



Graphics Development

# User Guide

Issue	04
Date	2016-05-15

**Copyright © HiSilicon Technologies Co., Ltd. 2014-2016. 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](mailto:support@hisilicon.com)



# About This Document

## Purpose

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



### NOTE

Unless otherwise specified, this document applies to the Hi3516A, Hi3516D, Hi3518E V200/V201, and Hi3516C V200.

Unless otherwise stated, the contents of Hi3516C V200 are consistent with those of Hi3518E V200/Hi3518E V201.

## Related Versions

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

Product Name	Version
Hi3516A	V100
Hi3516D	V100
Hi3518E	V200
Hi3518E	V201
Hi3516C	V200
Hi3519	V100
Hi3519	V101

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

## Change History

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

### Issue 04 (2016-05-15)

This issue is the fourth official release, which incorporates the following change:

The content related to Hi3519 V101 is added.

### Issue 03 (2015-08-20)

This issue is the third official release, which incorporates the following change:

The content related to Hi3519 V100 is added.

### Issue 02 (2015-05-29)

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

The content related to Hi3518E V200, Hi3518E V201, and Hi3516C V200 is added.

#### Chapter 1 Introduction to Graphics Layers

Section 1.1 is modified.

### Issue 01 (2014-12-20)

This issue is the first official release, which incorporates the following change:

The content related to the Hi3516D is added.



## **Issue 00B01 (2014-09-14)**

This issue is the first draft release.



# Contents

<b>About This Document.....</b>	<b>i</b>
<b>1 Introduction to Graphics Layers .....</b>	<b>1</b>
1.1 Overview .....	1
1.2 Architecture of Hi3516A/Hi3518E V200 /Hi3519 V100/Hi3519 V101 Graphics Layers .....	1
<b>2 Recommended Schemes for Graphics Development .....</b>	<b>3</b>
2.1 Overview .....	3
2.2 GUI Scheme with a Single Graphics Layer .....	3
2.2.1 Introduction.....	3
2.2.2 Derivative Scheme .....	5
2.2.3 Development Process.....	6
2.2.4 Application Scenarios .....	6
2.2.5 Advantages and Limitations.....	6



## Figures

---

<b>Figure 2-1</b> Schematic diagram of the scheme with a graphics layer .....	4
<b>Figure 2-2</b> Schematic diagram of the derivative scheme.....	5



## Tables

---

<b>Table 1-1</b> Relationships among the FB device files of the Hi3516A/Hi3518E V200/Hi3519 V100/Hi3519 V101, graphics layers, and output devices.....	1
--	---





# 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 Hi3516A/Hi3518E V200 /Hi3519 V100/Hi3519 V101 Graphics Layers

The Hi3516A/Hi3518E V200/Hi3519 V100/Hi3519 V101 supports one display devices one standard -definition (SD) display devices (SD0), and one graphics layers (G0).



### NOTE

For details about the interfaces and timings supported by each output device, see the description of the video display processor (VDP) in the *Hi35xx xx HD IP Camera SoC Data Sheet*.

[Table 1-1](#) shows the relationships among the FB device files of the Hi3516A/Hi3518E V200/Hi3519 V100/Hi3519 V101, graphics layers, and output devices.

**Table 1-1** Relationships among the FB device files of the Hi3516A/Hi3518E V200/Hi3519 V100/Hi3519 V101, graphics layers, and output devices

FB Device File	Graphics Layer	Corresponding Display Device
/dev/fb0	G0	G0 is displayed only on SD0.



**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 Hi3516A/Hi3518E V200/Hi3519 V100/Hi3519 V101 HiFB.



# 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 Hi3516A, Hi3518E V200, Hi3516C V200, Hi3519 V100 or Hi3519 V101 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.

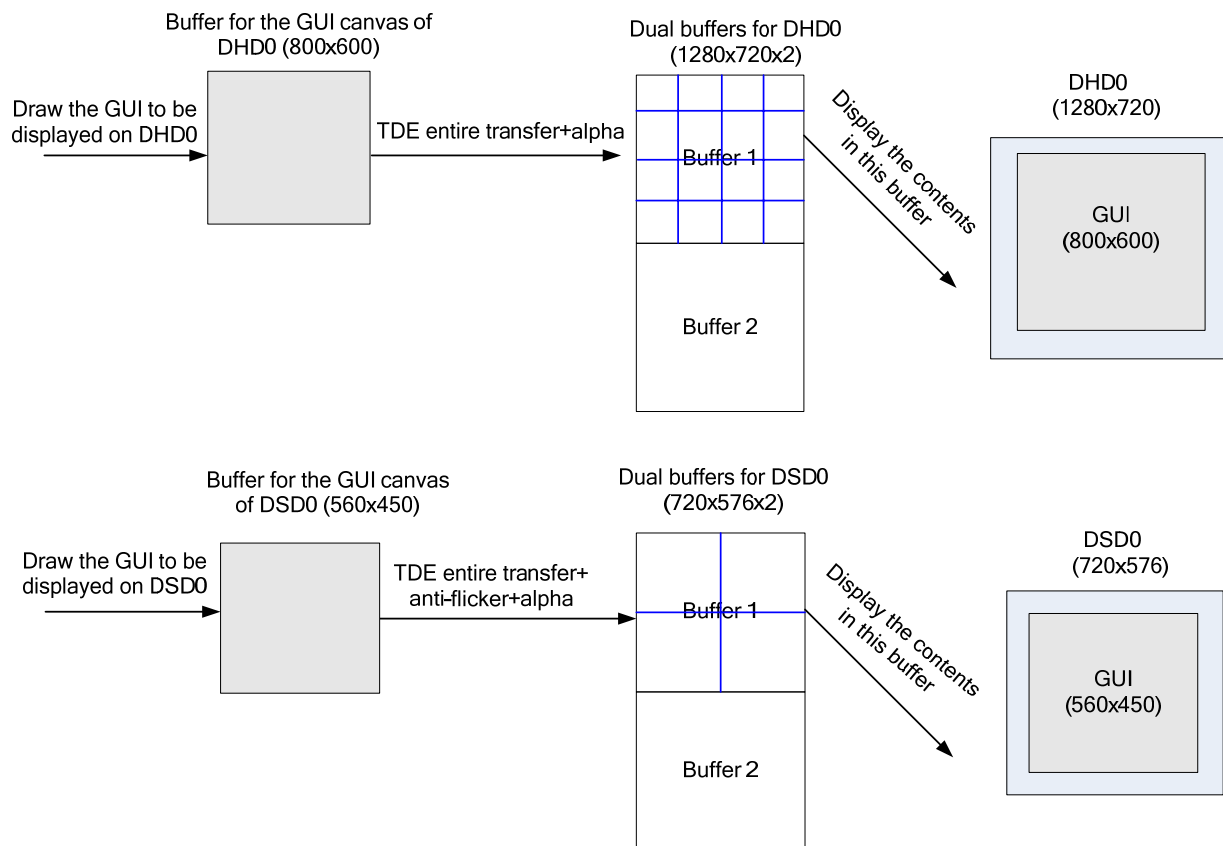
When a drawn canvas is entirely transferred to the corresponding display buffer, transparent blending between the GUI and OSD can be implemented through the TDE. Each time the GUI or OSD changes, the blended area between the GUI and OSD does not need to be calculated based on the local information, because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

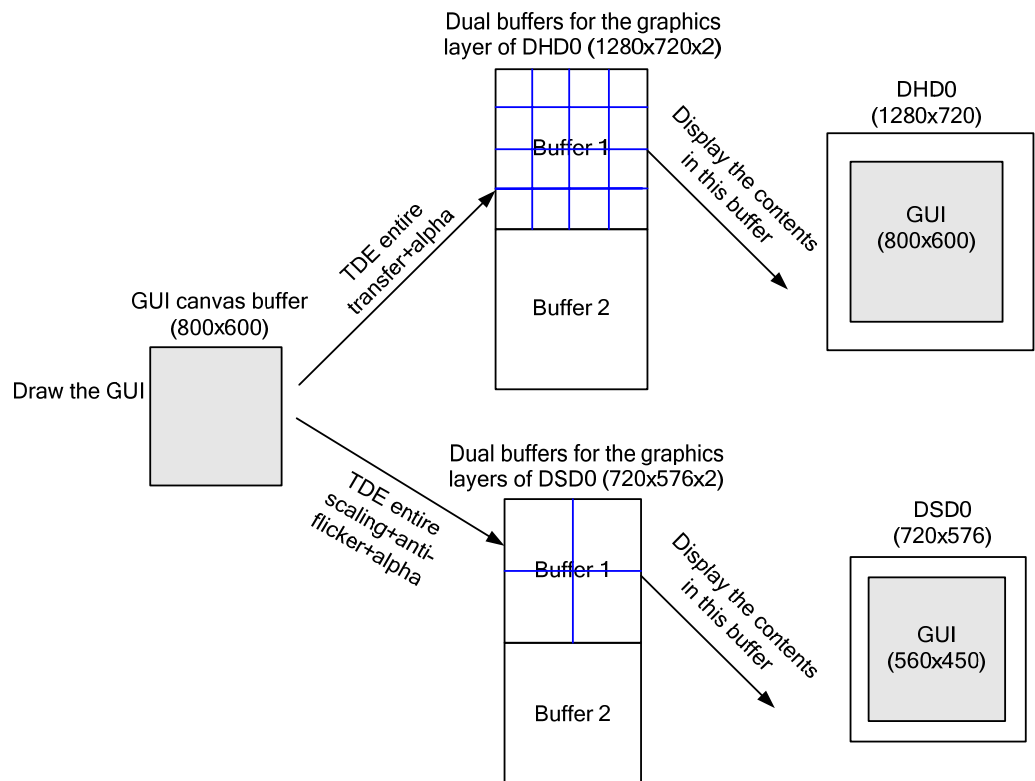
## 2.2.2 Derivative Scheme

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.
- 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.