



Graphics Development

User Guide

Issue 04

Date 2013-06-21

Copyright © HiSilicon Technologies Co., Ltd. 2011-2013. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of HiSilicon Technologies Co., Ltd.

Trademarks and Permissions



HISILICON, and other HiSilicon icons are trademarks of HiSilicon Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between HiSilicon and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the 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.

HiSilicon Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: <http://www.hisilicon.com>

Email: support@hisilicon.com



About This Document

Purpose

This document provides two schemes for graphics development. The schemes include scheme description, derivative scheme, development process, application scenarios, and advantages and limitations.

Related Versions

The following table lists the product versions related to this document.

Product Name	Version
Hi3531	V100
Hi3532	V100
Hi3521	V100
Hi3520A	V100
Hi3518	V100
Hi3520D	V100
Hi3515A	V100
Hi3515C	V100

Intended Audience






This document is intended for:

- Technical support personnel
- Software development engineers



Symbol Conventions

The symbols that may be found in this document are defined as follows.

Symbol	Description
 DANGER	Alerts you to a high risk hazard that could, if not avoided, result in serious injury or death.
 WARNING	Alerts you to a medium or low risk hazard that could, if not avoided, result in moderate or minor injury.
 CAUTION	Alerts you to a potentially hazardous situation that could, if not avoided, result in equipment damage, data loss, performance deterioration, or unanticipated results.
 TIP	Provides a tip that may help you solve a problem or save time.
 NOTE	Provides additional information to emphasize or supplement important points in the main text.

Change History

Changes between document issues are cumulative. Therefore, the latest document issue contains all changes made in previous issues.

Issue 04(2013-06-21)

The descriptions related to the Hi3515C are added.

Issue 03(2013-04-03)

The descriptions related to the Hi3515A are added.

Issue 02(2013-02-05)

The descriptions related to the Hi3520D are added.

Issue 01(2012-09-20)

This issue is the first official release.

Issue 00B50 (2012-08-09)

This issue is the sixth draft release, which incorporates the following changes:

Chapter 1 Introduction to Graphics Layers

Section 1.2.5 is added.

Issue 00B40 (2012-06-08)

This issue is the fifth draft release, which incorporates the following changes:



The descriptions of the Hi3520A are added.

Issue 00B30 (2012-04-20)

This issue is the fourth draft release, which incorporates the following changes:

The descriptions of the Hi3521 are added.

Issue 00B20 (2012-02-15)

This issue is the third draft release, which incorporates the following changes:

Chapter 1 Description of Graphics Layers

In Table 1-1, the description "G5 and G6 cannot be bound to the same display device" is added.

Issue 00B10 (2012-01-15)

This issue is the second draft release, which incorporates the following changes:

Chapter 2 Recommended Schemes for Graphics Layer Development

Section 2.3.1 is updated.

Issue 00B01 (2011-11-10)

This issue is the first draft release.



Contents

About This Document.....	i
1 Introduction to Graphics Layers	1
1.1 Overview	1
1.2 Architecture of Graphics Layers.....	1
1.2.1 Architecture of Hi3531 Graphics Layers	1
1.2.2 Architecture of Hi3532 Graphics Layers	2
1.2.3 Architecture of Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C Graphics Layers	2
1.2.4 Architecture of the Hi3518 Graphics Layer	3
2 Recommended Schemes for Graphics Development	4
2.1 Overview	4
2.2 GUI Scheme with a Single Graphics Layer	4
2.2.1 Introduction.....	4
2.2.2 Derivative Scheme	6
2.2.3 Development Process	7
2.2.4 Application Scenarios	7
2.2.5 Advantages and Limitations.....	7
2.3 GUI Scheme with Multiple Graphics Layers	8
2.3.1 Introduction.....	8
2.3.2 Development Process.....	10
2.3.3 Application Scenarios	11
2.3.4 Advantages and Limitations.....	11



Figures

Figure 2-1 Schematic diagram of the scheme with a graphics layer	5
Figure 2-2 Schematic diagram of the derivative scheme.....	6
Figure 2-3 Displaying the OSD, GUI, and cursor on DHD0 and displaying the OSD and cursor on DSD0	9
Figure 2-4 Switching the GUI and cursor to DSD0	10



Tables

Table 1-1 Relationships among the FB device files of the Hi3531, graphics layers, and output devices	1
Table 1-2 Relationships among the FB device files, graphics layers, and output devices of the Hi3521,Hi3520A , Hi3520D, Hi3515A or Hi3515C	3
Table 1-3 Relationship among the FB device file, graphics layer, and output device of the Hi3518	3



1 Introduction to Graphics Layers

1.1 Overview

The HiSilicon digital media processing platform (HiMPP) provides a set of mechanisms for developing graphical user interfaces (GUIs). The mechanisms consist of:

- Two-dimensional engine (TDE). It processes graphics through hardware acceleration.
- HiSilicon frame buffer (HiFB). It manages graphics layers. Besides the basic functions of the Linux FB, it also provides the extended functions such as inter-layer colorkey and inter-layer alpha.



NOTE

- For details on how to use the TDE, see the *TDE API Reference*.
- For details on how to use the HiFB, see the *HiFB Development Guide* and *HiFB API Reference*.

1.2 Architecture of Graphics Layers

1.2.1 Architecture of Hi3531 Graphics Layers

The Hi3531 supports eight display devices: two high-definition (HD) display devices (DHD0 and DHD1) and six standard-definition (DSD) display devices (DSD0–DSD5). The Hi3531 also supports seven graphics layers (G0–G6). Where, G5 and G6 are the cursor layers.



NOTE

For details about the interfaces and timings supported by each output device, see section 11.2 "VDP" in the *Hi3531 H.264 Codec Processor Data Sheet*.

[Table 1-1](#) shows the relationships among the FB device files of the Hi3531, graphics layers, and output devices.

Table 1-1 Relationships among the FB device files of the Hi3531, graphics layers, and output devices

FB Device File	Graphics Layer	Corresponding Display Device
/dev/fb0	G0	G0 is displayed only on DHD0.
/dev/fb1	G1	G1 is displayed only on DHD1.



FB Device File	Graphics Layer	Corresponding Display Device
/dev/fb2	G2	G2 is displayed only on DSD0.
/dev/fb3	G3	G3 is displayed only on DSD1.
/dev/fb4, /dev/fb5, /dev/fb6	G4, cursor layer0 (G5), cursor layer1 (G6)	<p>G4–G6 can be displayed on DHD0, DHD1, DSD0, and DSD1. You can specify the display device by calling the related binding interfaces. G5 and G6 cannot be bound to the same display device.</p> <p>G5 and G6 are the cursor layers and are always the uppermost layers when multiple layers are overlaid on a display device. G4 is the layer under G5. If the video layer, G0, G4, and G5 are overlaid on DHD0, the overlaid sequence from the bottom up is: video layer, G0, G4, and G5. G5 and G6 can act as the hardware cursor layers or software cursor layers, which is determined by the softcursor parameter. When G5 and G6 are the hardware cursor layers, the operations are the same as those of other graphic layers. When G5 and G6 are the software cursor layers, they must be operated by the software cursor dedicated interface that is provided by the HiFB.</p>

NOTE

To display graphics layers, you must configure and enable VO devices by calling the interfaces of the VOU, and operate graphics layers by calling the interfaces of the Hi3531 HiFB.

1.2.2 Architecture of Hi3532 Graphics Layers

The Hi3532 supports a display device (DHD0), a graphics layer (G0), and a cursor layer (G5). Except that G0 does not support compression and G5 is always displayed on DHD0 without binding, the other features are the same as those of the Hi3531.

NOTE

For details about the interfaces and timings supported by each output device, see section 10.2 "VDP" in the *Hi3532 H.264 Encoding Processor Data Sheet*.

1.2.3 Architecture of Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C Graphics Layers

The Hi3521, Hi3520A, Hi3520D, Hi3515A or Hi3515C supports three display devices (DHD0, DSD0, and DSD1) and four graphics layers (G0–G3). G3 is the cursor layer.

NOTE

For details about the interfaces and timings supported by each output device, see section 11.2 "VDP" in the *Hi35xx H.264 Codec Processor Data Sheet*.

[Table 1-2](#) shows the relationships among the FB device files, graphics layers, and output devices of the Hi3521.



Table 1-2 Relationships among the FB device files, graphics layers, and output devices of the Hi3521, Hi3520A, Hi3520D, Hi3515A or Hi3515C

FB Device File	Graphics Layer	Corresponding Display Device
/dev/fb0	G0	G0 is displayed only on DHD0.
/dev/fb1	G1	G1 is displayed only on DSD0.
/dev/fb2	G2	G2 is displayed only on DSD1.
/dev/fb3	cursor layer0 (G3)	<p>G3 is displayed on DHD0, DHD1, DSD0, or DSD1. You can specify the display device by calling the related binding interfaces.</p> <p>G3 are the cursor layers and are always the uppermost layers when multiple layers are overlaid on a display device. If the video layer, G0, and G3 are overlaid on DHD0, the overlaid sequence from the bottom up is: video layer, G0, and G3.</p> <p>G3 can act as the hardware cursor layers or software cursor layers, which is determined by the softcursor parameter. When G5 and G6 are the hardware cursor layers, they are used in the same way as that of other graphics layers are used. When G5 and G6 are the software cursor layers, they must be operated by using the software cursor dedicated interfaces that are provided by the HiFB.</p>

1.2.4 Architecture of the Hi3518 Graphics Layer

The Hi3518 supports only one SD display device (DSD1) and one graphics layer (G3).

[Table 1-3](#) shows the relationship among the FB device file, graphics layer, and output device of the Hi3518.

Table 1-3 Relationship among the FB device file, graphics layer, and output device of the Hi3518

FB Device File	Graphics Layer	Corresponding Display Device
/dev/fb0	G3	G2 is displayed only on DSD1.



2 Recommended Schemes for Graphics Development

2.1 Overview

In the surveillance field, the GUI contents of an output device include:

- Backend OSD: It includes the dividing lines of the displayed picture, channel IDs, and time that specify the display layout of multiple pictures.
- GUI: It includes various menus and progress bars. You can configure devices through the GUI.
- Cursor: It provides user-friendly and convenient menus.

The preceding GUI contents can be implemented through one or more graphics layers. As the Hi3531, Hi3532 or Hi3521 provides multiple graphics layers, this chapter describes the following recommended schemes to instruct you to use those graphics layers in a correct, reasonable, and effective manner, satisfying the requirements in various output GUI applications.

2.2 GUI Scheme with a Single Graphics Layer

2.2.1 Introduction

In this scheme, each device displays the backend OSD, GUI, and cursor through one graphics layer. The cursor can also be displayed at a cursor layer.

To be specific, each output device displays its backend OSD and GUI through one graphics layer. The GUI is drawn in an independent buffer, the backend OSD is drawn in the display buffer, and then alpha blending is performed by using the TDE. The cursor is displayed on a cursor layer or the shared graphics layers of the backend OSD and GUI. When the shared graphics layer is used, the cursor can be drawn in the GUI buffer.

The following mechanisms are used in this scheme:

- The backend OSD of each device is drawn in its display buffer.
For example, the dividing lines, channel ID, or time is drawn in the FB corresponding to each graphics layer.
- Each device has a GUI canvas that is updated partially when the GUI changes.

That is, each device draws the GUI by using an independent buffer (also known as GUI canvas). This canvas needs to be updated partially when the GUI changes.

- The GUI canvas is entirely transferred to the display buffer of the corresponding graphics layer.

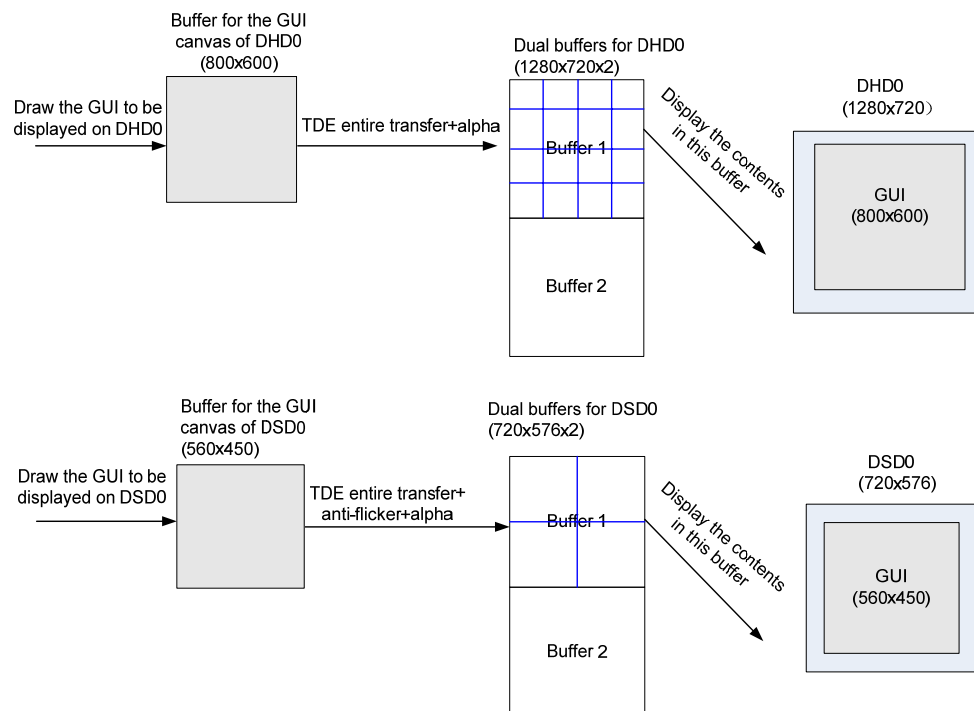
When a drawn canvas is entirely transferred to the corresponding display buffer, transparent blending between the GUI and OSD can be implemented through the TDE. Each time the GUI or OSD changes, the blended area between the GUI and OSD does not need to be calculated based on the local information, because blending is performed on the entire canvas and OSD.

- Dual display buffers

To avoid the drawing process being visible when a display buffer is used for drawing and displaying concurrently, dual display buffers, HIFB_LAYER_BUF_DOUBLE or HIFB_LAYER_BUF_DOUBLE_IMMEDIATE in the extended mode is recommended. That is, the HiFB module assigns dual display buffers with the same size for drawing and displaying alternatively. For example, when buffer 2 is displayed on the VOU, buffer 1 is used for drawing. For the FB standard mode, after PAN_DISPLAY and FBIOFLIP_SURFACE of the HiFB are called, the VO device is notified that buffer 1 should be displayed. For the FB extended mode, after FBIO_REFRESH of the HiFB is called, the VO device is notified that buffer 1 should be displayed.

Figure 2-1 shows the schematic diagram of the scheme with a graphics layer

Figure 2-1 Schematic diagram of the scheme with a graphics layer



When the backend OSD or GUI changes, the OSD or GUI needs to be drawn again in the display buffer.

- When the backend OSD changes (such as the 16-picture dividing line is switched to the 9-picture dividing line), you need to empty the display buffer, draw a new OSD, and then transfer the entire GUI to the display buffer.

- When the GUI changes, you also need to empty the display buffer, draw a new OSD, and then entirely transfer the new GUI to the display buffer.

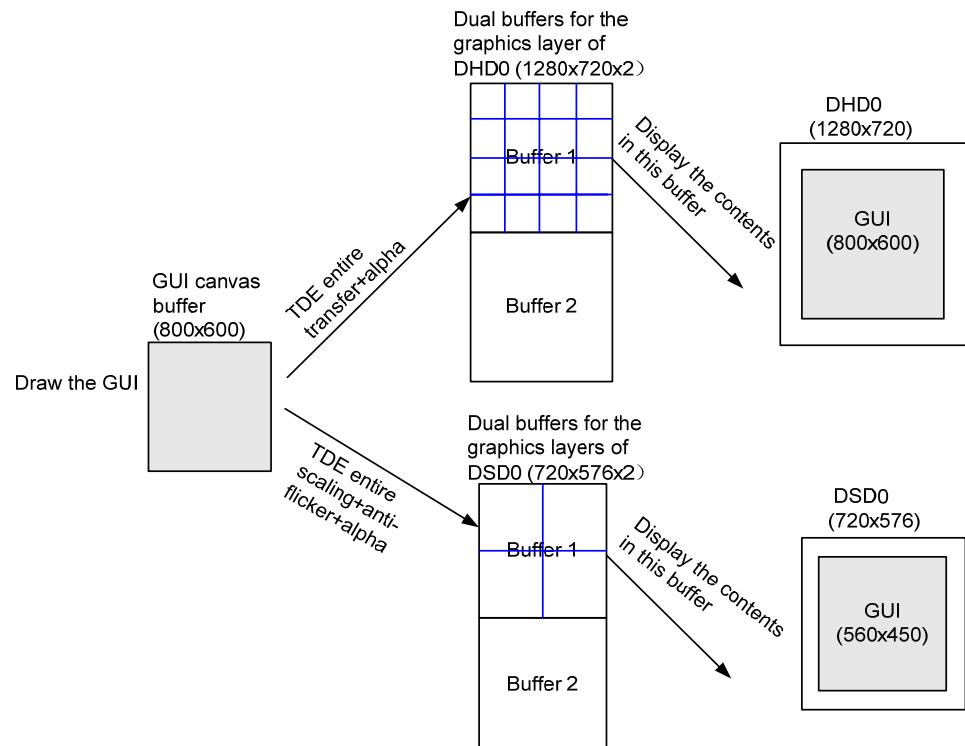
2.2.2 Derivative Scheme

When DSD0 and DHD0 display the same GUI, the preceding scheme can be simplified as a derivative scheme that requires only a GUI canvas buffer.

- The canvas size (800x600) is the same as that of DHD0 GUI. You can prepare a set of pictures based on the specifications (800x600) of DHD0 GUI. Each time the GUI changes, you need to draw the canvas partially. DSD0 GUI is obtained after scaling and anti-flicker operations are performed on the entire canvas. The DSD GUI is not as clear as the DHD GUI.
- For the HD device, each time after the canvas is updated, only the entire transfer operation by using the TDE is required, because the canvas size is the same as the GUI size. For DSD0, each time after the canvas is updated, the entire canvas needs to be scaled by using the TDE based on the size of display buffer for the graphics layer bound to DSD0, because DSD0 is an interlaced device. In addition, anti-flicker is performed.

Figure 2-2 shows the schematic diagram of the derivative scheme.

Figure 2-2 Schematic diagram of the derivative scheme





2.2.3 Development Process

Development Process of the Scheme with a Graphics Layer

This section takes the GUIs and OSDs of DHD0 and DSD0 as examples. The dividing lines can divide the entire screen into 16 even pictures for DHD0 or four even pictures for DSD0. In addition, DHD0 and DSD0 display the same GUI concurrently.

If the GUI changes, the scheme is implemented as follows:

Step 1 Clear the free buffers for the graphics layers corresponding to DHD0 and DSD0.

This example assumes that the free buffer is buffer 1 and the VO device displays the contents in buffer 2.

Step 2 Draw 16-picture dividing lines in buffer 1 for the graphics layer corresponding to DHD0.

Step 3 Draw 4-picture dividing lines in buffer 1 for the graphics layer corresponding to DSD0.

Step 4 Refresh the canvas partially.

Step 5 Transfer the entire canvas to buffer 1 for the graphics layer corresponding to DHD0 by using the TDE.

During this process, alpha transparency blending is supported for making the GUI semi-transparent.

Step 6 Scale the entire canvas by using the TDE based on the size of buffer 1 for the graphics layer corresponding to DSD0.

During this process, anti-flicker and alpha transparency blending are supported for making the GUI semi-transparent.

Step 7 Call PAN_DISPLAY to instruct DHD0 to display the contents in buffer 1 for the graphics layer bound to DHD0.

Step 8 Call PAN_DISPLAY to instruct DSD0 to display the contents in buffer 1 for the graphics layer bound to DSD0.

----End

2.2.4 Application Scenarios

This scheme applies to the following scenarios:

- Each device has its backend OSD. For example, the backend OSD is 16-picture layout for DHD0, 8-picture layout for DHD1, 4-picture layout for DSD0, and a single picture with the channel ID information for DSD1.
- Two or more output devices display the same or different GUIs.

2.2.5 Advantages and Limitations

The advantages of the scheme with a single graphics layer are as follows:

- Displaying GUIs on multiple devices at the same time.
- Updating the GUI canvas partially, which saves bus bandwidth and improves the TDE capability.



- Implementing transparency blending between the GUI and OSD through an easy-to-use process. Each time the GUI or OSD changes, the blended area between the GUI and OSD does not need to be calculated based on the local information, because blending is performed on the entire canvas and OSD.
- In the derivative scheme, only a set of GUI pictures are required. In this case, the GUI requirements of the devices with different solutions are met and the space of the flash memory is saved.

The limitation on the derivative scheme is as follows:

- The GUI displayed on the SD device is not as clear as that on the HD device, because the GUI displayed on the SD device is obtained after being scaled.

2.3 GUI Scheme with Multiple Graphics Layers

2.3.1 Introduction

Some chips may provide a dedicated layer for displaying the GUI. This layer is called GUI graphics layer. This scheme displays the backend OSD, GUI, and cursor separately by using three graphics layers.

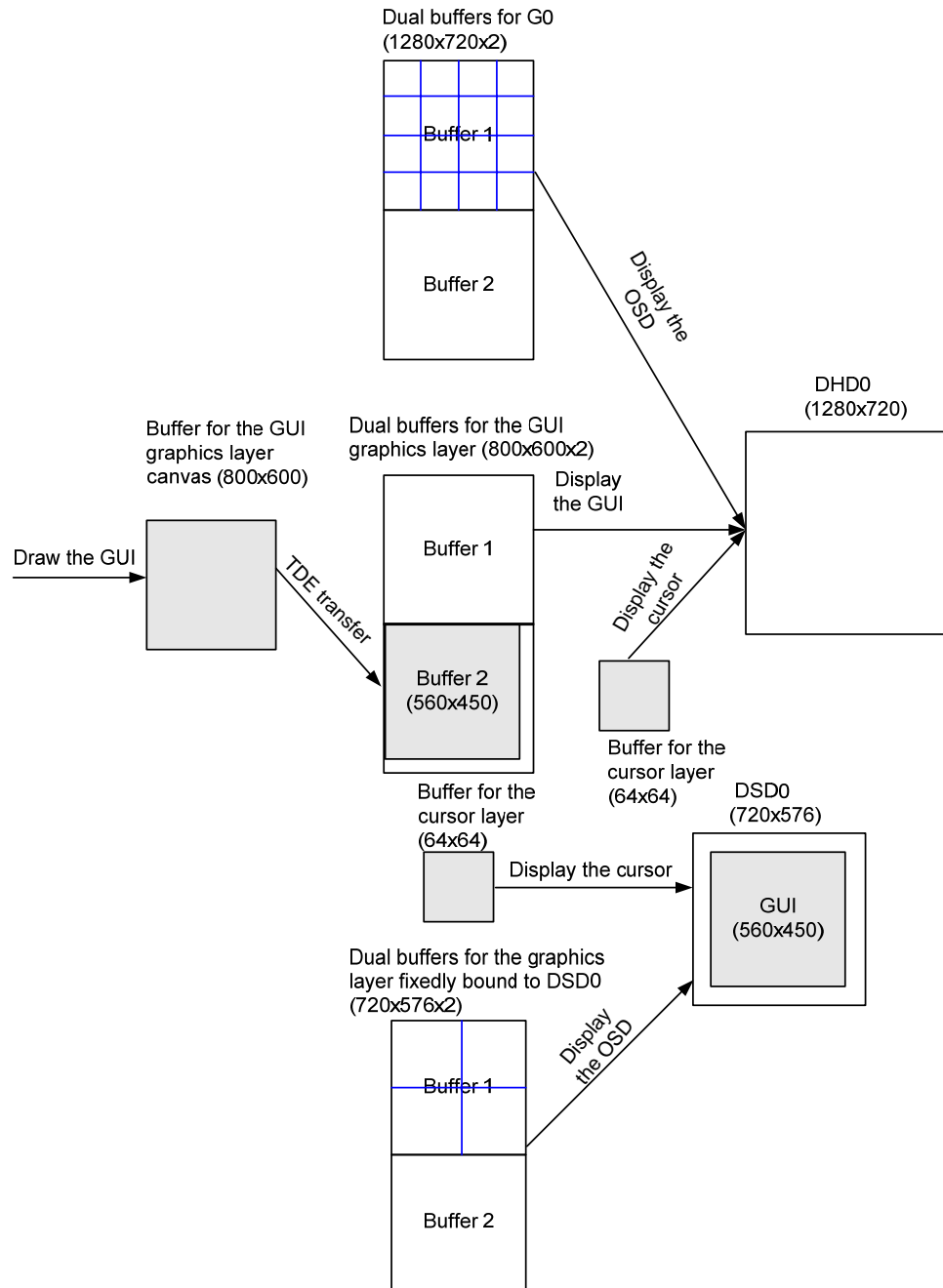
Each device displays its backend OSD by using only one graphics layer. This layer is the one that is fixedly bound to a device.

The GUI is displayed by using the GUI graphics layer. When the GUI is transferred to a device such as DHD0, you can bind DHD0 to both the GUI graphics layer and the graphics layer that is fixedly bound to DHD0. After alpha blending is performed on the two layers, the GUI becomes semi-transparent.

When the GUI is transferred from DHD0 to DHD1, you need to disable the GUI graphics layer, unbind the GUI graphics layer from DHD0, bind the GUI graphics layer to DHD1, and enable the GUI graphics layer by calling the related HiFB interface. Note that you do not need to disable the VO device when dynamically binding a layer.

The cursor is displayed by using a cursor layer. The cursor layer size is the same as the cursor picture size. The cursor can be moved by setting its display position in the FB corresponding to the cursor layer. The cursor can be dynamically moved among the display devices that support multiple layers. To use this function, you need to disable the cursor layer, cancels the existing binding relationship, bind the cursor layer to a new device, and then restart the cursor layer.

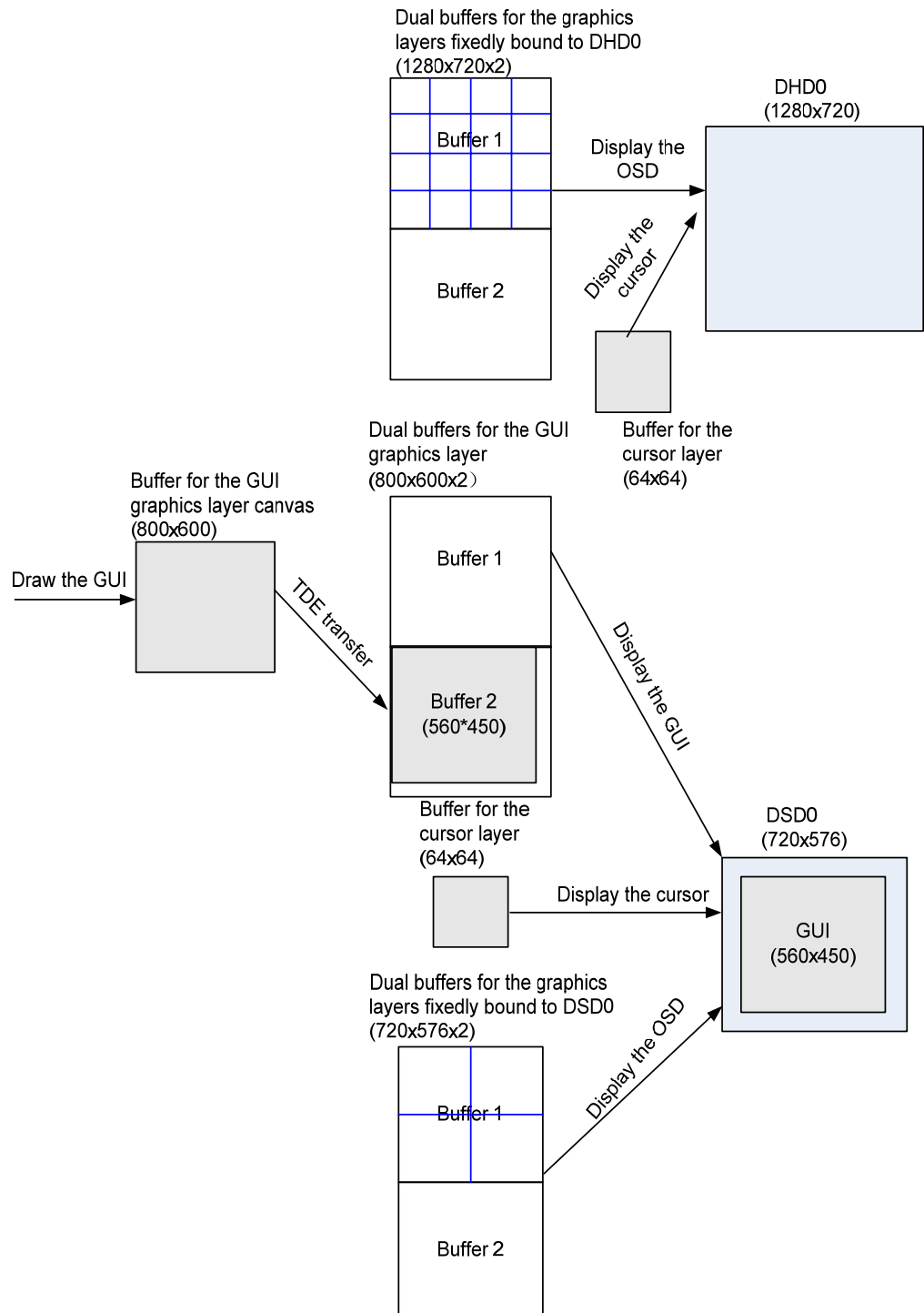
Figure 2-3 Displaying the OSD, GUI, and cursor on DHD0 and displaying the OSD and cursor on DSD0



As the display devices do not need to be disabled when the GUI graphics layer and cursor layer are switched, the screen does not go blank. This scheme is recommended, because the GUI can be switched dynamically.

Figure 2-4 shows the schematic diagram of switching the GUI and cursor to DSD0.

Figure 2-4 Switching the GUI and cursor to DSD0



2.3.2 Development Process

This section uses the GUIs and OSDs of the DHD and DSD as examples. The dividing lines can divide the entire screen into 16 even pictures for DHD0 or four even pictures for DSD0. In addition, the GUI is switched between DHD0 and DSD0.



The following example describes how to switch the GUI from DHD0 to DSD0.

Perform the following steps:

- Step 1** Refresh the canvas of the GUI graphics layer partially.
- Step 2** Disable the GUI graphics layer.
- Step 3** Bind the GUI graphics layer to DSD0.
- Step 4** Set the attributes of the GUI graphics layer and start it by calling HiFB interfaces, and scale the entire canvas based on the size of buffer 2 for the GUI graphics layer.

Note: The size of GUI displayed on DSD0 is smaller than the canvas size. You need to scale the canvas based on the GUI size by changing the resolution of the display buffer for the GUI graphics layer to 560x450.

----End

2.3.3 Application Scenarios

The scheme applies to the following scenarios:

- All devices have independent backend OSDs. For example, the backend OSD is 16-picture layout for DHD0, 8-picture layout for DHD1, 4-picture layout for DSD0, and a single picture with the channel ID information for DSD1.
- Only one device displays the GUI, and the GUI needs to be statically or dynamically switched between the display devices that support multiple layers.
- If the HiFB supports a maximum of n cursor layer, the cursor is displayed on n devices and the cursor can be dynamically switched between the display devices that support multiple layers.

2.3.4 Advantages and Limitations

The advantages of the scheme with multiple graphics layer are as follows:

- The GUI and backend OSD are displayed through different graphics layers. When the GUI or backend OSD changes, the graphics layer of the unchanged one does not need to be updated, which brings convenience to users.
- The cursor can be dynamically switched between the display devices that support multiple layers. That is, the output device does not need to be disabled during switching.
- The scheme allows the GUI to switch between the display devices that support multiple layers, which brings better user experience.
- With only a set of GUI pictures, the GUI requirements of the devices with different resolutions are met.
- Only a GUI canvas that can be updated partially is required.

The limitations on the scheme with multiple graphics layers are as follows:

- The GUI and OSD are displayed on two graphics layers. Compared with the scheme with a graphics layer, more system bandwidth is occupied.