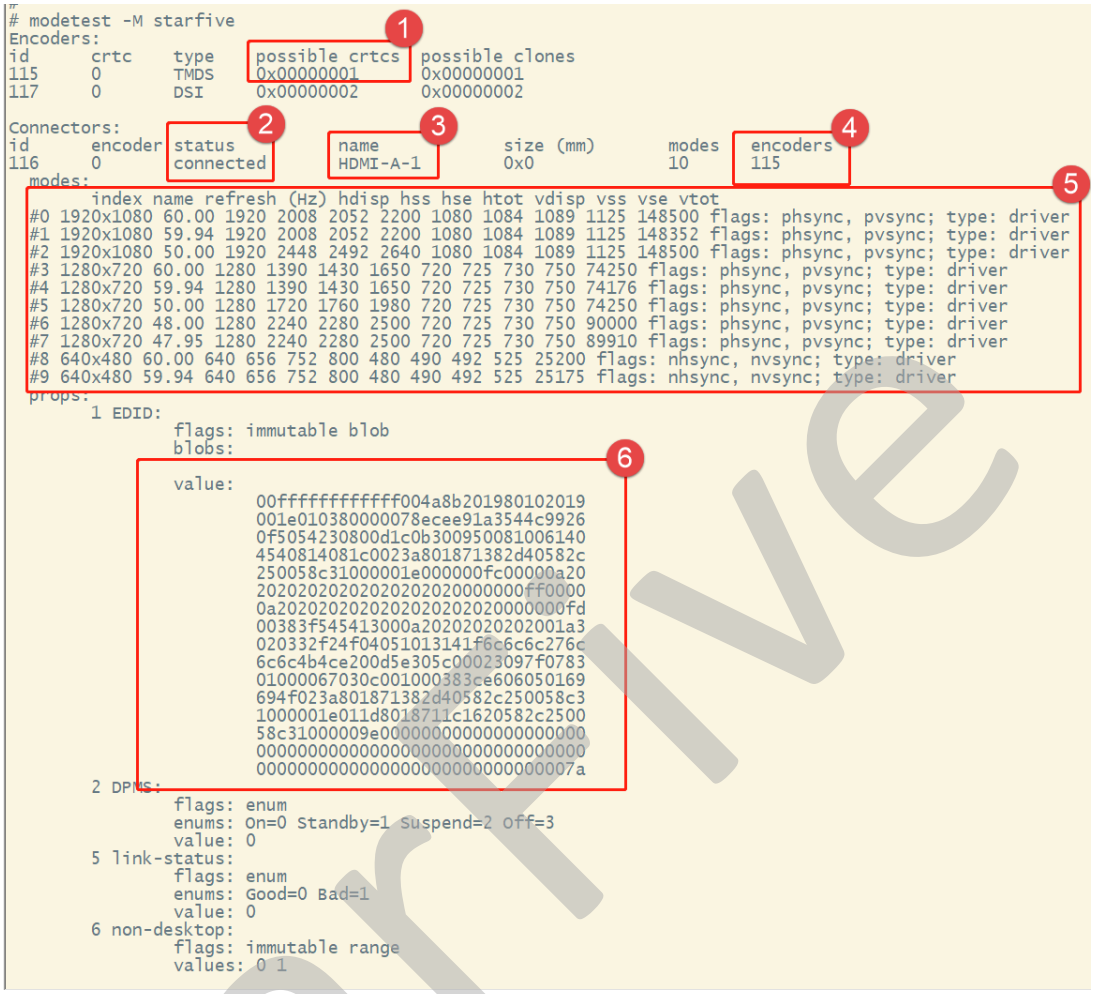
2. After you have completed the kernel start-up, use the following command to verify the display functions and connection status.

   ```
   modetest -M starfive
   ```

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

◦ Debug output 1:

Figure 3-7 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
                        00383f545413000a20202020202001a3
                        020332f24f04051013141f6c6c6c276c
                        6c6c4b4ce200d5e305c00023097f0783
                        01000067030c001000383ce606050169
                        694f023a801871382d40582c250058c3
                        1000001e011d8018711c1620582c2500
                        58c31000009e000000000000000000000
                        000000000000000000000000000000000
                        000000000000000000000000000000000
                        00000000000000000000000000000007a
        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 3-2 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 3-8 Debug Display 2

Table 3-3 Debug Display 2

Legend	Label	Description
①	**id**	The CRTC 0x00000001 mentioned in row ① of table Table 3-2 : Debug Display 1 *(on page 22)*, which means the CRTC is available for use.

Legend	Label	Description
②	**id**	The CRTC 0x00000002 mentioned in row ① of table Table 3-2 : Debug Display 1 *(on page 22)*, 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 3-9 Debug Display 3

Table 3-4 Debug Display 3

Legend	Description
①	The CRTC and its connected plane

3.4. Test Example

For HDMI Output

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

```
modetest -M starfive -D 0 -a -s 116@31:1920x1080 -P 39@31:1920x1080@RG16 -Ftiles
```

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

- **116@31:1920x1080** - <Connector ID>@<CRTC ID>: <Resolution>
- **39@31:1920x1080@RG16** - <Plane ID>@<CRTC ID>: <Resolution>@<Format>

For MIPI Output

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

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

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>

For RGB2HDMI Output

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

```
modetest -M starfive -D 0 -a -s 118@35:1920x1080 -P 74@35:1920x1080@RG16 -Ftiles
```

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

- **118@35:1920x1080** - <Connector ID>@<CRTC ID>: <Resolution>
- **74@35:1920x1080@RG16** - <Plane ID>@<CRTC ID>: <Resolution>@<Format>

For Both MIPI and RGB2HDMI Outputs

If your board is connected with both a MIPI and a RGB2HDMI output devices, the following commands show an example for testing on each of them.

- For MIPI:

  ```
  modetest -M starfive -D 0 -a -s 120@35:800x480 -P 74@35:800x480@RG16
  ```

  - **120@35:800x480** - <Connector ID>@<CRTC ID>: <Resolution>
  - **74@35:800x480@RG16** - <Plane ID>@<CRTC ID>: <Resolution>@<Format>

- For RGB2HDMI:

  ```
  modetest -M starfive -D 0 -a -s 118@35:1920x1080 -P 74@35:1920x1080@RG16 -Ftiles
  ```

  - **118@35:1920x1080** - <Connector ID>@<CRTC ID>: <Resolution>
  - **74@35:1920x1080@RG16** - <Plane ID>@<CRTC ID>: <Resolution>@<Format>

Output Result

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

Figure 3-10 Test Example