life.augmented

AN4943
应用笔记

在STM32L496xx/L4A6xx/L4Rxxx/L4Sxxx微控制器上使用Chorm-ART Accelerator ™来刷新LCD-TFT显示器

前言

本应用笔记旨在说明如何使用STM32L496xx/L4A6xx/L4Rxxx/L4Sxxx微控制器上的Chrom-ART Accelerator™，通过FSMC接口刷新LCD-TFT显示屏。

STM32L496xx/L4A6xx/L4Rxxx/L4Sxxx微控制器实现Chrom-Art Accelerator™（DMA2D），即专用于图像处理的专用DMA。

它可以执行下列操作：

- 用特定颜色填充目标图像的一部分或全部
- 通过像素格式转换将源图像的一部分或全部复制到目标图像的一部分或全部中
- 将像素格式不同的两个源图像部分和/或全部混合，再将结果复制到颜色格式不同的部分或整个目标图像中。

在STM32L496xx/L4A6xx/L4Rxxx/L4Sxxx微控制器上，使用灵活的静态存储控制器（FSMC）通过并行接口访问LCD-TFT显示屏。

本应用笔记说明了以下内容：

- 如何通过FSMC接口连接LCD-TFT显示屏
- 如何为LCD-TFT显示屏刷新配置DMA2D
- 如何使用DMA2D字节重新排序功能直接驱动Intel 8080显示器。

要充分利用本应用笔记，用户应熟悉STM32 Chrom-ART Accelerator™（DMA2D），如*STM32L4x6基于Arm®的高级32位MCU*参考手册（RM0351）和*STM32L4Rxxx/L4Sxxx基于Arm®*的高级32位MCU参考手册（RM0432）中所述内容（可从意法半导体的网站www.st.com获取）。

表1. 适用产品

类型	产品线和部件号
微控制器	STM32L496AE、STM32L496AG、STM32L496QE、STM32L496QG、STM32L496RE、STM32L496RG、STM32L496VE、STM32L496VG、STM32L496ZE、STM32L496ZG
	STM32L4A6AG、STM32L4A6QG、STM32L4A6RG、STM32L4A6VG、STM32L4A6ZG
	STM32L4R5/S5产品线，STM32L4R7/S7产品线，STM32L4R9/S9产品线

目录

表格索引

图片目录

1 参考文档

以下文档可从*www.st.com*获得。

- *STM32L4x6基于Arm®的高级32位MCU*（RM0351）参考手册
- *STM32L4Rxxx/L4Sxxx基于Arm®*的高级32位MCU（RM0432）参考手册
- 探索套件和*STM32L496AG MCU*用户手册（UM2160）
- STM32L4系列（STM32CubeL4）的嵌入式软件

本应用笔记适用于基于Arm®的器件。

2 Chrom-ART Accelerator™（DMA2D）应用用例概述

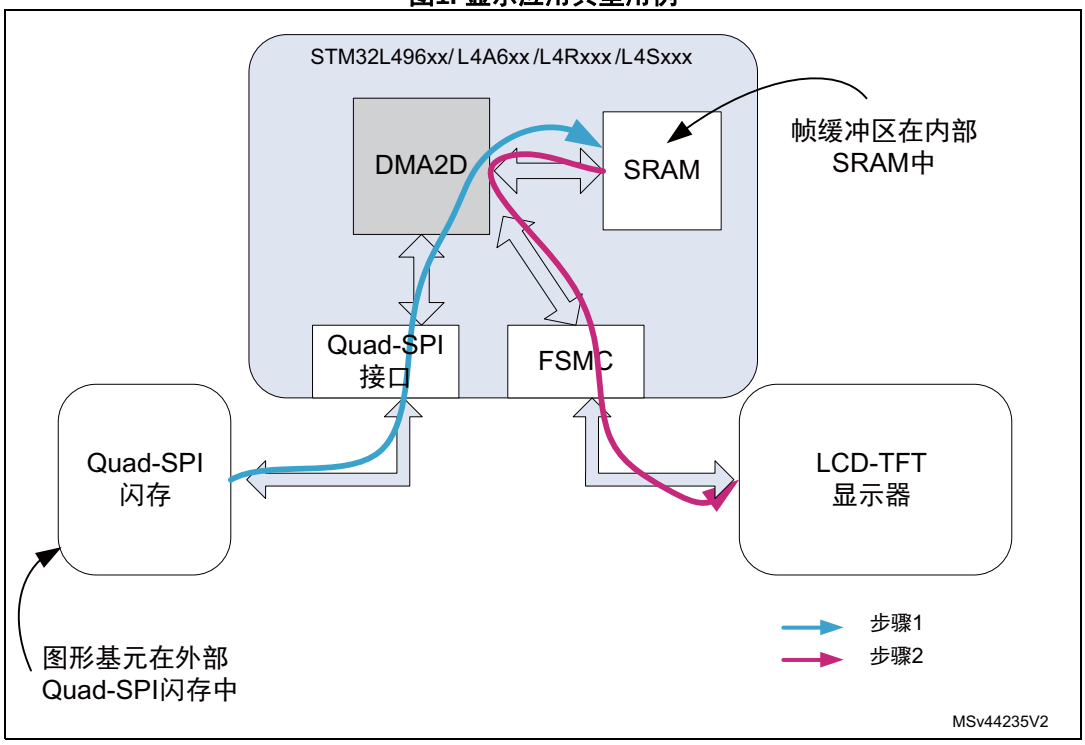
在LCD-TFT显示屏中显示图像的典型应用分2步实现。

- 第1步：创建帧缓冲器内容
  - 帧缓冲由图标、图片和字体等图形基元构成
  - 由运行图形库软件的CPU来完成此操作
  - 可通过图形库由与CPU一起使用的专用硬件进行加速（Chrom-ART Accelerator™（DMA2D））
  - 帧缓冲器的更新频率越高，动画越流畅
- 第2步：在LCD-TFT显示屏上显示帧缓冲
  - 帧缓冲通过专用硬件接口传输到显示屏
  - 可使用CPU、系统DMA或使用Chrom-ART Accelerator™（DMA2D）完成传输

在使用STM32L496xx/L4A6xx/L4Rxxx/L4Sxxx微控制器的典型显示应用示例中，灵活静态存储控制器（FSMC）用作LCD-TFT显示器的硬件接口，图形基元（例如图片、图标或字体）存储在外部Quad-SPI闪存中，帧缓冲存储在内部SRAM中。还可通过Chrom-ART Accelerator™（DMA2D）来管理帧缓冲向LCD-TFT显示屏的传输，因此不使用CPU或DMA资源。

如*图 1：显示应用典型用例*所示。

图1. 显示应用典型用例



Chrom-ART Accelerator™（DMA2D）既可以更新显示屏上的整个图像（完全刷新），也可以只更新它的一部分（部分刷新）。

通过使用高级HAL库功能进行特定寄存器的编程来完成Chrom-ART Accelerator™（DMA2D）的配置（完全或部分刷新），如*第 4 节：STM32CubeL4 中的Chrom-ART Accelerator™ （DMA2D）配置*所示。

3 通过FSMC访问LCD-TFT显示屏

3.1 硬件接口描述

使用以下信号将灵活的静态存储接口（FSMC）连接到LCD-TFT显示屏：

表2. FSMC 信号

信号名称	FSMC I/O	功能
A[25:0]	O	地址总线
D[15:0]	I/O	双向数据总线
NE[x]	O	芯片选择，x = 1..4
NOE	O	输出使能
NWE	O	写入使能

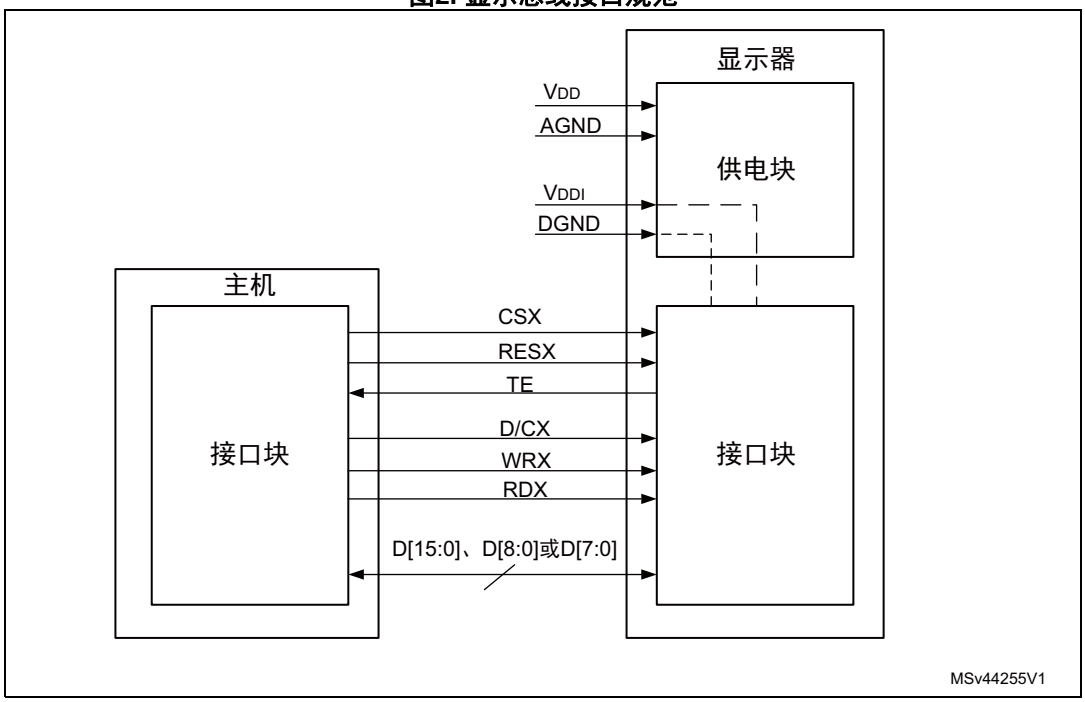
表3. LCD-TFT 信号

信号名称(1)	LCD-TFT I/O	功能
D/CX	I	数据/指令控制信号
D[15:0]	I/O	双向信息信号总线
CSX	I	芯片选择控制信号
RDX	I	读取控制信号
WRX	I	写入控制信号
TE	O	撕裂效应
RESX	I	复位

1. 根据MIPI联盟显示总线接口（DBI）标准所述的B类显示总线接口提供信号名称。

典型连接如*图 2*所示。

图2. 显示总线接口规范



3.2 显示指令集（DCS）软件界面

可根据MIPI联盟DCS规范定义的显示指令集（DCS）使用软件指令，通过物理接口（此处为FSMC总线）控制LCD-TFT显示屏。

DCS指令用于配置显示模块和将帧缓冲传输至显示屏。

3.3 通过STM32L496xx/L4A6xx/L4Rxxx/L4Sxxx微控制器控制D/CX信号

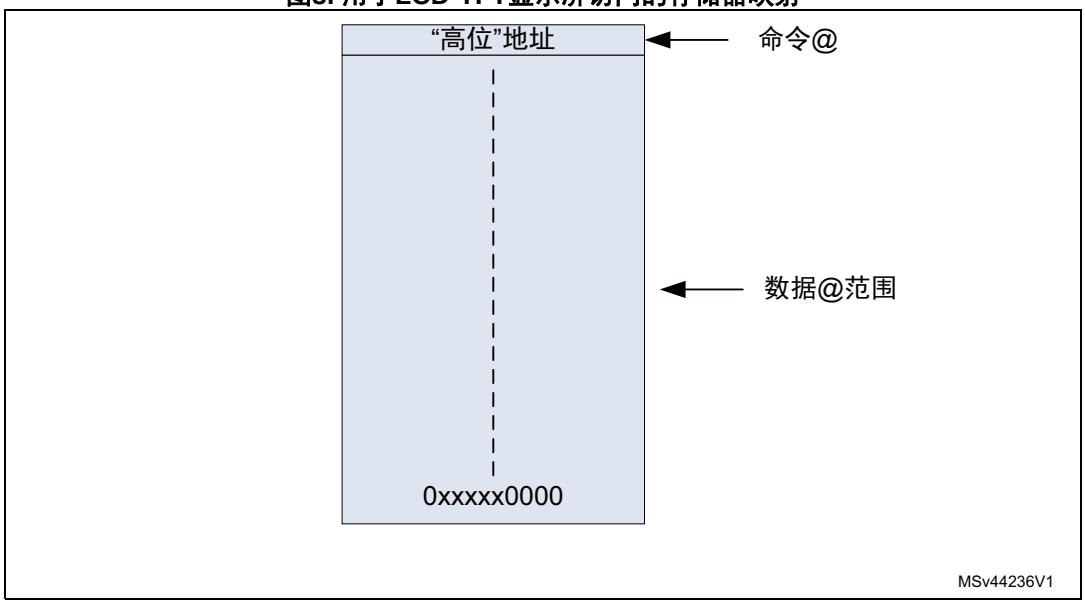
使用DBI协议的D/CX信号将指令（D/CX = 0）与数据（D/CX = 1）传输区分开来。

有2种控制“数据/指令控制”（D/CX）信号的方法：

1. **使用专用GPIO：**
   - 将“数据/指令控制”信号设置为“指令模式”（通过软件将连接到D/CX信号的GPIO置为“0”）
   - 发送指令
   - 将“数据/指令控制”信号设置为“数据模式”（通过软件将连接到D/CX信号的GPIO置为“1”）
   - 发送数据（帧缓冲）
2. **使用FSMC地址总线的地址位：**
   - 保留存储器映射中的“低位”地址用于指令传输
   - 保留较高的存储器映射范围用于数据传输

在使用DMA2D通过FSMC访问LCD-TFT显示屏时，需谨记即使LCD-TFT显示目标是一个固定地址，Chrom-ART Accelerator™（DMA2D）也会在每次访问时使所传输数据的地址总线递增（类似于存储器至存储器的访问）。因此，FSMC地址总线经过递增，可覆盖存储器映射中的完整数据范围地址。

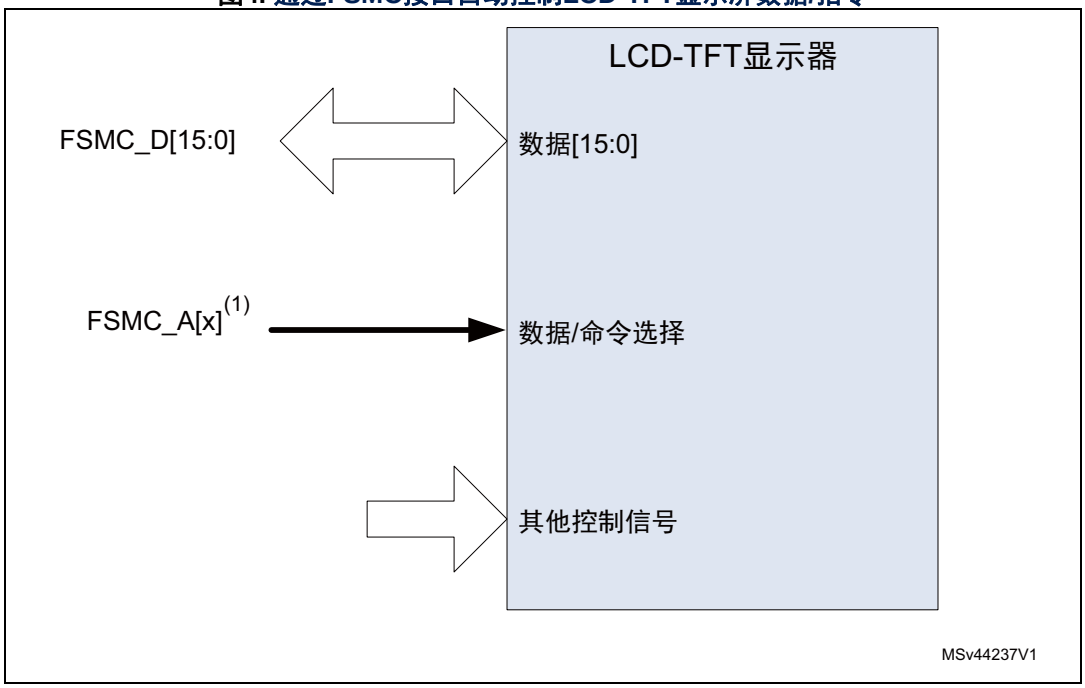
图3. 用于LCD-TFT显示屏访问的存储器映射



使用第2个选项“FSMC地址总线的地址位”会使软件比使用包含专用GPIO的第1个选项更简单，但是它需要使用“高位”地址来控制“数据或指令选择信号”。

- 用户无法使用FSMC地址LSB位（FSMC_A0）等控制“数据或指令选择信号”。
- 用户必须使用“足够高的”FSMC地址位，以确保整个图像帧缓冲传输过程中该位的值相同。

图4. 通过FSMC接口自动控制LCD-TFT显示屏数据/指令



1. 根据*表 4*，“x”应尽可能大。

例如，如果图像缓冲器大小为240x240像素并使用16位RGB565模式（每次访问向LCD传输1个像素）执行传输，则访问次数为240x240＝57600次访问，FSMC地址从0x0000 0000递增至0x0000 E0FF。

因此，在传输过程中不发生变化的第1个地址位为第16位。

就本例而言，可以使用FSMC_A16或更高地址位。

*表 4*显示了根据一些图像大小可以使用的最小FSMC地址位。

表4. 最小可使用FSMC地址位取决于图像大小（16位RGB565访问）

图像大小	像素数量	访问次数	最大地址	最小可使用FSMC地址位
VGA	640x480	307200	0x4AFFF	FSMC_A19
HVGA	480x320	153600	0x257FF	FSMC_A18
QVGA	320x240	76800	0x12BFF	FSMC_A17
-	240x240	57600	0x0E0FF	FSMC_A16

4 STM32CubeL4中的Chrom-ART Accelerator™ （DMA2D）配置

4.1 LCD部分刷新

STM32Cube示例中提供了配置DMA2D用于LCD部分刷新的示例：

STM32Cube_FW_L4\Firmware\Projects\STM32L496G-Discovery\Examples\DMA2D**DMA2D_MemToMemWithLCD**。

下面显示了用于配置和启动DMA2D的代码：

```
/* 在显示图像前配置LCD：设置第一个像素位置和图像大小 */
/* 在这里定义部分刷新的窗口的位置。屏幕中间的长方形 */
LCD_ImagePreparation((ST7789H2_LCD_PIXEL_WIDTH - LAYER_SIZE_X)/2,
(ST7789H2_LCD_PIXEL_HEIGHT - LAYER_SIZE_Y)/2, LAYER_SIZE_X, LAYER_SIZE_Y);
/*##-2- DMA2D配置 ################################################*/
DMA2D_Config();
/*##-3- 开始DMA2D传输 ############################################*/
hal_status = HAL_DMA2D_Start_IT(&Dma2dHandle,
(uint32_t)&RGB565_240x160,   /* RGB565格式、240x160大小的源缓冲 */
(uint32_t)&(LCD_ADDR->REG),  /* LCD数据地址 */
1, LAYER_SIZE_Y * LAYER_SIZE_X); /* 要传输的像素数量 */
OnError_Handler(hal_status != HAL_OK);
...
...
...
/**
* @brief DMA2D配置。
* @note  此功能配置DMA2D外设：
*       1) 配置传输模式：存储器至存储器
*       2) 将输出色彩模式配置为RGB565
*       3) 配置从FLASH到SRAM的传输
*       4) 配置数据大小：240x160（像素）
* @retval
*  无
*/
static void DMA2D_Config(void)
{
  HAL_StatusTypeDef hal_status = HAL_OK;
/* 配置DMA2D模式、色彩模式和输出偏移 */
Dma2dHandle.Init.Mode         = DMA2D_M2M; /* DMA2D模式存储器至存储器*/
```

```
Dma2dHandle.Init.ColorMode    = DMA2D_OUTPUT_RGB565; /* 输出色彩模式为RGB565：16 bpp */
Dma2dHandle.Init.OutputOffset = 0x0; /* 输出无偏移 */
Dma2dHandle.Init.RedBlueSwap   = DMA2D_RB_REGULAR;     /* 输出图像无R&B交换 */
Dma2dHandle.Init.AlphaInverted = DMA2D_REGULAR_ALPHA;   /* 输出图像无α反转 */

/* DMA2D回调配置 */
Dma2dHandle.XferCpltCallback  = TransferComplete;
Dma2dHandle.XferErrorCallback = TransferError;

/* 前景配置：第1层 */
Dma2dHandle.LayerCfg[1].AlphaMode = DMA2D_NO_MODIF_ALPHA;
Dma2dHandle.LayerCfg[1].InputAlpha = 0xFF; /* 完全不透明 */
Dma2dHandle.LayerCfg[1].InputColorMode = DMA2D_INPUT_RGB565; /* 前景层格式为RGB565：16bpp */
Dma2dHandle.LayerCfg[1].InputOffset = 0x0; /* 输入无偏移 */
Dma2dHandle.LayerCfg[1].RedBlueSwap  = DMA2D_RB_REGULAR;    /* 输入前景图像无R&B交换 */
Dma2dHandle.LayerCfg[1].AlphaInverted = DMA2D_REGULAR_ALPHA;   /* 输入前景图像无α反转 */
Dma2dHandle.Instance = DMA2D;

/* DMA2D初始化 */
hal_status = HAL_DMA2D_Init(&Dma2dHandle);
OnError_Handler(hal_status != HAL_OK);
hal_status = HAL_DMA2D_ConfigLayer(&Dma2dHandle, 1);
OnError_Handler(hal_status != HAL_OK);
}
```

完全刷新的方式相同，只是在（0, 0）处初始化LCD第1个像素，并将图像尺寸初始化为LCD尺寸。

```
LCD_ImagePreparation(0, 0, ST7789H2_LCD_PIXEL_WIDTH, ST7789H2_LCD_PIXEL_HEIGHT);
```

在DMA2D开始指令中更改要传输的像素数量：

```
hal_status =  HAL_DMA2D_Start_IT(&Dma2dHandle,
(uint32_t)&RGB565_240x240,   /* RGB565格式、240x240大小的源缓冲 */
(uint32_t)&(LCD_ADDR->REG),   /* LCD数据地址 */
1, ST7789H2_LCD_PIXEL_HEIGHT * ST7789H2_LCD_PIXEL_WIDTH); /* 要传输的像素数量  */
OnError_Handler(hal_status != HAL_OK);
```

5 新的DMA2D功能可支持Intel 8080显示器

在STM32微控制器上，像素数据以小端格式存储在帧缓冲存储器中。这意味着最低有效字节存储在最低地址，最高有效字节存储在最高地址。

例如，在RGB888像素格式的情况下，蓝色分量存储在地址0，而红色分量存储在地址2。

当像素数据通过FSMC传输到LCD显示器时，它首先从最低有效字节开始传输，这是本例中的蓝色部分。

这会造成与某些Intel 8080 LCD显示器颜色编码不匹配，那些显示器需要首先传输最高有效字节（在RGB888像素格式的情况下为红色分量）。

需要在通过FSMC传输像素数据之前进行额外的字节重新排序步骤，才能使这种不匹配获得正确的字节顺序。

这种新的DMA2D字节重新排序功能可对DMA2D输出FIFO中的数据进行重新排序，使其能够以传统RGB顺序从帧缓冲区直接驱动LCD显示器，无需任何额外的软件操作。

5.1 Intel 8080接口颜色编码

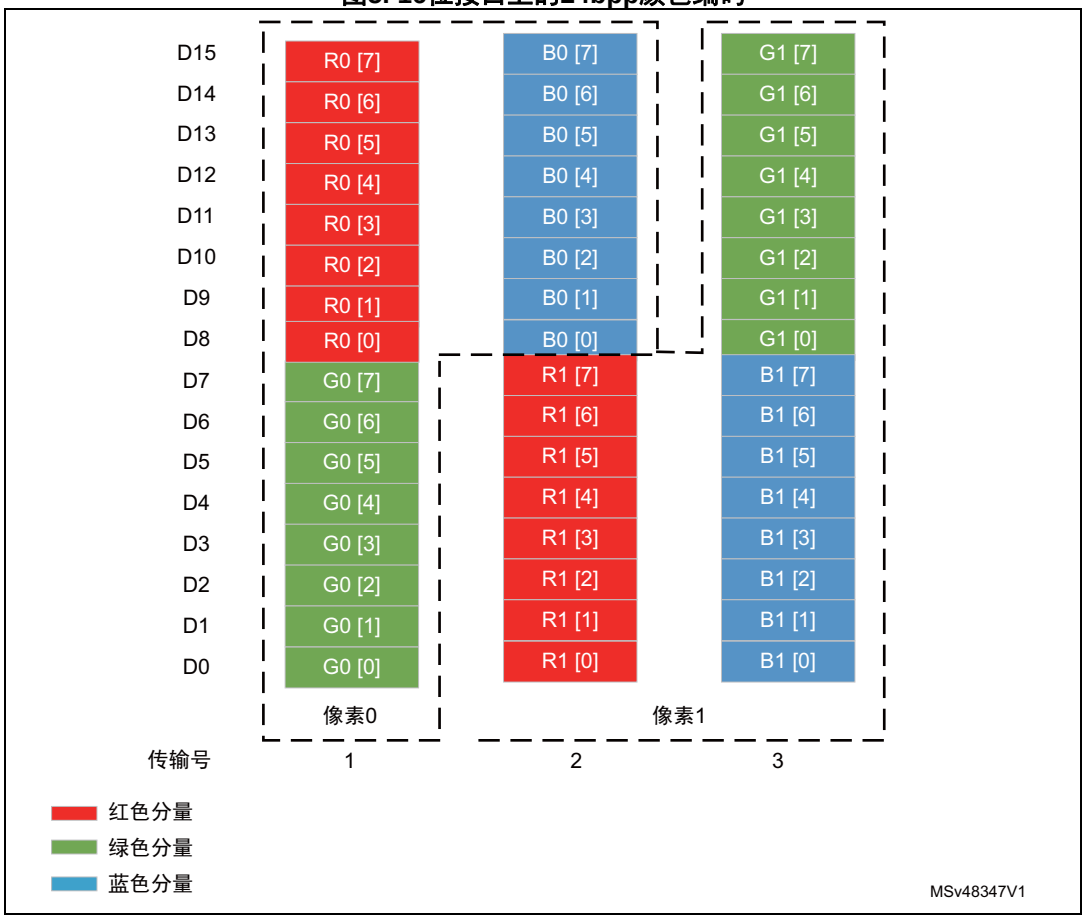
Intel 8080是LCD显示器的通用接口标准。它是可支持8、9、16和18位总线的并行总线接口。

本节介绍Intel 8080显示器颜色编码与STM32内存中传统RGB顺序不匹配的情况。

- 16位接口上24bpp（16.7M色）和18bpp（262k色）

*图 5*显示了通过Intel 8080显示器上的16位总线接口传输24bpp数据的颜色编码。

图5. 16位接口上的24bpp颜色编码

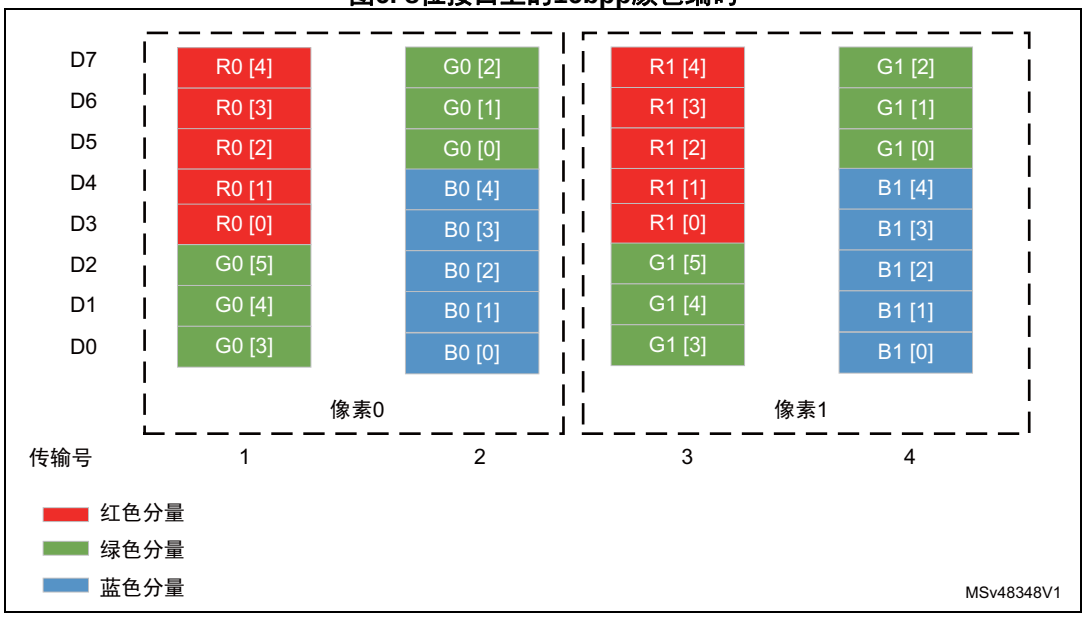


注： *18bpp显示器具有与之相同的颜色编码，不同的是在18bpp的情况下，R/G/B [6:0]位于总线的最高有效位中，且数据线D9、D8、D1和D0被忽略。*

- 8位接口上的16bpp（64k色）

*图 6*给出了8位总线接口上的16bpp显示器的像素颜色编码。

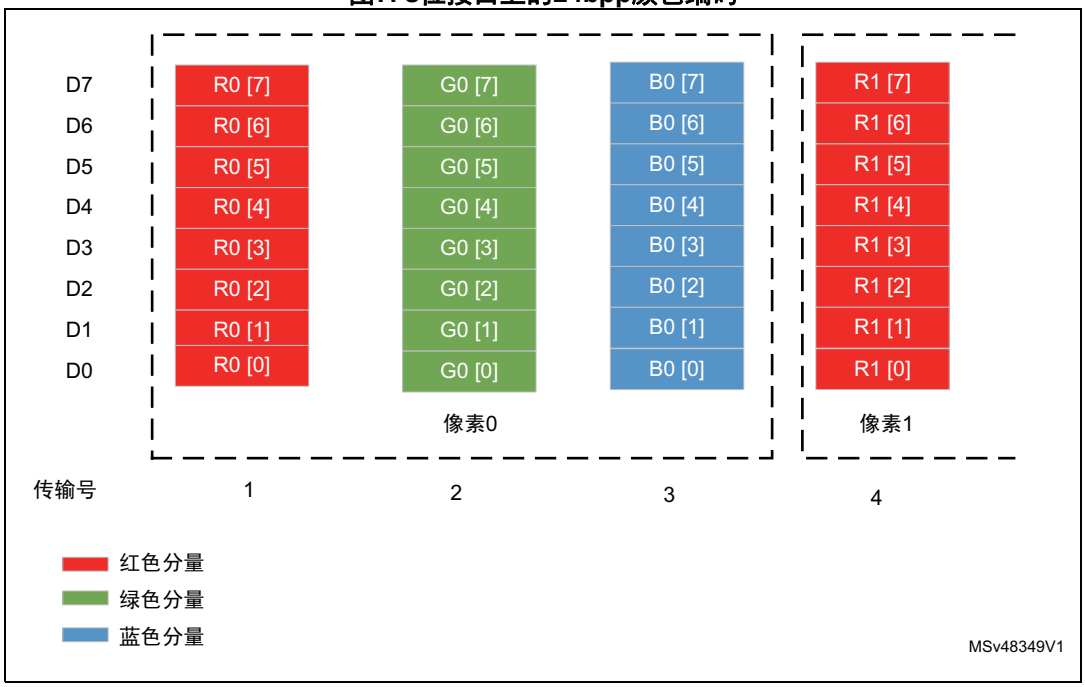
图6. 8位接口上的16bpp颜色编码



- 8位接口上的24bpp（1670万色）和18bpp（262k色）。

*图 7*给出了8位总线接口上的24bpp的像素颜色编码。

图7. 8位接口上的24bpp颜色编码



注： *18bpp显示器具有与之相同的颜色编码，不同的是在18bpp的情况下，R/G/B [6:0]位于总线的最高有效位中（D [7:2]），且数据线D0和D1被忽略。*

5.2 DMA2D重新排序功能

可以对DMA2D输出FIFO字节进行重新排序，以便直接利用来自DMA2D的并行接口（FSMC）来实现显示器帧缓冲区更新。用户可以对重新排序操作进行组合，以获得与显示颜色编码匹配的正确字节顺序。

5.2.1 红蓝交换

红色和蓝色分量可以交换，这可以通过设置DMA2D_OPFCCR寄存器中的RBS位来完成。

STM32L4系列和STM32L4 Plus系列产品具有此功能。

5.2.2 字节交换

半字的MSB和LSB字节可在输出FIFO中交换。这可以通过设置DMA2D_OPFCCR寄存器中的SB位来完成。

该功能仅在STM32L4 Plus系列产品上提供。

*表 5*显示了匹配LCD显示器颜色编码所需的交换操作，它取决于显示器色深和总线接口宽度。

表5. 交换操作

色深	接口总线宽度	所需操作	
		红蓝交换	字节交换
8bpp（256色）	8 位	N	N
	16 位	N	是
16bpp（64k色）	8 位	N	是
	16 位	N	N
18bpp（262k色）	8 位	是	N
	16 位	是	是
24bpp（16.7M色）	8 位	是	N
	16 位	是	是

5.3 DMA2D重新排序用例示例

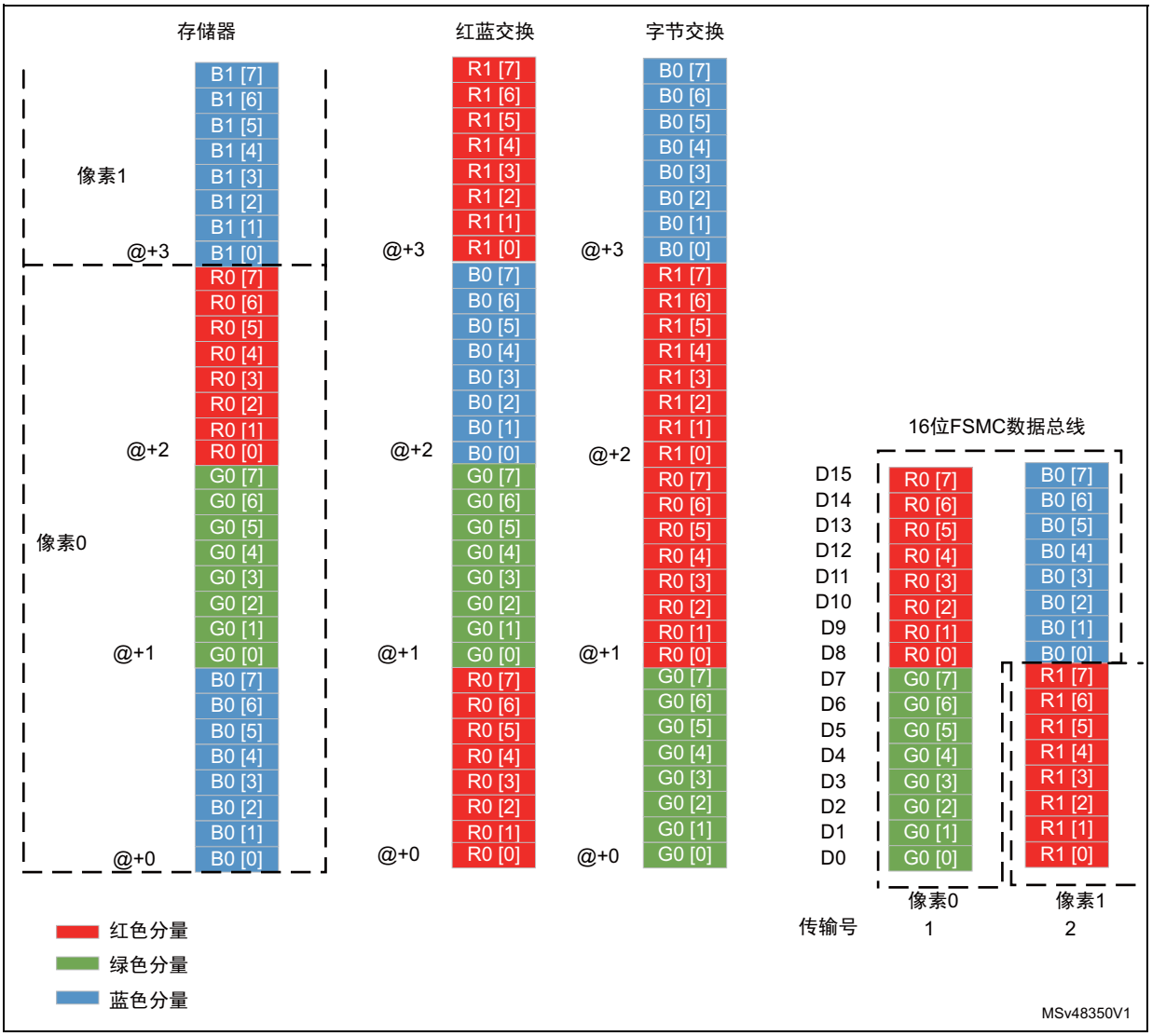
5.3.1 16位FSMC数据总线接口上的24bpp/18bpp

要支持使用8080标准的24bpp显示器，帧缓冲器数据需要进行两个操作：

- 红蓝交换
- 半字的MSB和LSB字节交换

*图 8*显示了由DMA2D执行的操作，该操作针对16位接口上24bpp色深，可以实现对应Intel 8080协议的良好字节顺序。

图8. 支持16位接口上24bpp的DMA2D操作



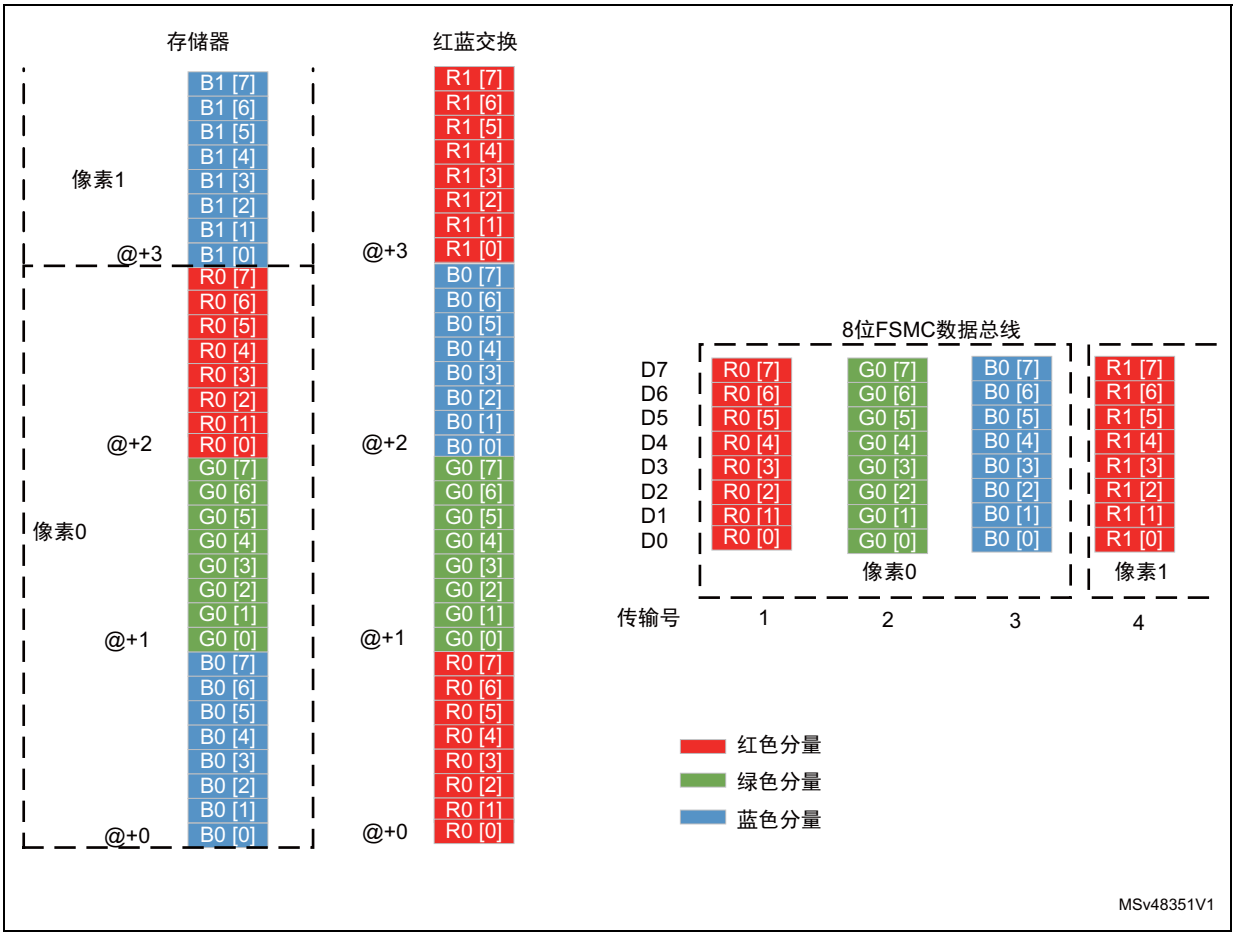
注： *在不支持字节交换的MCU上，可以通过交换板上LCD接口的数据线来实现硬件调整。显示器D [15:8]线连接到FSMC D [7:0]线，显示器D [7:0]线连接到FSMC D [15:8]线。*

5.3.2 8位FSMC数据总线接口上的24bpp/18bpp

对于使用8位数字总线的24bpp，需要进行红蓝交换来获得正确的字节顺序。

*图 9*显示了由DMA2D所做的红蓝交换操作，可以实现良好的字节顺序。

图9. 支持8位接口上24bpp的DMA2D操作

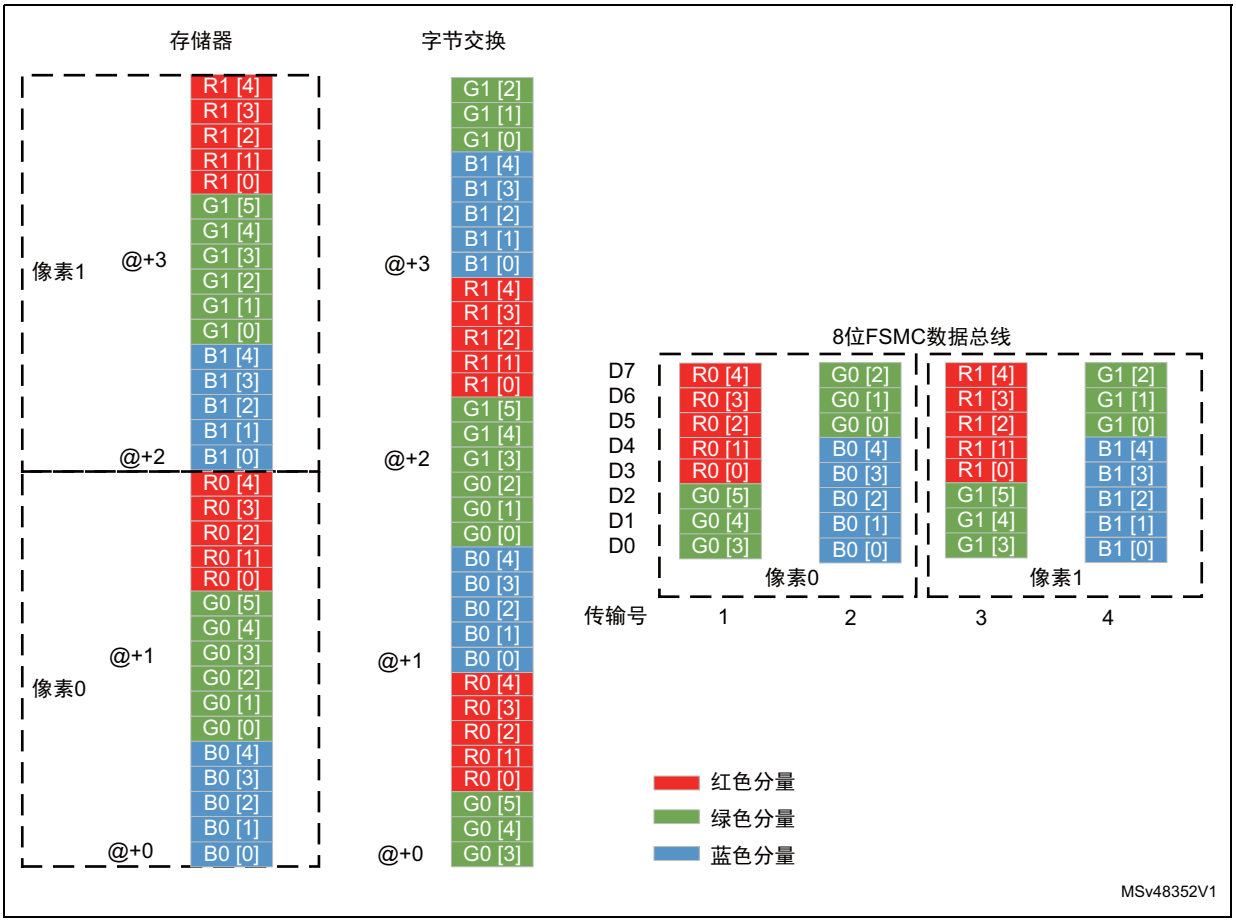


5.3.3 通过8位FSMC数据总线接口的16bpp

为了通过8位接口驱动16bpp Intel 8080显示器，必须交换半字的MSB和LSB字节。

*图 10*显示了交换操作如何实现良好的字节顺序。

图10. 支持8位接口上16bpp的DMA2D操作



6 结论

本应用笔记提供了使用Chrom-ART Accelerator™（DMA2D）通过FSMC接口将图像轻松传输至LCD-TFT显示屏的指南，期间无需使用CPU或DMA资源。重点说明了LCD-TFT显示屏的“数据/指令控制”信号的正确控制。此外，还提供了一些设置Chrom-ART Accelerator™（DMA2D）的代码示例。

本应用笔记还介绍了DMA2D的新型字节重新排序功能，该功能可直接通过FSMC来支持16.7M色和262k色Intel 8080显示器的更新。

7 版本历史

表6. 文档版本历史

日期	版本	变更
2017年1月27日	1	初始版本。
2017年10月23日	2	整个文档中增加了STM32L4Rxxx/L4Sxxx器件。 增加了*第 5节：新的DMA2D功能可支持Intel 8080显示器*。

表7. 中文文档版本历史

日期	版本	变更
2018年8月10日	1	中文初始版本。

重要通知 - 请仔细阅读

意法半导体公司及其子公司 （“ST”）保留随时对 ST 产品和 / 或本文档进行变更、更正、增强、修改和改进的权利，恕不另行通知。买方在订货之前应获取关于 ST 产品的最新信息。 ST 产品的销售依照订单确认时的相关 ST 销售条款。

买方自行负责对 ST 产品的选择和使用， ST 概不承担与应用协助或买方产品设计相关的任何责任。

ST 不对任何知识产权进行任何明示或默示的授权或许可。

转售的 ST 产品如有不同于此处提供的信息的规定，将导致 ST 针对该产品授予的任何保证失效。

ST 和 ST 徽标是 ST 的商标。所有其他产品或服务名称均为其各自所有者的财产。

本文档中的信息取代本文档所有早期版本中提供的信息。本文档的中文版本为英文版本的翻译件，仅供参考之用；若中文版本与英文版本有任何冲突或不一致，则以英文版本为准。