

STM32 MCU上的LCD-TFT显示控制器（LTDC）

前言

移动、工业和消费应用的发展使得对图形用户界面（GUI）的需求更强，并且所需硬件资源也有所增加。这些应用需要更高质量的图形、更多的硬件和软件资源（比如图形基元或帧缓冲区的内存）以及更高的处理性能。

为了应对这种日益增长的需求，通常会使用微处理器单元，而这导致了成本更高、设计更复杂，而且上市时间也更长。为了应对这些要求，STM32 MCU提供了丰富的图形产品组合。

由于采用了嵌入式LCD-TFT显示控制器（LTDC），STM32 MCU可以直接驱动高分辨率显示面板，无需任何CPU干预。另外，LTDC可以自主访问内部存储器或外部存储器来获取像素数据。

本应用笔记介绍了表 1中所列STM32微控制器的LCD-TFT显示控制器，并演示了如何使用和配置LTDC外设。还重点阐述了为获得最佳图形性能所需要的一些硬件、软件和架构考虑因素。

相关文档

意法半导体网站 www.st.com 提供以下文档：

- STM32F75xxx和STM32F74xxx基于32位MCU（RM0385）的高级ARM®
- STM32F76xxx和STM32F77xxx基于32位MCU（RM0410）的高级ARM®
- STM32F469xx和STM32F479xx基于32位MCU（RM0386）的高级ARM®
- STM32F405/415、STM32F407/417、STM32F427/437和STM32F429/439基于32位MCU（RM0090）的高级ARM®
- STM32F429/439、STM32F469/479、STM32F7x6、STM32F7x7、STM32F7x8、STM32F7x9数据表

表1. 适用产品

类型	产品线
微控制器	STM32F429/439、STM32F469/479、STM32F7x6、STM32F7x7、STM32F7x8、STM32F7x9

目录

1	显示器和图形概述	8
1.1	基本图形概念	8
1.2	显示接口标准	11
1.3	STM32 MCU支持的显示接口	13
2	LTDC控制器和STM32 MCU图形产品组合概述	15
2.1	STM32 MCU上的LCD-TFT显示控制器	15
2.2	LTDC可用性和STM32系列的图形产品组合	15
2.3	智能架构中的LTDC	16
2.4	使用STM32 LTDC控制器的优势	19
3	LCD-TFT (LTDC) 显示控制器说明	20
3.1	功能描述	20
3.1.1	LTDC时钟域	20
3.1.2	LTDC复位	20
3.2	灵活的时序和硬件接口	21
3.2.1	LCD-TFT引脚和信号接口	21
3.2.2	对于不同的显示器尺寸，其时序完全可编程	22
3.3	两个可编程LTDC层	25
3.3.1	灵活的窗口位置和尺寸配置	26
3.3.2	可编程层：颜色帧缓冲器	27
3.4	中断	29
3.5	低功耗模式	29
4	使用LTDC创建图形应用	31
4.1	确定图形应用要求	31
4.2	检查显示器尺寸和色深与硬件配置的兼容性	31
4.2.1	帧缓冲存储器大小要求和位置	31
4.2.2	考虑存储器时检查显示兼容性带宽要求	33
4.2.3	检查显示面板接口与LTDC的兼容性	39
4.3	STM32封装选择指南	40
4.4	LTDC与DMA2D和CPU同步	41
4.4.1	DMA2D 的用法	41

4.4.2	LTDC和DMA2D/CPU同步	42
4.5	图形性能优化	42
4.5.1	内存分配	42
4.5.2	优化从外部存储器读取LTDC帧缓冲器的过程（SDRAM或SRAM） ...	43
4.5.3	优化从SDRAM读取LTDC帧缓冲器的过程	47
4.5.4	在消隐周期中更新帧缓冲器内容	48
4.6	关于Cortex [®] -M7（STM32F7系列）的特别建议	48
4.6.1	如果不使用，就禁用FMC bank1	49
4.6.2	配置存储器保护单元（MPU）	49
4.7	LTDC外设配置	53
4.7.1	显示面板连接	53
4.7.2	LTDC时钟和时序配置	54
4.7.3	LTDC层配置	57
4.7.4	显示面板配置	57
4.8	存储图形基元	58
4.8.1	将图像转换为C文件	58
4.9	硬件注意事项	58
5	节省能耗	60
6	LTDC应用示例	61
6.1	实现示例和资源要求	61
6.1.1	单片MCU	61
6.1.2	带外部存储器的MCU	62
6.2	示例：创建基本图形应用	64
6.2.1	硬件说明	64
6.2.2	如何检查特定显示器尺寸是否匹配 硬件配置	66
6.2.3	LTDC GPIO配置	67
6.2.4	LTDC外设配置	71
6.2.5	显示来自内部闪存的图像	76
6.2.6	FMC SDRAM配置	81
6.2.7	MPU和高速缓存配置	81
6.3	带LCD-TFT面板的参考板	85
7	所支持的显示面板	87

8	常见问题	88
9	结论	89
10	版本历史	90

表格索引

表1.	适用产品	1
表2.	STM32 MCU支持的显示接口	13
表3.	嵌入了LTDC的STM32 MCU及其可用的图形产品组合	15
表4.	使用STM32 MCU LTDC控制器的优势	19
表5.	LTDC接口输出信号	21
表6.	LTDC时序寄存器	22
表7.	LTDC中断总结	29
表8.	LTDC外设状态与STM32低功耗模式的关系	30
表9.	不同屏幕分辨率的帧缓冲器大小	32
表10.	STM32F4x9, 其HCLK @ 180 MHz并且SDRAM @ 90 MHz 最大支持像素时钟与LTDC配置和SDRAM总线宽度的关系	37
表11.	STM32F7x6、STM32F7x7、STM32F7x8和STM32F7x9, 其HCLK @ 200 MHz并且 SDRAM@ 100 MHz时, 最大支持像素时钟与LTDC配置和SDRAM总线宽度的关系	38
表12.	特定STM32硬件配置下所支持的显示分辨率示例	39
表13.	带LTDC外设的STM32封装与RGB接口可用性的关系	40
表14.	从ROCKTECH RK043FN48H数据表中提取LCD-TFT时序	55
表15.	编程LTDC时序寄存器	56
表16.	使用不同硬件配置下STM32的图形移植示例	63
表17.	STM32参考板, 嵌入了LTDC并具有板上LCD-TFT面板	86
表18.	常见问题	88
表19.	文档版本历史	90
表20.	中文文档版本历史	90

图片目录

图1.	基本嵌入式图形系统	8
图2.	带嵌入式控制器和GRAM的显示模块	9
图3.	没有控制器和帧缓冲器的显示模块	10
图4.	不带控制器和GRAM、带外部帧缓冲器的显示模块	10
图5.	MIPI-DBI的A类或B类接口	11
图6.	MIPI-DBI C类接口	11
图7.	MIPI-DPI接口	12
图8.	MIPI-DSI接口	12
图9.	STM32F429/439和STM32F469/479系列产品中的LTDC AHB主设备智能架构	17
图10.	STM32F7x6、STM32F7x7、STM32F7x8和STM32F7x9中的LTDC AHB主设备智能架构	18
图11.	LTDC框图	20
图12.	LTDC信号接口	22
图13.	典型LTDC显示帧（有效宽度 = 480像素）	23
图14.	完全可编程的时序和分辨率	24
图15.	LTDC完全可编程显示分辨率总宽度达4096像素，总高度达2048行	25
图16.	两层与背景混合	26
图17.	层窗口可编程尺寸和位置	26
图18.	像素数据映射与颜色格式的关系	27
图19.	帧缓冲器中可编程颜色层	28
图20.	从RGB565输入像素格式到内部ARGB8888格式的像素格式转换	28
图21.	同时访问SDRAM的AHB主设备	34
图22.	使用外部SDRAM时的典型图形硬件配置	36
图23.	双缓冲：将LTDC与DMA2D或CPU同步	42
图24.	采用从属存储器分割的示例在STM32F4x9系列MCU上	43
图25.	跨越千字节边界进行突发访问	44
图26.	减少图层窗口和帧缓冲器行宽	46
图27.	添加虚拟字节使行宽为64字节的倍数	47
图28.	将两个缓冲区置于独立的SDRAM存储区中	48
图29.	默认系统存储器映射（MPU禁用）情况下，FMC SDRAM和NOR / PSRAM存储器交换	51
图30.	连接RGB666显示面板	53
图31.	低端图形实现示例	62
图32.	高端图形实现示例	63
图33.	STM32F746G-DISCO中的图形硬件配置	64
图34.	STM32F746G-DISCO板上的LCD-TFT连接	65
图35.	背光控制器模块	66
图36.	STM32CubeMX：LTDC GPIO配置	68
图37.	STM32CubeMX：PJ7引脚配置为LTDC_G0备用功能	68
图38.	STM32CubeMX：LTDC配置	69
图39.	STM32CubeMX：LTDC GPIO输出速度配置	69
图40.	STM32CubeMX：显示器使能引脚（LCD_DISP）配置	70
图41.	STM32CubeMX：将LCD_DISP引脚输出电平设置为高电平	70
图42.	STM32CubeMX：使能LTDC全局和错误中断	71
图43.	STM32CubeMX：时钟配置选项卡	72
图44.	STM32CubeMX：系统时钟配置	72
图45.	STM32CubeMX：LTDC像素时钟配置	73
图46.	STM32CubeMX：LTDC时间配置	74
图47.	STM32CubeMX：LTDC层1参数设置	76
图48.	LCD图像转换器：主页	77

图49.	LCD图像转换器：图像项目.....	78
图50.	LCD图像转换器：设置转换选项	78
图51.	LCD图像转换器：生成头文件.....	79
图52.	FMC SDRAM MPU 配置示例	82
图53.	Quad-SPI区域的MPU配置	83

1 显示器和图形概述

本节介绍了显示器和图形内容中使用的基本术语，对通用显示器和图形环境进行了概述。本节还总结了STM32 MCU所支持的显示接口。

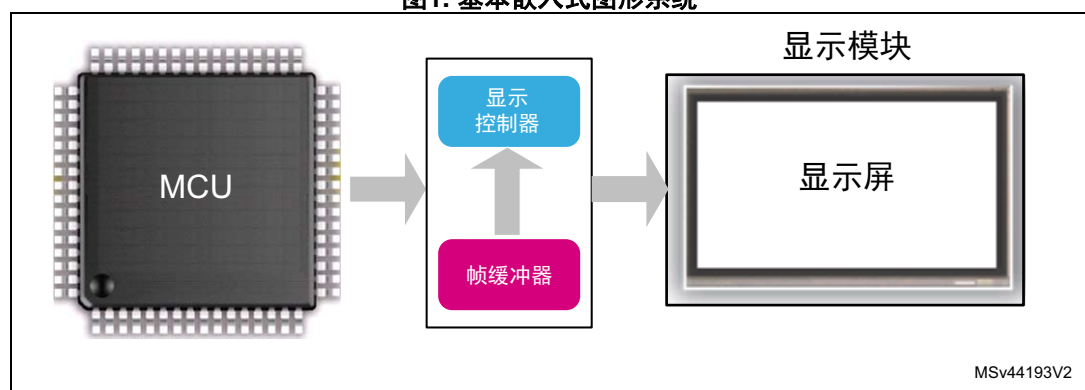
1.1 基本图形概念

本节介绍基本的嵌入式图形系统、显示模块类别和显示技术。

基本嵌入式图形系统

一个基本嵌入式图形系统可简化为如图1所示。

图1. 基本嵌入式图形系统



基本嵌入式图形系统由微控制器、帧缓冲器、显示控制器和显示屏组成。

- 微控制器对要在帧缓冲器中显示的图像进行计算，以组成图标或图像等图形基元。CPU通过运行图形库软件来执行此过程。该过程可以由图形库使用专用硬件（如DMA2D Chrom-Art Accelerator[®]）来加速。帧缓冲器更新的频率越高，动画越流畅（动画每秒帧数）。

- 帧缓冲器是一个易失性存储器，用于存储要显示图像的像素数据。该存储区通常称为图形RAM（GRAM）。所需帧缓冲器大小取决于显示器的分辨率和色深。关于所需帧缓冲器大小的更多信息，请参阅[第 4.2.1 节：帧缓冲存储器大小要求和位置](#)。
 - 双缓冲技术使用两个帧缓冲器，可以避免显示正在写入帧缓冲器的内容。
- 显示控制器持续“刷新”显示器，以每秒60次（60Hz）的速度将帧缓冲器内容传送到显示屏。显示控制器可以嵌入显示模块或MCU中。
- 显示屏由显示控制器来驱动，并负责显示图像（由像素矩阵组成）。
显示器特性为：
 - 显示尺寸（分辨率）：由显示像素数定义，表示为水平（像素数）×垂直（行数）。
 - 色深：定义可以绘制像素的颜色数量。它以每像素位数（bpp）来表示。对于24 bpp的色深（也可以用RGB888表示），一个像素可以有16777216种颜色表示。
 - 刷新率（以Hz为单位）：显示面板每秒刷新的次数。因为刷新率较低时产生的视觉效果不佳，所以显示器每秒钟刷新60次（60 Hz）。

显示模块类别

显示模块分为两大类，取决于它们是否嵌入了内部控制器和GRAM。

- 第一类对应于具有显示屏控制器和GRAM的显示器（参见[图 2](#)）。
- 第二类对应的显示器，其显示屏没有主控制器，仅有低电平时序控制器。
要连接无控制器和GRAM的显示器，所用帧缓冲器可以位于MCU的内部SRAM中（参见[图 3](#)）或位于外部存储器中（参见[图 4](#)）。

图2. 带嵌入式控制器和GRAM的显示模块

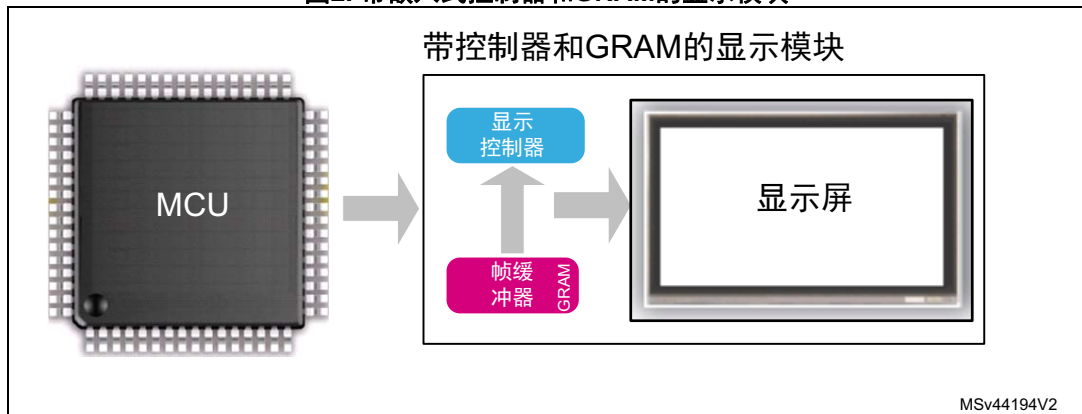


图3. 没有控制器和帧缓冲器的显示模块

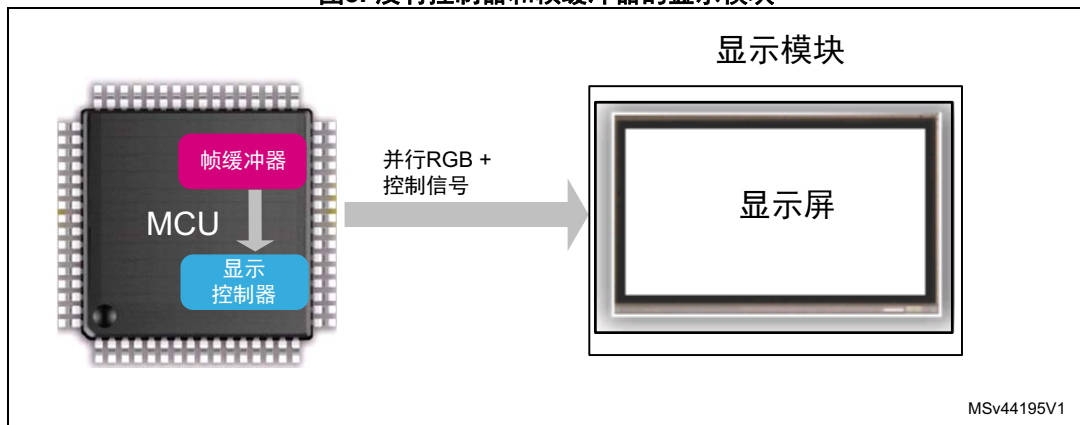
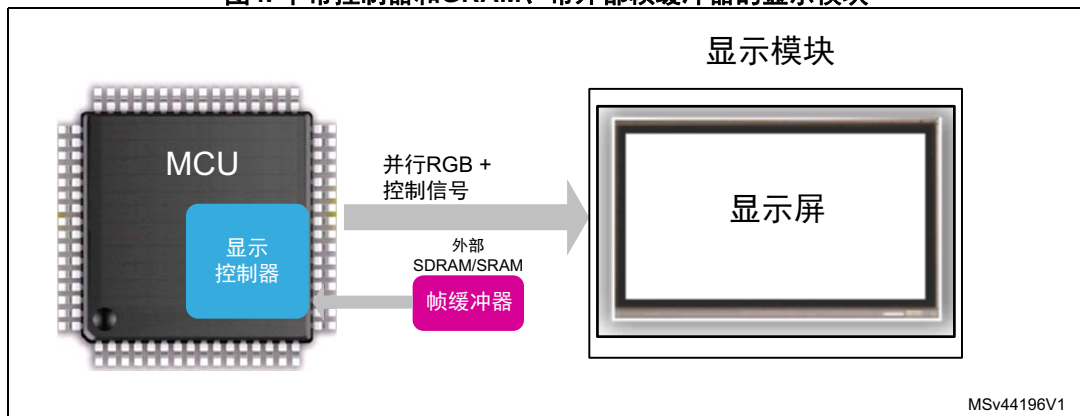


图4. 不带控制器和GRAM、带外部帧缓冲器的显示模块



显示技术

市场上有许多可用的显示技术，采用的两种主要技术如下所述：

- LCD-TFT显示器（液晶显示器-薄膜晶体管）：是一种LCD变体，它采用TFT技术提高了对每个像素的控制。得益于TFT技术，每个像素都可以通过晶体管进行控制，从而实现快速的响应时间和精确的色彩控制。
- OLED显示器（有机LED）：像素由直接发光的有机LED组成，可以实现更好的对比度并优化功耗。LED技术可以使用柔性显示器，不再需要玻璃屏或背光源。响应时间非常快，视角不受任何光线偏振的影响。

TFT和OLED技术中驱动显示模块的方式非常相似，它们的主要区别在于是否需要背光源，因为OLED不需要任何背光源。

1.2 显示接口标准

MIPI（移动行业处理器接口）联盟是一个致力于定义和推广移动设备接口规范的全球协作组织。MIPI联盟不仅开发了新标准，还将现有的显示接口进行了标准化：

MIPI显示总线接口（MIPI-DBI）

MIPI-DBI是MIPI联盟发布的第一个显示标准，用来规定显示接口。MIPI-DBI中定义了三类接口：

- A类：基于Motorola 6800总线
- B类：基于Intel® 8080总线
- C类：基于SPI协议

MIPI-DBI用来与带有集成图形RAM（GRAM）的显示器进行连接。像素数据在显示器的本地GRAM中进行更新。图 5中举例说明了一个MIPI-DBI的A类或B类显示接口示例。

图5. MIPI-DBI的A类或B类接口

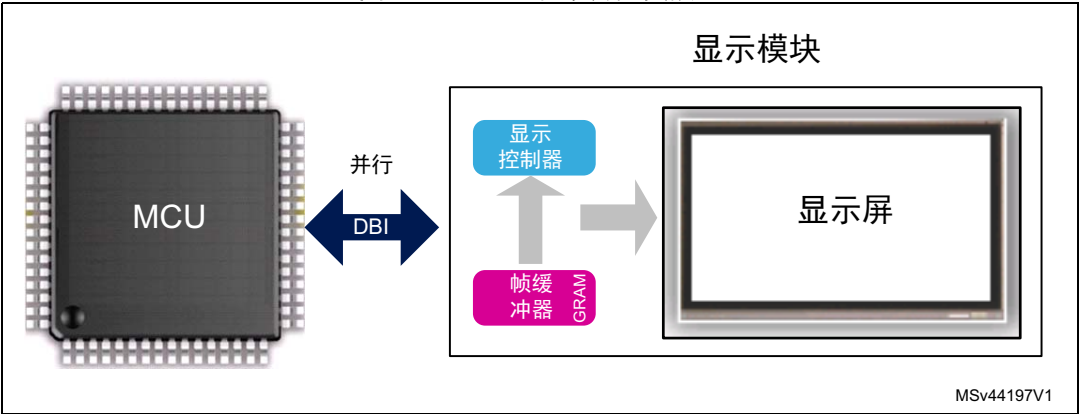
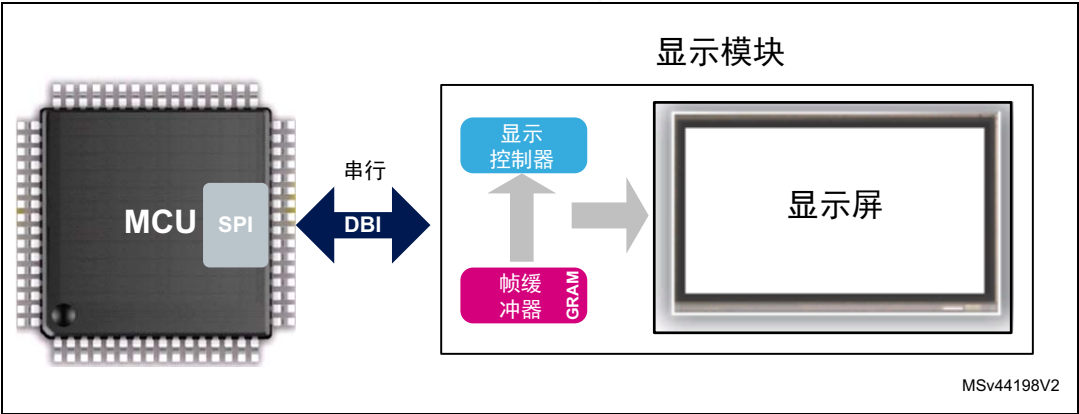


图 6中举例说明了一个MIPI-DBI的C类显示接口示例。

图6. MIPI-DBI C类接口



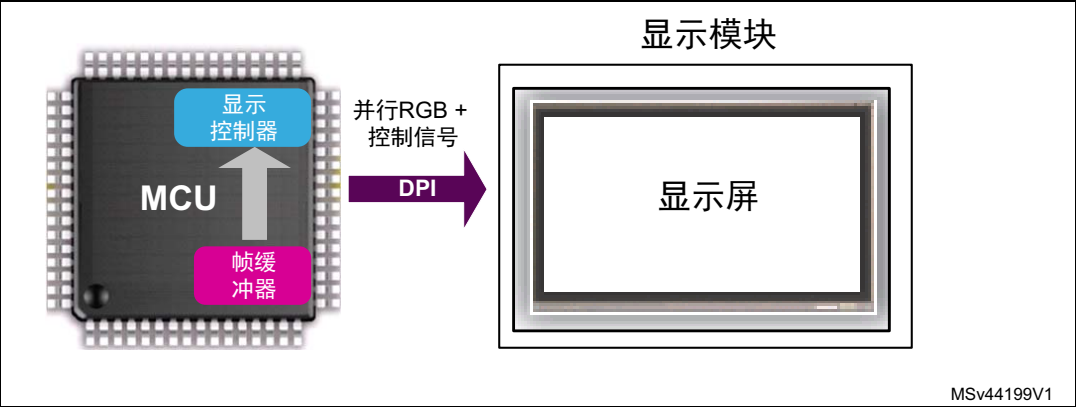
MIPI显示并行接口（MIPI-DPI）

DPI通过TFT控制器对接口进行标准化。一个例子是当16到24位RGB信令与同步信号（HSYNC，VSYNC，EN和LCD_CLK）结合使用时。

DPI用来与没有帧缓冲器的显示器进行连接。像素数据必须实时流式传输到显示器。

其实时性能非常好，但它要求MCU具有大带宽以支持显示。

图7. MIPI-DPI接口



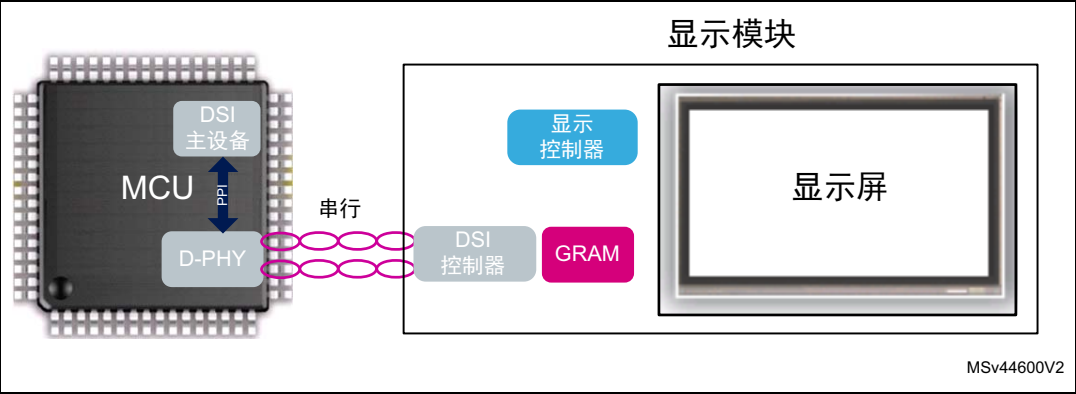
MIPI显示串行接口（MIPI-DSI）

为了减少连接显示器的线路数量，MIPI联盟对DSI进行了定义。DSI是高带宽多通道差分链路；它使用标准的MIPI D-PHY作为物理链路。

DSI封装了DBI或DPI信号，并通过PPI协议将它其发送到D-PHY。

图 8中举例说明了一个MIPI-DSI显示接口示例。

图8. MIPI-DSI接口



1.3 STM32 MCU支持的显示接口

- 下面总结了STM32 MCU支持的MIPI联盟显示接口：
- 所有STM32 MUC均支持MIPI-DBI C类（SPI）接口
 - 带F(S)MC的所有STM32 MCU均支持MIPI-DBI A类和B类接口
 - 带LTDC的STM32 MCU支持MIPI-DPI接口
 - 嵌入DSI主机的STM32 MCU支持MIPI-DSI接口
- 表 2中还阐述并总结了STM32微控制器所支持的显示接口。

表2. STM32 MCU支持的显示接口

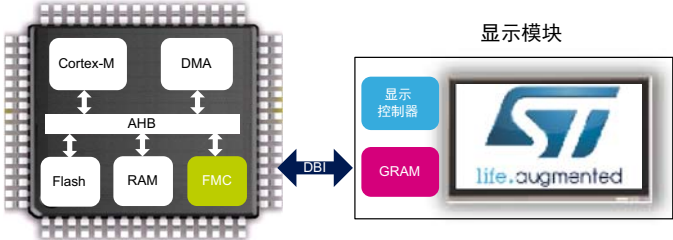
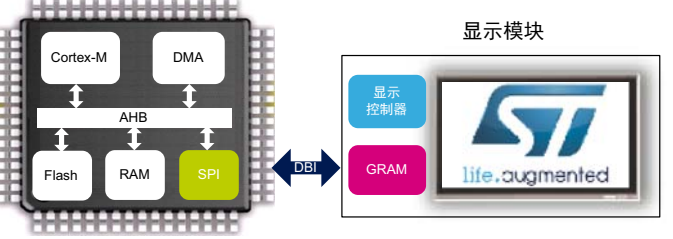
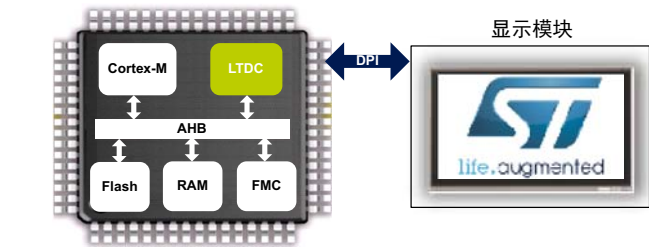
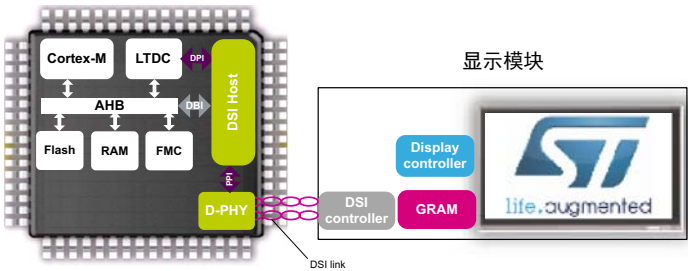
显示器界面		将显示面板连接到STM32 MCU ⁽¹⁾
DBI ⁽²⁾	Motorola 6800 DBI A类	 <p>MSv44647V1</p>
	Intel 8080 DBI B类	
	SPI DBI C类	 <p>MSv44648V1</p>

表2. STM32 MCU支持的显示接口（续）

显示器界面	将显示面板连接到STM32 MCU ⁽¹⁾
DPI: 使用LTDC的并行RGB ⁽³⁾	<div><p>MSv44649V1</p></div>
DSI ⁽⁴⁾	<div><p>MSv44650V1</p></div>

1. 紫色箭头表示显示器的像素数据路径。
2. 关于如何利用STM32的F(S)MC来支持Motorola 6800和Intel 8080的更多信息，请参考应用笔记与高密度STM32F10xxx FSMC相连接的TFT LCD（AN2790）。
3. 所有其他没有LTDC外设的STM32MCU均可以使用FSMC和DMA直接驱动LCD-TFT面板。请参考应用笔记使用STM32F10xx FSMC外设的直接驱动QVGA TFT-LCD（AN3241）。
4. 仅 表 3 中所嵌入DSI主设备的STM32 MCU可支持DSI接口。更多信息，请参考STM32微控制器上的DSI主设备（AN4860）。



2 LTDC控制器和STM32 MCU图形产品组合概述

本节说明LTDC控制器的优点并总结STM32微控制器图形产品组合。

2.1 STM32 MCU上的LCD-TFT显示控制器

STM32微控制器上的LTDC是片上LCD显示控制器，可提供高达24位的并行数字RGB信号，以便与各种显示面板连接。LTDC还可以像AMOLED显示器一样使用并行RGB接口来驱动其他显示技术。LTDC可以连接既不嵌入控制器也不嵌入图形RAM的低成本显示面板。

2.2 LTDC可用性和STM32系列的图形产品组合

表 3总结了嵌入LTDC的STM32，并详细介绍了相应的可用图形产品组合。

表3. 嵌入了LTDC的STM32 MCU及其可用的图形产品组合

STM32系列	闪存 (字节)	片上 SRAM (字节)	Quad- SPI ⁽¹⁾	最高AHB 频率 (MHz) ⁽²⁾	最高FMC SRAM和 SDRAM频率 (MHz)	最大 像素 时钟 (MHz) ⁽³⁾	JPEG 编解码 器	DMA2D ⁽⁴⁾	MIPI -DSI 主机 ⁽⁵⁾	图形库
STM32F429/ 439	最大 2 M	256 k	无	180	90	83	无	有	无	TouchGFX Embedded wizard SEGGER STemWin
STM32F469/ 479	最大 2 M	384 k	有	180	90	83	无	有	有	
STM32F7x6	最大 1 M	320 k	有	216	100	83	无	有	无	
STM32F7x7	最大 2 M	512 k	有	216	100	83	有	有	无	
STM32F7x8/ STM32F7x9			有	216	100	83	有	有	有	

1. Quad-SPI接口可以连接外部存储器以扩大应用规模。有关STM32MCUQSPI接口的更多详细信息，请参考应用笔记STM32微控制器上的Quad-SPI (QSPI) 接口 (AN476)。
2. LTDC以AHB速度获取图形数据。
3. 对于IO级的最大像素时钟，请参考 表 10；对于系统级的最大像素时钟，则请参考 表 11。像素时钟 (LCD_CLK) 形成了相关STM32数据表。
4. Chrom-Art加速器[®]
5. 集成MIPI-DSI控制器使得PCB设计更简单，引脚更少，EMI（电磁干扰）更低并降低了功耗。关于STM32的MIPI-DSI主机的更详细信息，请参见应用笔记AN4860。

2.3 智能架构中的LTDC

LTDC是AHB架构上的主设备，可以对内部和外部存储器进行读访问。LTDC有两个独立的层，每层都有自己的FIFO，从而使显示更加灵活。

LTDC控制器以AHB总线速度自动从帧缓冲器提取图形数据。然后将图形数据存储在其中FIFO内部层中，随后驱动到显示器。

该系统架构使图形可以在没有任何CPU介入的情况下构建并绘制到屏幕上。LTDC从帧缓冲器中检索属于图像的数据，而Chrom-ArtAccelerator[®]（DMA2D）则准备下一张图像。

LTDC接口集成在智能架构中，可以：

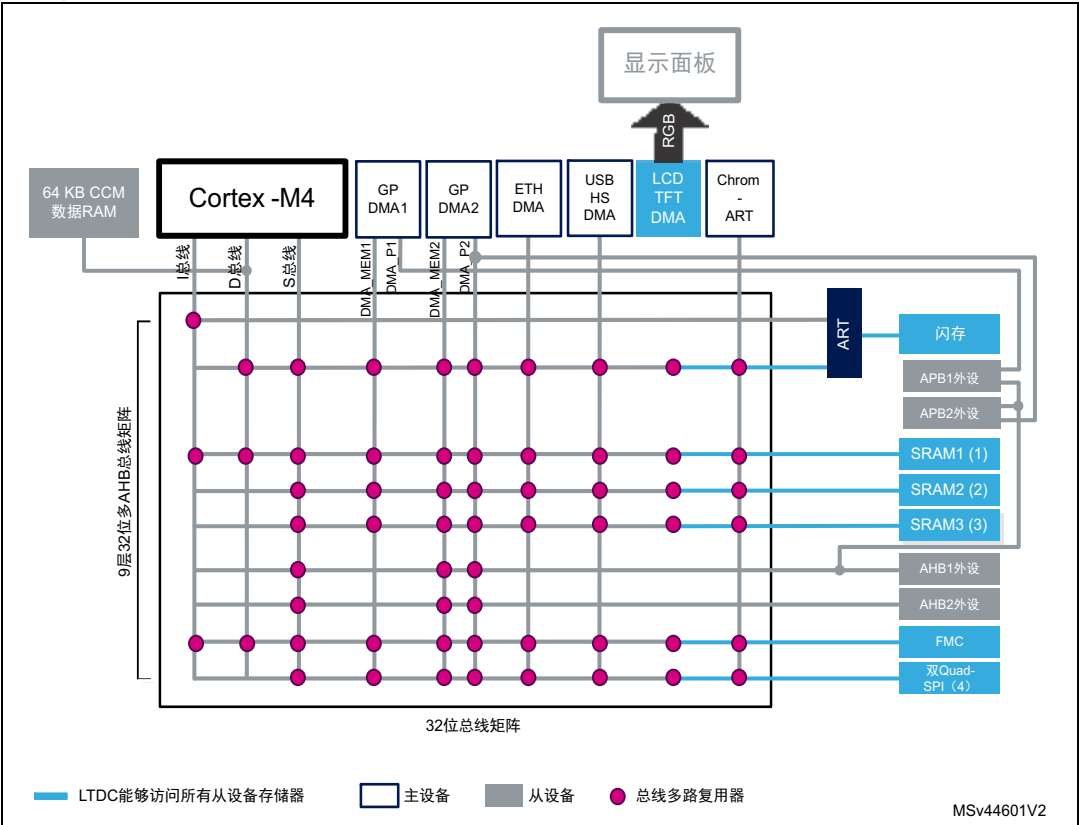
- LTDC自动从帧缓冲器（可以是内部存储器，如内部闪存、内部SRAM，或外部存储器，如FMC_SDRAM或Quad-SPI）读取图形数据并将其驱动到显示器。
- 作为AHB主设备的DMA2D可用于从图形密集型任务中为CPU减荷。
- 即使在CPU不运行时的睡眠模式下，LTDC也能够继续显示图形。
- 多层AHB总线架构提高了内存吞吐量及性能。

STM32F429/439和STM32F469/479微控制器上的系统架构

STM32F429/439系列和STM32F469/479系列的系统结构主要由32位多层AHB总线矩阵组成，它们将10个主设备和9个从设备（对于STM32F429/F439是8个从设备）互连起来。LTDC是AHB总线矩阵上的十个AHB主设备之一。

LTDC可以自动访问AHB总线矩阵上的所有存储器从设备，如FLASH、SRAM1、SRAM2、SRAM3、FMC或Quad-SPI，从而实现高效的数据传输，非常适合图形应用。[图 9](#) 显示了STM32F429/439和STM32F469/479系列产品系统中的LTDC互连。

图9. STM32F429/439和STM32F469/479系列产品中的LTDC AHB主设备智能架构



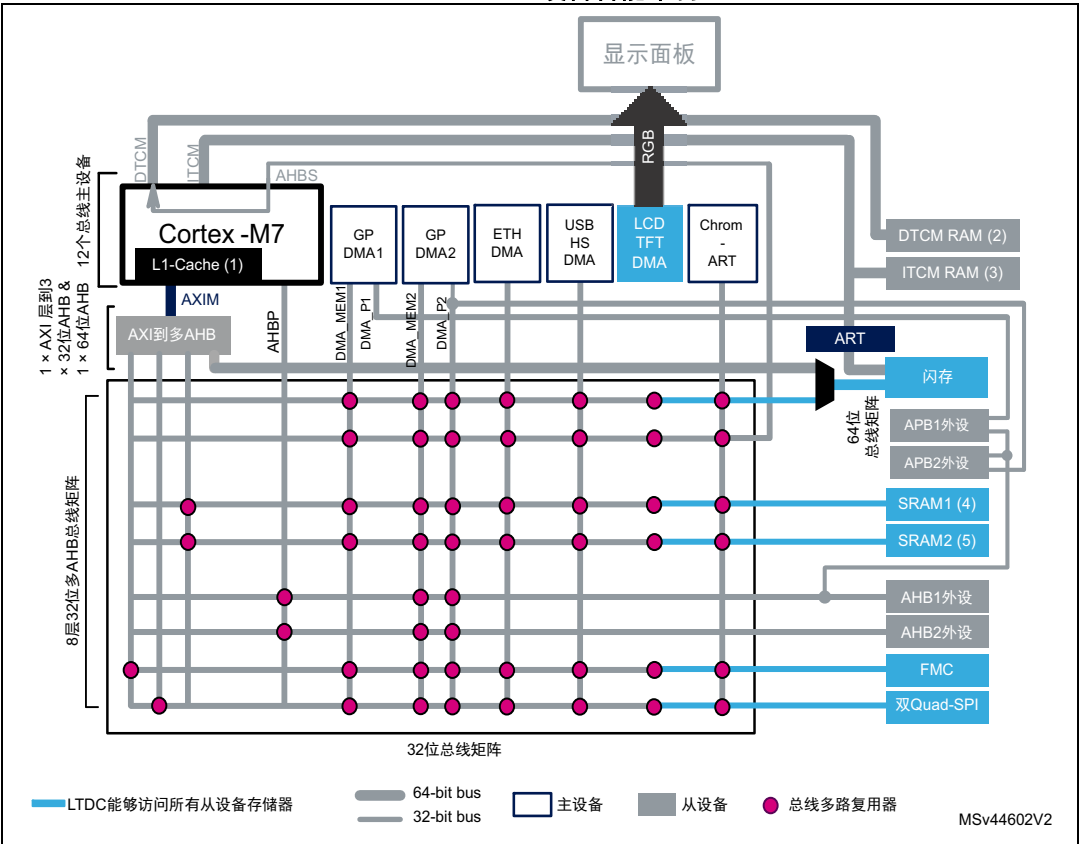
1. SRAM1大小 = 对于STM32F429/439为112 KB，对于STM32F469/479为160 KB
2. SRAM2大小 = 对于STM32F429/439为16 KB，对于STM32F469/479为32 KB
3. SRAM3大小 = 对于STM32F429/439为64 KB，对于STM32F469/479为128 KB
4. 双路Quad-SPI接口仅适用于STM32F469 / 479

STM32F7x6、STM32F7x7、STM32F7x8和STM32F7x9上的系统架构

STM32F7x6、STM32F7x7、STM32F7x8和STM32F7x9系列产品的系统架构主要由32位多层AHB总线矩阵组成，它们将12个主设备和8个从设备互连起来。LTDC是AHB总线矩阵上的十二个AHB主设备之一。

LTDC可以自动访问AHB总线矩阵上的所有存储器从设备，如FLASH、SRAM1、SRAM2、FMC或Quad-SPI，从而实现高效的数据传输，非常适合图形应用。图10显示了STM32F7x6、STM32F7x7、STM32F7x8和STM32F7x9系统中的LTDC互连。

图10. STM32F7x6、STM32F7x7、STM32F7x8和STM32F7x9中的
LTDC AHB主设备智能架构



1. I/D Cache大小 = 对于STM32F7x6为4 KB
I/D Cache大小 = 对于STM32F7x7、STM32F7x8和STM32F7x9为16 KB
2. DTCM RAM大小 = 对于STM32F7x6为64 KB
DTCM RAM大小 = 对于STM32F7x7、STM32F7x8和STM32F7x9为128 KB
3. ITCM RAM大小 = 对于STM32F7x6、STM32F7x7、STM32F7x8和STM32F7x9为16 KB
4. SRAM1大小 = 对于STM32F7x6为240 KB
SRAM1大小 = 对于STM32F7x7、STM32F7x8和STM32F7x9为368 KB
5. SRAM2大小 = 对于STM32F7x6、STM32F7x7、STM32F7x8和STM32F7x9为16 KB

2.4 使用STM32 LTDC控制器的优势

表 4总结了使用STM32嵌入式LTDC接口的主要优点。

表4. 使用STM32 MCU LTDC控制器的优势

优点	注释
节约成本	与其他DBI接口（SPI、Motorola 6800或Intel 8080）相比，LTDC可以连接任意无显示控制器和GRAM的低成本显示模块。
CPU被减荷。	LTDC是一款自身拥有DMA的AHB主设备，它可以在任意AHB存储器中自主获取数据，无需任何CPU干预。
不需要额外的应用层	LTDC硬件完全能够管理数据读取、RGB输出和信号控制，因此不需要额外的应用层。
完全可编程的分辨率可以支持自定义和标准显示	分辨率完全可编程，其总宽度达4096像素，总高度达2048行，像素时钟高达83 MHz。 可支持自定义和标准分辨率（QVGA、VGA、SVGA、WVGA、XGA、HD等）。
灵活的色彩格式	可以对每个LTDC层进行配置，使其能够以所需像素格式来提取帧缓冲器（参见第 3.3.2 节：可编程层：颜色帧缓冲器）。
灵活的并行RGB接口	灵活的并行RGB接口可以驱动16位、18位和24位显示器。
适用于智能手表等低功耗移动应用。	当CPU处于休眠模式时，LTDC能够继续读取图形数据和驱动显示。

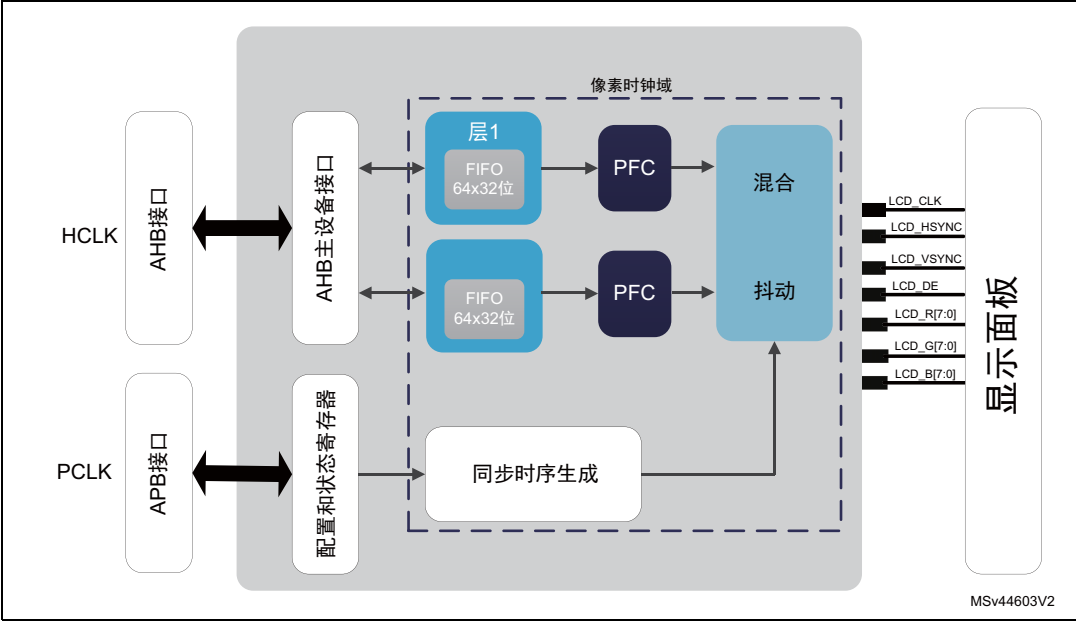
3 LCD-TFT (LTDC) 显示控制器说明

LTDC是以逐行方式读取图像数据的控制器。其存储器访问模式的长度为64字节，但当到达一行的结尾并且剩余数据少于64个字节时，LTDC将提取剩余的数据。

3.1 功能描述

在每个像素时钟上升沿或时钟下降沿，并在屏幕有效区域内，LTDC层从其FIFO中检索一个像素数据，将其转换为内部ARGB8888像素格式，并将其与背景和/或其他图层像素颜色进行混合。得到的像素以RGB888格式编码，通过抖动单元并被驱动到RGB接口中。像素便会显示到屏幕上。

图11. LTDC框图



3.1.1 LTDC时钟域

LCD-TFT控制器外设使用3个时钟域：

- AHB时钟域（HCLK）：用来将数据从存储器传输到FIFO层，反之亦然。
- APB时钟域（PCLK）：用来访问配置和状态寄存器。
- 像素时钟域（LCD_CLK）：用来生成LCD-TFT接口信号。LCD_CLK输出应通过PLL按照面板要求配置。

3.1.2 LTDC复位

LTDC可通过设置RCC_APB2RSTR寄存器中的LTDCRST位来复位。

3.2 灵活的时序和硬件接口

由于其时序和硬件接口的灵活性，LCD-TFT控制器能够驱动多台具有不同分辨率和信号极性的监控器。

3.2.1 LCD-TFT引脚和信号接口

为了驱动LCD-TFT显示器，LTDC利用简单的3.3V信号提供了多达28个信号，包括：

- 像素时钟LCD_CLK。
- 数据使能LCD_DE。
- 同步信号（LCD_HSYNC和LCD_VSYNC）。
- 像素数据RGB888。

注：如果接口兼容，那么LTDC控制器还可以支持其他显示技术。

LTDC接口输出信号如 表 6 中所示。

表5. LTDC接口输出信号 .

LCD-TFT 信号	说明
LCD_CLK	LCD_CLK用作LCD-TFT的数据有效信号。只有在LCD_CLK上升沿或下降沿才会显示该数据。
LCD_HSYNC	行同步信号（LCD_HSYNC）管理水平线扫描，作为行显示选通。
LCD_VSYNC	帧同步信号（LCD_VSYNC）管理垂直扫描，作为帧更新选通。
LCD_DE	DE信号向LCD-TFT指示RGB总线中的数据是有效的，并且该数据必须被锁存才能绘制出来。
像素RGB数据	可以对LTDC界面进行配置，使之输出多种色深。它最多可以使用24条数据线（RGB888）作为显示接口总线。

其它信号

通常情况下，显示面板接口还包含其他信号，这些信号不属于 表 5 中所述LTDC信号的一部分。显示模块要完全发挥作用，这些额外的信号 是必需的。LTDC控制器只能驱动 表 5 中所述的信号。

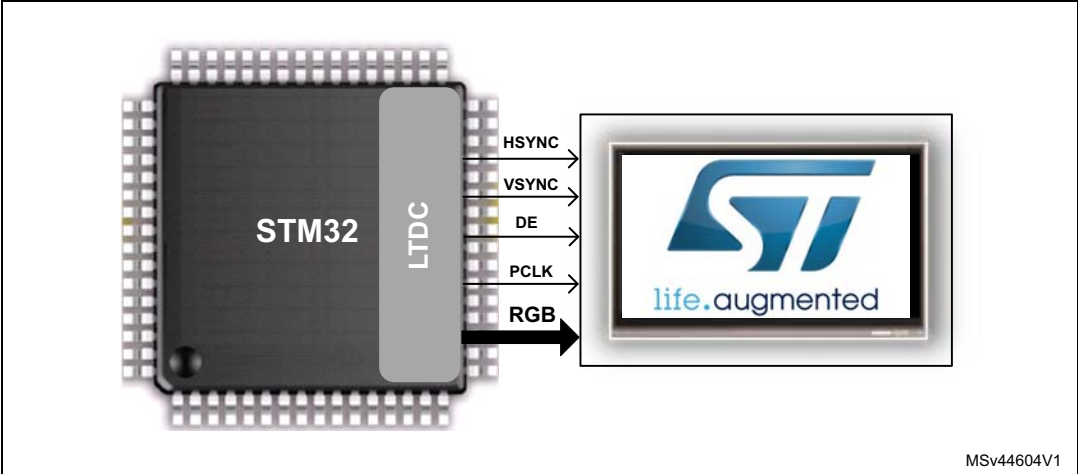
不属于LTDC的信号可以使用GPIO和其他外设进行管理，这可能需要特定的电路。

显示面板通常会嵌入背光单元，背光单元需要额外的背光控制电路和GPIO。

一些显示面板需要复位信号以及串行接口（如I2C或SPI）。这些接口通常用于显示器初始化命令或触摸面板控制。

图 12 显示了使用 表 5 中所示LTDC接口信号连接到STM32 MCU的显示面板。

图12. LTDC信号接口



LTDC可以按照以下并行格式输出数据：RGB565、RGB666和RGB888。因此可以连接16位的RGB565、18位的RGB888或24位的RGB888显示器。

LTDC信号极性可编程

LTDC控制信号的极性是可编程的，这使得STM32微控制器能够驱动任意RGB并行显示器。利用LTDC_GCR寄存器，可以将控制信号（Hsync, Vsync和数据使能DE）以及像素时钟（LCD_CLK）定义为高电平有效或低电平有效。

3.2.2 对于不同的显示器尺寸，其时序完全可编程

由于其“时序灵活性”，LTDC外设可以支持任意显示器尺寸，这些显示器尺寸可以满足寄存器中最大的可编程时序参数以及表 3、表 11和表 13中所述的最大支持像素时钟。

编程时用户应当考虑表 6中所述的时序寄存器，因为LTDC时序和同步信号应该被编程为与显示规格相匹配。

表 6概括了LTDC所支持的时序寄存器。

表6. LTDC时序寄存器

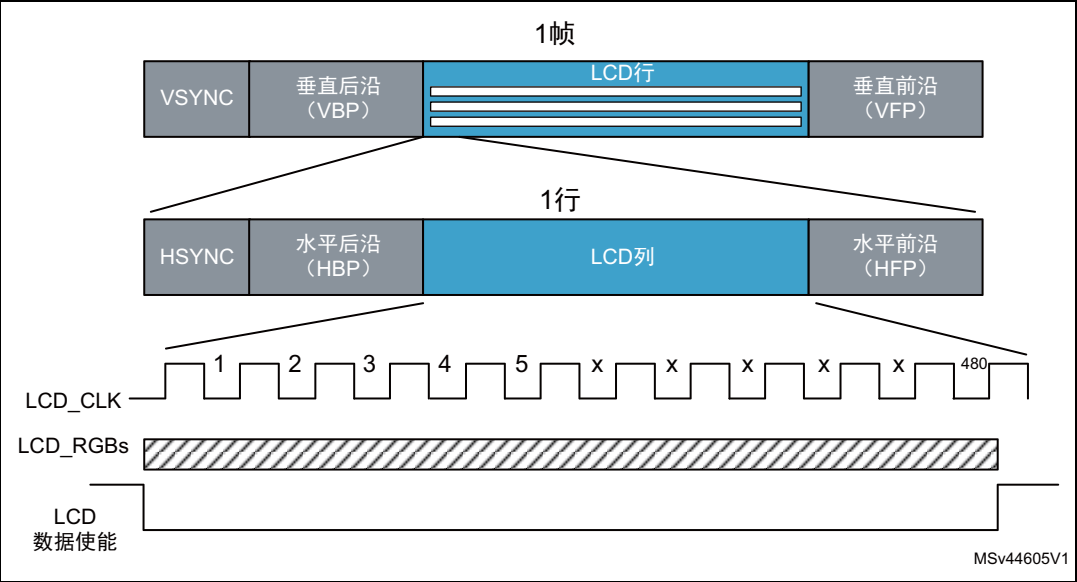
寄存器		时序参数	要编程的值
LTDC_SSCR ⁽¹⁾	HSW[11:0]	HSYNC宽度 - 1	从1至4096像素
	VSH[11:0]	VSYNC高度 - 1	从1至2048行
LTDC_BPCR	AHBP[11:0]	HSYNC宽度 + HBP - 1	从1至4096像素
	AVBP[10:0]	VSYNC高度 + VBP - 1	从1至2048行
LTDC_AWCR	AAW[11:0]	HSYNC宽度 + HBP + 有效宽度 - 1	从1至4096像素
	AAH[10:0]	VSYNC高度 + BVBP + 有效高度 - 1	从1至2048行
LTDC_TWCR	TOTALW[11:0]	HSYNC宽度 + HBP + 有效宽度 + HFP - 1	从1至4096像素
	TOTALH[10:0]	VSYNC高度 + BVBP + 有效高度 + VFP - 1	从1至2048行

1. 在HSW [11:0]中将HSYNC设置为0给出的是一个LCD_CLK的脉冲宽度。在VSW [11:0]中将VSYNC设置为0给出的是一整个行扫描周期

典型LTDC显示帧示例

图 13显示了一个典型LTDC显示帧示例，其中显示了表 6中所示的时序参数。

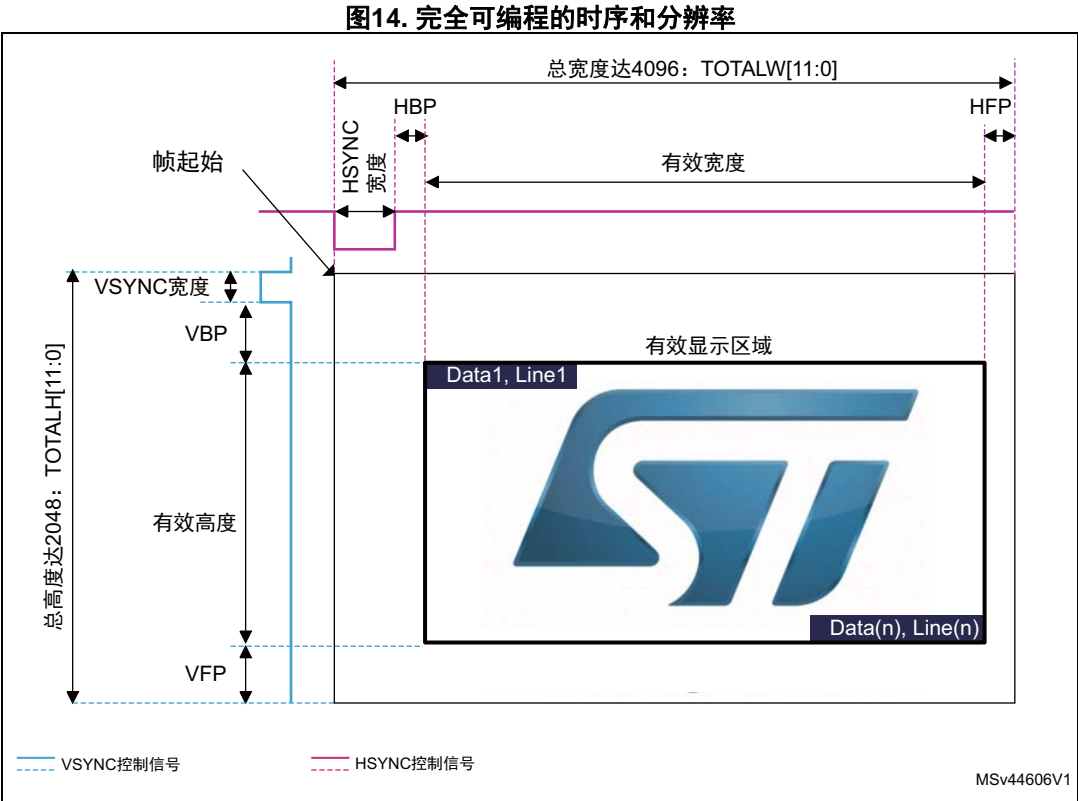
图13. 典型LTDC显示帧（有效宽度 = 480像素）



LTDC灵活时序

LTDC外设允许用户连接任意显示器尺寸，总宽度可达4096像素，总高度可达2048行（参见表 6）。

图 14中举例说明了完全可编程的时序和分辨率。

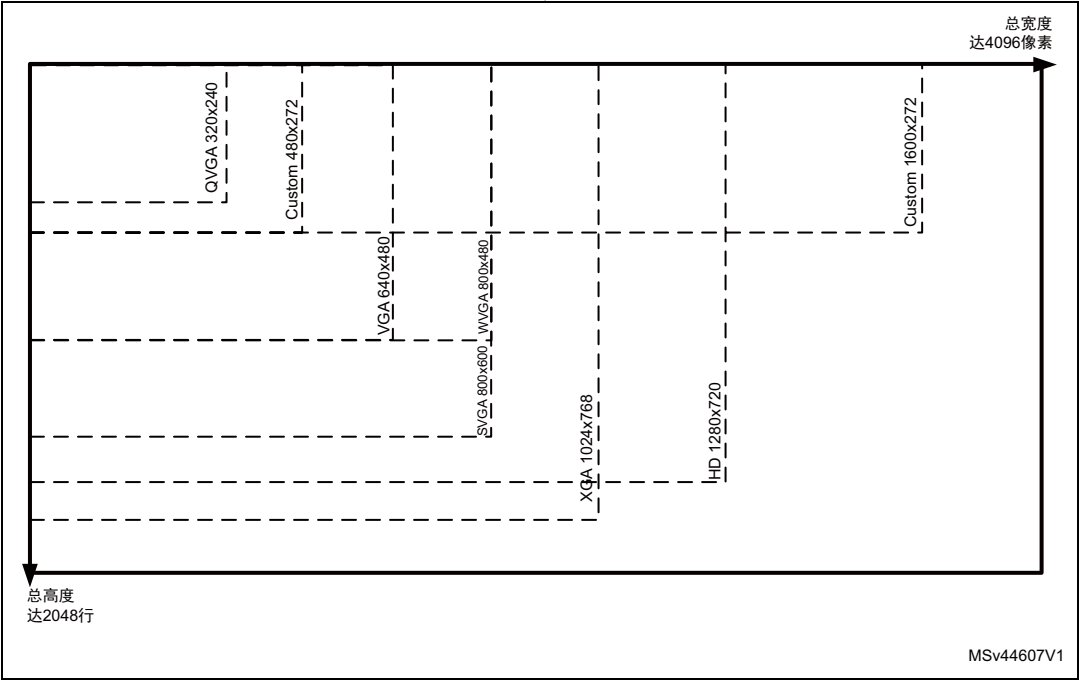


注意：只要满足以下条件，LTDC就可以支持图 15中所述最大总面积4096 x 2048之内的任何显示分辨率：

- 显示面板像素时钟不得超过表 2中的最大LTDC像素时钟
- 显示面板像素时钟不得超过帧缓冲器带宽的最大STM32像素时钟（见第 4.2节：检查显示器尺寸和色深与硬件配置的兼容性）。

图 15显示了一些由LTDC支持的属于最大4096 x 2048之内的自定义和标准分辨率。

图15. LTDC完全可编程显示分辨率
总宽度达4096像素，总高度达2048行



1. 此图中仅显示有效显示区域。

3.3 两个可编程LTDC层

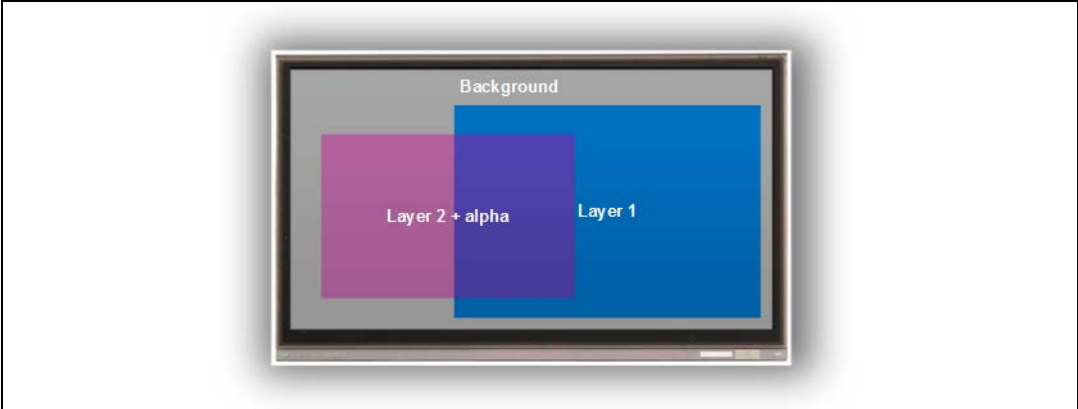
LTDC有两层，每层都可以分别启用、禁用和配置。图层显示的顺序是固定的，因此始终是由下至上的。如果使能两个层，则层 2 为顶部显示窗口。

LTDC具有可配置的混合因数。混合始终使用alpha值来激活。混合顺序固定，即由下至上。如果使能了两层，首先第 1 层将与背景色混合，随后第 2 层与第 1 层和背景的混合颜色结果再次混合。

背景颜色可通过LTDC_BCCR寄存器进行编程。可以在RGB888格式中编程常量背景颜色，其中BCRED[7:0]字段用于红色值，BCGREEN[7:0]用于绿色值，BCBLUE[7:0]用于蓝色值。

图 16描述了两层与背景的混合。

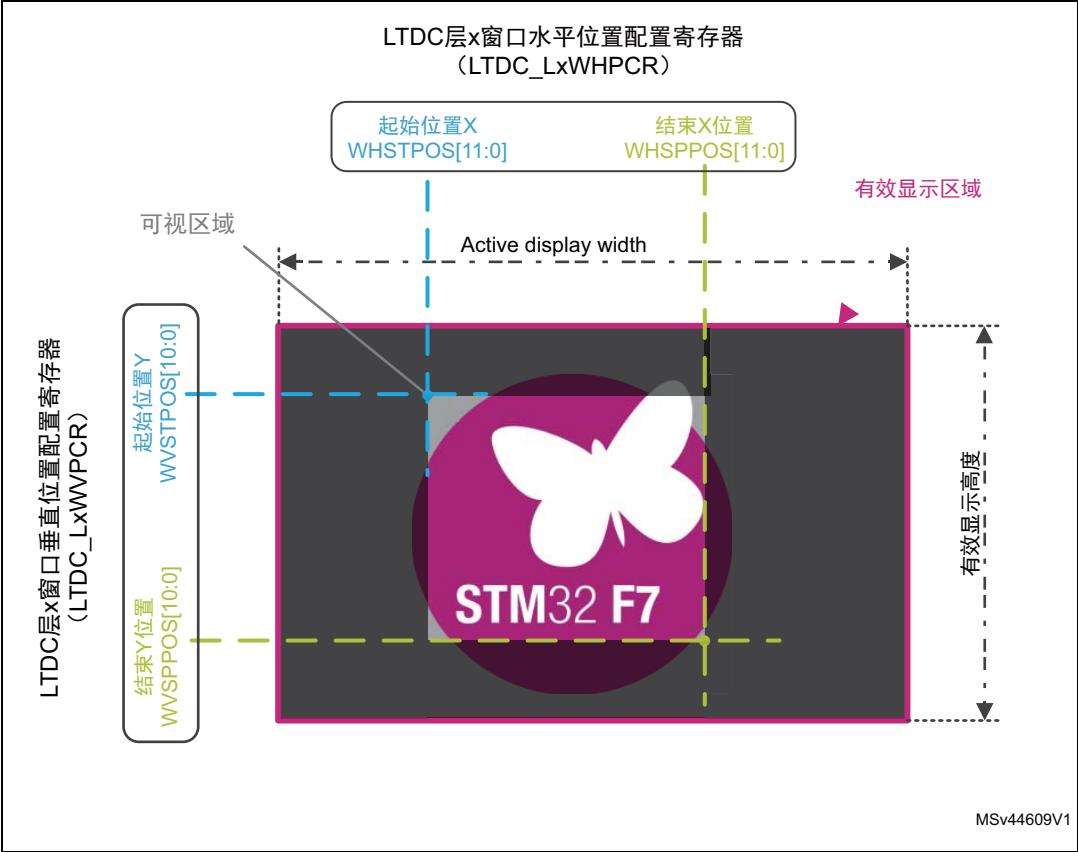
图16. 两层与背景混合



3.3.1 灵活的窗口位置和尺寸配置

每个图层都可在运行时进行定位和调整大小，并且必须位于有效显示区域内。可编程的图层位置和尺寸定义了一行中的第一个/最后一个可见像素和窗口中的第一个/最后一个可见行。它可以显示完整图像（所有有效显示区域）或仅显示图像帧的一部分。图 17显示了一个小窗口，其中只显示图像的一部分，而其余区域不显示。

图17. 层窗口可编程尺寸和位置



1. LTDC_LxWHPCR和LTDC_LxWVPCR分别是LTDC层x窗口水平和垂直位置配置寄存器，其中“x”可以指代层1或层2。

3.3.2 可编程层：颜色帧缓冲器

对于不同的颜色帧缓冲器和间距，每个图层都有可配置的特定行数和行长。

颜色帧缓冲区地址

每个层的颜色帧缓冲区均有一个起始地址，该地址通过 LTDC_LxCFBAR 寄存器进行配置。

颜色帧缓冲器长度（尺寸）

行长和行数参数用来停止在帧缓冲器末尾从FIFO层预取数据。

行长（以字节计）在LTDC_LxCFBLR寄存器中进行配置。

行数（以字节计）在LTDC_LxCFBLNR寄存器中进行配置。

颜色帧缓冲区间距

间距是指一行开始和下一行开始之间的距离，以字节为单位。它通过LTDC_LxCFBLR寄存器配置。

像素输入格式

可编程像素格式用于存储在每个LTDC层帧缓冲器中的所有数据。

对于每一层，可以分别配置特定的像素输入格式。LTDC每层最多可配置8种可编程输入颜色格式。

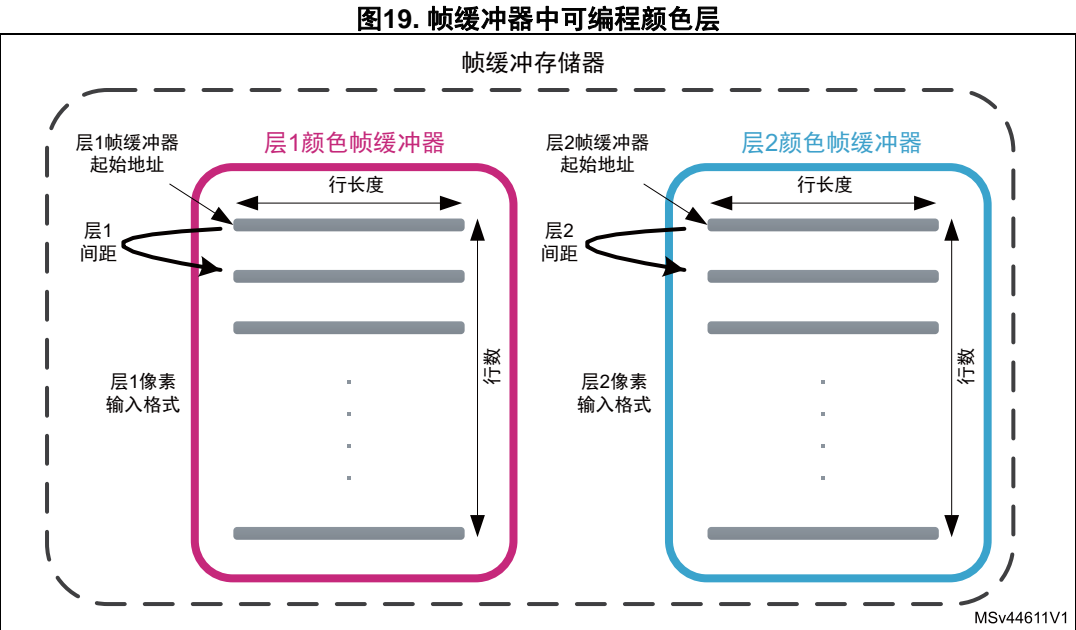
图 18说明了像素数据映射与所选输入颜色格式的关系。

图18. 像素数据映射与颜色格式的关系

	@ + 4	@ + 3	@ + 2	@
直接颜色	ARGB8888	Ax[7:0] Rx[7:0] Gx[7:0] Bx[7:0]		
	RGB888	Bx+1[7:0] Rx[7:0] Gx[7:0] Bx[7:0]		
	ARGB1555	Ax+1[0] Rx+1[4:0] Gx+1[4:3] Gx+1[2:0] Bx+1[4:0] Ax[0] Rx[4:0] Gx[4:3] Gx[2:0] Bx[4:0]		
	ARGB4444	Ax+1[3:0] Rx+1[3:0] Gx+1[3:0] Bx+1[3:0] Ax[3:0] Rx[3:0] Gx[3:0] Bx[3:0]		
间接颜色	L8	Lx+3[7:0] Lx+2[7:0] Lx+1[7:0] Lx[7:0]		
	AL88	Ax+1[7:0] Lx+1[7:0] Ax[7:0] Lx[7:0]		
	AL44	Ax+3[3:0] Lx+3[3:0] Ax+2[3:0] Lx+2[3:0] Ax+1[3:0] Lx+1[3:0] Ax[3:0] Lx[3:0]		

MSv44610V1

图 19总结了所有层的颜色帧缓冲器可配置参数。



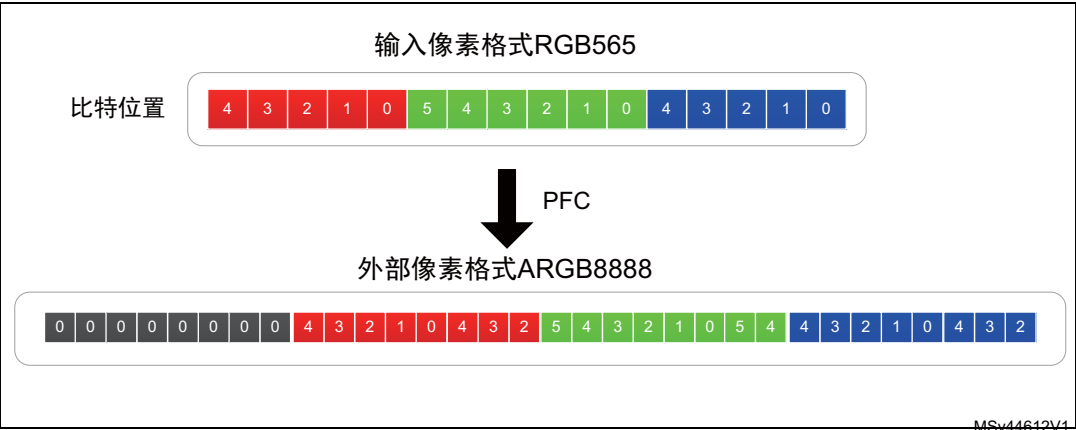
像素格式转换（PFC）

从帧缓冲器读取后，像素数据从配置的像素输入格式转换为内部ARGB8888格式。

宽度低于8位的分量按位重复扩展为8位。

选择8个MSB位。图 20显示从RGB565输入像素格式到内部ARGB8888格式的转换。

图20. 从RGB565输入像素格式到内部ARGB8888格式的像素格式转换



注：使用两个层时会在系统上产生带宽限制。在帧缓冲器计算期间，最好只用一个层，并使用Chrom-ArtAccelerator[®]进行合成（见第 4.2.2 节：考虑存储器时检查显示兼容性带宽要求）。

3.4 中断

LTDC外设支持两个全局中断：

- LTDC 全局中断。
- LTDC 全局错误中断。

每个全局中断都连接了两个LTDC中断（逻辑上相互分离），这两个LTDC中断可以通过特定的寄存器单独屏蔽。表 7总结了所有相关中断以及每个中断产生时的所有特例。

表7. LTDC中断总结

相关NVIC中断	中断事件	事件标志位 (LTDC_ISR寄存器)	使能位 (LTDC_IER寄存器)	清除位 (LTDC_ICR寄存器)	说明
LTDC 全局中断	行	LIF	LIE	CLIF	达到屏幕上定义的行时生成
	寄存器重载	RRIF	RRIE	CRRIF	在发生阴影重载时生成
LTDC 全局错误中断	FIFO 下溢 ⁽¹⁾	FUIF	FUIE	CFUIF	在FIFO为空的情况下请求像素时生成
	传输错误	TERRIF	TERRIE	CTERRIF	发生总线错误时生成

1. FIFO下溢中断用于确定显示器尺寸兼容性（见第 4.2.2 节：考虑存储器时检查显示兼容性带宽要求）。

3.5 低功耗模式

STM32电源状态对LTDC外设有直接影响。在睡眠模式下，LTDC不受影响，并持续将图形数据驱动到屏幕上。在待机和停止模式下，LTDC被禁用，不会驱动输出通过其并行接口。应在LTDC重新配置后退出待机模式。

由于STM32微控制器中嵌入了智能架构，这使得即使在睡眠模式下也可以启用所有外设，因此可以在CPU停止时的睡眠模式下驱动显示面板。此功能适合要求低功耗的可穿戴应用。

作为AHB主设备的LTDC可能会继续从FMC_SDRAM或Quad-SPI（当使用存储器映射模式时）获取数据，即使MCU进入休眠模式后也是如此。当到达屏幕上定义的行或发生阴影重载时，可以生成行事件或寄存器重载中断，来唤醒STM32。

有关降低功耗的更多信息，请参阅第 5 节。

表 8 总结了LTDC的状态与STM32的低功耗模式的关系。

表8. LTDC外设状态与STM32低功耗模式的关系

模式	说明
运行	激活
睡眠	激活。外设中断导致设备退出睡眠模式
停止	冻结。外设寄存器内容仍被保持
待机	掉电。在退出待机模式后，必须重新初始化外设



4 使用LTDC创建图形应用

本章说明使用LTDC进行图形应用开发之前和期间所需的不同步骤。用户首先应确定图形应用的要求，然后检查所需的显示器尺寸是否适合硬件配置。在图形应用兼容性检查阶段，用户可以使用表 17中所述的现有STM32参考板来评估其硬件和软件配置。

4.1 确定图形应用要求

确定图形应用要求是开始时的关键步骤。在开始创建图形应用之前，必须要定义的一些重要参数为：显示分辨率、色深以及待显示数据的性质（静态图像、文本或动画）。

当定义了上述基本参数后，用户应确定应用的图形硬件结构以及所需的硬件资源。用户应根据以下参数选择最适合的STM32封装（参见表 13）：

- 如果帧缓冲器需要外部存储器
- 外部帧缓冲器存储器总线宽度
- LTDC接口：RGB565、RGB666或RGB888，具体取决于显示模块
- 如果需要外部存储器来存储图形基元（QSPI或FMC_NOR）

4.2 检查显示器尺寸和色深与硬件配置的兼容性

当使用STM32微控制器开始图形应用开发时，用户通常已定义了所需的显示器尺寸和色深。在继续开发之前用户必须回答的一个关键问题是，*这种显示器尺寸和色深与具体硬件配置是否相匹配？*

要回答这个问题，用户应按照以下步骤进行：

1. 确定所需的帧缓冲器大小及其位置。
2. 检查显示器与帧缓冲存储器带宽要求的兼容性。
3. 检查显示面板接口与LTDC的兼容性。

4.2.1 帧缓冲存储器大小要求和位置

确定帧缓冲存储器大小及其位置是显示兼容性检查的关键参数。

RAM中支持帧缓冲器所需的内存空间应该是连续的，并且最小大小等于：

$$\text{帧缓冲器大小} = \text{像素数} \times \text{每像素位数}$$

如上面的公式所示，所需的帧缓冲器大小取决于显示分辨率及其色深。

帧缓冲器色深 (bpp) 不一定与显示色深相同。例如，可以使用RGB565帧缓冲器来驱动RGB888显示器。

注：双帧缓冲器配置时所需帧缓冲器大小也是双倍的。通常使用双缓冲区配置，其中一个图形缓冲区用于存储当前图像，而第二个缓冲区用于准备下一个图像。

表 9显示了不同像素格式下标准屏幕分辨率所需的帧缓冲器大小。

表9. 不同屏幕分辨率的帧缓冲器大小

屏幕分辨率	像素数	帧缓冲器大小 (KB) ⁽¹⁾			
		8 bpp	16 bpp	24 bpp	32 bpp
QVGA (320 x 240)	76800	75	150	225	300
自定义 (480 x 272) ⁽²⁾	130560	128	255	383	510
HVGA (480 x 320)	153600	150	300	450	600
VGA (640 x 480)	307200	300	600	900	1200
WVGA (800 x 480)	384000	375	750	1125	1500
SVGA (800 x 600)	480000	469	938	1407	1875
XGA (1024 x 768)	786432	768	1536	2304	3072
HD (1280 x 720)	921600	900	1800	2700	3600

- 1. 双帧缓冲器配置时所需帧缓冲器大小也是双倍的。
- 2. 自定义480 272显示器的一个示例是嵌入在STM32F746探索套件 (32F746GDISCOVERY) 中的ROCKTECH。

帧缓冲器位置

根据所需的帧缓冲器大小，它可以位于内部SRAM或外部SRAM/SDRAM中。
如果内部RAM不足以支持帧缓冲器，那么用户必须使用连接到FMC的外部SDRAM/SRAM。
因此，所需的帧缓冲器大小将决定是否需要使用外部存储器。所需的帧缓冲器大小取决于显示器的尺寸和色深。

帧缓冲器在内部SRAM中

根据帧缓冲器大小，它可以位于内部SRAM或外部SRAM或SDRAM中。
使用内部SRAM作为帧缓冲器可实现最高性能，并可避免LTDC的所有带宽限制问题。



使用内部SRAM而不是外部SRAM或SDRAM具有许多优点：

- 具有更高的吞吐量（0等待状态访问）。
- 减少了所需引脚数量，降低了PCB设计复杂性。
- 由于不需要外部存储器，因此降低了BOM，并因此降低了成本。

使用内部SRAM的唯一限制是其大小有限（几百KB）。当帧缓冲器大小超过可用存储器时，应使用外部SDRAM或SRAM（由FMC接口驱动）。但是，在处理外部存储器时，用户必须小心避免带宽限制。详情请参见 [第 4.5 节：图形性能优化](#)。

注： 色彩查找表CLUT可用于减少所需的帧缓冲器大小。（更多详细信息，请参考相关STM32 MCU 参考手册）。

4.2.2 考虑存储器时检查显示兼容性带宽要求

本节所述内容是解释在考虑帧缓冲存储器带宽时如何检查显示兼容性。为此，本节将介绍一些重要的带宽因素，并说明如何确定像素时钟和LTDC所需的带宽。最后，本节介绍了一种判断所需显示尺寸是否与特定硬件配置兼容的简单方法。

帧缓冲器存储器带宽因素

帧缓冲器位置固定时（无论是在内部存储器还是外部存储器中），用户应检查其带宽是否能够维持硬件配置。

为了检查存储器带宽能否维持LTDC所需的带宽，用户必须要考虑到对存储器的其他并发访问。

通常，位于内部RAM中的小尺寸帧缓冲器不需要高带宽。这是因为小尺寸帧缓冲器意味着低像素时钟，因此LTDC所需带宽较低。

要分析的更复杂的用例是帧缓冲器位于外部存储器（SDRAM或SRAM）时。

帧缓冲存储器总线并发性

- LTDC、DMA2D和CPU主设备

在使用外部SDRAM或SRAM存储器作为帧缓冲器的典型图形应用中，两个或三个主AHB主设备会同时使用相同的存储器。

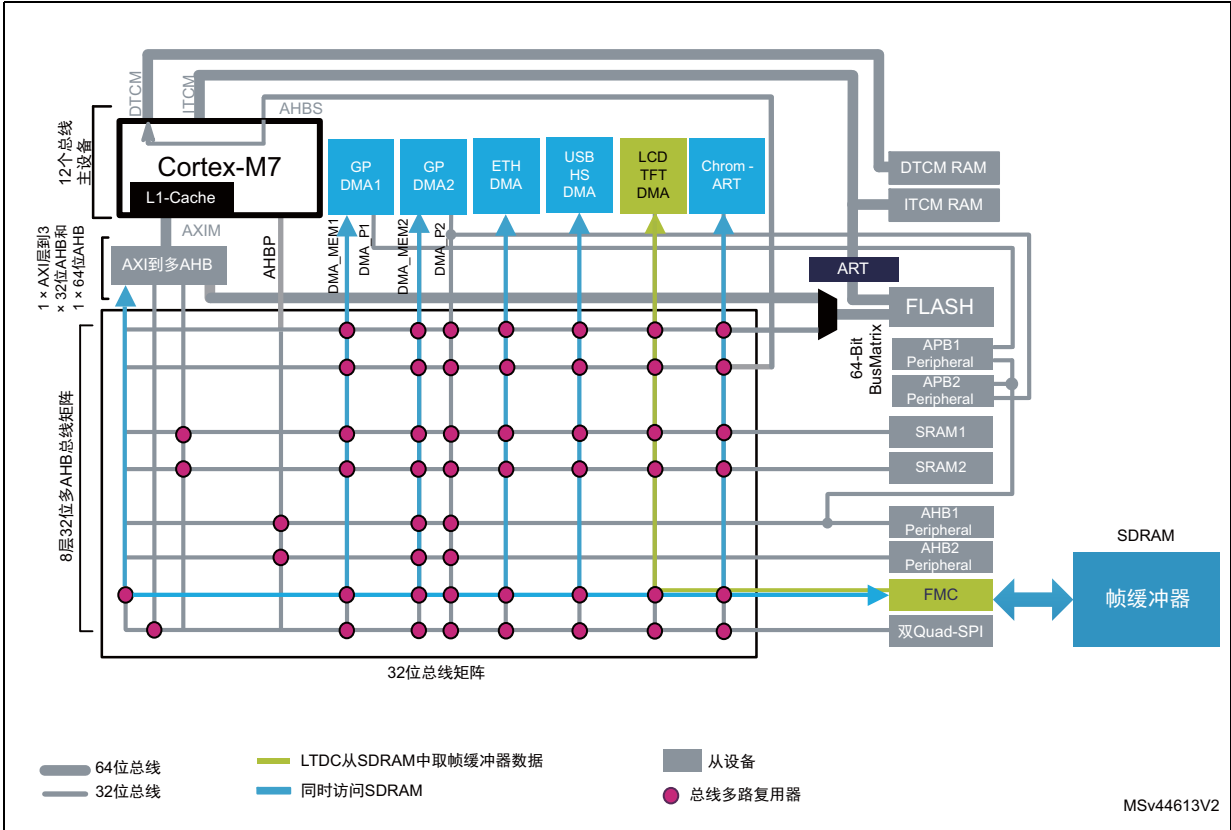
DMA2D（或CPU）更新要显示的下一个图像，而LTDC则获取并显示实际图像。存储器总线负载主要取决于LTDC所需带宽。

- 其他AHB主设备

外部SDRAM或SRAM存储器通常由其他主设备共享，而不仅仅由用于图形的设备共享。这种并发会导致总线负载较重，并可能影响图形性能。

[图 21](#)显示了所有可以同时访问SDRAM的AHB主设备。

图21. 同时访问SDRAM的AHB主设备



外部SDRAM/SRAM存储器总线宽度

当帧缓冲器位于外部SDRAMSRAM中时，用户应该考虑到外部存储器的运行频率大约是系统频率的一半或三分之一。这就是为什么存储器带宽被视为整个图形系统的瓶颈的原因。

检查显示器兼容性所需的一个参数是存储器总线宽度。对于SDRAM，用户可以采用8位、16位或32位配置。

如前所述，要分析的最复杂的用例是帧缓冲器置于外部存储器时：
主设备在同一外部存储器上并行访问会导致更多延迟并影响其吞吐量。

确定像素时钟和LTDC所需带宽

像素时钟计算

像素时钟是用于检查显示器尺寸与特定硬件配置的兼容性的关键参数。
要得到显示器的典型像素时钟，请参考显示器的数据表。计算出的像素时钟应符合显示器规格。

利用以下公式计算特定刷新速率的像素时钟：

$$\text{LCD_CLK (MHz)} = \text{总屏幕尺寸} \times \text{刷新速率}$$

其中，总屏幕尺寸 = 总宽度 x 总高度。

LTDC所需带宽

LTDC所需带宽主要取决于三个因素：

- 所用LTDC层数
- LTDC层色深
- 像素时钟（取决于显示面板分辨率和刷新速率）

最大所需带宽可以按如下所述进行计算：

- 如果仅使用了一个LTDC层
 - LTDC所需带宽 = LCD_CLK x BppL1
- 如果使用了两个LTDC层
 - LTDC所需带宽 = LCD_CLK x (BppL1 + BppL2)

这里BppL1和BppL2分别为LTDC层1和层2的色深。

LTDC所需带宽不能超过存储器的可用带宽，否则将出现显示问题，并且FIFO下溢运行标志将被置位（如果使能了FIFO下溢运行中断）。

注：如果用于存储帧缓冲器的存储器还用于其他应用，那可能会影响系统的图形性能。

检查使用的显示分辨率是否适合硬件配置

检查特定色深的显示器尺寸是否与存储器带宽兼容的一般方法是：

1. 根据显示器尺寸来计算像素时钟，或从显示器数据表中提取像素时钟。
2. 检查显示器的像素时钟是否未超过 [表 10](#)或 [表 11](#)中所述系统支持的最大像素时钟。用户应该使用以下参数从 [表 10](#)或 [表 11](#)中提取所用硬件配置对应的最大支持像素时钟：
 - a) 所用LTDC层数。
 - b) 所用系统的时钟速度HCLK和帧缓冲器存储器速度。
 - c) 外部帧缓冲器存储器总线宽度。
 - d) 同时访问外部帧缓冲存储器的AHB主设备的数量。

3. 用户应进行一些测试来确认硬件与所需显示器尺寸和色深的兼容性。要做到这一点，用户应监控LTDC_ISR寄存器中的LTDC FIFO下溢中断标志。
- 如果FIFO下溢中断标志始终是复位的，则用户可以确定所需的显示尺寸与硬件配置兼容。
- 如果FIFO下溢标志被置位，那么用户应检查以下几点：
- 验证他是否从表 10或表 11提取了对应正确硬件的最大像素时钟（例如，用户正在使用16位SDRAM，但他提取的是与32位SDRAM相对应的像素时钟，这是错误的）。
 - 颜色帧缓冲器行宽不是64字节对齐的（见第 4.5.2节：优化从外部存储器读取LTDC帧缓冲器的过程（SDRAM或SRAM））。
 - 对于STM32F7系列产品，MPU未正确配置，不能避免Cortex®-M7推测性读取访问SDRAM（参见第 4.6节：关于Cortex®-M7（STM32F7系列）的特别建议）。
 - 如果有两个以上的AHB主设备同时访问外部存储器而导致FIFO下溢仍然置位，则用户可以使用以下建议来放宽存储器带宽：
 - 仅使用一个LTDC层
 - 使用最大存储器总线宽度（使用32位SDRAM/SRAM，而不是16位或8位的）
 - 在LTDC没有取数据的消隐期间更新帧缓冲器内容
 - 使用最高系统时钟HCLK和最高存储器速度（FMC_SDRAM / FMC_SRAM）
 - 降低图像色深（BPP）
 - 关于存储器带宽优化的更多详细信息，请参考第 4.5节：图形性能优化。

注：要评估特定硬件配置下的STM32图形功能，用户可以使用表 17：STM32参考板，嵌入了LTDC并具有板上LCD-TFT面板中所述的STM32板。

图 22显示了典型的图形硬件配置，其中外部SDRAM连接到用于帧缓冲器的FMC。SDRAM存储器带宽取决于总线宽度和工作时钟。

SDRAM总线宽度可以是32位、16位或8位，而工作时钟取决于系统时钟HCLK和配置的预分频器（ $HCLK / 2$ 或 $HCLK / 3$ ）。

图22. 使用外部SDRAM时的典型图形硬件配置

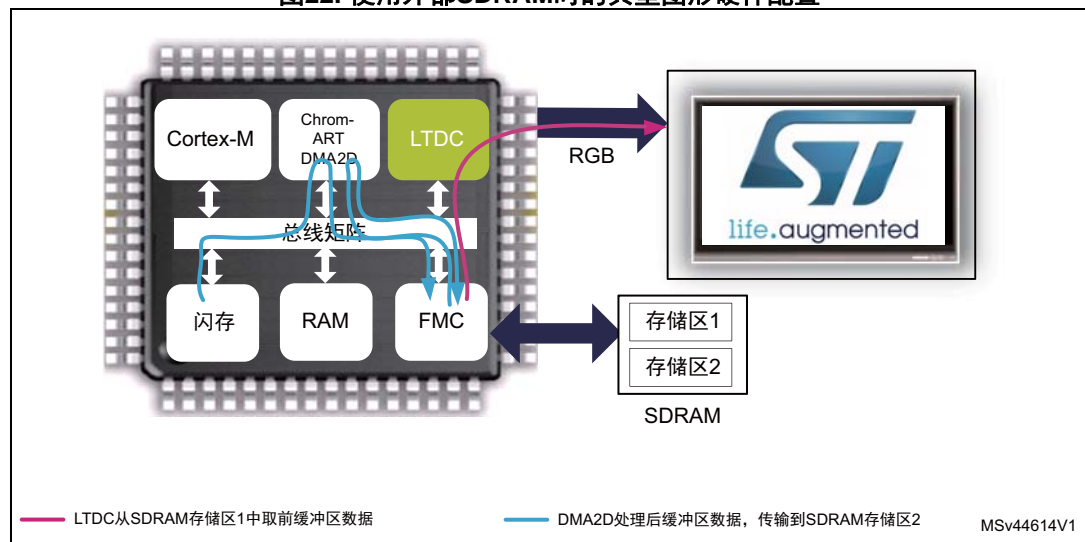


表 10列出了STM32F4x9系列产品的系统级最大支持像素时钟，表 11列出了以下条件
STM32F7x6、STM32F7x7、STM32F7x8和STM32F7x9系列的系统级最大支持像素时钟：

- 对于STM32F4x9系列，系统时钟HCLK运行于@ 180 MHz，SDRAM运行于 @ 90 MHz。
- 对于STM32F7x6、STM32F7x7、STM32F7x8和STM32F7x9系列，系统时钟HCLK运行于@ 200 MHz，SDRAM运行于 @ 100 MHz。
- 一个或两个AHB主设备同时访问SDRAM（LTDC或LTDC + DMA2D）。
- SDRAM总线宽度为16位或32位。
- 仅使用一个LTDC层或使用了两个LTDC层。
- LTDC层色深为8 bpp、16 bpp、24 bpp或32 bpp。

**表10. STM32F4x9，其HCLK @ 180 MHz并且SDRAM @ 90 MHz
最大支持像素时钟与LTDC配置和SDRAM总线宽度的关系**

所用 LTDC 层	色深 (bpp)	最大像素时钟 (MHz)			
		LTDC		LTDC + DMA2D	
		16 位 SDRAM	32 位 SDRAM	16 位 SDRAM	32 位 SDRAM
1层	32	38	67	22	35
	24	51	83	30	47
	16	76	83	45	70
	8	83	83	83	83
2层	32/32	19	33	NA	18
	32/24	22	38	13	21
	32/16	25	44	15	25
	32/8	30	53	19	30
	24/24	26	44	15	24
	24/16	31	53	18	30
	24/8	38	67	23	38
	16/16	39	67	22	37
	16/8	51	83	31	50
	8/8	78	83	46	74

表11. STM32F7x6、STM32F7x7、STM32F7x8和STM32F7x9，其HCLK @ 200 MHz并且SDRAM @ 100 MHz时，最大支持像素时钟与LTDC配置和SDRAM总线宽度的关系

所用 LTDC 层	色深 (bpp)	最大像素时钟 (MHz)			
		LTDC		LTDC + DMA2D	
		16 位 SDRAM	32 位 SDRAM	16 位 SDRAM	32 位 SDRAM
1层	32	42	74	25	39
	24	56	83	34	52
	16	83	83	51	78
	8	83	83	83	83
2层	32/32	21	37	12	20
	32/24	24	42	14	23
	32/16	28	49	17	28
	32/8	34	59	21	34
	24/24	29	49	17	27
	24/16	34	59	20	33
	24/8	42	74	26	42
	16/16	43	74	25	41
	16/8	57	83	34	56
	8/8	83	83	51	82

注：降低系统时钟（HCLK，然后LTDC）会导致图形性能下降。

STM32F4x9系列和STM32F7系列支持的显示分辨率示例

表 12列出了STM32F4x9系列和STM32F7系列在以下条件下支持的一些标准和自定义显示尺寸的示例：

- 对于STM32F4x9系列，系统时钟HCLK运行于@ 180 MHz，SDRAM运行于@ 90 MHz。
- 对于STM32F7x6、STM32F7x7、STM32F7x8和STM32F7x9系列，系统时钟HCLK运行于@200 MHz，SDRAM运行于@ 100 MHz。
- 仅使用了一个LTDC层。
- 两个AHB主设备同时访问SDRAM（LTDC + DMA2D）。

表12. 特定STM32硬件配置下所支持的显示分辨率示例

显示特性				STM32的LTDC配置	
分辨率	刷新速率 (Hz)	像素时钟 (MHz)	显示标准	色深	
				16 位 SDRAM	32 位 SDRAM
320 x 240 (QVGA)	60	5.6	定制	达32 bpp	
480 x 272		9.5			
640 x 480 (VGA)		25.175	工业标准	达24 bpp	达32 bpp
800 x 600 (SVGA)		40.000	VESA指南 ⁽¹⁾	达16 bpp	达24 bpp
1024 x 768 (XGA)		65		8 bpp	达16 bpp
1280 x 768		68.250	CVT R.B ⁽²⁾		达16 bpp ⁽⁴⁾
1280 x 720 (HD)		74.25	CEA ⁽³⁾		
1920 x 1080	30	74.25	CEA ⁽³⁾		

1. VESA（视频电子标准协会）是一个计算机显示标准的技术标准组织，提供了显示监视时序（DMT）标准。
2. CVT R.B：调节的视频时序降低了VESA的消隐标准。
3. CEA = 消费电子协会。
4. STM32F4x9微控制器最高可达8 bpp

4.2.3 检查显示面板接口与LTDC的兼容性

用户应根据应用需求来选择LCD面板。选择LCD面板时需要考虑的两个主要因素是分辨率和色深。这两个因素对以下参数有直接影响：

- 所需GPIO数。
- 帧缓冲器大小和位置。
- 显示器的像素时钟。

当选择了一个显示面板时，用户应：

- 确保显示器接口与LTDC（带控制信号的并行RGB）兼容。
- 检查控制信号是否可以由LTDC控制（有时需要额外的GPIO）。
- 确保显示信号电平与LTDC接口信号电平（VDD从1.8 V至3.6 V）匹配。
- 确保显示器的像素时钟可以被相关STM32微控制器数据表中定义的LTDC最大像素时钟支持。
- 确定LTDC时序可以支持显示时序参数（参见 [表 6：LTDC时序寄存器](#)）。
- 检查显示器的尺寸和色深是否可由LTDC支持（请参阅 [第 4.2.2 节：考虑存储器时检查显示兼容性带宽要求](#)）。

4.3 STM32封装选择指南

在图形应用开发的这个阶段，用户已经根据GPIO确定了应用的需求：

- 是否需要外部存储器，哪个是总线宽度。
- 使用哪种LTDC配置：RGB565、RGB666或RGB888。

在选择STM32封装时，用户必须考虑RGB接口可用性和应用在GPIO数量方面的要求。用户必须参考STM32的相关数据表来获得可用的GPIO封装。

有一个简单的方法，可以检查用户感兴趣的STM32封装是否与应用需要的GPIO数相匹配，就是使用STM32CubeMX（引脚分配选项卡）。

[表 13](#)总结了嵌入LTDC的STM32 MCU的可用封装和RGB接口。

表13. 带LTDC外设的STM32封装与RGB接口可用性的关系⁽¹⁾

产品	LQFP100	TFBGA100	LQFP144	UFBGA169	UFBGA176	LQFP176	LQFP208	TFBGA216	WLCSP143	WLCSP168	WLCSP180
STM32F429 SRM32F439	18	NA	18	24	24	24	24	24	18	NA	NA
STM32F469 STM32F479 ⁽²⁾	18	NA	18	24	24	24	24	24	NA	24	NA
STM32F7x6	18	18	24	NA	24	24	24	24	24	NA	NA
STM32F7x7	18	NA	24	NA	24	24	24	24	NA	NA	NA
STM32F7x8 ⁽²⁾	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	24
STM32F7x9 ⁽²⁾	NA	NA	NA	NA	NA	24	24	24	NA	NA	24

1. 带有“NA”的灰色单元格 = 该封装不适用于该指定产品。
值为“18”的单元格 = 仅支持RGB565和RGB666并行输出。
值为“24”的单元格 = 支持所有RGB565、RGB666和RGB888输出。
2. 集成MIPI-DSI控制器使得PCB设计更简单，引脚更少，有关STM32 MIPI-DSI主设备的更多详细信息，请参考应用笔记AN4860。

4.4 LTDC与DMA2D和CPU同步

4.4.1 DMA2D 的用法

DMA2D是AHB总线矩阵上的主设备，将图形数据传输到内存。建议使用DMA2D来为CPU减荷。

DMA2D执行四项基本任务：

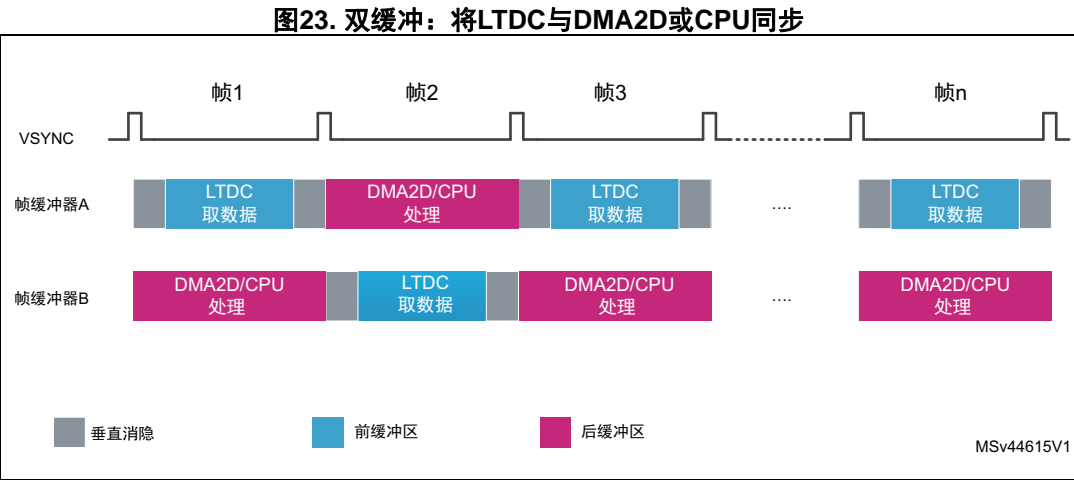
- 填充独特颜色的矩形形状。
- 将一帧或一帧的矩形部分从一个存储器复制到另一个存储器。
- 转换一帧或一帧的矩形部分的像素格式，同时将其从一个存储器传输到另一个存储器。
- 混合两种不同尺寸和像素格式的图像，并将结果图像存储在一个结果存储器中。

4.4.2 LTDC和DMA2D/CPU同步

当仅使用一个帧缓冲器时，会出现帧缓冲器计算显示在屏幕上的风险。通常使用多重缓冲技术（例如双缓冲）来避免在屏幕上显示帧缓冲器计算。

即使使用双缓冲技术，由于LTDC和帧缓冲器更新（利用CPU或DMA2D）之间的非同步，也可能出现撕裂效应。解决此问题的一个方法是使用VSYNC信号来同步这两个主设备（LTDC与CPU或DMA2D）的工作流程。

LTDC从缓冲区（称为前缓冲区）获取图形数据，而DMA2D在另一个缓冲区中准备下一帧（称为后缓冲区）。VSYNC周期指示实际帧显示的结束以及应该翻转的两个缓冲区。



- LTDC提供了同步该工作流程的不同选项：
- 如果使用屏幕最后一行的值来编程行中断，那么中断处理程序会翻转帧缓冲器并开始下一个帧缓冲器计算。
 - 将阴影重载寄存器（LTDC_SRCR）编程为垂直消隐重载，来更改VSYNC周期上的LTDC帧缓冲器地址，并轮询LTDC_CDSR寄存器的VSYNC位来解除对DMA2D的阻塞。

4.5 图形性能优化

如前文所述，帧缓冲器存储器带宽是图形应用最重要的参数。本节提供了基于帧缓冲存储器的带宽优化来优化图形性能的一些建议。

4.5.1 内存分配

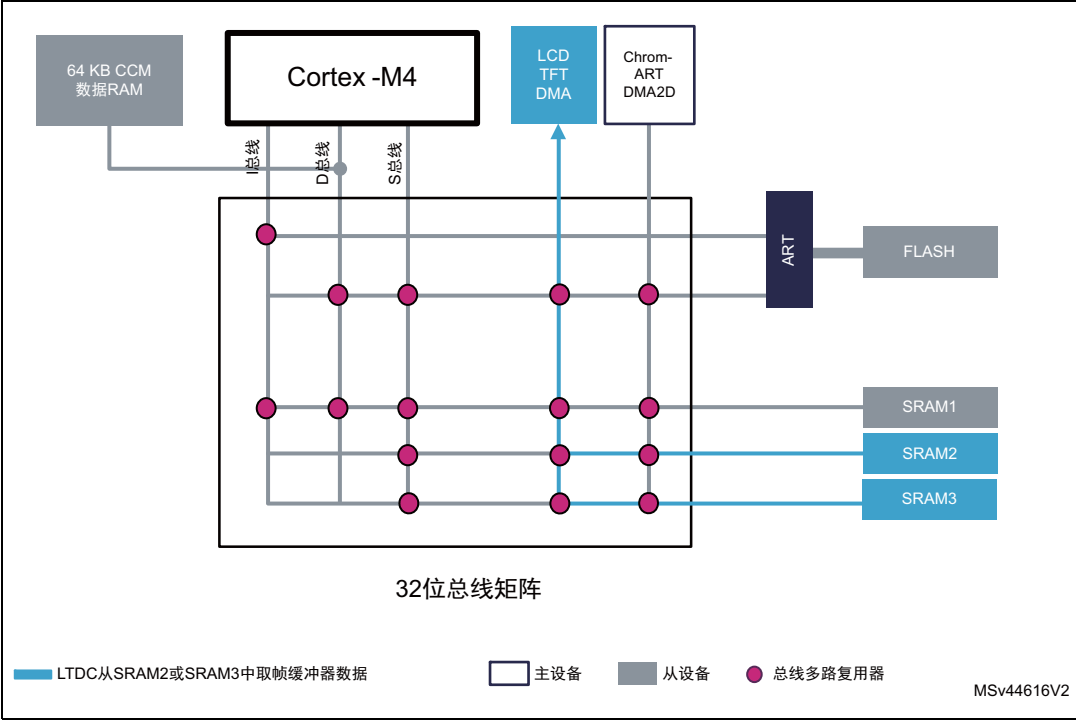
当使用分割为两个或更多从设备的内部SRAM存储器时，STM32 MCU的智能架构能够显著提高系统性能。

在不同主设备同时访问同一个SRAM时，将这些主设备之间的从属存储器分离，可以减少它们之间的竞争。此操作还会产生额外的系统总线带宽。



如 图 24 中所述示例所示，SRAM2和SRAM3专门用于帧缓冲器图形，而SRAM1则用于CPU。

图24. 采用从属存储器分割的示例
在STM32F4x9系列MCU上



4.5.2 优化从外部存储器读取LTDC帧缓冲器的过程（SDRAM或SRAM）

关于SDRAMSRAM的另一个考虑因素是帧缓冲器的位置和行长度数据大小。由于AHB总线矩阵禁止跨越一个千字节边界对存储器进行突发访问，因此当LTDC执行64字节的突发读取时，如果帧缓冲器内容位于一个千字节边界处的地址上，那么就会将该突发读取拆分成单点访问，这会严重影响图形性能。

当一行像素的数据大小不是64字节的倍数时，可能会发生同样的问题。在这些情况下，经过多次访问后，LTDC突发读取会跨越一千字节边界，这会将突发读取拆分为单点访问。

因此，当LTDC不产生突发操作时，每个访问都会被CPU或其他主设备访问（Chrom-Art Accelerator[®]，以太网或其他）中断。该中断极大地降低了高延迟存储器（如导致下溢运行的外部SDRAM）上的LTDC带宽。

要解决上述问题，用户可以选择不会导致上述问题的色深，或者使用以下两种方法之一：

- 减少图层窗口和帧缓冲器行宽。
- 在每行像素的末尾添加若干虚拟字节，来匹配最接近的64字节倍数的帧行宽。

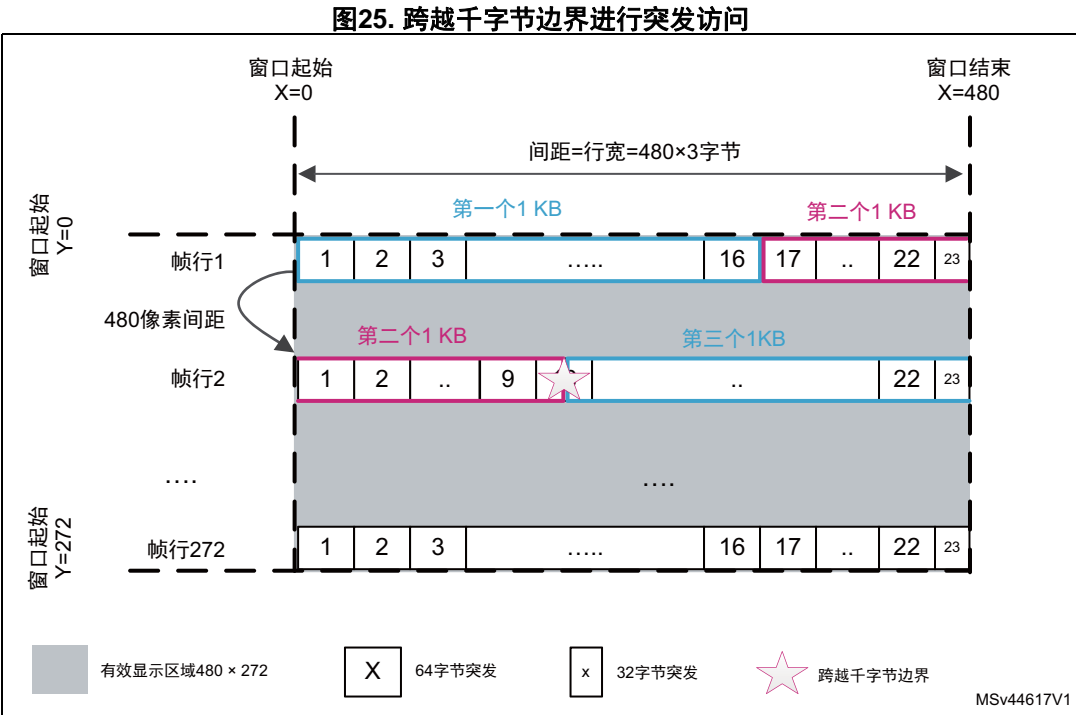
示例：24 bpp的480 x 272显示器

对于480 x 272的显示器（帧行宽为480像素）和24 bpp的色深，行宽等于1440字节，不是64字节的倍数。

注：对于该分辨率，要使行宽大小为64字节的倍数，用户可以使用其他色深，例如RGB565。

由于帧行由22个64个字节的突发和一个32字节的突发组成，所以帧第二行的第10个突发会跨越千字节边界。这会导致读取操作被拆分为单点访问。

图 25说明了给定示例的千字节边界跨越问题。



对于这个例子，我们描述了解决跨越千字节边界问题的两种方法：

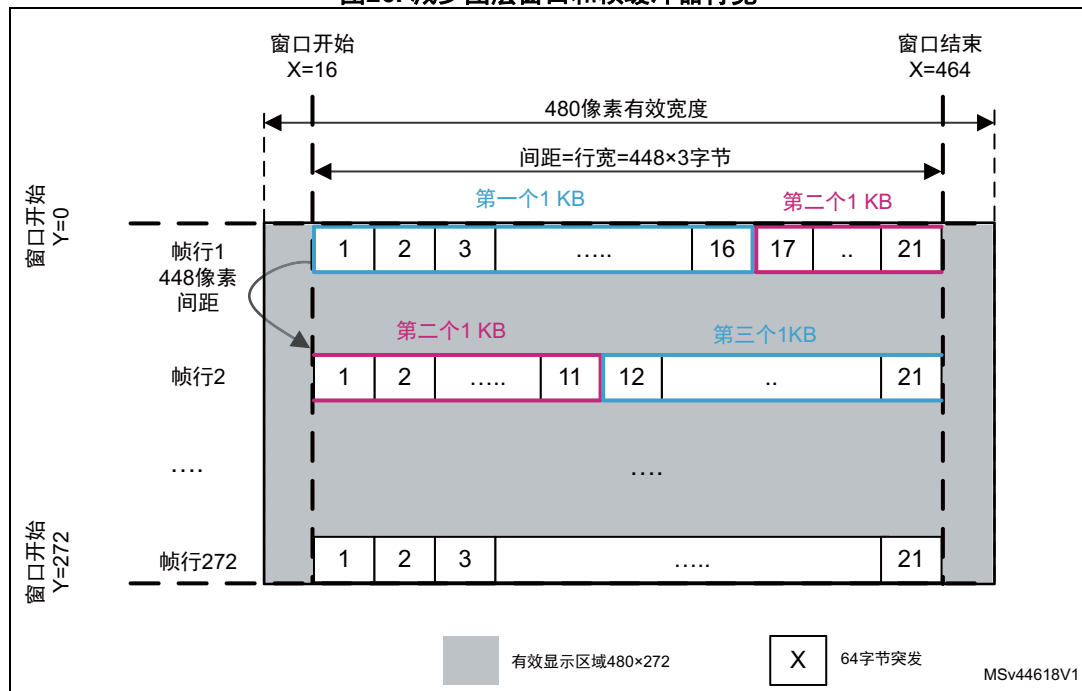
- 减少图层窗口和帧缓冲器行宽：使用LTDC图层窗口功能，通过减小窗口大小来匹配最接近64字节倍数的帧行宽。由于窗口宽度减小，帧缓冲器的大小也会减小，因为所有帧行的额外22和23突发不会被LTDC获取或显示。此方法解决了千字节边界跨越问题，并且窗口宽度稍有减小（参见图 26）。以下代码基于HAL驱动程序，显示了图 26中所述设置间距的示例：

/* 将层1窗口设置为448x272，位置在X = 16且Y = 0 */

```
pLayerCfg.WindowX0 = 16;
pLayerCfg.WindowX1 = 464;
pLayerCfg.WindowY0 = 0;
pLayerCfg.WindowY1 = 272;
```

```
pLayerCfg.PixelFormat = LTDC_PIXEL_FORMAT_RGB888;
pLayerCfg.Alpha = 255;
pLayerCfg.Alpha0 = 0;
pLayerCfg.BlendingFactor1 = LTDC_BLENDING_FACTOR1_PAxCA;
pLayerCfg.BlendingFactor2 = LTDC_BLENDING_FACTOR1_PAxCA;
/* 帧缓冲器起始地址：LTDC直接从内部闪存获取图像，实际图像宽度为448像素。仅显示448个像素
*/
pLayerCfg.FBStartAdress = (uint32_t)&image_data_Image_RGB888_448x272;
pLayerCfg.ImageWidth = 448;
pLayerCfg.ImageHeight = 272;
pLayerCfg.Backcolor.Blue = 0;
pLayerCfg.Backcolor.Green = 0;
pLayerCfg.Backcolor.Red = 0;
if (HAL_LTDC_ConfigLayer(&hltdc, &pLayerCfg, 0) != HAL_OK)
{
    Error_Handler();
}
```

图26. 减少图层窗口和帧缓冲器行宽



- 在每行像素的末尾添加若干虚拟字节，来匹配最接近的64字节倍数的帧行宽。这可以使用LTDC层间距来实现（请参阅第 3.3节：两个可编程LTDC层）。为此，用户必须考虑以下两点：
 - 帧缓冲器应该包含虚拟字节（如图 27中所述）：将数据写入帧缓冲器时，这可以通过将DMA2D的输出偏移量编程为等于最接近的突发倍数与实际行长度数据大小之间的差值来实现。
 - LTDC行长度应始终等于有效数据大小，但LTDC间距应编程为最接近64字节倍数的字节数。

Hal_ltdc驱动程序下提供的HAL_LTDC_SetPitch函数可用于编程所需的间距值（以像素数来计）。对于前面的例子，传递给这个函数的间距值应等于512（512是每行像素数，对应于1536字节的行长度，这是64字节的倍数）。

以下代码基于HAL驱动程序，显示了图 27中所述设置间距的示例：

/* 将层1窗口设置为480x272，位置在X = 0且Y = 0 */

```
pLayerCfg.WindowX0 = 0;
pLayerCfg.WindowX1 = 480;
pLayerCfg.WindowY0 = 0;
pLayerCfg.WindowY1 = 272;
pLayerCfg.PixelFormat = LTDC_PIXEL_FORMAT_RGB888;
pLayerCfg.Alpha = 255;
pLayerCfg.Alpha0 = 0;
pLayerCfg.BlendingFactor1 = LTDC_BLENDING_FACTOR1_PAxCA;
pLayerCfg.BlendingFactor2 = LTDC_BLENDING_FACTOR1_PAxCA;
```

/* 帧缓冲器起始地址：LTDC直接从内部闪存中获取图像，实际图像宽度为480像素，但每行增加了32像素以获得512像素的间距。

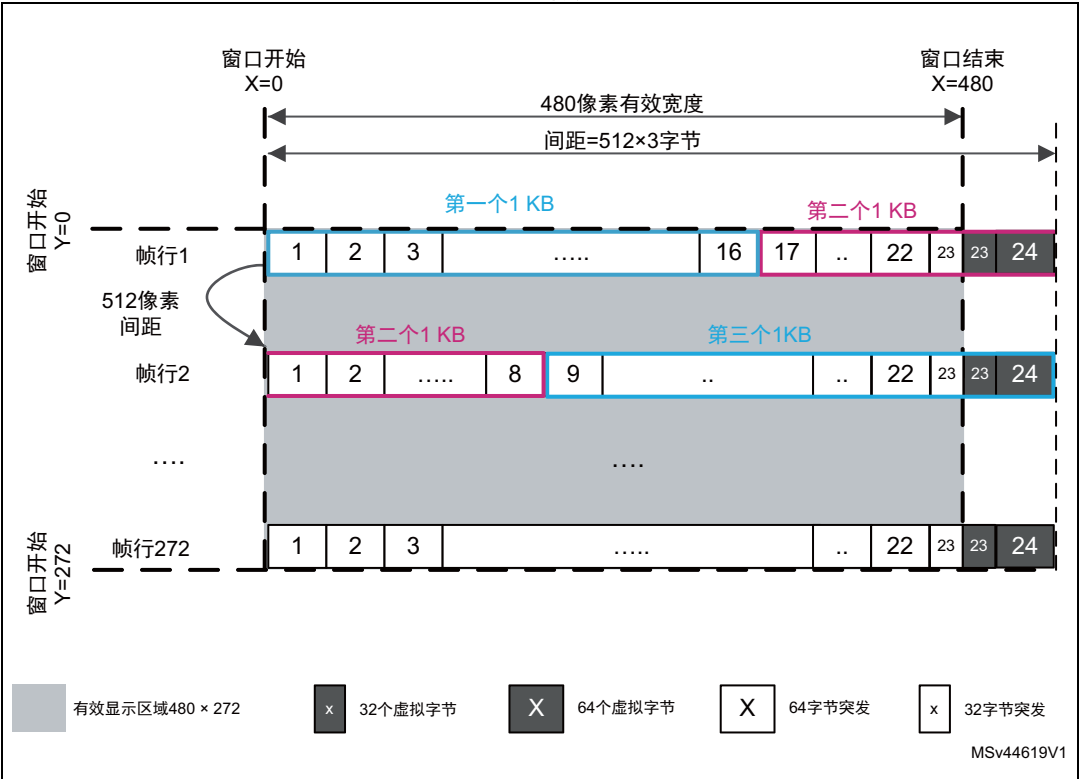
仅显示480个像素 */

```
pLayerCfg.FBStartAddress = (uint32_t)&image_data_Image_RGB888_512x272;
```

```
pLayerCfg.ImageWidth = 480;
pLayerCfg.ImageHeight = 272;
pLayerCfg.Backcolor.Blue = 0;
pLayerCfg.Backcolor.Green = 0;
pLayerCfg.Backcolor.Red = 0;
if (HAL_LTDC_ConfigLayer(&hltdc, &pLayerCfg, 0) != HAL_OK)
{
    Error_Handler();
}
/* 将层1（索引0表示层1）间距设置为512像素 */
HAL_LTDC_SetPitch(&hltdc, 512, 0);
```

图 27描述了解决上述问题后的存储器管理原理。

图27. 添加虚拟字节使行宽为64字节的倍数



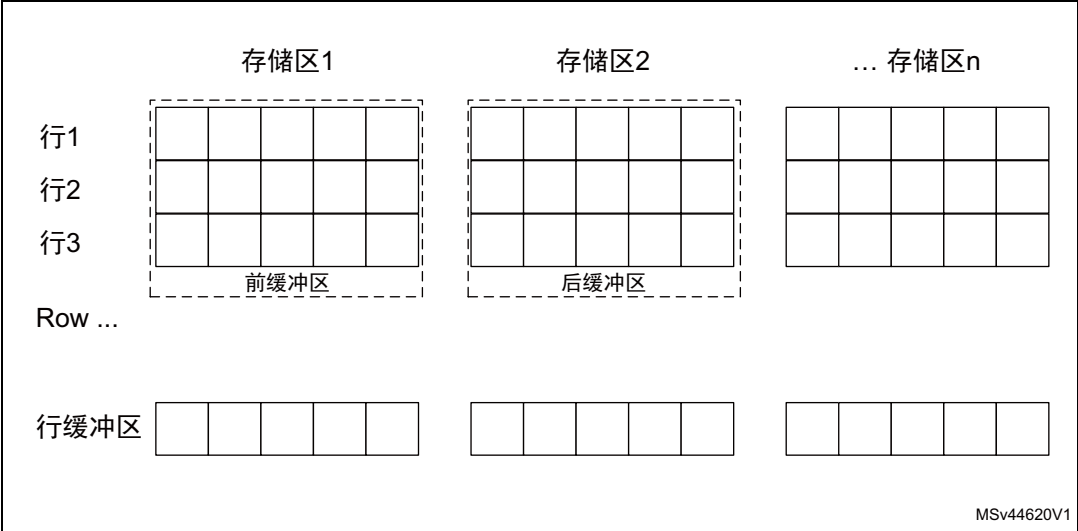
4.5.3 优化从SDRAM读取LTDC帧缓冲器的过程

随机访问存储区会产生一些预充电循环，从而增加见于LTDC的SDRAM延迟。由于LTDC执行顺序访问，因此重要的是，要没有其他主设备访问同一SDRAM存储区。

外部SDRAM由多个存储区组成。鉴于在存储区上进行随机访问会产生一些预充电并激活一些周期，帧缓冲器必须位于仅限LTDC访问的独立存储区中。此操作可减少外部存储器延迟并提高吞吐量。因此，当使用双帧缓冲技术时，建议将这些缓冲区放在两个单独的存储区中。

将前缓冲区存储在SDRAM的第一个地址中，并通过添加一个存储区大小的偏移来寻址后缓冲区，可以实现此操作。请参见图 28。

图28. 将两个缓冲区置于独立的SDRAM存储区中



例如，当SDRAM存储区大小等于4 MB时，可以使用以下行代码：

```
/* 帧缓冲器地址在SDRAM中 */
/* 前缓冲区位于SDRAM内存的存储器1中 */
uint32_t FrontBuffer = LCD_FB_START_ADRESS;
/* 后缓冲区位于SDRAM内存的存储器2中 */
uint32_t BackBuffer = LCD_FB_START_ADRESS + 1024 * 1024 * 4;
```

SDRAM RBURST

另一个可以优化SDRAM读取性能的有趣功能是使用RBURST。

SDRAM控制器添加一个深度为6个32位行的可缓存读取FIFO。读取FIFO在读取突发使能时使用，并允许在CAS延迟期间提前进行下一次读取访问。

4.5.4 在消隐周期中更新帧缓冲器内容

优化图形性能（尤其是在性能瓶颈为帧缓冲存储器带宽时）的一种方法是在消隐期间更新帧缓冲器内容。由于在此期间LTDC不从帧缓冲器获取任何像素数据，所以总线带宽放宽并且可以更新帧缓冲器。

4.6 关于Cortex[®]-M7（STM32F7系列）的特别建议

本节介绍对嵌入了Cortex[®]-M0PU的STM32F7系列的一些建议。这些建议针对Cortex[®]-M7，因为它与Cortex[®]-M4 CPU相比具有以下特点：

- Cortex[®]-M7对正常内存区域进行一些推测性读取访问。
当在SDRAM或Quad-SPI等外部存储器上执行这些推测读取访问时，可能会导致高延迟或系统错误。这会影响访问FMC或Quad-SPI的AHB主设备（如LTDC），尤其是会降低图形性能，并可能导致系统错误（如果LTDC帧缓冲器位于外部存储器中和/或Quad-SPI存储器用于图形显示时）。

- Cortex[®]-M7 CPU嵌入一个L1缓存（参见图 10）。
由于缓存设置不合适可能会引起一些图形问题；如果缓存维护没有正确执行，可能会出现不良的图形视觉效果。如果没有使用合适的缓存维护方式，则图形性能可能会受影响。

4.6.1 如果不使用，就禁用FMC bank1

复位后，FMC bank1始终处于使能状态，可以引导进入外部存储器。Cortex[®]-M7在进行一些推测时，它会对第一个FMC存储区进行推测性读取访问。

默认的FMC配置非常缓慢，这种推测性访问会长时间阻止其他AHB主设备访问FMC，导致LTDC端运行下溢。

为了防止FMC bank1上的CPU推测性读取访问，建议在不使用时将其禁用。这可以通过复位FMC_BCR1寄存器中的MBKEN位来完成，默认情况下该位在复位后使能。

要禁用FMC Bank1，用户可以使用以下代码：

```
/* 禁用FMC Bank1：复位后 FMC_BCR1 = 0x000030DB  
这里MBKEN = 1b意味着FMC_Bank1使能  
而MTYP[1:0]= 10意味着存储器类型设置为NOR Flash/OneNAND Flash*/  
FMC_Bank1->BTCR[0] = 0x000030D2;
```

有关FMC配置的更多详细信息，请参考相关的STM32参考手册。

4.6.2 配置存储器保护单元（MPU）

本节定义STM32F7系列的系统存储器属性和基本MPU概念。还介绍了如何配置MPU以避免由Cortex[®]-M7推测性读取访问和缓存维护引起的图形性能问题。

注： 本节仅介绍配置所需的一些必要的基本MPU概念。有关MPU和缓存的更多详细信息，请参阅以下文档：

- AN4838应用笔记“管理STM32 MCU中的存储器保护单元（MPU）”
- AN4839应用笔记“STM32F7系列产品上的一级缓存”
- STM32F7系列产品Cortex[®]-M7处理器编程手册（PM0253）
- ARM Cortex[®]-M7技术参考手册。

MPU属性配置

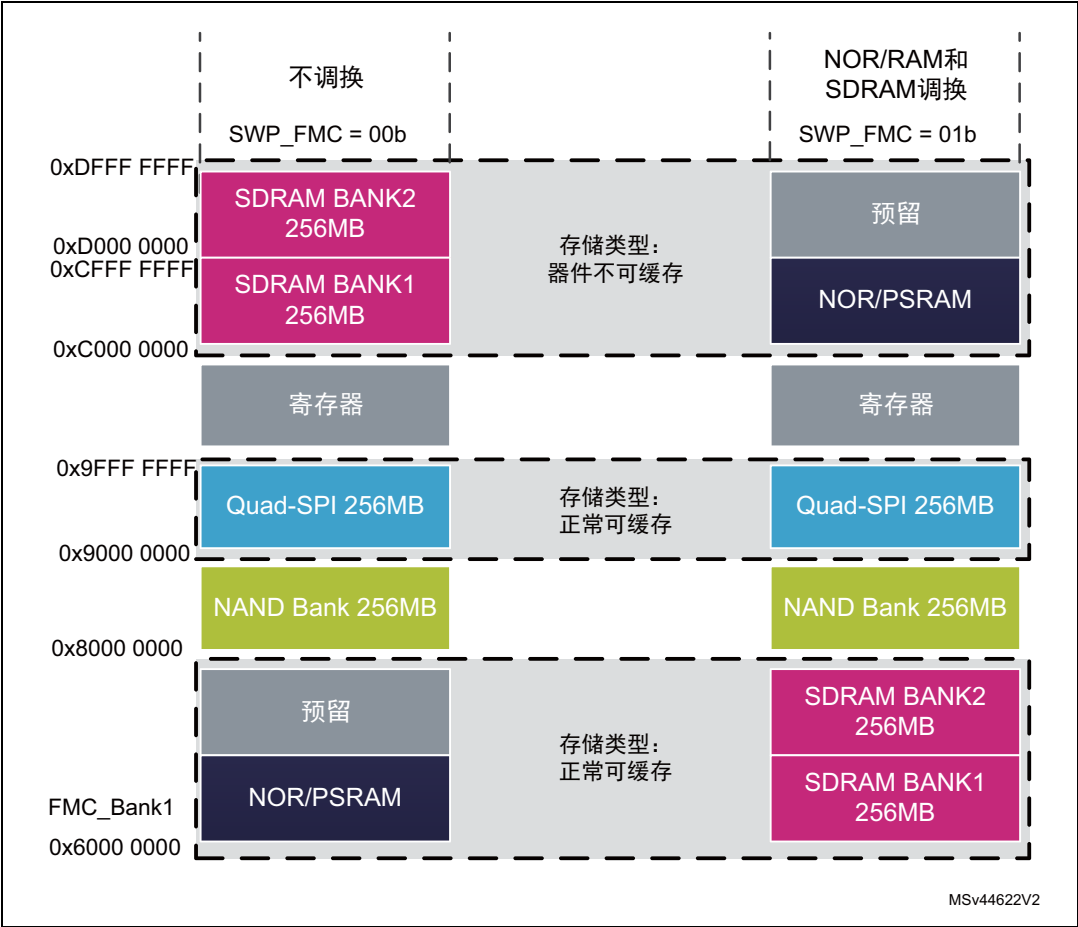
为了防止由Cortex®-M7推测性读取访问引起的图形性能问题，用户应检查应用的所有存储器映射并根据硬件来配置MPU。因此，用户必须设置以下配置：

- 定义帧缓冲器MPU区域和其他应用MPU区域。
- MPU必须根据应用使用的内存大小进行配置。
- 未使用区域的MPU属性必须配置为强排序永不执行（XN）。例如，对于Quad-SPI，如果连接了8 MB的存储器，则剩余的248 MB未使用空间（总共256 MB可寻址空间）必须设置为强排序XN。见[第 6.2.7节](#)中示例。
- 为防止Cortex-M7推测性读取访问外部SDRAM / SRAM（如果启用FMC交换，参见[图 29](#)），必须将SDRAM / SRAM MPU区域设置为永不执行（XN）。
- 如果Cortex®-M7CPU用于帧缓冲器处理（写入SDRAM/SRAM），则应将帧缓冲器MPU属性设置为正常缓存，并具有读取和写入访问权限。

注： *帧缓冲器MPU区域属性应设置为永不执行，因为它仅用于图形内容创建。*

[图 29](#)描述了默认系统存储器映射下STM32F7的FMC存储区和Quad-SPI MPU存储器属性

图29. 默认系统存储器映射（MPU禁用）情况下，
FMC SDRAM和NOR / PSRAM存储器交换



MPU和高速缓存策略配置

使用Cortex®-M7高速缓存可以提高系统和图形性能。尤其是当CPU访问SDRAM或Quad-SPI等外部存储器时，会发现此性能增益。

在图形应用中，当CPU用于帧缓冲器处理时，建议使用缓存，特别是如果帧缓冲器位于外部存储器（如SDRAM或SRAM）中时。这种情况下，用户在使用缓存时应考虑以下几点：

- MPU内存区域可缓存性
如之前在图 29 中所示，在默认系统存储区域MPU属性下，一些系统存储区域是正常可高速缓存的，而其他系统存储区域是不可高速缓存的。
当CPU用于帧缓冲器处理时，用户应该将帧缓冲器MPU属性更改为正常缓存（或者执行FMC交换，见图 29）。

- 缓存维护和数据一致性：无缓存维护操作时的WBWA视觉影响
使用启用了L1缓存的Cortex[®]-M7 CPU和WBWA缓存策略执行帧缓冲器处理时，经常遇到数据一致性问题。当多个主控制器（如Cortex[®]-M7和LTDC）共享相同的区域（帧缓冲器）并且未执行缓存维护时，会发生此问题。
当CPU处理帧缓冲器（写入帧缓冲器）并且帧缓冲器具有回写缓存策略时，处理结果（待显示的图像）不会出现在帧缓冲器（可能是SRAM或SDRAM）中，因此不能显示。
要避免此问题，用户应使用以下方法：
 - 将帧缓冲器缓存属性配置为透写（WT），在这种情况下，每次写入操作都在高速缓存和帧缓冲器上执行。
 - 将帧缓冲器缓存属性配置为回写写分配（WBWA），并通过软件执行缓存维护。
透写对数据一致性更安全，但可能会影响图形性能。
- 缓存维护可能会影响图形性能
用户应该使用适合其应用的缓存策略。每种方法都有其优缺点。所以对于每种方法，用户都应该考虑其以下特点：
 - 透写：管理起来非常简单（不需要通过软件执行缓存维护），并且对数据一致性更安全，但它会产生向帧缓冲器的大量单次写入操作，从而影响LTDC访问。
 - 回写写分配：它更适合使用WBWA和软件例程，并使缓存维护操作与LTDC在消隐期间相同步。这可以在帧缓冲存储区（SRAM或SDRAM）上产生额外的带宽。缓存维护操作应在数据写入帧缓冲存储区域之后，通过软件来执行；这可以通过使用CMSIS函数SCB_CleanDCache()强制执行D-高速缓存清理操作来完成。因此缓存中的所有被污染的行都会写回到帧缓冲器。

注： 用户还应该考虑到，即使不使用CPU进行帧缓冲器处理，缓存维护也可能会影响图形性能。因此，在某些应用中，CPU访问外部SDRAM或SRAM，是为了启用缓存的图形以外的其他用途。这种情况下，缓存维护可能会影响LTDC访问。

MPU 配置示例

第 6.2.7 节中介绍了MPU配置的一个示例，其中介绍了如何在使用CPU（启用缓存）时设置帧缓冲MPU属性以进行图形操作。所描述的例子是针对STM32F746G-DISCO板硬件配置而创建的，其中外部SDRAM用于帧缓冲器，而外部QSPI闪存则包含有图形基元。

4.7 LTDC外设配置

本节介绍配置LTDC外设所需的步骤。

注： 建议在开始配置之前重置LTDC外设，并且建议确保外设处于复位状态。LTDC可通过设置RCC_APB2RSTR寄存器中的相应位来复位，这可以复位三个时钟域。

4.7.1 显示面板连接

LTDC硬件接口为每条颜色总线8位，非常适合用于RGB888真彩色面板。LTDC硬件接口还提供了时序信号：LCD_HSYNC，LCD_VSYNC，LCD_DE和LCD_CLK。

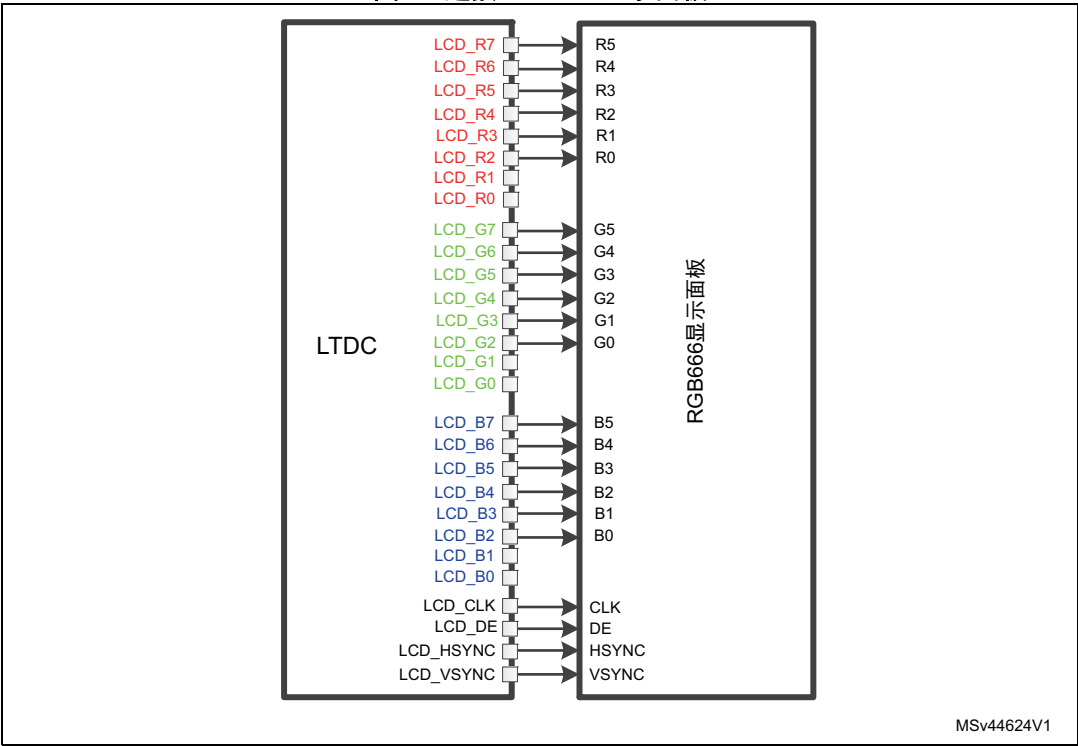
LTDC GPIO应配置为相应备用功能。关于LTDC备用功能可用性与GPIO的关系的更多详细信息，请参阅相关数据表中的备用功能映射表。

注： 所有GPIO都必须配置为高速模式。

连接弱调色板显示面板

对于具有弱调色板（如RGB666和RGB565）的显示面板，应使用数据信号的最高有效位来实现总线连接。图 30显示了连接RGB666显示面板的示例。

图30. 连接RGB666显示面板



MSv44624V1

使用STM32CubeMX工具进行GPIO配置

要将显示面板连接到STM32 MCU，用户应对GPIO进行配置，使之可用于连接。

使用STM32CubeMX工具是配置LTDC外设及其GPIO的一种非常简单方便且快速的方式，因为它可以使用预配置的LTDC来生成项目。

章节 [第 6.2.3节：LTDC GPIO配置](#) 提供了有关如何配置LTDC GPIO的指南。

显示模块特定引脚的配置

某些显示模块可能需要其他信号才能完全正常工作。用户可以使用GPIO和一些外设来控制这些信号。

使用GPIO来控制显示面板上 *显示器使能* 引脚（LCD_DISP）的示例如章节 [第 6.2.3节：LTDC GPIO配置](#) 中所述。

启用LTDC中断

为了能够使用LTDC中断，用户应该在NVIC端启用LTDC全局中断。然后，通过使能相应的使能位，将每个中断分别使能。LTDC中断使能位可在LTDC_IER寄存器中找到，如 [表 7：LTDC中断总结](#) 中所述。

注： FIFO 下溢运行和传输错误中断可在在 `hal_ltdc` 驱动程序 `HAL_LTDC_Init()` 函数中使能
利用STM32CubeMX使能LTDC中断的示例如 [第 6.2.3节：LTDC GPIO配置](#) 中所述。

4.7.2 LTDC时钟和时序配置

本节介绍配置LTDC时钟和时序并使其符合显示器规格所需的步骤。它还为嵌入在STM32F746G-DISCO板上的ROCKTECH（RK043FN48H）显示器提供了配置示例。

系统时钟配置

建议使用最高的系统时钟，以获得最佳的图形性能。此建议也适用于外部存储器帧缓冲器。因此，如果外部存储器用于帧缓冲器，则应使用所允许的最高时钟速度来获得最佳存储器带宽。

例如，对于STM32F4x9微控制器，最高系统速度为180 MHz，因此如果外部SDRAM连接到FMC，则最大SDRAM时钟为90 MHz（HCLK/2）。

对于STM32F7系列产品，最高系统速度为216MHz，但是使用此速度和HCLK/2预分频器时，SDRAM速度超过了最大允许速度（更多详细信息，请参见产品数据手册）。因此，要获得最大的SDRAM，建议将HCLK配置为200 MHz，然后将SDRAM速度设置为100 MHz。

具有最高性能的时钟配置是：

- STM32F4x9器件：HCLK @ 180 MHz 并且 SDRAM @ 90 MHz。
- STM32F7系列：HCLK @ 200 MHz 并且 SDRAM @ 100 MHz。

利用STM32CubeMX配置LTDC的示例如 [第 6.2.4 节：LTDC外设配置](#) 中所述。

像素时钟和时序配置

在图形应用开发的这个阶段，用户应该已经检查并确认了所需的显示器尺寸和色深与硬件配置相兼容。因此，要配置的像素时钟应该是已知的，或者从显示器数据表中提取或计算（参见 [第 4.2.2 节：考虑存储器时检查显示兼容性带宽要求](#)）。

示例：在STM32F746G-DISCO板上嵌入了ROCKTECH（RK043FN48H）显示器的LTDC时序配置

首先，用户应该从显示器的数据表中提取时序参数（参见 [表 14](#)）。建议使用典型的显示时序。

表14. 从ROCKTECH RK043FN48H数据表中提取LCD-TFT时序 ⁽¹⁾

项目		符号	最小值	典型值	最大值	单位
DCLK频率		Fclk	5	9	12	MHz
DCLK周期		Tclk	83	110	200	ns
Hsync	周期时间	TH	490	531	605	DCLK
	显示周期	Thdisp	-	480	-	DCLK
	后沿	Thbp	8	43	-	DCLK
	前沿	Thfp	2	8	-	DCLK
	脉冲宽度	Thw	1	-	-	DCLK
Vsync	周期时间	Tv	275	288	335	H
	显示周期	Tvdisp	-	272	-	H
	后沿	Tvbp	2	12	-	H
	前沿	Tvfp	1	4	-	H
	脉冲宽度	Tvw	1	10	-	H

1. 灰色单元格突出显示了以下示例中使用的值。

基于表 14，提取时序参数为：

- 显示周期（有效宽度）= 480像素
- 后沿 HBP = 43像素
- 前沿 HBP = 8像素
- 脉冲宽度 HSYNC = 1像素（最小值）
- 显示周期（有效高度）= 272像素
- 垂直后沿 VBP = 12像素
- 前沿 VFP = 4像素
- 脉冲宽度 VSYNC = 10像素

程序时序参数：一旦时序参数被提取，它们就被用来对LTDC时序寄存器进行编程，表 15中汇总了所有要编程的参数。

使用STM32CubeMX进行时序参数配置：使用STM32CubeMX编程时序参数非常容易，用户只需简单地将提取的参数填充到LTDC配置窗口中即可（参考章节第 6.2.4节：LTDC外设置）。

表15. 编程LTDC时序寄存器

寄存器			要编程的值
LTDC_SSCR	HSW[11:0]	HSYNC宽度 - 1	0
	VSH[11:0]	VSYNC高度 - 1	9
LTDC_BPCR	AHBP[11:0]	HSYNC宽度 + HBP - 1	43
	AVBP[10:0]	VSYNC高度 + VBP - 1	21
LTDC_AWCR	AAW[11:0]	HSYNC宽度 + HBP + 有效宽度 - 1	523
	AAH[10:0]	VSYNC高度 + BVBP + 有效高度 - 1	293
LTDC_TWCR	TOTALW[11:0]	HSYNC宽度 + HBP + 有效宽度 + HFP - 1	531
	TOTALH[10:0]	VSYNC高度 + BVBP + 有效高度 + VFP - 1	297

利用STM32CubeMX进行像素时钟配置：像素时钟以60 Hz刷新率计算，如下所示：

$$LCD_CLK = TOTALW \times TOTALH \times \text{刷新率}$$

基于表 15，TOTALW = 531 并且 TOTALH = 297。

对于本例：

$$LCD_CLK = 531 \times 297 \times 60 = 9.5 \text{ MHz}$$

参考第 6.2.4节中的LTDC像素时钟配置STM32CubeMX示例。

LTDC控制信号极性配置

LTDC控制信号（HSYNC、VSYNC、DE和LCD_CLK）极性必须根据显示器规格进行配置。用户应注意，只有DE控制信号应该与显示数据表中指示的DE极性相反。其他控制信号的配置应与显示数据表完全相同。

4.7.3 LTDC层配置

本节介绍配置LTDC层相关显示器尺寸和色深所需的步骤。

如先前 [第 3.3.2 节](#) 中所述，LTDC具有两个可独立配置的层，其中用户可以启用一层或两层。默认情况下，这两个层都被禁用，因此只显示配置的背景色（默认颜色为黑色）。

用户可以显示层1 + 背景或显示层1 + 层2 + 背景。

仅显示背景

如果未启用图层，则只显示背景。如果未配置背景色，则显示默认的背景黑色（LTDC_BCCR = 0x00000000）。

要设置蓝色背景色，LTDC_BCCR寄存器应设置为0x000000FF。

图层参数配置

当LTDC GPIO、时钟和时序正确设置时，用户应配置以下LTDC层参数。每个LTDC层都有自己的参数，因此应该单独配置。

- 窗口大小和位置
- 像素输入格式
- 帧缓冲器起始地址
- 帧缓冲器大小（图像宽度和图像高度）和间隔
- ARGB8888格式下的图层默认颜色
- 用于混合的图层常量alpha
- 图层混合factor1和factor2

利用STM32CubeMX配置LTDC层参数的示例如 [第节：LTDC图层参数配置第 75页](#)中所述。

注：除CLUT外，所有层参数均可实时修改。新配置必须通过配置LTDC_SRCR寄存器立即重载或在垂直消隐周期内重载。

4.7.4 显示面板配置

某些显示器需要使用串行通信接口（如I2C或SPI）进行配置。

例如，STM32F429I-DISCO嵌入了ILI9341显示模块，它通过SPI接口来初始化。

此显示模块（ili9341.c）的专用驱动程序（包括初始化和配置命令）可在STM32Cube固件包中找到，在以下目录下：

STM32Cube_FW_F4_Vx.xx.x\Drivers\BSP\Components\ili9341

STM32F429I探索板的STM32Cube示例中包含了一个显示初始化序列的示例，该示例基于ili9341_Init()函数

STM32Cube_FW_F4_Vx.xx.x\Projects\STM32F429I-Discovery\
Examples\LTDC\LTDC_Display_2Layers

4.8 存储图形基元

图形基元是基本元素（如图像或字体），组合起来就构成了所显示的帧缓冲器内容。

静态数据应放置在非易失性存储器中。当要存储的数据量相对较少时，可以使用内部FLASH存储器。否则，应将图形内容放置在外部存储器中。

STM32微控制器为外部NOR闪存提供了并行（FMC）或串行（QSPI）接口（见表 3）。

为了构建帧缓冲器内容，DMA2D可以直接从并行NOR闪存或Quad-SPI闪存读取图形基元。

有关在QSPI存储器上存储图形内容的更多详细信息，请参考应用笔记STM32微控制器上的Quad-SPI（QSPI）接口（AN4760）。

4.8.1 将图像转换为C文件

要将图形基元添加到用户项目中，应将它们转换为C或头文件。要做到这一点，用户可以使用一些特定的工具来生成C或*.h文件。

警告： 考虑到配置的像素输入格式，用户必须将图像转换为C文件，如第节：像素输入格式第 27页中所述。某些工具可能会产生红色和蓝色调换的C或*.h文件。为了避免这个问题（显示红蓝调换的图像），用户可以使用LCD图像转换工具。

LCD图像转换器是一个可定制的自由工具，可以将图像转换为C文件，并能够以所需格式生成C文件。第 6.2.5节：显示来自内部闪存的图像 给出了一个示例。

4.9 硬件注意事项

本节提供了图形应用的硬件注意事项。一般应小心设计两个重要的硬件接口：LTDC接口和外部存储器接口（用于帧缓冲器），例如FMC_SDRAM或FMC_SRAM。

LTDC 并行接口

当像素时钟低于40 MHz（SVGA）时，可以使用简单的3.3 V信号。

如果负载和导线长度减小（例如，在同一块PCB上连接LVDS变压器或HDMI收发器），则并行RGB可以达到83 MHz。

建议将LTDC GPIO配置为最大工作速度OSPEEDRy[1:0]= 11b。有关GPIOx_SPEEDR GPIO端口输出速度寄存器的说明，请参见相关STM32参考手册。

FMC SDRAM/SRAM接口

当使用外部SRAM或SDRAM内存作为帧缓冲器时，FMC-SDRAM和FMC-SRAM接口速度取决于诸多因素，包括电路板布局和焊盘速度。良好的PCB设计下可以达到表 10和表 11所述的最大像素时钟。

为了获得最佳性能，布线应尽可能良好。要获得有关PCB布线指南的更多信息，请参考以下应用笔记（可在意法半导体网站上获取）：

- 应用笔记*STM32F7系列MCU硬件开发入门*（AN4661）的章节“灵活内存控制器（FMC）接口”
- 应用笔记*STM32F4xxxxMCU硬件开发入门*（AN4488）的章节“灵活内存控制器（FMC）接口”

5 节省能耗

当应用处于空闲状态并且仅显示屏幕保护程序时，将STM32芯片驱动为睡眠模式以降低功耗非常重要。在睡眠模式下，所有外设（例如FMC-SDRAM和LTDC）都可以使能，而CPU停用。

有需要时外部存储器（如SDRAM或QSPI FLASH）可以驱动为低功耗模式，以免费电。

如果应用处于低功耗状态，但需要显示图形，则LTDC可以保持活动状态，SDRAM可以进入自刷新模式（以节省功耗）。如果应用设置为掉电模式，则可节省更多电能。

如果在运行应用时不需要显示，也可以禁用显示器或将其置于低功耗模式。

6 LTDC应用示例

本节提供了：

- 考虑了资源需求时的一些图形实现示例
- 关于如何创建基本图形应用的示例
- 嵌入LTDC并具有板载LCD-TFT面板的STM32参考板总结

6.1 实现示例和资源要求

本节提供了一些详细说明了硬件资源要求的图形实现示例。

6.1.1 单片MCU

得益于其集成的SRAM，STM32 MCU可用于图形应用，无需使用外部SDRAM/SRAM存储器作为帧缓冲器。此外，由于其内部闪存空间较大（高达2 MB），用户可以使用它们来存储图形基元。使用内部存储器可以减少所用引脚，简化PCB设计并节省成本。

要将单芯片MCU用于图形应用，用户可以使用以下硬件配置：

- 内部闪存最大2 MB：存储用户应用程序代码和图形基元。
- 帧缓冲器应位于内部SRAM中，因此，根据每个STM32 MCU的内部SRAM大小，用户可以连接相应的显示尺寸和色深，如下所示：
 - STM32F7x7系列：使用SRAM1（368 KB）以支持分辨率400 x 400 16 bpp（313 KB）或480 x 272 16 bpp（255 KB）。
 - STM32F7x6系列：使用SRAM1（240 KB）以支持320x320分辨率，且具有16 bpp（200 KB）。
 - STM32F469/F479系列：使用SRAM1（160 KB）以支持320 x 240分辨率，且具有16 bpp（154 KB）。
 - STM32F429/F439系列：使用SRAM1（112 KB）以支持320 x 240分辨率，且具有8 bpp（75 KB）。
 - STM32 MCU封装：LQFP100或TFBGA100


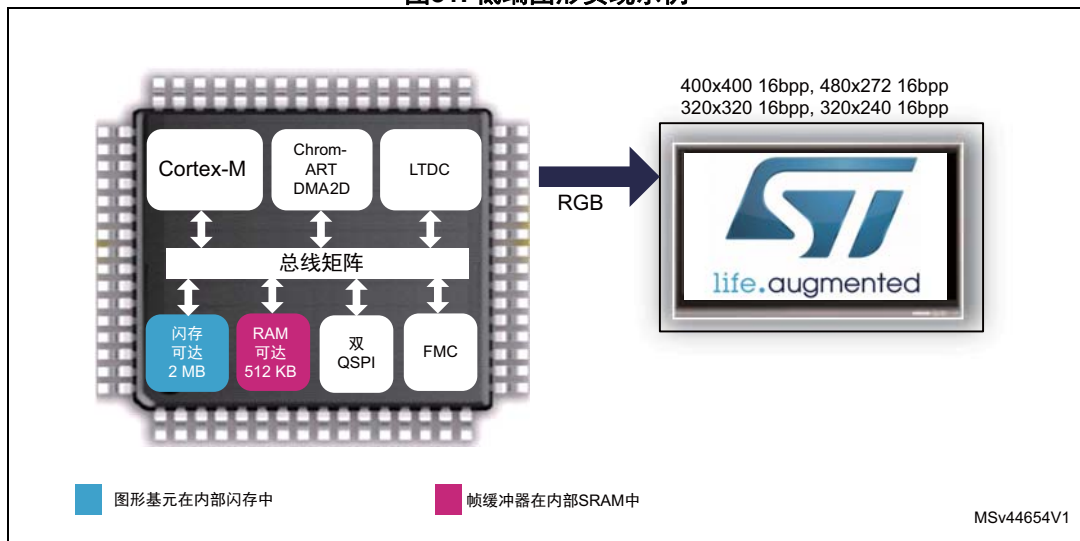
 31说明了一个图形实现示例，其中仅使用了单个芯片，没有外部存储器。

图31. 低端图形实现示例



6.1.2 带外部存储器的MCU

为了与更高分辨率的显示器连接，需要一个连接到FMC的外部存储器用于帧缓冲器。外部QSPI闪存可用于存储图形基元。

对于中端或高端图形应用，用户可以使用以下硬件配置示例：

- 具有256 MB可寻址内存映射的外部QSPI闪存可用来存储图形基元。
- 用于帧缓冲器的外部SDRAM 32位内存。
- STM32 MCU封装：UFBGA169, UFBGA176, LQFP176, LQFP208, TFBGA216, WLCSP168和WLCSP180。

[图 32](#)说明了一个图形实现示例，其中两个外部存储器连接到STM32 MCU，一个用于帧缓冲器，另一个用于图形基元。

图32. 高端图形实现示例

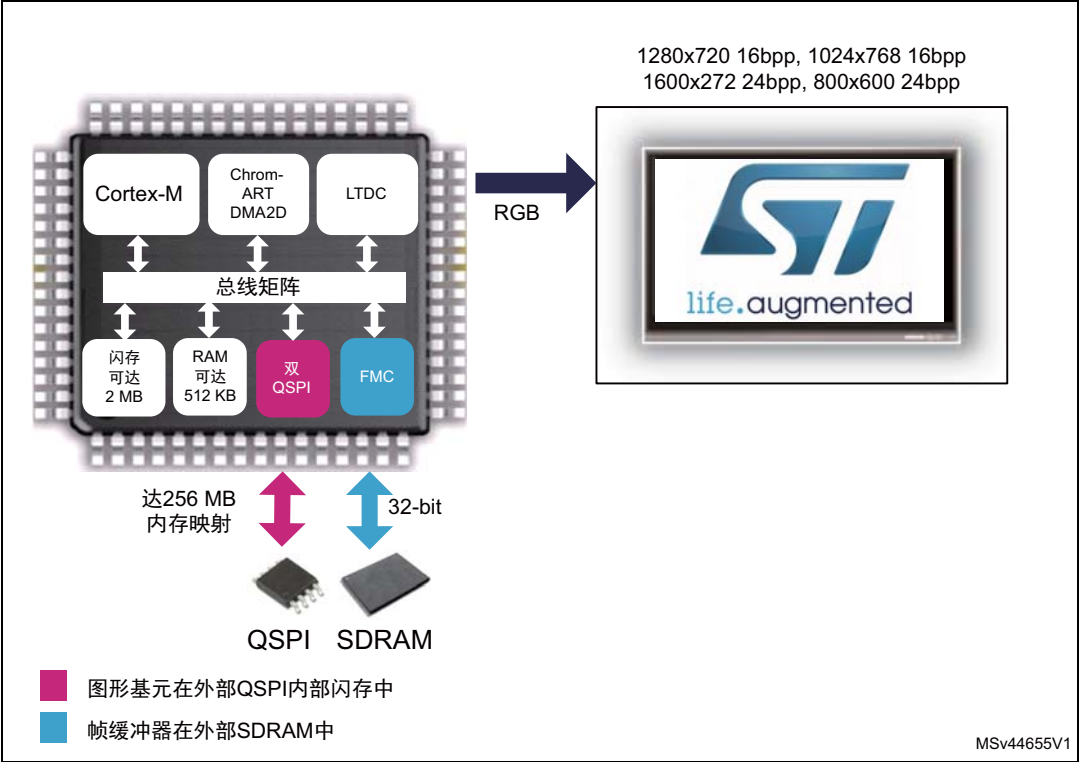


表 16总结了不同STM32硬件配置下的图形实现示例。

表16. 使用不同硬件配置下STM32的图形移植示例

型号	显示器尺寸	色深	外部存储器 - SDRAM	显示器界面	STM32封装 ⁽¹⁾
高端	1280 x 720	16 bpp	32 位	RGB888	UFBGA176 TFBGA216/UFBGA169/ LQFP176/LQFP208 WLCSP180/WLCSP168
	1024 x 768				
	1600 x 272	24 bpp			
	800 x 600				
中端	800 x 600	16 bpp	16 位	RGB666	LQFP144/WLCSP143
	800 x 480	24 bpp			
	640 x 480				
	400 x 400 ⁽²⁾				
低端	400 x 400 ⁽²⁾	16 bpp	无	RGB666	LQFP100/TFBGA100
	480 x 272				
	320 x 320 ⁽²⁾				
	320 x 240				

1. 表 13中总结了嵌入LTDC的STM32 MCU的封装可用性。
2. 400x400和320 x 320是通常用于智能手表的特定显示分辨率。

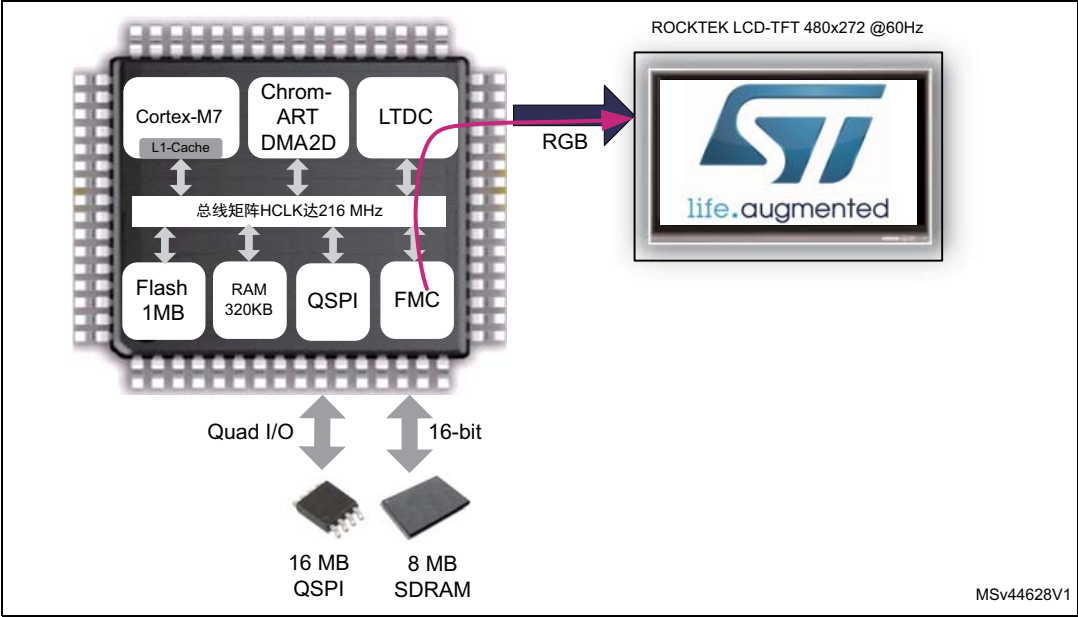
6.2 示例：创建基本图形应用

本节提供了一个基于STM32F746G-DISCO板的示例，其中介绍了创建基本图形应用所需的步骤。

6.2.1 硬件说明

本例中使用STM32F746G-DISCO板上嵌入的硬件资源。[图 33](#)描述了要使用的图形硬件资源：

图33. STM32F746G-DISCO中的图形硬件配置



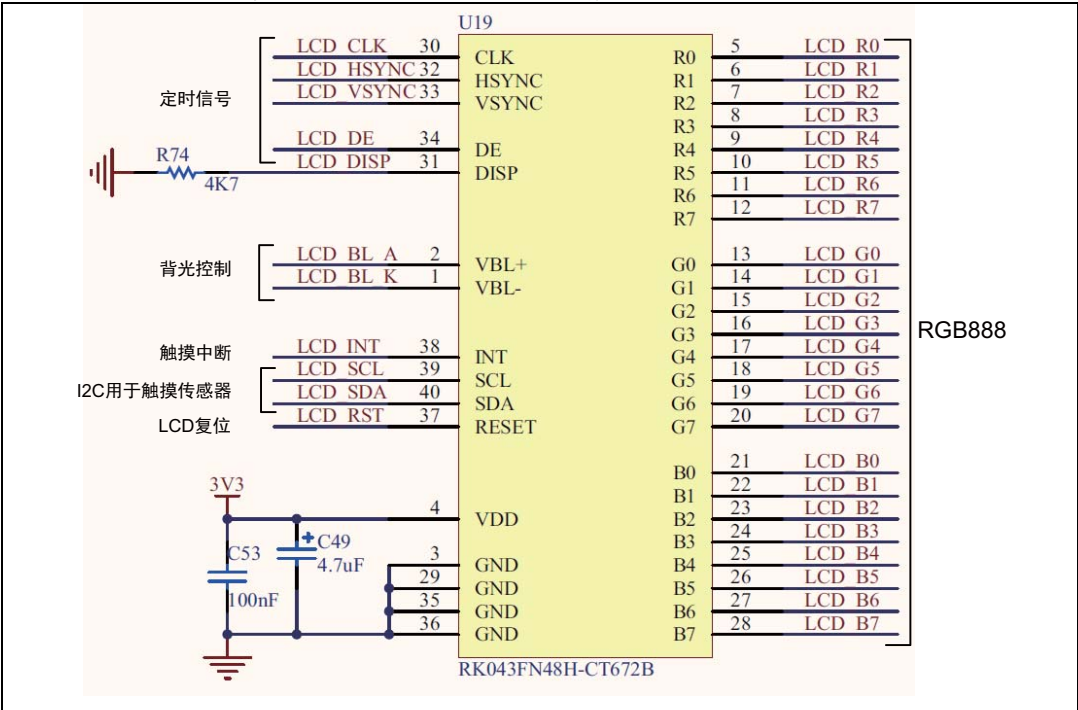
1. 粉色箭头表示显示器的像素数据路径。

STM32F746G-DISCO板嵌入了480×272分辨率的并行真彩色RGB888 LCD-TFT面板。

有关STM32F746G-DISCO板的更多详细信息，请参考意法半导体网站上的用户手册 *带STM32F746NG MCU的STM32F7系列产品探索套件* (UM1907)。

[图 34](#)显示了连接到STM32F746 MCU的ROCKTECH RK043FN48H真彩色面板 (RGB888)。

图34. STM32F746G-DISCO板上的LCD-TFT连接



如 图 34 中所示，显示器模块通过两种不同的引脚策略连接到MCU：

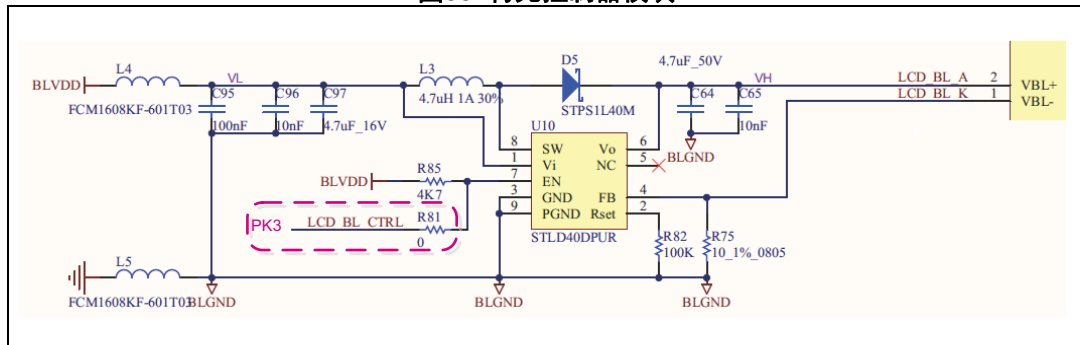
- LTDC接口引脚：
 - 24位RGB接口。
 - 时序信号：LCD_HSYNC，LCD_VSYNC，LCD_DE和LCD_CLK。
- 其他特定引脚：
 - LCD_DISP用来启用/禁用显示待机模式。
 - INT中断行：允许触摸传感器产生中断。
 - I2C接口用来控制触摸传感器。
 - LCD_RST复位引脚可以复位LCD-TFT，它连接到全局MCU复位引脚（NRST）。
 - 用于LED背光控制的LCD_BL_A和LCD_BL_K引脚：背光由STLD40DPUR电路控制。

背光控制器： 图 35 中所述的STLD40DPUR电路是一款升压转换器，工作电压范围为3.0 V至5.5 V。它可提供高达37 V的输出电压，并可驱动多达10个串联的白光LED。有关背光控制器的更多信息，请参考STLD40D数据表。

LCD_BL_CTRL（PK3）信号上的高电平可以点亮背光，而低电平则将其关闭。

注： 通过向STLD40D电路的EN引脚7施加一个低频（1至10 kHz）PWM信号，可以改变显示亮度（调暗背光强度）。由于PK3引脚上没有定时器PWM输出备用功能，所以用户需要去掉R81电阻，并将另一个GPIO引脚连接到PWM输出备用功能，因此需要重新设计。

图35. 背光控制器模块



6.2.2 如何检查特定显示器尺寸是否匹配硬件配置

本节假定在此阶段用户需要480 x 272 @ 60Hz的显示尺寸以及24 bpp的色深，下一步是要选择正确的硬件配置。

所需显示面板

所需显示器为ROCKTECH RK043FN48H-CT672B显示器：

- 显示器分辨率：480 x 272像素，带LED背光和电容式触控面板。
- 显示器接口：24位RGB888（共计28个信号）。

确定帧缓冲器的大小和位置

根据其大小和内部可用的SRAM大小，帧缓冲器可以位于内部SRAM或外部SDRAM中。STM32F746NGH6 MCU的嵌入SRAM总大小为320 KB，这里可以使用SRAM1（240KB）（参见图 10：STM32F7x6、STM32F7x7、STM32F7x8和STM32F7x9中的LTDC AHB主设备智能架构）。

帧缓冲器大小按以下方式计算：

- 对于24 bpp
 - 帧缓冲器（KB）= $480 \times 272 \times 3 / 1024 = 382.5$
- 对于16 bpp
 - 帧缓冲器（KB）= $480 \times 272 \times 2 / 1024 = 255$
- 对于8 bpp：
 - 帧缓冲器（KB）= $480 \times 272 / 1024 = 128$

因此，根据这些结果，8 bpp所需的帧缓冲器大小约为128 KB。这种情况下，帧缓冲器可以位于内部SRAM1（240 KB）中。这对于双帧缓冲器情况是无效的，因为128 x 2 KB的大小超过了内部SRAM大小。

对于16 bpp色深和双帧缓冲器配置，所需的帧缓冲器大小（2 x 255 KB）超过了内部SRAM大小，因此此配置下必须使用外部SRAM或SDRAM。

对于24 bpp色深和双帧缓冲器配置，所需的帧缓冲器大小（2 x 382.5 KB）超过了内部SRAM大小，因此此配置下必须使用外部SRAM或SDRAM。

下一步是检查SDRAM 16位总线宽度是否可以维持所需的分辨率和色深。

检查24 bpp的480 x 272分辨率是否适合SDRAM 16位配置

在这个阶段，用户应当已经决定了使用外部SDRAM，但仍然需要检查SDRAM 16位总线宽度（探索板中的实际硬件实现）是否与480 x 272 @ 60 Hz显示器尺寸和24 bpp色深相匹配。

为了确定这样的硬件配置是否可以支持所需的显示器尺寸和色深，用户首先应计算像素时钟。

计算出的LCD_CLK约为9.5 MHz（用于计算像素时钟，参考第 6.2.3节：LTDC GPIO配置）。

然后，用户应该根据以下参数，检查计算出的像素时钟是否不高于表 11中所示的最大LCD_CLK。

- 所用LTDC层的数量：在这个例子中只使用了一层。
- 系统时钟速度HCLK和帧缓冲存储器速度：HCLK@200MHz和SDRAM@100MHz。
- 外部帧缓冲器存储器总线宽度SDRAM 16位。
- 同时访问外部SDRAM的AHB主设备数量：2个主设备（DMA2D和LTDC）。

参考“LTDC + DMA2D”列和一个图层行中的像素时钟表 11，对于16位的SDRAM，像素时钟可以达到34MHz。

因此，SDRAM 16位总线宽度足以维持480x272 @ 60Hz分辨率（LCD_CLK=9.5MHz）及24 bpp的色深。

6.2.3 LTDC GPIO配置

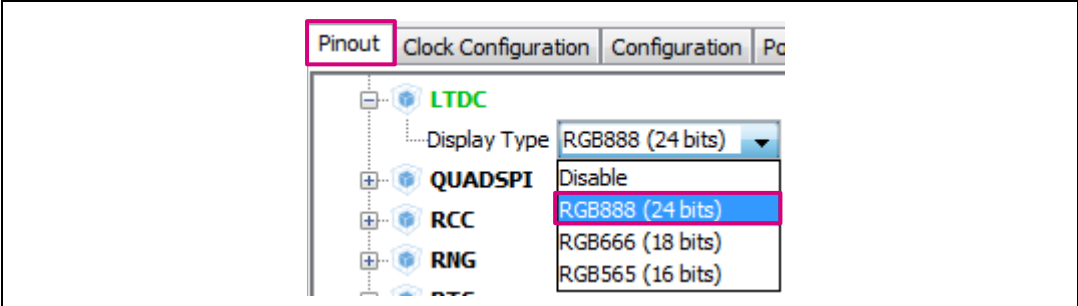
如图 34中所示，ROCKTECH RK043FN48H显示器使用24位并行RGB888连接到STM32F746xx。

LTDC RGB接口引脚配置

创建了STM32CubeMX项目后，在引脚分配选项卡中从所列硬件配置中选择一个。图 36显示了如何使用STM32CubeMX选择RGB888硬件配置。

用户也可以通过为每个GPIO逐个设置正确的备用功能来配置所有GPIO。

图36. STM32CubeMX：LTDC GPIO配置



如果在选择一个硬件配置（如 图 36中所示的RGB888）后，所用GPIO与显示面板连接板不匹配，用户可以更改所需的GPIO并直接在引脚上配置备用功能。图 37举例说明了如何手动将PJ7引脚配置为LTDC_G0备用功能。

图37. STM32CubeMX：PJ7引脚配置为LTDC_G0备用功能

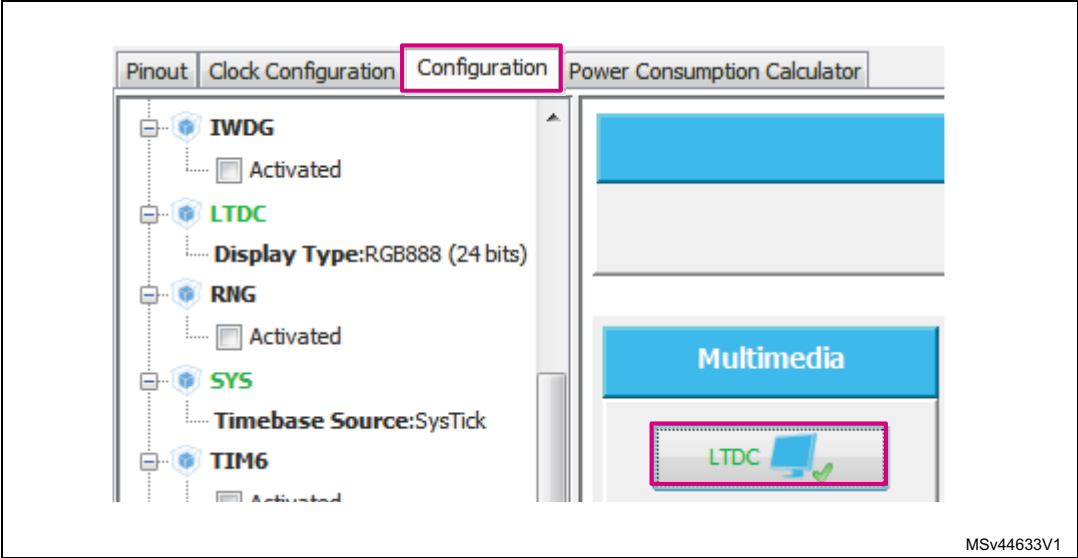


当所有的LTDC接口GPIO配置正确时，所用引脚将以绿色突出显示。

当所有LTDC GPIO管脚都配置好以后，用户应将其速度设置得非常高。

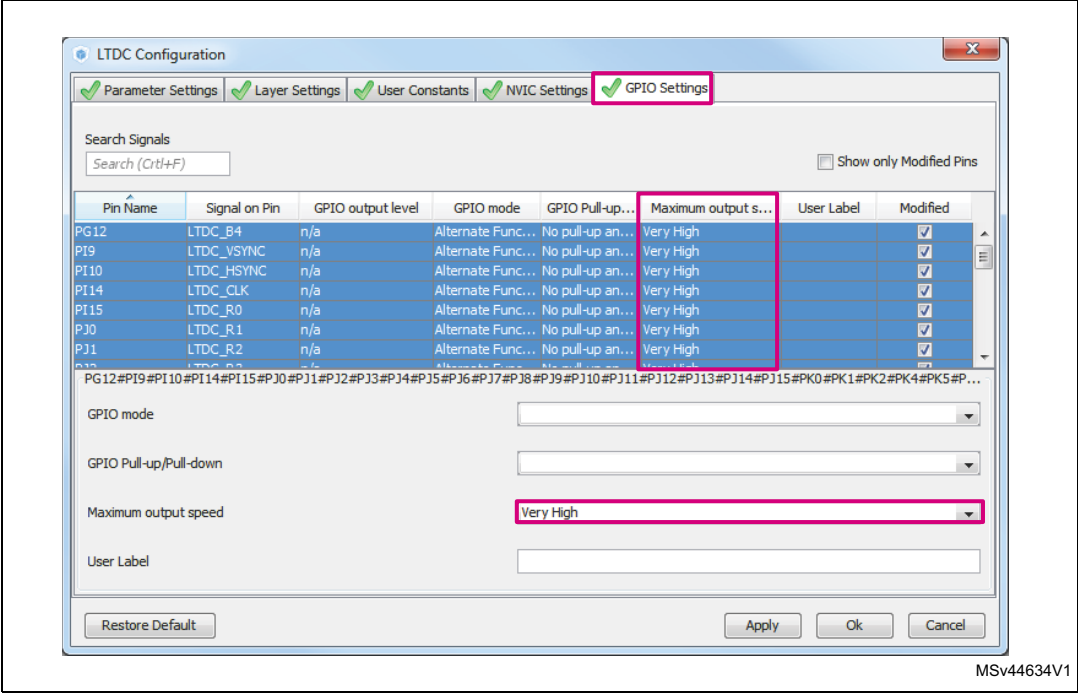
要使用STM32CubeMX设置GPIO的速度，请选择配置选项卡，然后点击LTDC按钮，如 图 38 中所示。

图38. STM32CubeMX: LTDC配置



在图 39中所述的LTDC配置窗口中，选择所有LTDC引脚，然后将最大输出速度设置为非常高。

图39. STM32CubeMX: LTDC GPIO输出速度配置



显示模块的特定引脚配置

当所有LTDC接口引脚都按照LCD-TFT面板连接正确配置后，用户应配置连接到显示器的其他特定引脚（LCD_DISP，INT引脚和I2C接口）

必须将LCD_DISP引脚（PI12引脚）配置为高电平的输出推挽，来启用显示器，否则显示器将保持待机模式。

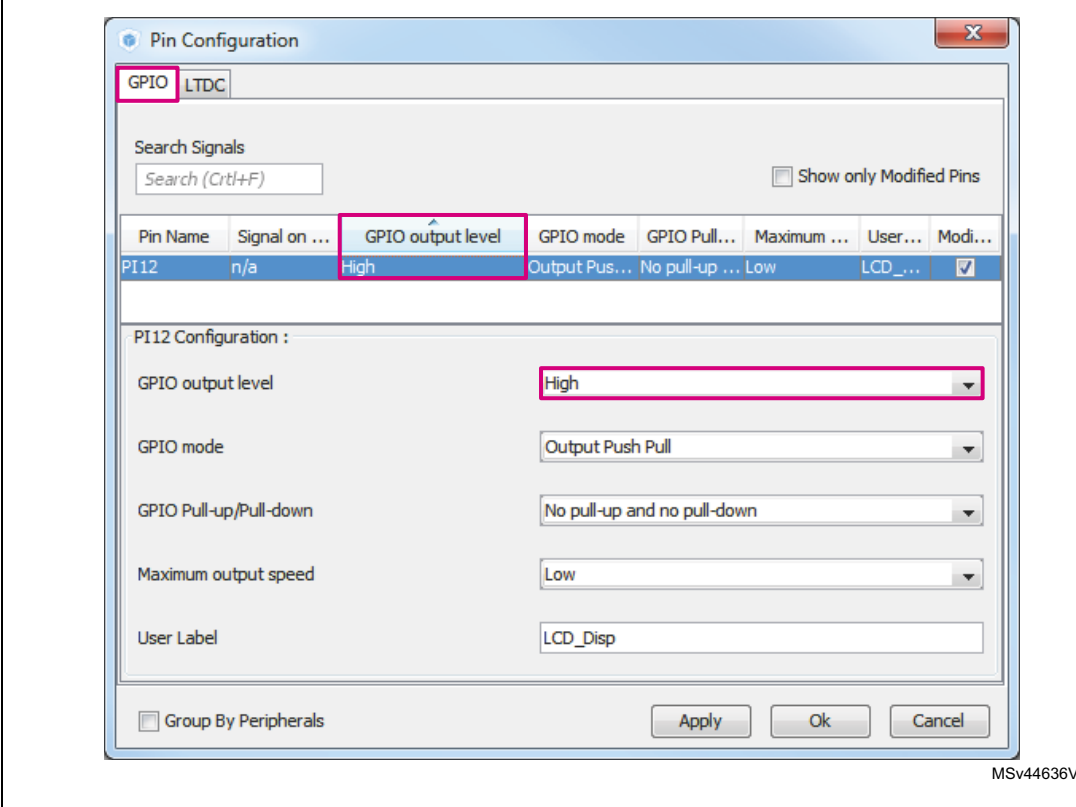
要使用STM32CubeMX配置输出模式下的LCD_DISP引脚，应在引脚分配选项卡中点击PI12引脚，然后选择GPIO_Output（参见图 40）。

图40. STM32CubeMX：显示器使能引脚（LCD_DISP）配置



然后，应将LDC_DISP（PI12）引脚配置为高电平，在配置选项卡中单击GPIO按钮可实现此配置。然后在引脚配置窗口中，将GPIO输出电平设置为高电平，如图 41中所述。

图41. STM32CubeMX：将LCD_DISP引脚输出电平设置为高电平



MSv44636V

由于存在R85上拉电阻，所以如果LCD_BL_CTRL（PK3）引脚保持悬空状态，则默认情况下背光处于最高电平，因此无需配置此引脚。

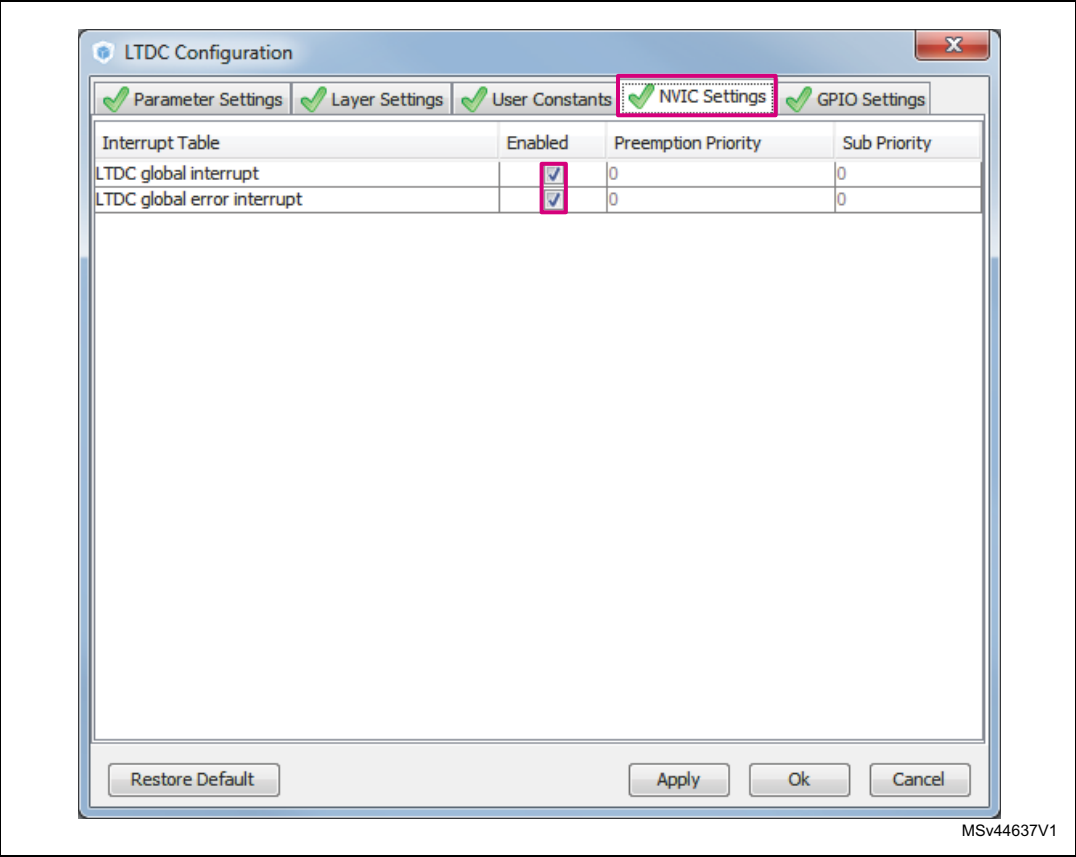
启用LTDC中断

FIFO下溢运行和传输错误中断可在在hal ltdc驱动程序HAL_LTDC_Init()函数中使能。所以用户应该在NVIC端启用LTDC全局中断。

要使用STM32CubeMX使能LTDC全局中断，请选择配置选项卡，然后点击LTDC按钮，如图 38 中所示。

在 图 42 中所示的LTDC配置窗口中选择NVIC设置选项卡，检查LTDC全局中断，然后单击OK按钮。

图42. STM32CubeMX：使能LTDC全局和错误中断



6.2.4 LTDC外设配置

本节介绍如何使用STM32CubeMX工具配置LTDC时钟/时序和图层参数。

LTDC时钟和时序配置

系统时钟配置

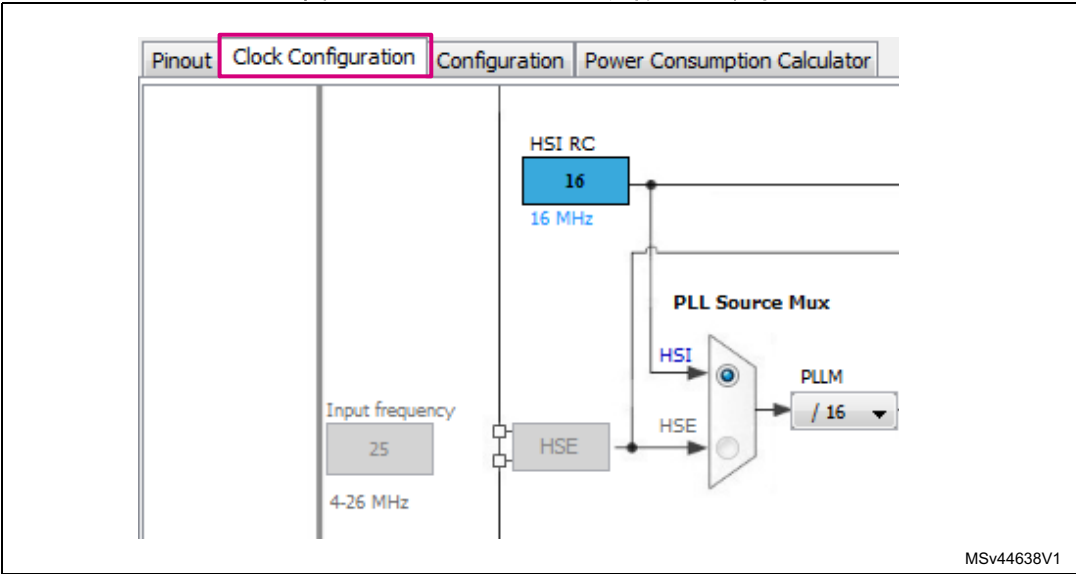
在这个例子中，系统时钟的配置如图 44所示，配置如下：

- 使用内部HSI RC，其中主PLL用作系统源时钟。
- HCLK @ 200 MHz，因此Cortex®-M7和LTDC均运行于@ 200 MHz。

注： HCLK设置为200 MHz而非216 MHz，这是为了使用HCLK / 2预分频器将SDRAM_FMC设置为其最大速度100 MHz。

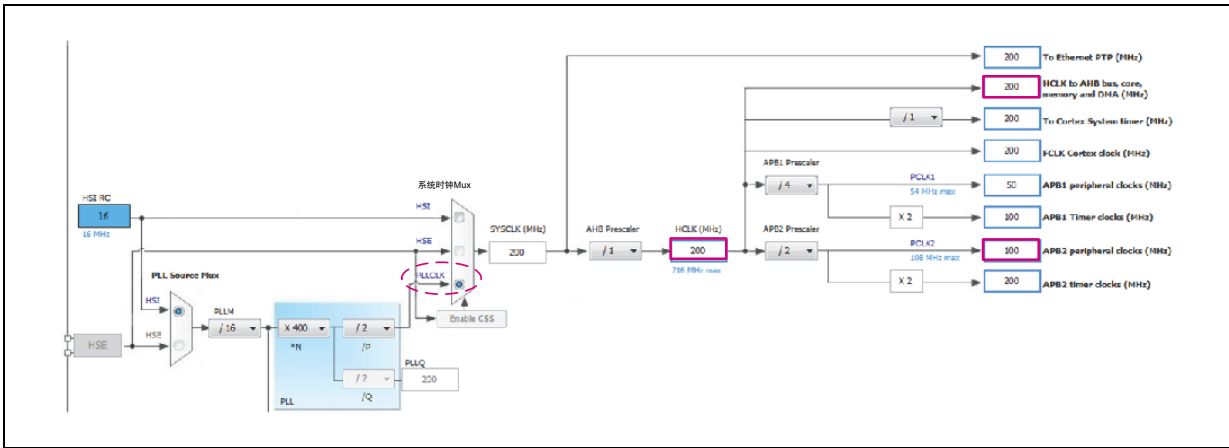
为了使用STM32CubeMX配置系统时钟，请选择时钟配置选项卡，如图 43所示。

图43. STM32CubeMX：时钟配置选项卡



然后，要实现系统时钟HCLK@200MHz，请在时钟配置选项卡中设置PLL和预分频器，如图 44所示。

图44. STM32CubeMX：系统时钟配置



像素时钟配置

应使用显示数据表中的参数来计算LCD_CLK。
为了进行计算，用户应该确定总宽度和总高度。

像素时钟以60 Hz刷新率计算，如下所示：

$LCD_CLK = TOTALW \times TOTALH \times \text{刷新率}$ （见第 4.7.2 节：LTDC时钟和时序配置中的抽取显示器时序参数）

$$LCD_CLK = 531 \times 297 \times 60 = 9.5 \text{ MHz}$$

要使用STM32CubeMX将LTDC像素时钟配置为9.5 MHz，请选择时钟配置选项卡，然后如图 45所示设置PLLSAI和预分频器。

图45. STM32CubeMX：LTDC像素时钟配置



时序参数配置

为了使用STM32CubeMX配置显示器时序，用户应该从器件的数据表中提取时序参数时序寄存器。对于本例，请参阅表 14上ROCKTECH数据表中的提取操作。建议使用典型的显示时序。

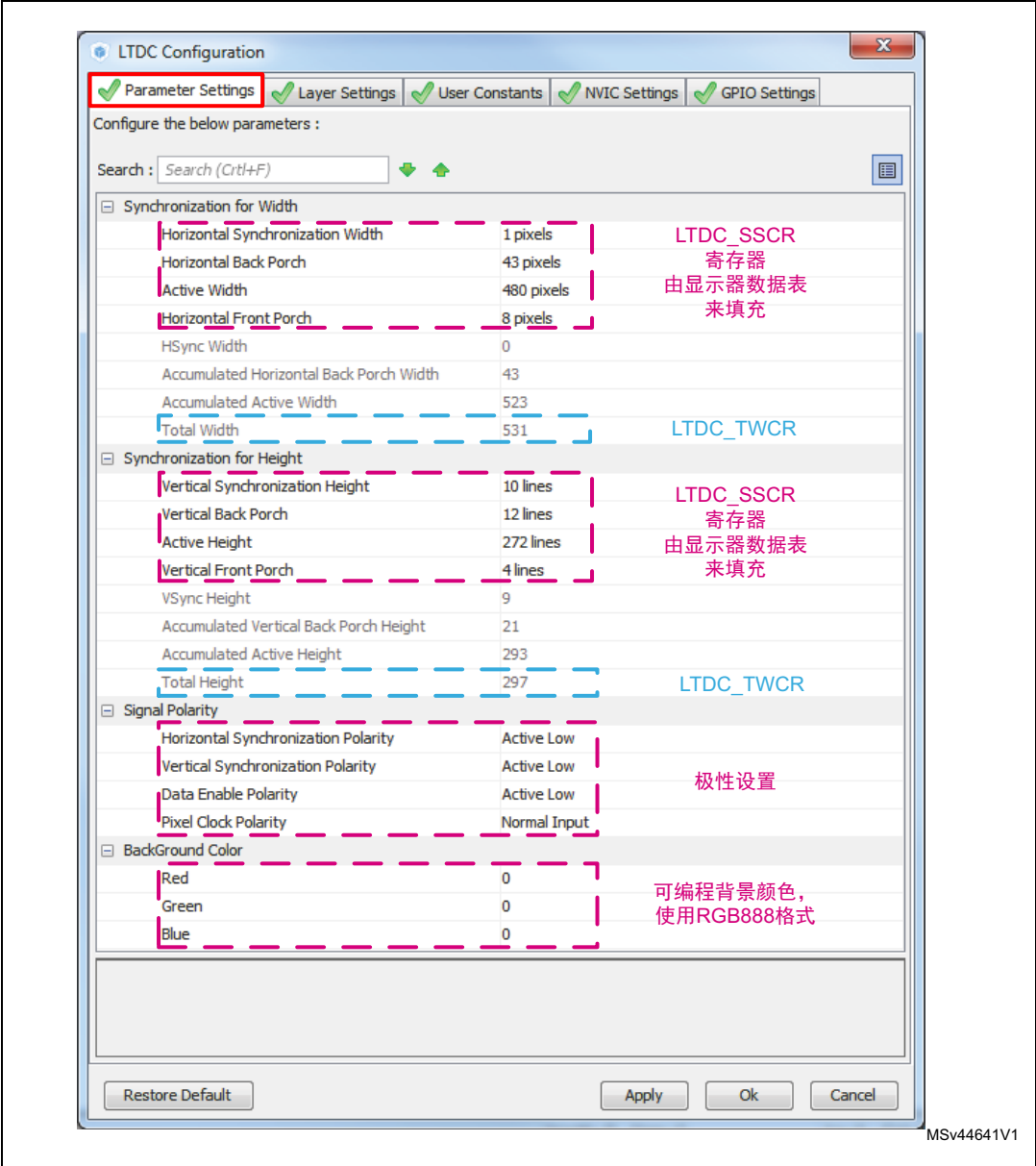
为了配置显示时间，用户必须按照图 38中指示转到配置选项卡，然后单击LTDC按钮。在LTDC配置窗口中，用户应选择参数设置选项卡并填写时间值（参考图 46）。

LTDC控制信号极性配置

参考显示数据表，HSYNC和VSYNC应该是低电平有效，DE信号应该为高电平有效。由于DE信号在输出中是反转的，所以它也应被设置为低电平有效。LCD_CLK信号不应该反转。

图 46显示了按照ROCKTECH显示器数据表进行的控制信号极性配置以及LTDC配置。

图46. STM32CubeMX：LTDC时间配置



MSv44641V1

LTDC图层参数配置

在这个阶段，所有的LTDC时钟和时序都应该在STM32CubeMX项目中进行设置。

用户应根据显示器尺寸和色深来配置LTDC layer1参数。

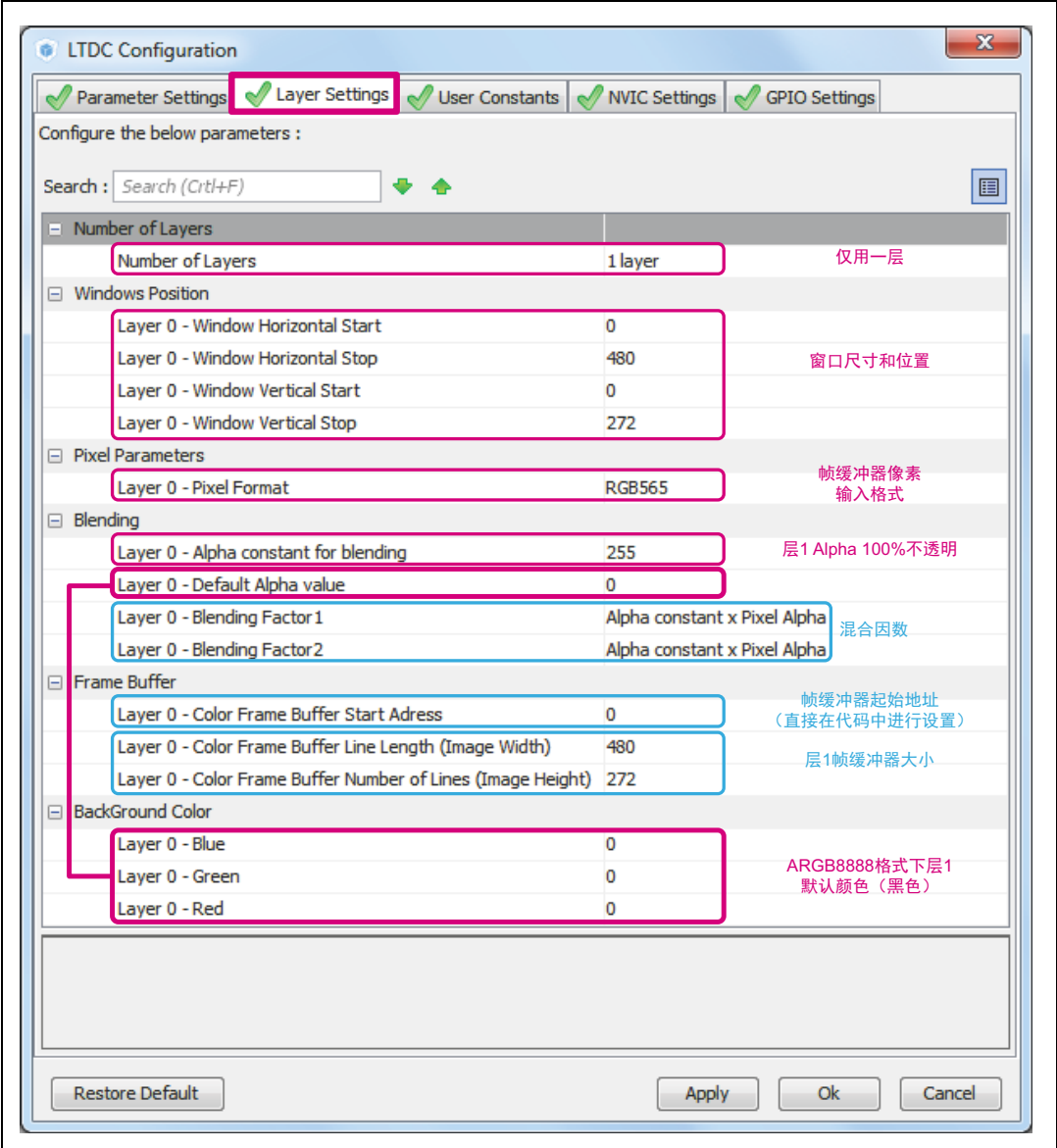
如果需要，用户还可以通过将LTDC配置窗口中的“图层数”字段设置为“2层”来启用层2，如 [图 47](#) 所示。

要使用STM32CubeMX设置LTDC层1，用户必须选择配置选项卡，然后点击LTDC按钮，如 [图 37](#) 中所示。

在 [图 47](#) 中所示的LTDC配置窗口中，用户必须选择图层设置选项卡，设置LTDC层1参数，然后单击OK按钮。

在这一步，用户可以通过点击“项目 ->生成代码”来生成具有所需工具链的项目

图47. STM32CubeMX：LTDC层1参数设置



6.2.5 显示来自内部闪存的图像

为确保LTDC根据显示面板的规格进行了正确配置，显示来自内部闪存的图像非常重要。要做到这一点，用户应首先将图像转换为C或头文件并将其添加到项目中。

使用LCD图像转换工具将图像转换为头文件

用户必须根据所配置的LTDC图层像素输入格式RGB565来生成头文件（参见 [像素输入格式第 27页](#)和） [第4.8节：存储图形基元第 58页](#)。

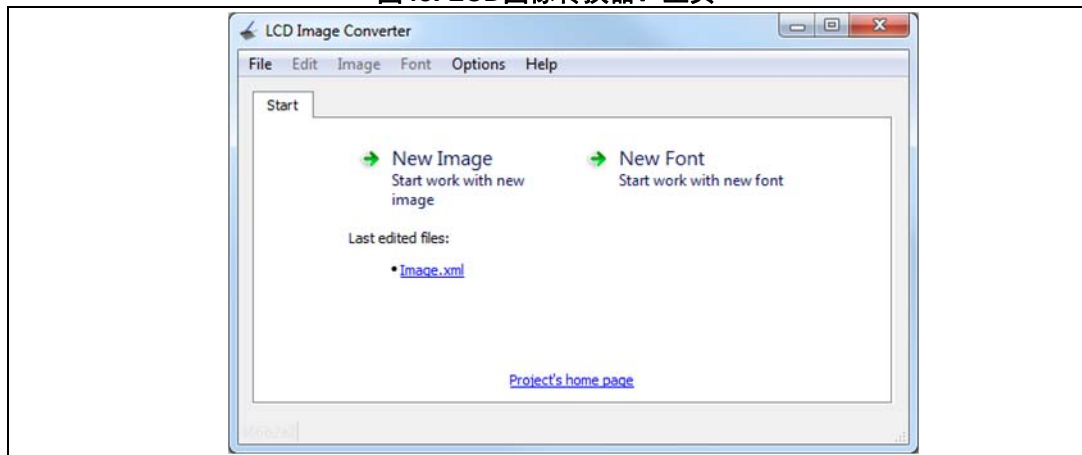
本例中使用了LCD-Image-Converter-20161012工具（关于此工具的更多细节，请参考第 4.8 节）。

要转换图像，用户必须首先运行LCD-Image-Converter工具，然后在图 48中所示的主页中单击“文件 -> 打开”，然后选择要转换的图像文件。

使用的图像尺寸应与LTDC layer1配置（480 x 272）一致。如果使用的图像尺寸与LTDC layer1配置不一致，用户可以调整图像大小，转到图像 -> 调整大小或选择正确尺寸的其他图像。

在这个例子中，使用的图像尺寸是480 x 272，并显示ST标志（见图 49）。

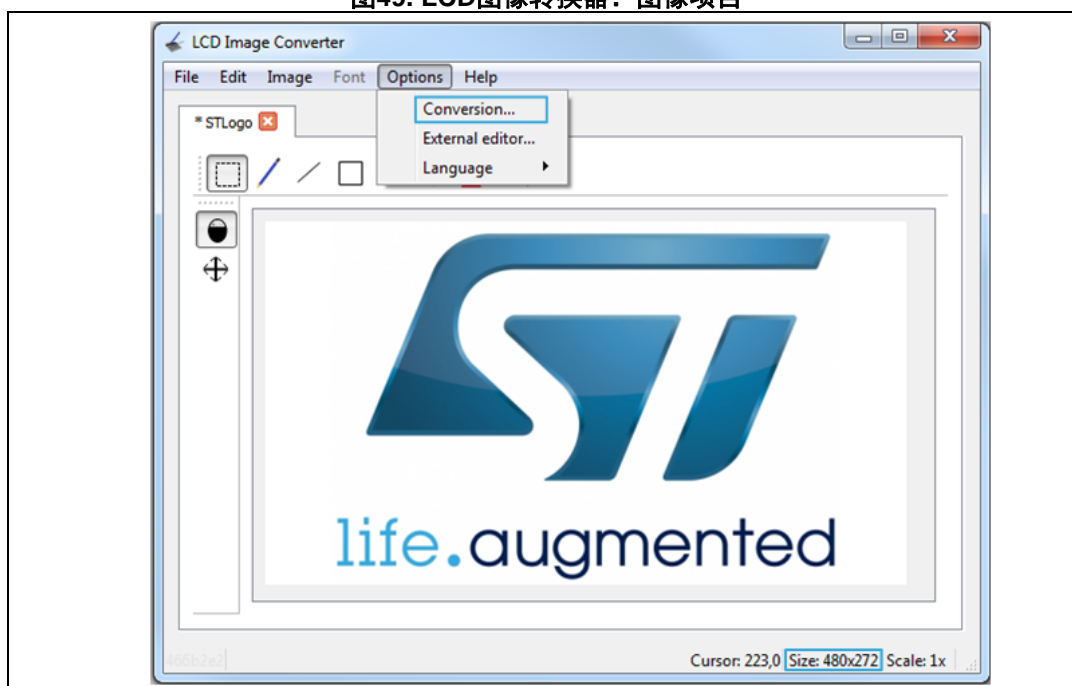
图48. LCD图像转换器：主页



图像随后将显示在工具的主页上，如图 49中所述。

要将图像转换为头文件以避免第 4.8 节：存储图形基元中所述的红蓝调换问题，用户应配置该工具，从而将图像转换为32位字的表格。要实现这一点，可以在主页菜单中点击选项 -> 转换，如图 49中所示。

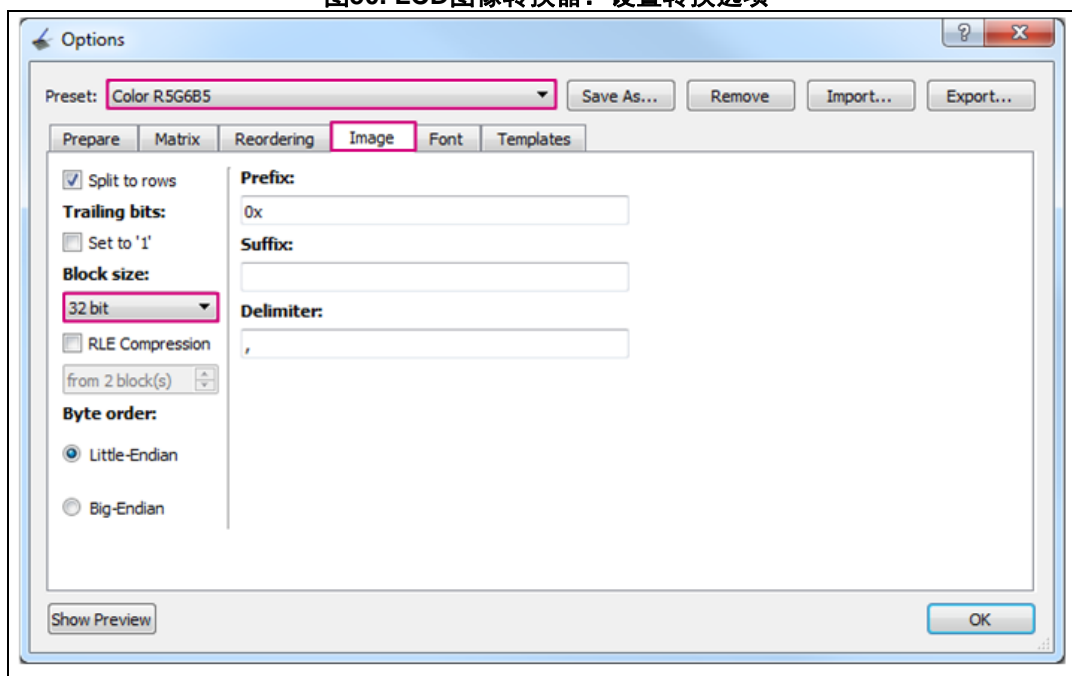
图49. LCD图像转换器：图像项目



在图 50中所示的选项窗口中，选择“图像”选项卡，然后在预设字段中选择“RGB565颜色”，然后将块大小字段设置为32位并单击OK按钮。

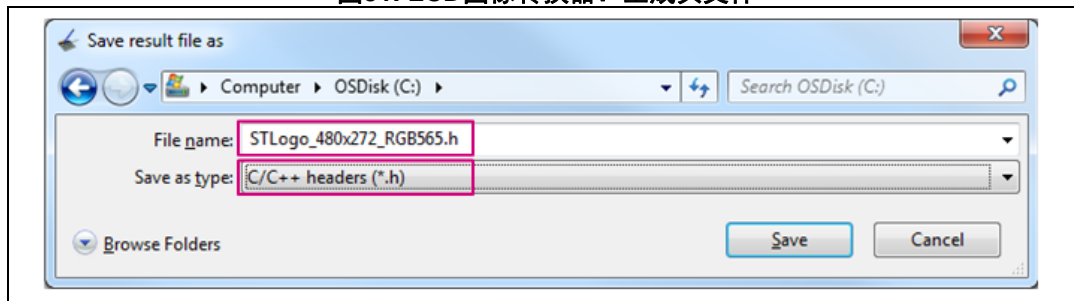
注：用户也可以将图像转换为字节表，但在这种情况下，应该在转换窗口矩阵选项卡中交换红蓝颜色。

图50. LCD图像转换器：设置转换选项



要生成头文件，请点击文件->转换。然后在如图 51 中所示的显示窗口中，将文件类型设置为“C/C++头文件 (*.h)”，然后通过单击保存按钮将*.h文件保存在包含“\Inc”的目录中（与main.h文件的位置相同）

图51. LCD图像转换器：生成头文件



生成的头文件应包含在main.c文件中。它包括了一个32位字表，其中每个字代表两个像素。

在这个头文件中，用户必须对位于表后面的结构体定义进行注释，并只保留表定义，如下所示：

```
/* 转换后的图像: image_data_STLogo定义 */
const uint32_t image_data_STLogo[65280] = {0xffffffff, 0xffffffff, .....};
```

将LTDC帧缓冲器的Layer1起始地址设置为内部闪存（闪存中的图像地址）

由STM32CubeMX生成的项目应该在main.c文件中包含MX_LTDC_Init()函数，该函数能够对LTDC外设进行配置。

为了显示图像，用户应该将LTDC Layer1帧缓冲器起始地址设置为图像在内部闪存中的地址。

下面介绍MX_LTDC_Init()函数的帧缓冲器起始地址设置。

```
/* 由STM32CubeMX工具生成的LTDC配置函数 */
static void MX_LTDC_Init(void)
{
    LTDC_LayerCfgTypeDef pLayerCfg;

    hltdc.Instance = LTDC;
    /* LTDC控制信号极性设置 */
    hltdc.Init.HSPolarity = LTDC_HSPOLARITY_AL;
    hltdc.Init.VSPolarity = LTDC_VSPOLARITY_AL;
    hltdc.Init.DEPolarity = LTDC_DEPOLARITY_AL;
    hltdc.Init.PCPolarity = LTDC_PCPOLARITY_IPC;
    /* 时序配置 */
    hltdc.Init.HorizontalSync = 0;
    hltdc.Init.VerticalSync = 9;
    hltdc.Init.AccumulatedHBP = 43;
    hltdc.Init.AccumulatedVBP = 21;
```

```
hltdc.Init.AccumulatedActiveW = 523;
hltdc.Init.AccumulatedActiveH = 293;
hltdc.Init.TotalWidth = 531;
hltdc.Init.TotalHeigh = 297;
/*背景颜色*/
hltdc.Init.Backcolor.Blue = 0;
hltdc.Init.Backcolor.Green = 0;
hltdc.Init.Backcolor.Red = 0x0;
if (HAL_LTDC_Init(&hltdc) != HAL_OK)
{
    Error_Handler();
}
/* Layer1窗口大小和位置设置 */
pLayerCfg.WindowX0 = 0;
pLayerCfg.WindowX1 = 480;
pLayerCfg.WindowY0 = 0;
pLayerCfg.WindowY1 = 272;
/* Layer1像素输入格式设置 */
pLayerCfg.PixelFormat = LTDC_PIXEL_FORMAT_RGB565;
/* Layer1常量Alpha设置100%不透明 */
pLayerCfg.Alpha = 255;
/* Layer1混合因数设置 */
pLayerCfg.BlendingFactor1 = LTDC_BLENDING_FACTOR1_PAxCA;
pLayerCfg.BlendingFactor2 = LTDC_BLENDING_FACTOR2_PAxCA;
/* 用户应设置帧缓冲器起始地址（如果使用外部SDRAM，则可以为0xC0000000） */
pLayerCfg.FBStartAdress = (uint32_t)&image_data_STLogo;
pLayerCfg.ImageWidth = 480;
pLayerCfg.ImageHeight = 272;
/* Layer1默认颜色设置 */
pLayerCfg.Alpha0 = 0;
pLayerCfg.Backcolor.Blue = 0;
pLayerCfg.Backcolor.Green = 0;
pLayerCfg.Backcolor.Red = 0;
if (HAL_LTDC_ConfigLayer(&hltdc, &pLayerCfg, 0) != HAL_OK)
{
    Error_Handler();
}
}
```

当在项目中正确配置了LTDC后，用户应该构建项目并运行它。

6.2.6 FMC SDRAM配置

由于外部SDRAM包含了LTDC帧缓冲器，因此必须对其进行配置。要配置安装在STM32746G-Discovery板上的FMC_SDRAM和SDRAM存储设备，用户可以使用STM32CubeMX或使用现有的BSP驱动程序。

要使用BSP驱动程序配置FMC_SDRAM，请遵循以下步骤：

1. 将以下文件添加到项目中：BSP stm32746g_discovery_sdram.c和stm32746g_discovery_sdram.h。main.c文件中包含stm32f7xx_hal_sdram.h。将stm32f7xx_hal_sdram.c和stm32f7xx_ll_fmc.c HAL驱动程序添加到项目中
2. 通过取消注释SDRAM模块定义来启用stm32f7xx_hal_conf.h文件中的SDRAM模块。main.c文件中包含了stm32f7xx_hal_sdram.h文件。
3. 调用main()函数中的BSP_SDRAM_Init()函数。

6.2.7 MPU和高速缓存配置

如第 4.6节中所示，要正确配置MPU属性以避免由Cortex®-M7推测性读取访问和缓存维护引起的图形性能问题。

本节介绍对应于STM32F746G-DISCO板硬件配置的MPU属性配置示例。

使用STM32CubeMX可以轻松配置MPU存储器属性。本节最后描述了使用STM32CubeMX生成的MPU配置的代码示例。

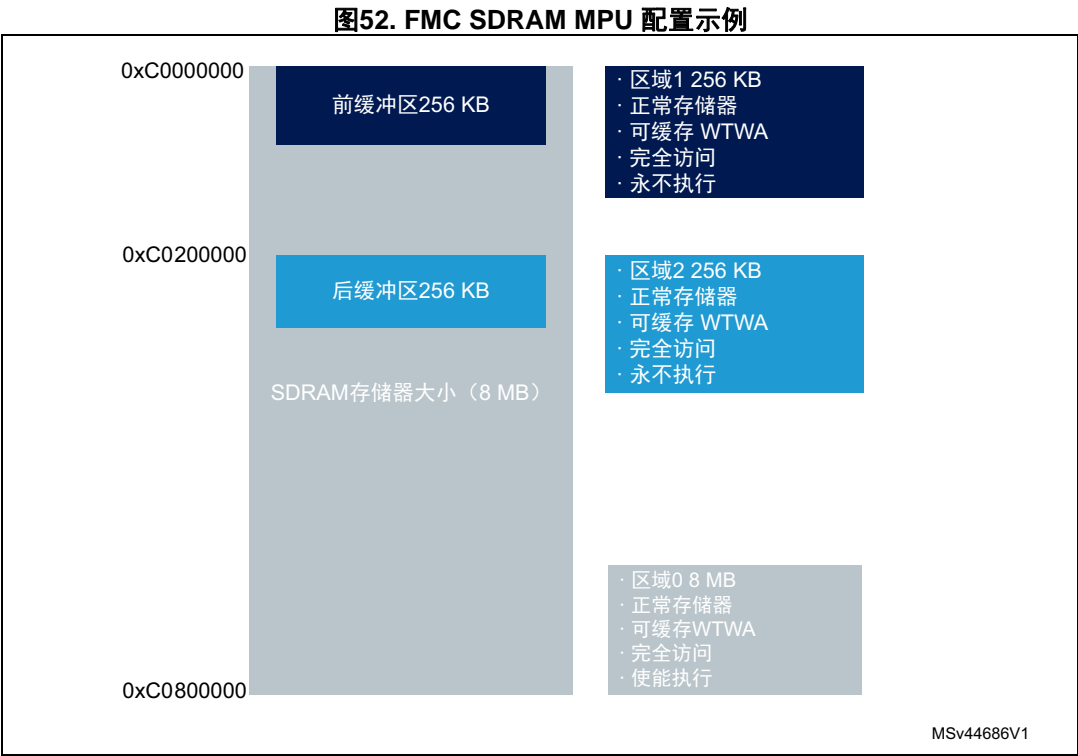
MPU配置示例：FMC_SDRAM

在此配置示例中，使用了双帧缓冲器技术；前缓冲区位于SDRAMbank1中，而后缓冲区位于SDRAM bank2中，符合第 4.5.3节：优化从SDRAM读取LTDC帧缓冲器的过程中描述的SDRAM带宽优化。

创建了以下MPU区域（没有锯齿的FMC）：

- Region0：定义了SDRAM存储器大小为8 MB
- Region1：定义了前缓冲区为256 KB（16 bpp x 480 x 272），它与region0重叠
- Region2：定义了后缓冲区为256 KB（16 bpp x 480 x 272），它与region0重叠

图 52说明了SDRAM区域的MPU配置。



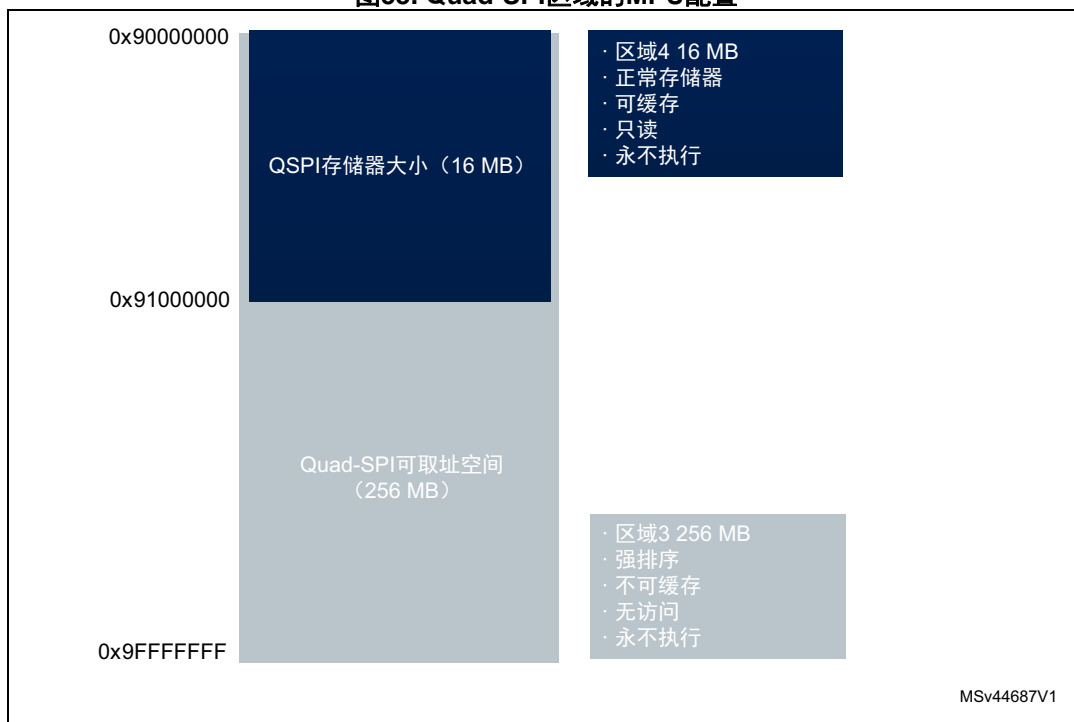
MPU配置示例：存储器映射中的Quad-SPI

此示例说明如何为Quad-SPI接口配置MPU。QSPI存储器包含图形基元，它可以通过Cortex[®]-M7、DMA2D或LTDC进行访问。为此，应将Quad-SPI接口设置为存储器映射模式，并且MPU区域必须按如下所述进行配置：

- Region3: 定义整个Quad-SPI可寻址空间，它必须设置为强禁止任何CPU推测性读取访问该区域。
- Region4: 定义实际QSPI内存空间，反映任意主机可以访问的存储器大小。

[图 53](#)说明了Quad-SPI区域的MPU配置。

图53. Quad-SPI区域的MPU配置



SDRAM和Quad-SPI MPU配置示例

以下代码（由STM32CubeMX生成）显示了如何设置前面所述配置下FMC_SDRAM和Quad-SPI的MPU属性。

```
/* MPU配置 */
void MPU_Config(void)
{
    MPU_Region_InitTypeDef MPU_InitStruct;

    /* 禁用MPU */
    HAL_MPU_Disable();

    /* 配置区域0的MPU属性 */
    /* 对于SDRAM，将MPU属性配置为正常存储器 */
    MPU_InitStruct.Enable = MPU_REGION_ENABLE;
    MPU_InitStruct.Number = MPU_REGION_NUMBER0;
    MPU_InitStruct.BaseAddress = 0xC0000000;
    MPU_InitStruct.Size = MPU_REGION_SIZE_8MB;
    MPU_InitStruct.SubRegionDisable = 0x0;
    MPU_InitStruct.TypeExtField = MPU_TEX_LEVEL1;
    MPU_InitStruct.AccessPermission = MPU_REGION_FULL_ACCESS;
    MPU_InitStruct.DisableExec = MPU_INSTRUCTION_ACCESS_ENABLE;
    MPU_InitStruct.IsShareable = MPU_ACCESS_NOT_SHAREABLE;
    MPU_InitStruct.IsCacheable = MPU_ACCESS_CACHEABLE;
    MPU_InitStruct.IsBufferable = MPU_ACCESS_BUFFERABLE;
```

```
HAL_MPU_ConfigRegion(&MPU_InitStruct);

/* 配置区域1的MPU属性 */
/* 对于前缓冲区, 将MPU属性配置为正常存储器 */
MPU_InitStruct.Enable = MPU_REGION_ENABLE;
MPU_InitStruct.Number = MPU_REGION_NUMBER1;
MPU_InitStruct.BaseAddress = 0xC0000000;
MPU_InitStruct.Size = MPU_REGION_SIZE_256KB;
MPU_InitStruct.SubRegionDisable = 0x0;
MPU_InitStruct.TypeExtField = MPU_TEX_LEVEL1;
MPU_InitStruct.AccessPermission = MPU_REGION_FULL_ACCESS;
MPU_InitStruct.DisableExec = MPU_INSTRUCTION_ACCESS_DISABLE;
MPU_InitStruct.IsShareable = MPU_ACCESS_NOT_SHAREABLE;
MPU_InitStruct.IsCacheable = MPU_ACCESS_CACHEABLE;
MPU_InitStruct.IsBufferable = MPU_ACCESS_BUFFERABLE;

HAL_MPU_ConfigRegion(&MPU_InitStruct);

/* 配置区域2的MPU属性 */
/* 对于后缓冲区, 将MPU属性配置为正常存储器 */
MPU_InitStruct.Enable = MPU_REGION_ENABLE;
MPU_InitStruct.Number = MPU_REGION_NUMBER2;
MPU_InitStruct.BaseAddress = 0xC0200000;
MPU_InitStruct.Size = MPU_REGION_SIZE_256KB;
MPU_InitStruct.SubRegionDisable = 0x0;
MPU_InitStruct.TypeExtField = MPU_TEX_LEVEL1;
MPU_InitStruct.AccessPermission = MPU_REGION_FULL_ACCESS;
MPU_InitStruct.DisableExec = MPU_INSTRUCTION_ACCESS_DISABLE;
MPU_InitStruct.IsShareable = MPU_ACCESS_NOT_SHAREABLE;
MPU_InitStruct.IsCacheable = MPU_ACCESS_CACHEABLE;
MPU_InitStruct.IsBufferable = MPU_ACCESS_BUFFERABLE;

HAL_MPU_ConfigRegion(&MPU_InitStruct);

/* 配置区域3的MPU属性 */
/* 将Quad-SPI区域的MPU属性配置为强排序存储器 */
MPU_InitStruct.Enable = MPU_REGION_ENABLE;
MPU_InitStruct.Number = MPU_REGION_NUMBER3;
MPU_InitStruct.BaseAddress = 0x90000000;
MPU_InitStruct.Size = MPU_REGION_SIZE_256MB;
MPU_InitStruct.SubRegionDisable = 0x0;
MPU_InitStruct.TypeExtField = MPU_TEX_LEVEL0;
MPU_InitStruct.AccessPermission = MPU_REGION_NO_ACCESS;
MPU_InitStruct.DisableExec = MPU_INSTRUCTION_ACCESS_DISABLE;
```

```

MPU_InitStruct.IsShareable = MPU_ACCESS_NOT_SHAREABLE;
MPU_InitStruct.IsCacheable = MPU_ACCESS_NOT_CACHEABLE;
MPU_InitStruct.IsBufferable = MPU_ACCESS_NOT_BUFFERABLE;

HAL_MPU_ConfigRegion(&MPU_InitStruct);

/* 配置区域4的MPU属性 */
/* 将QSPI存储器的MPU属性配置为正常存储器 */
MPU_InitStruct.Enable = MPU_REGION_ENABLE;
MPU_InitStruct.Number = MPU_REGION_NUMBER4;
MPU_InitStruct.BaseAddress = 0x90000000;
MPU_InitStruct.Size = MPU_REGION_SIZE_16MB;
MPU_InitStruct.SubRegionDisable = 0x0;
MPU_InitStruct.TypeExtField = MPU_TEX_LEVEL0;
MPU_InitStruct.AccessPermission = MPU_REGION_PRIV_RO;
MPU_InitStruct.DisableExec = MPU_INSTRUCTION_ACCESS_DISABLE;
MPU_InitStruct.IsShareable = MPU_ACCESS_NOT_SHAREABLE;
MPU_InitStruct.IsCacheable = MPU_ACCESS_CACHEABLE;
MPU_InitStruct.IsBufferable = MPU_ACCESS_NOT_BUFFERABLE;

HAL_MPU_ConfigRegion(&MPU_InitStruct);

/* 启用MPU */
HAL_MPU_Enable(MPU_PRIVILEGED_DEFAULT);
}

```

6.3 带LCD-TFT面板的参考板

ST提供了广泛的参考板，如NUCLEO、Discovery和EVAL板，其中有许多嵌入式显示面板。对于具有板上显示器但不嵌入LTDC的STM32参考板，DBI（FMC或SPI）接口用于将STM32与显示器连接。

对于其他STM32板，LTDC用于连接显示面板。

这些参考板可用于评估特定硬件/软件配置下的图形功能。

表 17总结了嵌入LTDC并具有板载LCD-TFT面板的STM32参考板

表17. STM32参考板，嵌入了LTDC并具有板上LCD-TFT面板

产品	板	TFT-LCD板					内部 SRAM (KB)	外部 SDRAM	外部 SRAM	QSPI (MB)
		接口	大小 (Inch)	分辨率	色深	触摸 传感器				
STM32 F429xx/ STM32 F439xx	32F429I 探索	DPI	2.4	240 x 320	RGB666	电阻	256	16 位	NA	NA
	STM32439I- EVAL2	DPI	5.7	640 x 480	RGB666	电容性		32 位	16 位	NA
	STM32429I- EVAL1	DPI	4.3	480 x 272	RGB888	电阻				
STM32 F469xx/ STM32 F479xx	32F469IDISC OVERY	MIPI-DSI	4	800 x 480	RGB888	电容性	384	32 位	NA	16
	STM32469I- EVAL ⁽¹⁾	MIPI-DSI	4	800 x 480	RGB888	电容性		32 位	16 位	64
STM32 F7x6线	32F746GDIS COVERY	DPI	4.3	480 x 272	RGB888	电容性	320	16 位	NA	16
	STM32746G- EVAL	DPI	5.7	640 x 480	RGB666	电容性		32 位	16 位	64
		DPI	4.3	480 x 272	RGB888	电阻				
STM32 F7x9线 ⁽¹⁾	STM32F769I- DISCO ⁽²⁾	MIPI-DSI	4	800x4 80	RGB888	电容性	512	32 位	NA	64
	STM32F779I- EVAL STM32F769I- EVAL	MIPI-DSI	4	800x4 80	RGB888	电容性		32 位	16 位	64

1. 可用板B-LCDAD-HDMI1（可以单独购买）能够将DSI转换为HDMI格式以连接HDMI用户显示器。DSI到LCD适配器板B-LCDAD-RPI1（可单独购买）提供了灵活的连接器，包括微控制器主板到标准显示器连接器（TE 1-1734248）。
2. 另一个可用的探索板是STM32F769I-DISC1，但其不带嵌入显示器，显示器可以按照B-LCD40-DSI1订购代码来单独购买。

7 所支持的显示面板

显示控制器嵌入了一个非常灵活的接口，可提供以下功能，使STM32MCU能够支持市场上的多种并行显示面板（如TFT-LCD和OLED显示器）：

- 不同的信号极性。
- 可编程的时序和分辨率。

显示面板的像素时钟（如制造商数据表中所示）不得高于STM32的最大像素时钟。所以用户应该参考显示器数据表，以确保面板的运行时钟低于最大像素时钟。

8 常见问题

本节总结了有关LTDC使用和配置的常见问题。

表18. 常见问题

问题	回答
LTDC所支持的最大分辨率是多少？	没有绝对的最大分辨率，因为它取决于几个参数，例如： – 色深 – 所用SDRAM总线宽度 – 系统运行速度（HCLK） – 同时访问用于帧缓冲的存储器的AHB主设备数量。 请参见第 4.2.2 节：考虑存储器时检查显示兼容性带宽要求。
STM32F4系列或STM32F7系列是否支持1280 x 720p 60 Hz分辨率？	是，见第 4.2.2 节：考虑存储器时检查显示兼容性带宽要求中示例。
对于特定的分辨率，应使用哪个SDRAM总线宽度？	没有一个确切的特定总线宽度，它取决于分辨率、色深以及是否与其他AHB主设备共享SDRAM。 SDRAM总线宽度越高越好。32位SDRAM性能最佳。
如何得到特定硬件的最大支持分辨率？	请参见第 4.2.2 节：考虑存储器时检查显示兼容性带宽要求。
LTDC支持OLED显示器吗？	是的，如果OLED显示屏具有并行RGB接口，则可以支持。
LTDC支持STN显示器吗？	不能，LTDC不支持STN显示器。
为什么图像以红蓝调换显示？	这是因为图像没有按照所配置的像素输入格式存储到内存中（参见第 4.8.1 节：将图像转换为C文件）。
LTDC是否支持灰度？	支持，使用L8模式并使用正确的CLUT（R = G = B）即可实现灰度。
为什么显示不佳（显示不好的视觉效果）？	很多因素都会导致视觉效果不良，用户可以进行以下检查： – 检查所用显示器是否被正确初始化/配置（某些显示器需要初始化/配置时序） – 检查LTDC时序和图层参数是否设置正确（参见第 6.2.4 节中示例） – 直接从内部闪存显示图像（参见第 6.2.5 节中示例） – 检查LTDC和帧缓冲器更新（利用DMA2D或CPU）之间是否存在非同步，参见第 4.4.2 节。



9 结论

STM32MCU提供了非常灵活的显示控制器，可以以较低成本与各种显示器连接，并提供高性能。

由于其智能架构的集成功能，LTDC可以自动从帧缓冲器获取图形数据并将其驱动到显示屏，无需任何CPU干预。

LTDC能够在CPU处于SLEEP模式时继续获取图形数据并驱动显示器，这使其非常适用于智能手表等低功耗和移动应用。

本应用笔记介绍了STM32的图形功能，并提出了一些注意事项和建议以充分利用该系统的智能架构。

10 版本历史

表19. 文档版本历史

日期	版本	变更
2017年2月10日	1	初始版本。
2017年2月10日	2	SDRAM和Quad-SPI MPU配置示例 上更新了代码

表20. 中文文档版本历史

日期	版本	变更
2018年8月10日	1	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。

© 2018 STMicroelectronics - 保留所有权利