



GR5525刷屏指南

版本： 1.0

发布日期： 2023-08-30

版权所有 © 2023 深圳市汇顶科技股份有限公司。保留一切权利。

非经本公司书面许可，任何单位和个人不得对本手册内的任何部分擅自摘抄、复制、修改、翻译、传播，或将其全部或部分用于商业用途。

商标声明

GOODIX 和其他汇顶商标均为深圳市汇顶科技股份有限公司的商标。本文档提及的其他所有商标或注册商标，由各自的所有人持有。

免责声明

本文档中所述的器件应用信息及其他类似内容仅为您提供便利，它们可能由更新之信息所替代。确保应用符合技术规范，是您自身应负的责任。

深圳市汇顶科技股份有限公司（以下简称“GOODIX”）对这些信息不作任何明示或暗示、书面或口头、法定或其他形式的声明或担保，包括但不限于针对其使用情况、质量、性能、适销性或特定用途的适用性的声明或担保。GOODIX对因这些信息及使用这些信息而引起的后果不承担任何责任。

未经GOODIX书面批准，不得将GOODIX的产品用作生命维持系统中的关键组件。在GOODIX知识产权保护下，不得暗或以其他方式转让任何许可证。

深圳市汇顶科技股份有限公司

总部地址：深圳市福田区腾飞工业大厦B座12-13层

电话：+86-755-33338828 邮编：518000

网址：www.goodix.com

前言

编写目的

本文介绍GR5525芯片用于穿戴设备的刷屏模块，有助于用户快速理解该模块的功能及特性，加速穿戴类产品的开发和性能优化。

读者对象

本文适用于以下读者：

- GR5525用户
- 开发人员
- 测试人员
- 开发爱好者
- 文档工程师

版本说明

本文档为第1次发布，对应的产品系列为GR5525。

修订记录

版本	日期	修订内容
1.0	2023-08-30	首次发布

目录

前言.....	I
1 概述.....	1
1.1 刷屏要素.....	1
1.2 刷屏模型.....	2
2 DMA典型特性.....	4
2.1 DMA 典型应用.....	4
2.1.1 DMA基础数据传输.....	4
2.1.2 DMA链式数据传输.....	4
2.1.3 DMA分散传输.....	5
2.1.4 DMA聚合传输.....	5
2.1.5 DMA同时进行分散/聚合传输.....	6
2.2 DMA 传输通道特点.....	6
3 QSPI典型特性.....	7
3.1 QSPI工作模式.....	7
3.2 QSPI数据端序.....	8
3.2.1 写操作下的数据端序.....	8
3.2.2 读操作下的数据端序.....	9
3.2.3 内存映射模式下静态数据端序读取规则.....	10
3.2.4 内存映射模式下动态数据端序读取规则.....	11
3.3 QSPI驱动使用说明.....	13
3.3.1 公共函数.....	13
3.3.1.1 app_qspi_init.....	13
3.3.1.2 app_qspi_deinit.....	15
3.3.1.3 app_qspi_get_handle.....	15
3.3.1.4 app_qspi_dma_init.....	15
3.3.1.5 app_qspi_dma_deinit.....	15
3.3.2 寄存器模式类接口.....	16
3.3.2.1 app_qspi_command_sync.....	16
3.3.2.2 app_qspi_command_async.....	16
3.3.2.3 app_qspi_dma_command_async.....	16
3.3.2.4 app_qspi_command_receive_sync.....	17
3.3.2.5 app_qspi_command_receive_async.....	17
3.3.2.6 app_qspi_dma_command_receive_async.....	17
3.3.2.7 app_qspi_command_transmit_sync.....	17
3.3.2.8 app_qspi_command_transmit_async.....	18
3.3.2.9 app_qspi_dma_command_transmit_async.....	18
3.3.2.10 app_qspi_transmit_sync_ex.....	18
3.3.2.11 app_qspi_transmit_async_ex.....	19

3.3.2.12 app_qspi_dma_transmit_async_ex.....	19
3.3.2.13 app_qspi_receive_sync_ex.....	20
3.3.2.14 app_qspi_receive_async_ex.....	20
3.3.2.15 app_qspi_dma_receive_async_ex.....	20
3.3.3 内存映射模式类接口.....	21
3.3.3.1 app_qspi_config_memory_mapped.....	21
3.3.3.2 app_qspi_active_memory_mapped.....	22
3.3.3.3 app_qspi_get_xip_base_address.....	22
3.3.3.4 app_qspi_mmap_set_endian_mode.....	22
3.3.4 特种功能接口.....	23
3.3.4.1 app_qspi_async_draw_screen.....	23
3.3.4.2 app_qspi_async_veri_draw_screen.....	23
3.3.4.3 app_qspi_mmap_blit_image.....	24
3.3.4.4 app_qspi_async_llp_draw_block.....	24
3.4 QSPI使用建议.....	25
4 显示控制和渲染逻辑.....	26
4.1 显示屏时序.....	26
4.2 色彩格式支持.....	27
4.2.1 色彩和色深.....	27
4.2.2 Alpha混合.....	28
4.3 帧缓冲区.....	28
4.3.1 基础概念.....	28
4.3.2 帧缓冲区数量.....	29
4.3.3 参考设计.....	30
4.3.3.1 GUI框架.....	30
4.3.3.2 穿戴示例（Watch Demo）工程.....	31

1 概述

GR5525系列芯片提供QFN68和QFN56两种封装，具体的芯片配置如下表所示：

表 1-1 GR5525系列芯片配置

产品型号	GR5525RGNI	GR5525IGNI	GR5525IENI	GR5525IONI
CPU	Cortex [®] -M4F	Cortex [®] -M4F	Cortex [®] -M4F	Cortex [®] -M4F
RAM	256 KB	256 KB	256 KB	256 KB
SiP Flash	1 MB	1 MB	512 KB	N/A
I/O数量	50	39	39	39
I/O电压	1.8 V ~ 3.3 V	1.8 V ~ 3.3 V	1.8 V ~ 3.3 V	跟随外部Flash
封装 (mm)	QFN68 (7.0 x 7.0 x 0.85)	QFN56 (7.0 x 7.0 x 0.75)	QFN56 (7.0 x 7.0 x 0.75)	QFN56 (7.0 x 7.0 x 0.75)

1.1 刷屏要素

GR5525为穿戴类产品提供了丰富的刷屏要素，如下表所示。

表 1-2 GR5525刷屏要素概览1

系统主频	X-Flash	外设QSPI频率	SRAM	QSPI0	QSPI1
96 MHz	512 KB/1 MB或外接Flash	48 MHz	256 KB	• NOR Flash（XIP读） • Nand-Flash（寄存器模式访问） • Q-PSRAM（XIP读写） • Display • DMA0（寄存器模式） • DMA0/DMA1（XIP模式） • 动态自适应端序 • 多种静态端序 • 1/2/4线模式	• NOR Flash（XIP读） • Nand-Flash（寄存器模式访问） • Q-PSRAM（XIP读写） • Display • DMA0/DMA1（寄存器模式） • DMA0/DMA1（XIP模式） • 动态自适应端序 • 多种静态端序 • 1/2/4线模式

表 1-3 GR5525刷屏要素概览2

QSPI2	DMA	SPIM	DSPI
• NOR Flash（XIP读） • Nand-Flash（寄存器模式访问） • Q-PSRAM（XIP读写） • Display • DMA1（寄存器模式） • DMA0/DMA1（XIP模式） • 动态自适应端序 • 多种静态端序 • 1/2/4线模式	• 2个实例 • Linked List • Scatter • Gather • 高FIFO	48 MHz	48 MHz

主要的刷屏要素分类如下：

1. 算力提供
 - (1) CPU
2. 素材/纹理数据持久化存储
 - (1) NOR Flash
 - (2) Nand Flash
 - (3) XQSPI Flash（代码空间剩余部分，可部分用作数据存储空间）
3. 计算及缓存空间
 - (1) 256 KB片内SRAM
 - (2) 最大外扩512 Mbit QSPI PSRAM 存储空间
4. 数据传输增强/帧率加速
 - (1) DMA0/DMA1
5. 显示接口（涵盖MIPI DBI Type-C接口）
 - (1) QSPI接口
 - (2) SPI Master接口
 - (3) Display SPI接口（3-wire/4-wire SPI）

1.2 刷屏模型

根据不同穿戴产品的市场定位，可以使用不同刷屏要素的排列组合构建出丰富的刷屏模型。

本文将重点介绍DMA、QSPI等模块的典型特征和用法。各模块的综合介绍，可参考《GR5525 Datasheet》。

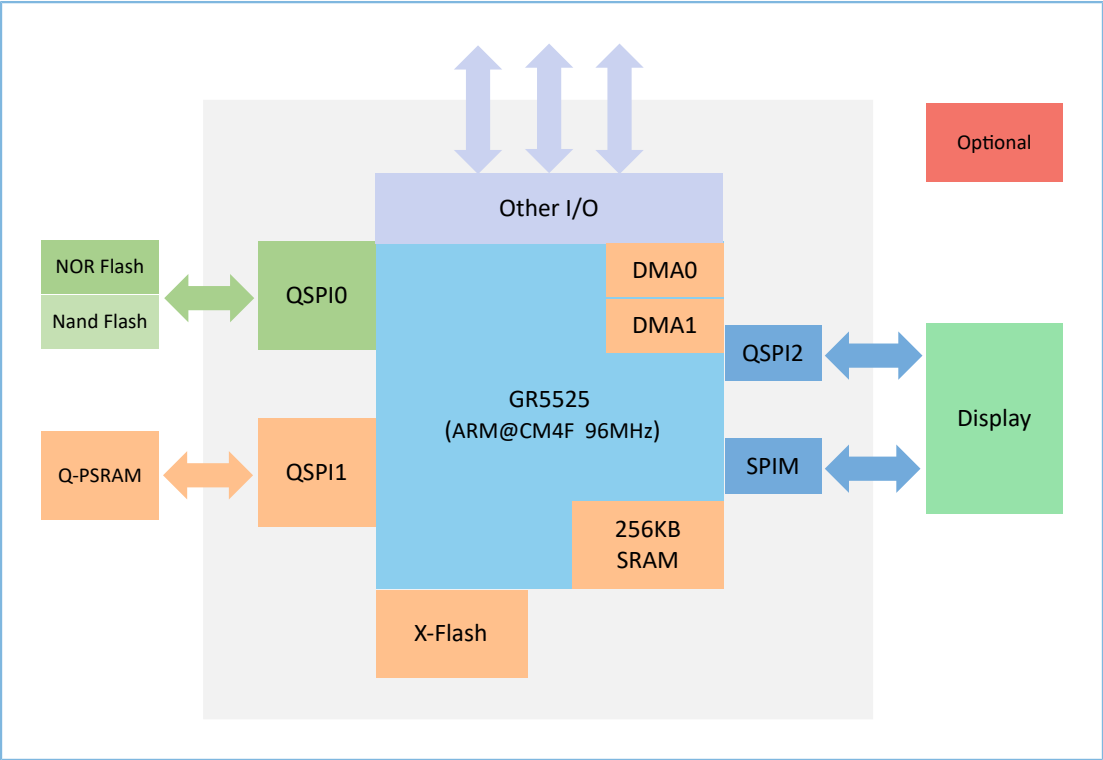


图 1-1 GR5525典型刷屏模型框图

2 DMA典型特性

DMA（Direct Memory Access，直接存储器访问）的通用使用介绍，请参考《GR5525 Datasheet》“DMA”章节。

2.1 DMA 典型应用

2.1.1 DMA基础数据传输

DMA可用于Memory至Memory、Memory至外设、外设至Memory、外设至外设间的数据传输。

当DMA进行基础的数据传输时，一次性最大传输4095拍数据。“拍”指DMA传输数据的位宽度。

- 如果使用的数据位宽是字节，DMA单次最多传输数据4095字节
- 如果使用的数据位宽是半字，DMA单次最多传输数据4095*2字节
- 如果使用的数据位宽是字，DMA单次最多传输数据4095*4字节

如果DMA一次基础传输时，数据量超过了4095拍，则会给出错误的DMA传输中断提示。

 说明:

DMA传输地址需要注意地址对齐。

2.1.2 DMA链式数据传输

为增强DMA以下场景的传输能力，GR5525 SoC扩展了DMA链式数据传输能力，即可以使用指针链表的方式，将多个数据块进行连接，在一次DMA传输周期内进行传输。

- 突破单次传输4095拍的限制
- 在单次传输周期中传输不连续地址空间的数据
- 在单次传输周期使用不同的传输配置进行数据传输

具体的传输形式为：将需要传输的所有数据区块，通过指针链表方式进行管理。上一个区块发送完成后，根据Next链表指针自动对下一个区块进行加载，直到最后Next链表指针为空。

如下图所示，每一个链节点主要包含当前节点的DMA传输配置信息、节点数据块、以及指向下一个链节点的指针信息，直到最后一个传输链节点为NULL。这些链节点会在一次DMA传输中完成，需要每个节点遵守数据量不大于4095拍的规则即可。

DMA链式数据传输可以广泛用在大数据块传输、不连续数据传输、刷屏等应用场景。

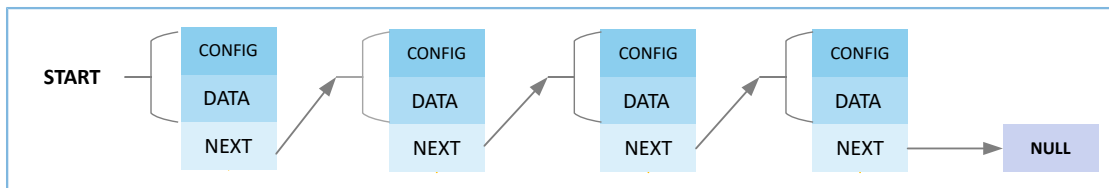


图 2-1 DMA链式数据传输

2.1.3 DMA分散传输

DMA分散（Scatter）传输指将连续地址空间的数据通过一定的规则，分散到不连续的地址空间的传输。

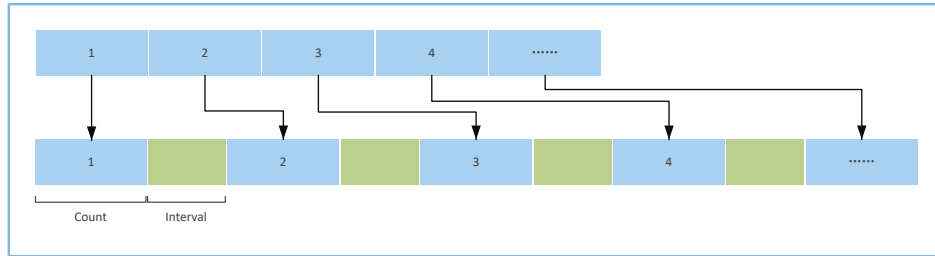


图 2-2 DMA分散传输示意图

1. 连续数据被等分为若干数据块，最后一块数据可以不是等分数
2. 每块数据包含的数据数，标记为Count，单位为拍
3. 数据块之间的间隔地址数，标记为Interval，单位为拍

举例：拟使用DMA Scatter传输1000字节数据，Count为4，Interval为2

- 如果传输宽度为8-bit，则数据总拍数为1000拍，每块数据大小为4字节，分散间隔地址为2字节
- 如果传输宽度为16-bit，则数据总拍数为500拍，每块数据大小为8字节，分散间隔地址为4字节
- 如果传输宽度为32-bit，则数据总拍数为200拍，每块数据大小为16字节，分散间隔地址为8字节

2.1.4 DMA聚合传输

DMA聚合（Gather）传输指将不连续地址空间的数据，聚合到连续的地址空间的传输。可视作Scatter的逆向传输过程。

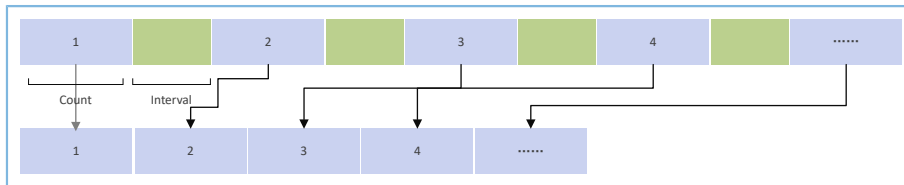


图 2-3 DMA聚合传输示意图

1. 若干个等长的数据块，按照相同的地址间隔均匀排布（最后一块数据长度可以和前面的不等）
2. 每块数据包含的数据数，标记为Count，单位为拍
3. 数据块之间的间隔地址数，标记为Interval，单位为拍

举例：拟使用DMA Gather 聚合传输1000字节数据，Count为4，Interval为2

- 如果传输宽度为8-bit，则数据总拍数为1000拍，每块数据大小为4字节，块数据的间隔地址为2字节
- 如果传输宽度为16-bit，则数据总拍数为500拍，每块数据大小为8字节，块数据的间隔地址为4字节
- 如果传输宽度为32-bit，则数据总拍数为200拍，每块数据大小为16字节，块数据的间隔地址为8字节

2.1.5 DMA同时进行分散/聚合传输

DMA可以同时启用分散（Scatter）/聚合（Gather）传输功能，将不连续地址空间的数据，传输到另一个不连续的地址空间。

2.2 DMA 传输通道特点

GR5525提供DMA0和DMA1两个可供用户使用的DMA实例，共计6个DMA通道，其中通道0 FIFO深度为32，通道1 ~ 5 的FIFO深度为4。

由于通道0具备很深的FIFO深度，在DMA传输时候能提供极深的数据缓存，从而提高DMA传输吞吐，因此在穿戴类应用中，请优先将DMA0/DMA1的通道0分配给高吞吐外设使用，如用于Flash访问、PSRAM访问、大块Memory搬运、Display显示等。

3 QSPI典型特性

QSPI的通用介绍，请参考《GR5525 Datasheet》“QSPI”章节。

3.1 QSPI工作模式

根据工作时数据线的参与数量和时序行为，GR5525芯片的QSPI分为3种工作模式：

- 1-Line SPI 模式：工作时使用SCLK、CS#、MOSI、MISO信号
- 2-Line DualSPI 模式：工作时使用SCLK、CS#、IO0、IO1信号
- 4-Line QuadSPI模式：工作时使用SCLK、CS#、IO0、IO1、IO2、IO3信号

根据外设访问方式的不同，GR552 芯片的QSPI分为2种工作模式：

- 寄存器模式（Register Mode）：在完成QSPI模块及外设的初始化后，QSPI对外设的读写访问通过操作QSPI控制寄存器进行，这些操作在SDK中按照功能封装成函数，而具体的访问行为基于调用具体功能函数（组）的形式进行
- 内存映射模式（Memory Mapped Mode）：又称XIP模式，在完成QSPI模块及外设的初始化时，进一步将外设的内存空间映射到系统总线地址空间，通过总线寻址的方式进行外设的访问

一般而言，任何支持QSPI时序协议的外设，均可以通过寄存器模式进行读写访问，如QSPI接口的NOR Flash设备、Nand Flash设备、PSRAM设备、Display设备及其他支持QSPI时序的外设。

QSPI NOR Flash、QSPI PSRAM存储设备，除寄存器模式外，还可以通过内存映射模式进行访问。其中，由于不同的访问特征，QSPI NOR Flash只支持通过内存映射模式进行读操作；而QSPI PSRAM 设备可以支持内存映射模式的读和写操作。

说明:

一般支持内存映射访问模式的存储设备，只能在DualSPI/QuadSPI 模式下工作，且为了获得更好的访问效率，建议尽量在QuadSPI模式下工作。

下表梳理了一般穿戴外设可支持的访问模式，相关建议如下：

1. 实际外设的支持情况需要根据具体选用外设的数据手册进行确认
2. 只要外设的访问方式支持内存映射访问模式，建议优先采用这种模式，让读取存储设备的程序代码更简洁、访问效率更高效

表 3-1 GR5525 QSPI对穿戴类外设支持工作模式查询表

工作模式		NOR Flash		Nand Flash		QSPI PSRAM		Display/LCD	
		Read	Write	Read	Write	Read	Write	Read	Write
寄存器 模式	SPI	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	DualSPI	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	QuadSPI	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

工作模式		NOR Flash		Nand Flash		QSPI PSRAM		Display/LCD	
		Read	Write	Read	Write	Read	Write	Read	Write
内存映射模式	SPI	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	DualSPI	Yes	N/A	N/A	N/A	Yes	Yes	N/A	N/A
	QuadSPI	Yes	N/A	N/A	N/A	Yes	Yes	N/A	N/A

3.2 QSPI数据端序

3.2.1 写操作下的数据端序

使用CPU或DMA将相同的内存数据，经不同总线访问宽度通过QSPI写入外设时，会得到不同的字节端序，如下图所示。

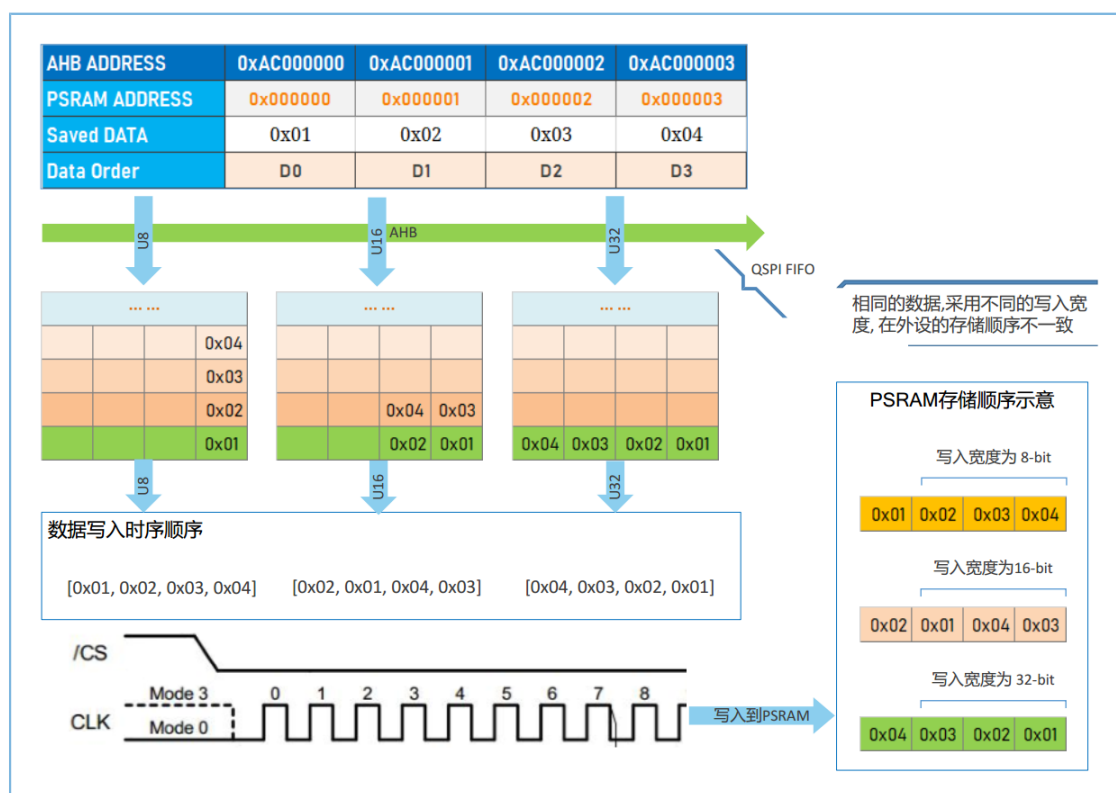


图 3-1 Synopsys QSPI 控制器默认写过程演示

端序转换过程如下：

1. 将数组{0x01, 0x02, 0x03, 0x04}通过QSPI向外发送。
2. 进入QSPI FIFO队列时：
 - (1) 如使用字节类型写入，则占用4个FIFO深度，0x01,0x02,0x03,0x04依次排列
 - (2) 如使用半字类型写入，数据在总线上转换为0x0201和0x0403，进入QSPI占用2个FIFO深度，{0x02,0x01} 和 {0x04,0x03} 依次排列

- (3) 如使用字类型写入，数据在总线上转换为0x04030201，进入QSPI占用1个FIFO深度，为{0x04, 0x03, 0x02, 0x01}

3. QSPI FIFO采用高字节到低字节的方式向数据线输出数据，则进入数据线的数据：

- (1) 字节类型写入时，时序线中的数据显示为 0x01,0x02,0x03,0x04
- (2) 半字类型写入时，时序线中的数据显示为 0x02,0x01,0x04,0x03
- (3) 字类型写入时，时序线中看到的数据显示为 0x04,0x03,0x02,0x01

4. 外设按照不同的数据端序进行存储或处理。

说明:

默认情况下，寄存器模式和内存映射模式的数据写入操作均遵守以上端序转换行为。

3.2.2 读操作下的数据端序

相对于写入操作，读操作是其逆向过程，针对位于外设地址空间相同的数据，使用不同的读访问宽度，会得到不同字节端序的数据，如下图所示。

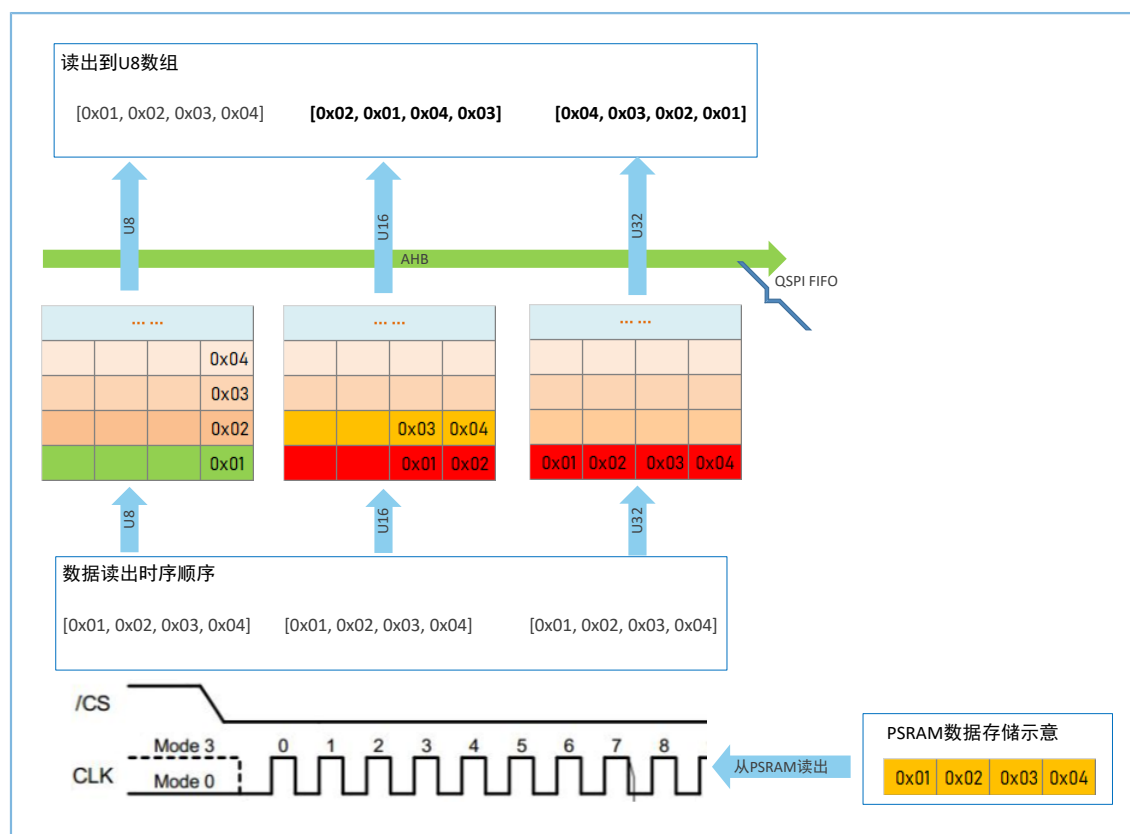


图 3-2 读操作下的数据端序

端序转换过程如下：

- 使用字节宽度，读取外设空间的{0x01,0x02,0x03,0x04}至内存空间，结果为{0x01,0x02,0x03,0x04}
- 使用半字宽度，读取外设空间的{0x01,0x02,0x03,0x04}至内存空间，结果为{0x02,0x01,0x04,0x03}
- 使用字宽度，读取外设空间的{0x01,0x02,0x03,0x04}至内存空间，结果为{0x04,0x03,0x02,0x01}

默认情况下，寄存器模式和内存映射模式的数据读操作均遵守以上端序转换行为，为了获取相同的字节端序，读写操作的访问宽度需要保持相同。

然而，在穿戴产品应用场景中，图片、字体等资源，由于颜色格式及色深的不同，其最小的访问单位可能会是Byte、Half Word、Word及其组合，因此在软件访问过程中，会涉及到字节端序的处理问题，如果通过软件进行端序调整则会消耗大量的CPU算力。GR5525从IC层设计了支持不同访问场景的端序模式。

3.2.3 内存映射模式下静态数据端序读取规则

静态数据端序是根据不同的寄存器配置，输出对应的数据端序（此规则下文简称为静态端序规则）。

- 静态端序规则只适用于QSPI工作在内存映射模式
- 静态端序规则的应用场景为QSPI NOR Flash设备数据的读取

在静态端序规则下，对固定的数据存储顺序，可以通过寄存器配置来实现不同的数据读出端序，下表描述了存储于外设存储设备的数据。

表 3-2 待读取外设数据

外设地址	0x000000	0x000001	0x000002	0x000003
存储数据	0x01	0x02	0x03	0x04

在外设地址空间0x000000 ~ 0x000003依次存储了数据0x01、0x02、0x03、0x04。

下表描述了在不同端序规则下，使用不同的数据类型访问上述外设数据时，实际获得的数据。

表 3-3 QSPI读操作的静态端序规则

访问类型	Byte				Half Word		Word
访问地址	0xAC000000	0xAC000001	0xAC000002	0xAC000003	0xAC000000	0xAC000002	0xAC000000
端序模式0	0x01	0x02	0x03	0x04	0x0102	0x0304	0x01020304
端序模式1	0x01	0x02	0x03	0x04	0x0201	0x0403	0x02010403
端序模式2	0x01	0x02	0x03	0x04	0x0201	0x0403	0x04030201

静态端序规则说明如下：

- 假定外设地址0x000000 ~ 0x000003在总线地址空间的映射地址为0xAC000000 ~ 0xAC000003。
- 当配置为静态端序模式0时：
 - 使用Byte（对应C语言的 uint8_t*）操作访问0xAC000000时，返回为0x01
 - 使用Half Word（对应C语言的 uint16_t*）操作访问0xAC000000时，返回为0x0102
 - 使用Word（对应C语言的 uint32_t*）操作访问0xAC000000时，返回为0x01020304

- 当配置为静态端序模式1时：
 - 使用Byte（对应C语言的 `uint8_t*`）操作访问0xAC000000时，返回为0x01
 - 使用Half Word（对应C语言的 `uint16_t*`）操作访问0xAC000000时，返回为0x0201
 - 使用Word（对应C语言的 `uint32_t*`）操作访问0xAC000000时，返回为0x02010403
- 当配置为静态端序模式2时：
 - 使用Byte（对应C语言的 `uint8_t*`）操作访问0xAC000000时，返回为0x01
 - 使用Half Word（对应C语言的 `uint16_t*`）操作访问0xAC000000时，返回为0x0201
 - 使用Word（对应C语言的 `uint32_t*`）操作访问0xAC000000时，返回为0x04030201

可根据实际情况，设置合适的端序规则使QSPI在最佳效率下工作。

设置静态端序规则的函数：`app_qspi_mmap_set_endian_mode`（详细描述参考[3.3.3.4 app_qspi_mmap_set_endian_mode](#)章节）。

3.2.4 内存映射模式下动态数据端序读取规则

动态数据端序是在内存映射模式下根据不同的数据访问类型，输出相应端序的数据读取规则（此规则下文简称为动态端序规则）。

- 动态端序规则只适用于QSPI工作在内存映射模式
- 动态端序规则典型的应用场景为QSPI PSRAM设备数据的读写
- 动态端序规则优先级高于静态端序规则，如果同时启用了动态/静态端序规则，系统只会响应动态端序规则

在动态端序规则下，发生读写访问时，可通过修改数据进入FIFO的行为，自动适应不同数据类型的混合访问。

下图演示了启用动态端序规则后，执行读写操作时不同数据类型进入QSPI FIFO的行为。动态端序规则帮助保持不同基础数据类型的写入一致性。

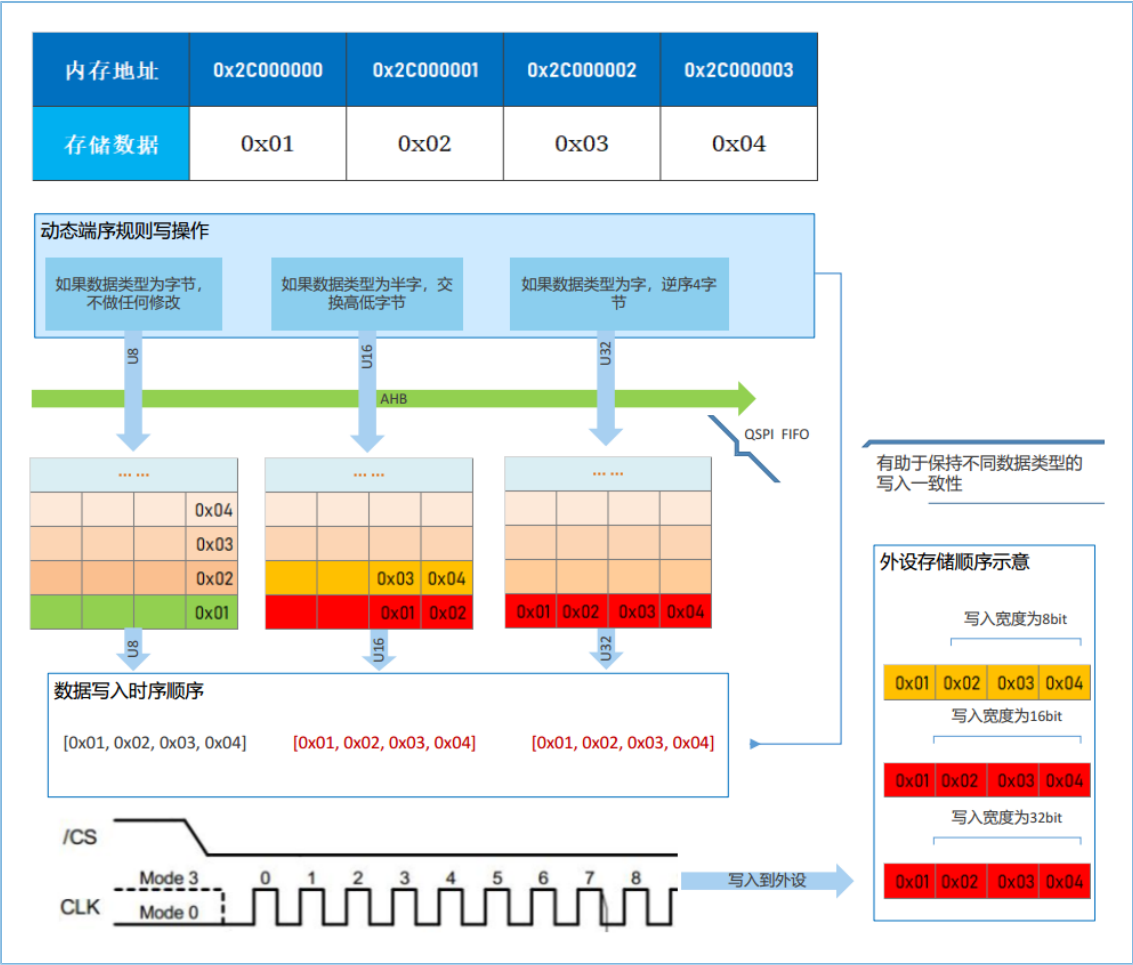


图 3-3 写操作下不同数据类型进入QSPI FIFO的行为

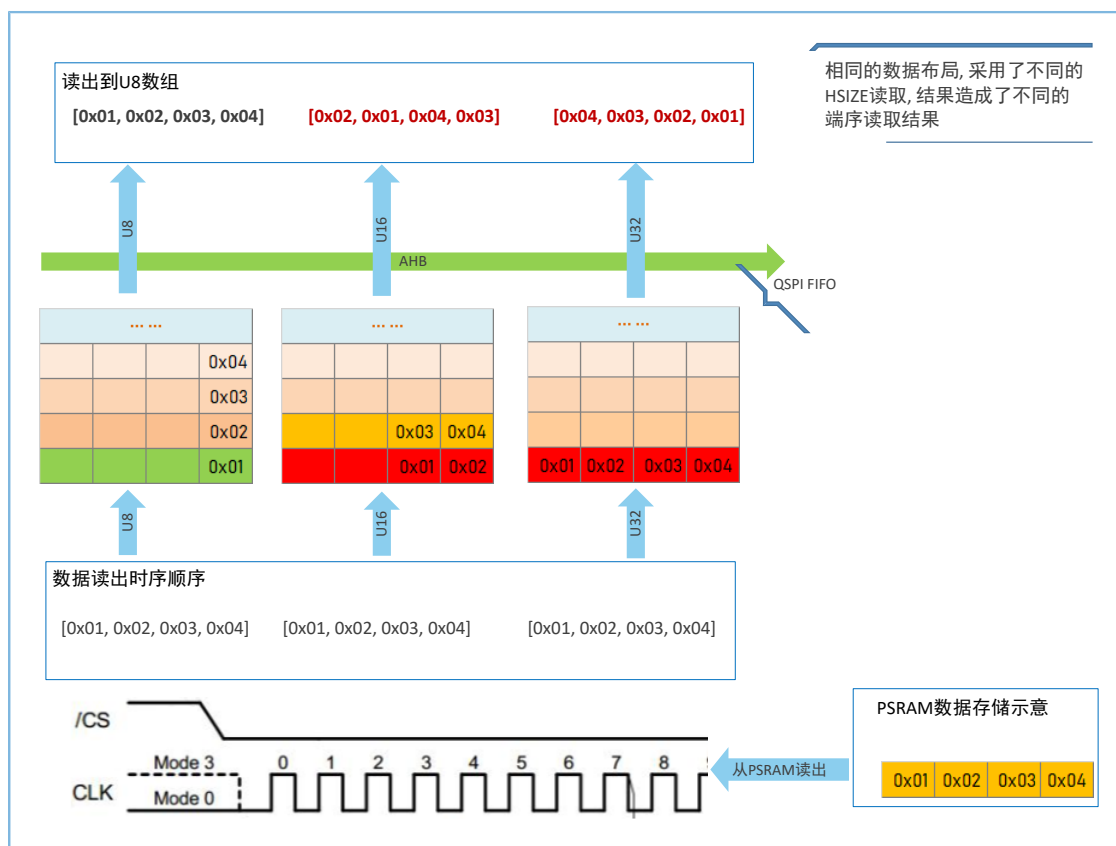


图 3-4 读操作下不同数据类型进入QSPI FIFO的行为

在QSPI PSRAM 设备进行内存映射模式初始化时，GR5525驱动会自动使能动态端序规则，使PSRAM 的访问行为和SRAM访问行为一致。

3.3 QSPI驱动使用说明

3.3.1 公共函数

3.3.1.1 app_qspi_init

表 3-4 app_qspi_init接口

函数原型	uint16_t app_qspi_init(app_qspi_params_t *p_params, app_qspi_evt_handler_t evt_handler)
功能说明	用于QSPI模块的初始化配置
输入参数	- p_params: 初始化参数配置 - evt_handler: 注册用于异步接口回调事件的通知函数
返回值	参考源码APP_DRV_ERR_CODE
备注	

结构体app_qspi_params_t定义及注解:

```
typedef struct
{
    app_qspi_id_t    id;
```

```

    app_qspi_pin_cfg_t    pin_cfg;
    app_qspi_dma_cfg_t    dma_cfg;
    qspi_init_t           init;
} app_qspi_params_t;

```

id

- APP_QSPI_ID_0: QSPI0
- APP_QSPI_ID_1: QSPI1
- APP_QSPI_ID_2: QSPI2

pin_cfg

- cs: CS片选脚配置
 - type: 引脚类型
 - mux: 引脚复用功能配置
 - pin: 引脚号
 - mode: 功能模式
 - pull: 上下拉模式
 - enable: 使能此引脚
- clk: Clock脚配置
- io_0: QSPI IO0 (SPI MOSI)
- io_1: QSPI IO1 (SPI MISO)
- io_2: QSPI IO2
- io_3: QSPI IO3

驱动默认提供三组QSPI，IO默认配置定义（g_qspi_pin_groups），一般情况下可以根据需要直接引用，也可以根据需要自行定义。

dma_cfg

- dma_instance: 如果配置DMA模式，需要指定DMA实例，QSPI0支持DMA0，QSPI1支持DMA0/DMA1，QSPI2支持DMA2
- dma_channel: 如果配置DMA模式，需要指定DMA实例的通道
- wait_timeout_ms: 用于轮询接口的超时等待
- extend: 待扩展参数，暂未使用

init

- clock_prescaler: QSPI 时钟分频系数，支持2 ~ 65535之间的偶数，时钟基准为系统外设时钟
- clock_mode: 由时钟采样边沿和相位构成的4种时钟模式

- rx_sample_delay: 接收采样延迟，单位为传输时钟的周期。采用最大频率时，一般设置为1，表示延时一个时钟周期，其他频率设置为0。

3.3.1.2 app_qspi_deinit

表 3-5 app_qspi_deinit接口

函数原型	uint16_t app_qspi_deinit(app_qspi_id_t id)
功能说明	用于QSPI模块的反初始化
输入参数	id: QSPI模块ID
返回值	参考源码APP_DRV_ERR_CODE
备注	

3.3.1.3 app_qspi_get_handle

表 3-6 app_qspi_get_handle接口

函数原型	qspi_handle_t *app_qspi_get_handle(app_qspi_id_t id)
功能说明	根据ID获取QSPI控制句柄
输入参数	id: QSPI模块ID
返回值	QSPI控制句柄指针
备注	

3.3.1.4 app_qspi_dma_init

表 3-7 app_qspi_dma_init接口

函数原型	uint16_t app_qspi_dma_init(app_qspi_params_t *p_params)
功能说明	初始化QSPI DMA模式
输入参数	p_params: 初始化参数的结构体指针
返回值	APP_DRV_xxx: 详见SDK_Folder\drivers\inc\app_drv_error.h宏定义
备注	

3.3.1.5 app_qspi_dma_deinit

表 3-8 app_qspi_dma_deinit接口

函数原型	uint16_t app_qspi_dma_deinit(app_qspi_id_t id)
功能说明	反初始化QSPI DMA模式
输入参数	id: QSPI模块ID
返回值	APP_DRV_xxx: 详见SDK_Folder\drivers\inc\app_drv_error.h宏定义
备注	

3.3.2 寄存器模式类接口

此类接口主要在QSPI寄存器模式下作为通用功能接口，进行各类QSPI外设的访问。

- 根据使用环境，接口分为带Command和不带Command控制的两大类接口。
- 根据数据传输方式，分为轮询方式传输接口、中断方式传输接口和DMA方式传输接口，分别为app_xxx_transmit_sync、app_xxx_transmit_async和app_xxx_dma_transmit_async。

3.3.2.1 app_qspi_command_sync

表 3-9 app_qspi_command_sync接口

函数原型	uint16_t app_qspi_command_sync(app_qspi_id_t id, app_qspi_command_t *p_cmd, uint32_t timeout)
功能说明	QSPI轮询方式（同步）发送命令
输入参数	• id: QSPI模块ID • p_cmd: 待发送命令的buffer • timeout: 超时时间，以ms为单位
返回值	APP_DRV_xxx: 详见SDK_Folder\drivers\inc\app_drv_error.h宏定义
备注	

3.3.2.2 app_qspi_command_async

表 3-10 app_qspi_command_async接口

函数原型	uint16_t app_qspi_command_async(app_qspi_id_t id, app_qspi_command_t *p_cmd)
功能说明	QSPI中断方式（异步）发送命令
输入参数	• id: QSPI模块ID • p_cmd: 待发送命令的buffer
返回值	APP_DRV_xxx: 详见SDK_Folder\drivers\inc\app_drv_error.h宏定义
备注	

3.3.2.3 app_qspi_dma_command_async

表 3-11 app_qspi_dma_command_async接口

函数原型	uint16_t app_qspi_dma_command_async(app_qspi_id_t id, app_qspi_command_t *p_cmd)
功能说明	QSPI以DMA方式（异步）发送命令
输入参数	• id: QSPI模块ID • p_cmd: 待发送命令的buffer
返回值	APP_DRV_xxx: 详见SDK_Folder\drivers\inc\app_drv_error.h宏定义
备注	

3.3.2.4 app_qspi_command_receive_sync

表 3-12 app_qspi_command_receive_sync接口

函数原型	uint16_t app_qspi_command_receive_sync(app_qspi_id_t id, app_qspi_command_t *p_cmd, uint8_t *p_data, uint32_t timeout)
功能说明	QSPI轮询方式（同步）读取数据，带控制命令封装
输入参数	- id: QSPI模块ID - p_cmd: 封装的控制命令 - p_data: 读取数据存放的buffer - timeout: 超时时间，以ms为单位
返回值	APP_DRV_xxx: 详见SDK_Folder\drivers\inc\app_drv_error.h宏定义
备注	

3.3.2.5 app_qspi_command_receive_async

表 3-13 app_qspi_command_receive_async接口

函数原型	uint16_t app_qspi_command_receive_async(app_qspi_id_t id, app_qspi_command_t *p_cmd, uint8_t *p_data)
功能说明	QSPI中断方式（异步）读取数据，带控制命令封装
输入参数	- id: QSPI模块ID - p_cmd: 封装的控制命令 - p_data: 读取数据存放的buffer
返回值	APP_DRV_xxx: 详见SDK_Folder\drivers\inc\app_drv_error.h宏定义
备注	

3.3.2.6 app_qspi_dma_command_receive_async

表 3-14 app_qspi_dma_command_receive_async接口

函数原型	uint16_t app_qspi_dma_command_receive_async(app_qspi_id_t id, app_qspi_command_t *p_cmd, uint8_t *p_data)
功能说明	QSPI以DMA方式（异步）读取数据，带控制命令封装
输入参数	- id: QSPI模块ID - p_cmd: 封装的控制命令 - p_data: 读取数据存放的buffer
返回值	APP_DRV_xxx: 详见SDK_Folder\drivers\inc\app_drv_error.h宏定义
备注	

3.3.2.7 app_qspi_command_transmit_sync

表 3-15 app_qspi_command_transmit_sync接口

函数原型	uint16_t app_qspi_command_transmit_sync(app_qspi_id_t id, app_qspi_command_t *p_cmd, uint8_t *p_data, uint32_t timeout)
------	---

功能说明	QSPI轮询方式（同步）发送数据，带控制命令封装
输入参数	• id: QSPI模块ID • p_cmd: 封装的控制命令 • p_data: 待发送数据存放的buffer • timeout: 超时时间，以ms为单位
返回值	APP_DRV_xxx: 详见SDK_Folder\drivers\inc\app_drv_error.h宏定义
备注	

3.3.2.8 app_qspi_command_transmit_async

表 3-16 app_qspi_command_transmit_async接口

函数原型	uint16_t app_qspi_command_transmit_async(app_qspi_id_t id, app_qspi_command_t *p_cmd, uint8_t *p_data)
功能说明	QSPI中断方式（异步）发送数据，带控制命令封装
输入参数	• id: QSPI模块ID • p_cmd: 封装的控制命令 • p_data: 待发送数据存放的buffer
返回值	APP_DRV_xxx: 详见SDK_Folder\drivers\inc\app_drv_error.h宏定义
备注	

3.3.2.9 app_qspi_dma_command_transmit_async

表 3-17 app_qspi_dma_command_transmit_async接口

函数原型	uint16_t app_qspi_dma_command_transmit_async(app_qspi_id_t id, app_qspi_command_t *p_cmd, uint8_t *p_data)
功能说明	QSPI以DMA方式（异步）发送数据，带控制命令封装
输入参数	• id: QSPI模块ID • p_cmd: 封装的控制命令 • p_data: 待发送数据存放的buffer
返回值	APP_DRV_xxx: 详见SDK_Folder\drivers\inc\app_drv_error.h宏定义
备注	

3.3.2.10 app_qspi_transmit_sync_ex

表 3-18 app_qspi_transmit_sync_ex接口

函数原型	uint16_t app_qspi_transmit_sync_ex(app_qspi_id_t id, uint32_t qspi_mode, uint32_t data_width, uint8_t *p_data, uint32_t length, uint32_t timeout)
功能说明	QSPI轮询方式（同步）发送数据，可设置时序模式和数据位宽
输入参数	• id: QSPI模块ID • qspi_mode: 数据传输采用的时序模式，可选QSPI_DATA_MODE_SPI（标准SPI模式）、QSPI_DATA_MODE_DUALSPI（Dual SPI模式）、QSPI_DATA_MODE_QUADSPI（Quad SPI模式）

	• data_width: 数据位宽（支持QSPI_DATASIZE_08_BITS、QSPI_DATASIZE_16_BITS、QSPI_DATASIZE_32_BITS） • p_data: 需要传输数据的buffer • length: 待发送数据的长度，以Byte为单位 • timeout: 超时时间，以ms为单位
返回值	APP_DRV_xxx: 详见SDK_Folder\drivers\inc\app_drv_error.h宏定义
备注	

3.3.2.11 app_qspi_transmit_async_ex

表 3-19 app_qspi_transmit_async_ex接口

函数原型	uint16_t app_qspi_transmit_async_ex(app_qspi_id_t id, uint32_t qspi_mode, uint32_t data_width, uint8_t *p_data, uint32_t length)
功能说明	QSPI中断方式（异步）发送数据，可设置时序模式和数据位宽
输入参数	• id: QSPI模块ID • qspi_mode: 数据传输采用的时序模式，可选QSPI_DATA_MODE_SPI（标准SPI模式）、QSPI_DATA_MODE_DUALSPI（Dual SPI模式）、QSPI_DATA_MODE_QUADSPI（Quad SPI模式） • data_width: 数据位宽（支持QSPI_DATASIZE_08_BITS、QSPI_DATASIZE_16_BITS、QSPI_DATASIZE_32_BITS） • p_data: 需要发送数据的buffer • length: 待发送数据的长度，以Byte为单位
返回值	APP_DRV_xxx: 详见SDK_Folder\drivers\inc\app_drv_error.h宏定义
备注	

3.3.2.12 app_qspi_dma_transmit_async_ex

表 3-20 app_qspi_dma_transmit_async_ex接口

函数原型	uint16_t app_qspi_dma_transmit_async_ex(app_qspi_id_t id, uint32_t qspi_mode, uint32_t data_width, uint8_t *p_data, uint32_t length)
功能说明	QSPI以DMA方式（异步）发送数据，可设置时序模式和数据位宽
输入参数	• id: QSPI模块ID • qspi_mode: 数据传输采用的时序模式，可选QSPI_DATA_MODE_SPI（标准SPI模式）、QSPI_DATA_MODE_DUALSPI（Dual SPI模式）、QSPI_DATA_MODE_QUADSPI（Quad SPI模式） • data_width: 数据位宽（支持QSPI_DATASIZE_08_BITS、QSPI_DATASIZE_16_BITS、QSPI_DATASIZE_32_BITS） • p_data: 需要发送数据的buffer • length: 待发送数据的长度，以Byte为单位
返回值	APP_DRV_xxx: 详见SDK_Folder\drivers\inc\app_drv_error.h宏定义
备注	

3.3.2.13 app_qspi_receive_sync_ex

表 3-21 app_qspi_receive_sync_ex接口

函数原型	uint16_t app_qspi_receive_sync_ex(app_qspi_id_t id, uint32_t qspi_mode, uint32_t data_width, uint8_t *p_data, uint32_t length, uint32_t timeout, uint32_t timeout)
功能说明	QSPI轮询方式（同步）接收数据，可设置时序模式和数据位宽
输入参数	- id: QSPI模块ID - qspi_mode: 数据传输采用的时序模式，可选QSPI_DATA_MODE_SPI（标准SPI模式）、QSPI_DATA_MODE_DUALSPI（Dual SPI模式）、QSPI_DATA_MODE_QUADSPI（Quad SPI模式） - data_width: 数据位宽（支持QSPI_DATASIZE_08_BITS、QSPI_DATASIZE_16_BITS、QSPI_DATASIZE_32_BITS） - p_data: 需要接收数据的buffer - length: 待接收数据的长度，以Byte为单位 - timeout: 超时时间，以ms为单位
返回值	APP_DRV_xxx: 详见SDK_Folder\drivers\inc\app_drv_error.h宏定义
备注	

3.3.2.14 app_qspi_receive_async_ex

表 3-22 app_qspi_receive_async_ex接口

函数原型	uint16_t app_qspi_receive_async_ex(app_qspi_id_t id, uint32_t qspi_mode, uint32_t data_width, uint8_t *p_data, uint32_t length)
功能说明	QSPI中断方式（异步）接收数据，可设置时序模式和数据位宽
输入参数	- id: QSPI模块ID - qspi_mode: 数据传输采用的时序模式，可选QSPI_DATA_MODE_SPI（标准SPI模式）、QSPI_DATA_MODE_DUALSPI（Dual SPI模式）、QSPI_DATA_MODE_QUADSPI（Quad SPI模式） - data_width: 数据位宽（支持QSPI_DATASIZE_08_BITS、QSPI_DATASIZE_16_BITS、QSPI_DATASIZE_32_BITS） - p_data: 需要接收数据的buffer - length: 待接收数据的长度，以Byte为单位
返回值	APP_DRV_xxx: 详见SDK_Folder\drivers\inc\app_drv_error.h宏定义
备注	

3.3.2.15 app_qspi_dma_receive_async_ex

表 3-23 app_qspi_dma_receive_async_ex接口

函数原型	uint16_t app_qspi_dma_receive_async_ex(app_qspi_id_t id, uint32_t qspi_mode, uint32_t data_width, uint8_t *p_data, uint32_t length)
功能说明	QSPI以DMA方式（异步）接收数据，可设置时序模式和数据位宽
输入参数	- id: QSPI模块ID - qspi_mode: 数据传输采用的时序模式，可选QSPI_DATA_MODE_SPI（标准SPI模式）、QSPI_DATA_MODE_DUALSPI（Dual SPI模式）、QSPI_DATA_MODE_QUADSPI（Quad SPI模式）

	• data_width: 数据位宽（支持QSPI_DATASIZE_08_BITS、QSPI_DATASIZE_16_BITS、QSPI_DATASIZE_32_BITS） • p_data: 需要接收数据的buffer • length: 待接收数据的长度，以Byte为单位
返回值	APP_DRV_xxx: 详见SDK_Folder\drivers\inc\app_drv_error.h宏定义
备注	

3.3.3 内存映射模式类接口

除了工作在寄存器模式外，NOR Flash和PSRAM 设备可以配置为内存映射模式，使访问更加简洁高效。

3.3.3.1 app_qspi_config_memory_mapped

表 3-24 app_qspi_config_memory_mapped接口

函数原型	bool app_qspi_config_memory_mapped(app_qspi_id_t id, app_qspi_mmap_device_t dev);
功能说明	主要用于块数据的发送
输入参数	• id: QSPI 模块ID • dev: 内存映射模式设备配置
返回值	true/false
备注	

结构体app_qspi_mmap_device_t定义及注解

```
typedef struct {
    app_qspi_device_e                dev_type;
    app_qspi_psram_mmap_wr_cmd_e     psram_wr;
    union {
        app_qspi_flash_mmap_rd_cmd_e flash_rd;
        app_qspi_psram_mmap_rd_cmd_e psram_rd;
    } rd;
    void * set;
} app_qspi_mmap_device_t;
```

dev_type

- APP_QSPI_DEVICE_FLASH: NOR Flash 设备
- APP_QSPI_DEVICE_PSRAM: PSRAM 设备

psram_wr

若为PSRAM设备，指定Write指令，否则可不用设置，支持：

- PSRAM_MMAP_CMD_QWRITE_02H
- PSRAM_MMAP_CMD_QWRITE_38H

rd

若为NOR Flash设备，配置 flash_rd 读命令：

- FLASH_MMAP_CMD_DREAD_3BH: DualSPI下3B命令
- FLASH_MMAP_CMD_2READ_BBH: DualSPI下BB命令
- FLASH_MMAP_CMD_QREAD_6BH: QuadSPI下6B命令
- FLASH_MMAP_CMD_4READ_EBH: QuadSPI下EB命令

若为PSRAM设备，配置psram_rd读命令：

- PSRAM_MMAP_CMD_QREAD_0BH: QuadSPI下0B命令
- PSRAM_MMAP_CMD_QREAD_EBH: QuadSPI下EB命令

3.3.3.2 app_qspi_active_memory_mapped

表 3-25 app_qspi_active_memory_mapped接口

函数原型	bool app_qspi_active_memory_mapped(app_qspi_id_t id, bool is_active)
功能说明	用于内存映射模式的激活/去活
输入参数	• id: QSPI模块ID • is_active: 激活/去活内存映射模式
返回值	true/false
备注	

3.3.3.3 app_qspi_get_xip_base_address

表 3-26 app_qspi_get_xip_base_address接口

函数原型	uint32_t app_qspi_get_xip_base_address(app_qspi_id_t id)
功能说明	获取QSPI内存映射在系统总线的基地址
输入参数	id: QSPI模块ID
返回值	QSPI内存映射在系统总线的基地址
备注	

3.3.3.4 app_qspi_mmap_set_endian_mode

表 3-27 app_qspi_mmap_set_endian_mode接口

函数原型	bool app_qspi_mmap_set_endian_mode(app_qspi_id_t id, app_qspi_mmap_endian_mode_e mode)
功能说明	配置内存映射模式下设备的读取端序模式
输入参数	• id: QSPI模块ID • mode: 读取端序模式
返回值	true/false
备注	

3.3.4 特种功能接口

特种功能接口，主要组合QSPI/DMA的一些重要特性，在刷屏等应用场景抽象出特种高效接口。

3.3.4.1 app_qspi_async_draw_screen

表 3-28 app_qspi_async_draw_screen接口

函数原型	<pre>bool app_qspi_async_draw_screen (app_qspi_id_t screen_id, app_qspi_id_t storage_id, const app_qspi_screen_command_t * const p_screen_cmd, const app_qspi_screen_info_t * const p_screen_info, app_qspi_screen_scroll_t * p_scroll_config, bool is_first_call)</pre>
功能说明	异步绘屏接口
输入参数	• screen_id: 屏幕ID • storage_id: 指定素材所在的QSPI存储器ID • p_screen_cmd: 指向刷屏控制命令的指针 • p_screen_info: 屏幕信息 • p_scroll_config: 屏幕滚动控制信息 • is_first_call: 用户调用请设置为true
返回值	true/false
备注	• 如果图片素材不是在QSPI设备上，storage_id需设定为APP_STORAGE_RAM_ID • storage_id和screen_id不能相同 • 入参p_screen_info的scrn_pixel_depth只能设为2，即该接口只能处理16 bits的显示输出 • 驱屏接口单独抽离在SDK_Folder\drivers\inc\app_graphics_qspi.h

3.3.4.2 app_qspi_async_veri_draw_screen

表 3-29 app_qspi_async_veri_draw_screen接口

函数原型	<pre>bool app_qspi_async_veri_draw_screen(app_qspi_id_t screen_id, app_qspi_id_t storage_id, const app_qspi_screen_command_t * const p_screen_cmd, const app_qspi_screen_info_t * const p_screen_info, app_qspi_screen_veri_link_scroll_t * p_link_scroll, bool is_first_call)</pre>
功能说明	异步链式刷屏专用接口（垂直链）
输入参数	• screen_id: 屏幕ID • storage_id: 指定素材所在的QSPI存储器ID • p_screen_cmd: 指向刷屏控制命令的指针 • p_screen_info: 屏幕信息 • p_link_scroll: 链式滚动参数配置 • is_first_call: 用户调用设置为true
返回值	• true: 成功

	• false: 失败
备注	• 如果图片素材不是在QSPI设备上，storage_id需设置为APP_STORAGE_RAM_ID • storage_id和screen_id不能相同 • 入参p_screen_info的scrn_pixel_depth只能设置为2，即该接口只能处理16 bits的显示输出 • 驱屏接口单独抽离在SDK_Folder\drivers\inc\app_graphics_qspi.c中

3.3.4.3 app_qspi_mmap_blit_image

表 3-30 app_qspi_mmap_blit_image接口

函数原型	<pre>bool app_qspi_mmap_blit_image(app_qspi_id_t storage_id, blit_image_config_t * p_blit_config, blit_xfer_type_e xfer_type)</pre>
功能说明	使用DMA进行二维数据搬运接口
输入参数	• storage_id: 指定素材所在的QSPI存储器ID • p_blit_config: blit传输配置 • xfer_type: 传输类型，支持SG和LLP
返回值	• true: 成功 • false: 失败
备注	• 驱屏接口单独抽离在SDK_Folder\drivers\inc\app_graphics_qspi.c中 • 使用该接口，需要将app_qspi_user_config.h中的宏定义QSPI_BLIT_RECT_IMAGE_SUPPORT设置为大于0的值（默认为0）

3.3.4.4 app_qspi_async_llp_draw_block

表 3-31 app_qspi_async_llp_draw_block接口

函数原型	<pre>bool app_qspi_async_llp_draw_block(app_qspi_id_t screen_id, app_qspi_id_t storage_id, const app_qspi_screen_command_t *const p_screen_cmd, const app_qspi_screen_info_t *const p_screen_info, app_qspi_screen_block_t *p_block_info, bool is_first_call)</pre>
功能说明	使用DMA-LLP（DMA链式）方式刷新屏幕的块区域，链表的节点为待刷新图像数据块的每行数据
输入参数	• screen_id: 屏幕ID • storage_id: 指定素材所在的QSPI存储器ID • p_screen_cmd: 指向刷屏控制命令的指针 • p_screen_info: 屏幕信息 • p_block_info: 刷屏的图像区域块 • is_first_call: 是否第一次调用，第一次调用设置为true（设置为true表明一帧图像的开始，需要发送命令字节）
返回值	• true: 成功 • false: 失败
备注	当传输结束，会产生APP_QSPI_EVT_TX_CPLT事件

3.4 QSPI使用建议

1. QSPI的时钟源为系统外设时钟。分频基于此时钟进行，分频系数支持2 ~ 65535之间的偶数。
2. 一般在SRAM和工作在内存映射模式下的QSPI设备移动数据时，如果数据量小于1 KB，可优先使用memcpy函数进行；如果数据量大于1 KB，建议使用DMA进行搬运。数据量越大，DMA会展现越大的优势
3. 在内存映射模式下访问数据时，如果需要使用memcpy/memset函数，Keil工程下请优先编译使用系统标准库，而不要选用microlib；前者能更好的激活总线Burst传输行为，获得更好的访问效率
4. 部分QSPI设备时序上需要较大的Tcsu（CS Setup Delay Time），否则不能获得正确稳定的访问数据：
 - 如果设备支持时钟模式3，可尝试设置时钟模式3。时钟模式3会比模式0有更大的Tcsu
 - 可通过寄存器配置增加Tcsu时间
5. 一般QSPI PSRAM 设备需要定时释放片选以完成内部数据保持电路的自刷新，最长的片选时间为 t_{CEM} ，尤其对于写访问，需要严格遵守。
 - 内存映射模式下写访问时，从设计上考虑了 t_{CEM} ，不用再关注此参数
 - 寄存器模式下，从驱动层考虑了 t_{CEM} ，但寄存器模式接口复杂度较高。建议QSPI PSRAM设备不在寄存器模式下使用
6. QSPI 存在预取模式，当进行QSPI读访问时，可以进一步提高QSPI的读效率，但此模式只能配合DMA工作，不能在CPU访问时开启。
7. GR5525在设计时对各个QSPI的FIFO进行了差异化优化，一般情况下外设混接QSPI不存在问题；同时，GR5525在设计时也对DMA的FIFO进行了差异化优化，为了发挥GR5525的最佳效率，使用建议如下：
 - QSPI0优先用于连接Flash设备，可搭配DMA0的通道0工作
 - QSPI1优先用于连接PSRAM设备，可搭配DMA0/DMA1的通道0工作
 - QSPI2优先用于连接Display设备，可搭配DMA1的通道0工作
 - 请将DMA0/DMA1的通道0优先留给QSPI0/QSPI1/QSPI2 等高速外设使用，且尽量不同时混用

4 显示控制和渲染逻辑

4.1 显示屏时序

GR5525提供QSPI、Display SPI和SPIM模块用于外接显示屏，支持的屏幕接口规范主要为MIPI DBI（Display Bus Interface）Type-C，涵盖的时序协议主要包括：

- SPI（8-bit版本）**

包含 SPI_CS（片选信号）、SPI_SCLK（时钟信号）、SPI_SD（数据输出）三根信号线，控制命令、地址和色彩数据均通过SPI_SD 信号线发送。
- 4-Wire SPI（扩展DCX信号线）**

包含 SPI_CS（片选信号）、SPI_SCLK（时钟信号）、SPI_SD（数据输出）、和SPI_DC（数据/命令指示信号）4根信号线，其中命令字（Command Word）由8-bit构成，而数据字（Data Word）由若干位像素数据位构成。

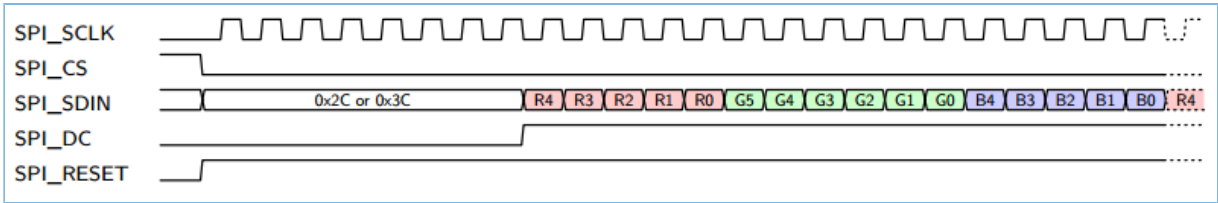


图 4-1 4-Wire SPI - RGB565 - Option0格式参考时序

- Dual SPI**

包含 SPI_CS（片选信号）、SPI_SCLK（时钟信号）、SPI_SD（数据输出）、和SPI_SD1（数据输出1）4根信号线，命令字（Command Word）一般由SPI_SD线发送而数据由 SPI_SD和SPI_SD1共同发送；其中命令字（Command Word）由9-bit构成，包含1-bit的数据/命令指示位和8-bit的命令（Command）；而数据字（Data Word）由1-bit数据/命令指示位和若干位像素数据位构成。

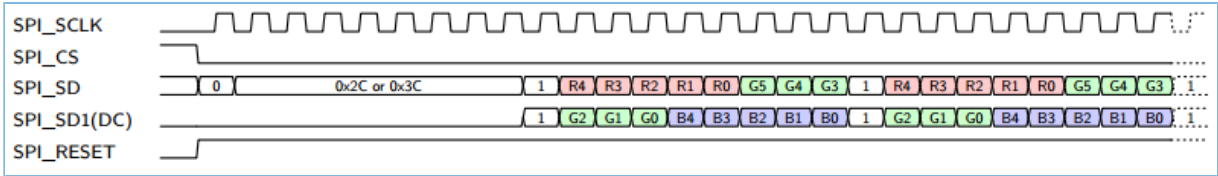


图 4-2 Dual SPI - RGB565 - Option0格式参考时序

- Quad SPI**

包含 SPI_CS（片选信号）、SPI_SCLK（时钟信号）、SPI_SD（数据输出）、SPI_SD1（数据输出1）、SPI_SD2（数据输出2）、SPI_SD3（数据输出3）6根信号线，命令类型可以被SPI_SD或所有SPI_SDx传输。时序一般由 8-bit命令头、24-bit命令和若干像素数据组成。

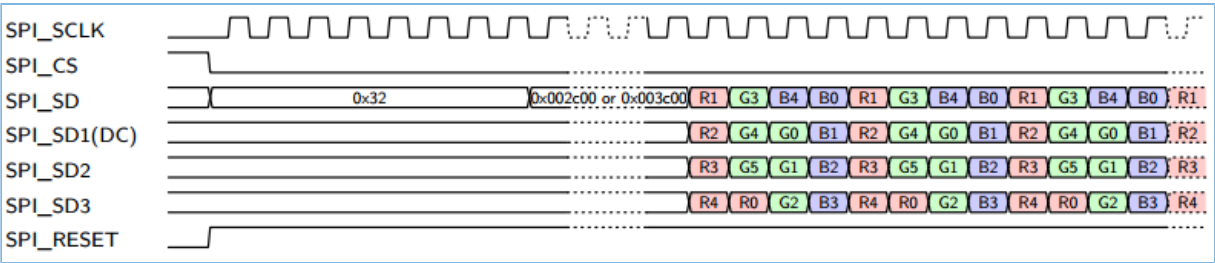


图 4-3 Quad SPI - RGB565 - Option0格式参考时序

4.2 色彩格式支持

4.2.1 色彩和色深

色彩（即RGB色彩）是由红色（Red）、绿色（Green）和蓝色（Blue）三个分量组成的三元组，分量决定颜色强度或亮度，不同颜色格式用于表示RGB元组分量的色彩深度（色深）也不同。色深是指帧缓冲存储的用于描述每种颜色的位数，也可以表示为每像素位数（bpp）。较高的色深意味着更精细的颜色细节和更多的颜色选择。

GR5525支持的色彩格式为RGB888、RGB565，对应色彩分量和色深如下：

1. RGB888（24-bit色深）：每个颜色分量都用8 bits表示，共24 bits，即24 bpp，所以RGB888可以表示16777216（ 2^{24} ）种不同颜色。分量范围为0 ~ 255，分量越大表示颜色强度/亮度越大，也就是“0”表示无颜色，255表示颜色处于最大值。
2. RGB565（16-bit色深）：R和B分量均用5 bits表示，分量范围为0 ~ 31；G分量则用6 bits表示，分量范围为0 ~ 63。所以，RGB565可以表示65536（ 2^{16} ）种不同颜色。

如下所示，全黑色的RGB色彩为（0,0,0），全白色的RGB色彩为（255,255,255），亮绿色为（0,255,0）。

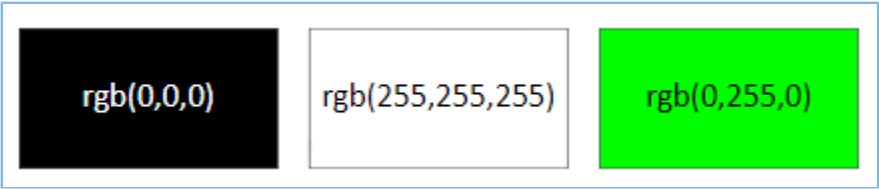


图 4-4 RGB元组分量效果图

对于1 bpp格式，只能表示2（ 2^1 ）种不同色彩，适用于只有黑色和白色的应用。

GR5525应用于穿戴产品时，建议优先使用RGB565（65K色）色彩格式。

Pixel 0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0
Byte 1								Byte 0							

图 4-5 RGB565 16-bit格式

4.2.2 Alpha混合

Alpha混合是一种常用的图像处理技术，用于实现图像特效和图像合成，通过控制图像中每个像素的透明度将多个图像叠加在一起。每个像素的透明度值被用来决定两个图像混合的比例，从而实现不同程度的透明效果。

使用Alpha混合技术可以创建各种视觉效果，例如半透明遮罩、图像渐变、阴影等。通过调整不同图像的透明度，可以获得实现更丰富多样的效果。

图片的Alpha混合需要消耗较多的运算周期，在渲染过程中会长时间占用CPU资源。因此，建议UI/UX工程师使用设计技巧来达到良好的图形效果，或者只针对特殊场景或者小范围的图片使用Alpha混合计算。

4.3 帧缓冲区

4.3.1 基础概念

帧缓冲区（Frame Buffer）是用于存储渲染结构的一段内存区域，一般从SRAM空间申请。作为渲染逻辑和显示逻辑的交互枢纽，其交互流程如下：

1. CPU等算力设备将图片资源从Flash等外设加载到帧缓冲区，通过GUI框架再按照设计的UI/UX原则进行运算。
2. 等待帧画面运算完成后，使用DMA等搬运模块将帧缓冲区的数据通过QSPI等显示外设接口，送到外部屏幕进行显示。

帧缓冲区的基本单位是像素（Pixel）。一般可以按照二维数组的数据结构方式来组织帧缓冲区，通过以下三个主要参数构建一个帧缓冲区：

- 宽度：显示区域的横向像素数量。一般情况下，该值和产品中显示设备的宽度相同。特殊场景除外，比如显示设备的宽度是奇数，但为了运算方便帧缓冲区宽度可能取偶数。
- 高度：显示区域的竖向像素数量。该值根据GUI框架和SRAM空间而定，一般不大于产品显示设备的高度。
- 深度：颜色格式的像素深度，表示一个像素占用多少字节。该值可以作为二维数组元素的基本单位，例如RGB565的深度就等于2 Bytes。

帧缓冲区的宽度、高度、深度共同决定其占用空间大小，计算公式为：Frame Buffer Size = Width x Height x Depth。

帧缓冲区也可以用平面坐标的方式进行索引，左上角为坐标原点，参考如下：

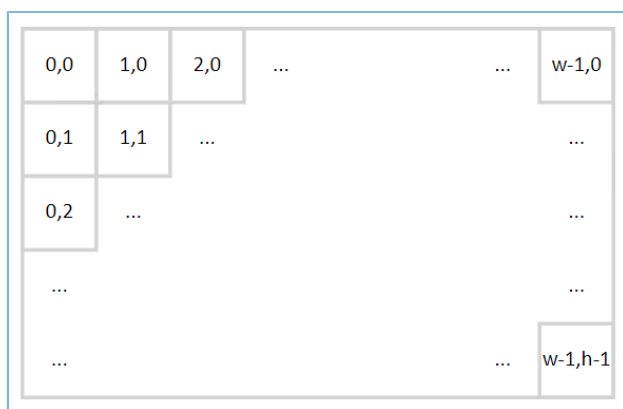


图 4-6 帧缓冲区坐标索引参考

4.3.2 帧缓冲区数量

图形软件架构下所需的帧缓冲区数量由产品显示屏分辨率、颜色格式深度、可用SRAM空间、显示帧率等主要参数共同决定。一般情况下，有下面几种设计参考：

- 一个完整缓冲区

该逻辑架构设计最为简单：一个帧缓冲区要容纳待传输到显示屏的所有像素；渲染、刷屏依次串行进行，即渲染时候刷新屏幕逻辑等待，反之亦然。



图 4-7 刷屏和渲染交互逻辑（一个完整缓冲区）

如果对显示帧率要求不高或者屏幕较小有足够的SRAM提供完整的帧缓冲区，可以考虑该设计方案。由于刷屏和渲染不能同时进行，这种架构不能获得更好的帧率性能，因此较少采用。

- 两个完整缓冲区

刷屏和渲染可以同时进行，并行任务能充分利用芯片性能，很好地提高刷新帧率，该方案对SRAM Size的要求较高。GR5525 SRAM资源有限，因此对于一定分辨率以上的显示屏无法提供两个完整的帧缓冲区。

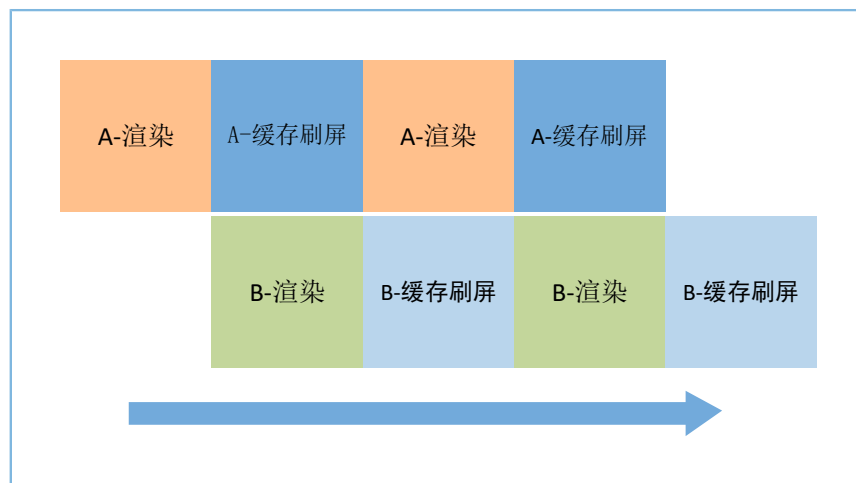


图 4-8 刷屏和渲染交互逻辑（两个完整缓冲区）

- 两个局部缓冲区

局部缓冲区架构下，渲染和刷屏可以并行工作以提高帧率性能、节省SRAM 资源。

每次渲染和刷新的区域为（显示宽度 * 局部高度 * 像素深度），需要若干次渲染刷新才能完成一屏数据的显示，软件架构较为复杂。

如果显示屏幕自身不带SRAM显存，这种方案有一定局限性。

4.3.3 参考设计

4.3.3.1 GUI框架

GR5525应用于穿戴产品等显示设备时，用户可以根据产品需求自行构建一套GUI框架，主要考虑以下内容：

- 基础的周期性调度框架
- 渲染、刷屏的并行设计
- 事件输入设备（Touch/Key）的数据处理
- 是否构建简单的图形组件
- 对资源的合理利用
- 最大化芯片的性能和使用率

此外，用户也可以选用第三方开源的GUI框架，比如 [LVGL](#)。LVGL是一个针对嵌入式设备的图形框架，在有限资源的芯片上可以实现较好的显示效果。但是，原生LVGL对UI控件的实现依然会较多消耗嵌入式设备宝贵的算力和SRAM资源，实际应用时用户也要按需优化。GR5525 SDK包提供了深度优化的LVGL版本，针对穿戴设备可以实现更好的帧率，供用户进行软件设计参考。

4.3.3.2 穿戴示例（Watch Demo）工程

GR5525 SDK提供了一个面向穿戴设备的示例工程（SDK_Folder\projects\peripheral\graphics\graphics_lvgl_831_std_demo），可以通过GProgrammer下载到GR5525 SK板运行。示例工程主要涵盖以下方面，供用户参考借鉴：

说明:

SDK_Folder为GR5525 SDK的根目录。

- 对LVGL的移植和优化
- 基于LVGL进行的UI/UX 参考界面设计
- 扩展的自定义设计和窗口管理系统
- 外设的驱动和使用参考
- 蓝牙功能的使用